



DMK
ИЗДАТЕЛЬСТВО


ЛАНЬ®

Кольер Р., Монтонен К., Азарми Б.

Машинное обучение в Elastic Stack



Рич Коьер, Камилла Мондони, Бахаалдин Азарми

Машинное обучение в Elastic Stack



Machine Learning with the Elastic Stack



Gain valuable insights from your data
with Elastic Stack's machine
learning features

Rich Collier
Camilla Montonen
Bahaaldine Azarmi



Packt

BIRMINGHAM – MUMBAI

Машинное обучение в Elastic Stack

Получите максимальную отдачу от ваших данных благодаря уникальному сочетанию передовых технологий

Рич Кольер
Камилла Монтонен
Бахаалдин Азарми



Москва, 2022

УДК 004.4
ББК 32.972
К62



Кольер Р., Монтонен К., Азарми Б.

К62 Машинное обучение в Elastic Stack / пер. с англ. В. С. Яценкова. – М.: ДМК Пресс, 2021. – 380 с.: ил.

ISBN 978-5-93700-107-8

В книге подробно рассматривается работа с Elastic Stack – обширной экосистемой компонентов, которые служат для сбора, поиска и обработки данных. Вы ознакомитесь с общими принципами машинного обучения, узнаете о методах автоматического обнаружения аномалий, проверке целостности и анализа данных из разрозненных источников, научитесь истолковывать результаты обнаружения и прогнозирования аномалий и использовать их в своих целях, а также выполнять анализ временных рядов для различных типов данных.

Издание адресовано специалистам, которые работают с данными и хотят интегрировать машинное обучение с эффективными приложениями для мониторинга, обеспечения безопасности и аналитики в области данных.



УДК 004.4
ББК 32.972

First published in the English language under the title 'Machine Learning with the Elastic Stack. Second Edition (978-1-80107-003-4). The Russian-Language 1st edition Copyright © 2021 by DMK Press Publishing under license by No Starch Press Inc. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-1-80107-003-4 (англ.)
ISBN 978-5-93700-107-8 (рус.)

© Packt Publishing, 2021
© Перевод, оформление, издание,
ДМК Пресс, 2021

Содержание

https://t.me/it_books

От издательства	12
Об авторах	13
О рецензентах	14
Предисловие	15
Часть I. ЗНАКОМСТВО С МАШИНЫМ ОБУЧЕНИЕМ И ELASTIC STACK	18
Глава 1. Машинное обучение в информационных технологиях	19
Преодоление исторических вызовов в IT.....	19
Что нам делать с потоком данных?.....	20
Причины появления автоматического обнаружения аномалий.....	21
Машинное обучение без учителя и с учителем.....	23
Использование машинного обучения без учителя для обнаружения аномалий	24
Что такое необычность?.....	24
Изучение того, что является нормой.....	26
Вероятностные модели	26
Обучение моделей	27
Выявление и устранение тенденций	30
Оценка степени необычности	31
Роль времени	32
Применение машинного обучения с учителем в аналитике фреймов данных	33
Процесс обучения с учителем.....	33
Заключение	35
Глава 2. Подготовка и использование Elastic ML	36
Технические требования.....	36
Включение функций Elastic ML.....	36
Включение машинного обучения в собственном кластере	37
Включение машинного обучения в облаке – Elasticsearch Service.....	39
Обзор операционализации Elastic ML	46
Узлы ML.....	46
Задания.....	47
Сегментирование данных в анализе временных рядов	48

Загрузка данных в Elastic ML	49
Служебные хранилища	51
.ml-config.....	51
.ml-state-*	51
.ml-notifications-*	52
.ml-annotations-*	52
.ml-stats-*	52
.ml-anomalies-*	52
Оркестровка обнаружения аномалий.....	52
Снимки модели обнаружения аномалий.....	53
Заключение	54

Часть II. АНАЛИЗ ВРЕМЕННЫХ РЯДОВ – ОБНАРУЖЕНИЕ И ПРОГНОЗИРОВАНИЕ АНОМАЛИЙ

Глава 3. Обнаружение аномалий	56
Технические требования.....	56
Типы заданий Elastic ML.....	56
Устройство детектора	58
Функция	59
Поле.....	59
Поле partition	59
Поле by	60
Поле over	60
Формула детектора.....	60
Обнаружение изменений частотности событий.....	61
Подробнее о функциях count	61
Другие функции подсчета	73
Ненулевой подсчет	73
Раздельный подсчет.....	74
Обнаружение изменений значений показателей.....	75
Метрические функции	76
min, max, mean, median и metric	76
varp.....	76
sum и non-null sum.....	76
Обзор расширенных функций детектора.....	77
Функция rare.....	78
Функция freq_rare	79
Функция info_content.....	79
Функции геолокации.....	79
Функции времени.....	80
Разделение анализа по категориальным признакам.....	80
Настройка поля разделения	80
Разница между разделением с использованием partition и by_field	82
Является ли двойное разделение пределом возможного?.....	83
Обзор временного и популяционного анализов.....	84

Категоризация в анализе неструктурированных сообщений.....	86
Типы сообщений, подходящие для категоризации.....	88
Предварительная категоризация.....	88
Анализ категорий.....	89
Пример задания по категоризации.....	90
Когда не следует использовать категоризацию.....	94
Управление Elastic ML через API.....	95
Заключение.....	97
Глава 4. Прогнозирование.....	98
Технические требования.....	98
Ключевое различие между предсказаниями и пророчествами.....	98
Для чего применяется прогнозирование?.....	100
Как работает прогнозирование?.....	100
Прогнозирование одиночного временного ряда.....	103
Просмотр результатов прогнозирования.....	114
Прогнозирование нескольких временных рядов.....	119
Заключение.....	122
Глава 5. Интерпретация результатов.....	123
Технические требования.....	123
Просмотр хранилища результатов Elastic ML.....	123
Оценка аномалий.....	128
Оценка на уровне сегмента.....	129
Нормализация.....	131
Оценка на уровне фактора влияния.....	131
Факторы влияния.....	133
Оценка на уровне записи.....	135
Описание схемы хранилища результатов.....	136
Результаты на уровне сегмента.....	137
Результаты на уровне записи.....	140
Результаты на уровне факторов влияния.....	143
Аномалии в нескольких сегментах.....	145
Пример аномалии в нескольких сегментах.....	145
Оценка аномалии в нескольких сегментах.....	146
Результаты прогноза.....	148
Запрос результатов прогноза.....	149
API результатов Elastic ML.....	151
Конечные точки API результатов.....	152
API обобщения сегментов.....	152
API категорий.....	153
Пользовательские панели мониторинга и рабочие панели Canvas.....	155
Панель инструментов встраивания.....	155
Аномалии как аннотации в TSVB.....	156
Настройка рабочих панелей Canvas.....	159
Заключение.....	162

Глава 6. Создание и использование оповещений	163
Технические требования.....	163
Определение и принцип работы оповещений.....	164
Аномалии не обязательно нуждаются в оповещениях.....	164
Точное время имеет значение.....	165
Создание оповещений из интерфейса машинного обучения.....	168
Определение заданий по обнаружению аномалий.....	168
Создание оповещений для пробных заданий.....	174
Моделирование аномального поведения в реальном времени.....	179
Получение и просмотр оповещений.....	180
Создание оповещений с помощью Watcher.....	183
Использование устаревшего варианта watch.....	183
trigger.....	184
input.....	184
condition.....	187
action.....	188
Пользовательские шаблоны watch с уникальной функциональностью.....	189
Связанный ввод и сценарий условий.....	189
Передача информации между связанными входами.....	190
Заключение.....	191
Глава 7. Выявление истинных причин аномалий	192
Технические требования.....	192
Настоящее значение термина AIOps.....	192
Значимость и ограничения KPI.....	194
Выходя за рамки KPI.....	197
Организация данных для анализа.....	198
Настраиваемые запросы для каналов данных.....	199
Дополнение получаемых данных.....	202
Использование контекстной информации.....	203
Аналитическое разделение.....	203
Статистические факторы влияния®.....	204
Анализ первопричин аномалии.....	205
История проблемы.....	205
Корреляция и общие факторы влияния.....	207
Заключение.....	212
Глава 8. Другие приложения Elastic Stack для обнаружения аномалий	213
Технические требования.....	213
Обнаружение аномалий в Elastic APM.....	213
Включение обнаружения аномалий для APM.....	214
Просмотр результатов задания по обнаружению аномалий.....	219
Создание заданий машинного обучения с помощью распознавателя данных.....	222
Обнаружение аномалий в приложении Logs.....	224

Категории журналов.....	224
Журнал аномалий.....	225
Обнаружение аномалий в приложении Metrics.....	227
Обнаружение аномалий в приложении Uptime.....	230
Обнаружение аномалий в приложении Elastic Security.....	233
Готовые задания по обнаружению аномалий.....	233
Оповещения на основе заданий обнаружения аномалий.....	235
Заключение.....	237

Часть III. АНАЛИЗ ФРЕЙМОВ ДАННЫХ..... 238

Глава 9. Введение в анализ фреймов данных..... 239

Технические требования.....	240
Основы преобразования данных.....	240
Чем полезны преобразования?.....	240
Анатомия преобразований.....	241
Использование преобразований для анализа заказов интернет-магазина.....	242
Более сложные конфигурации сводной таблицы и агрегирования.....	246
Различие между пакетными и непрерывными преобразованиями.....	248
Анализ данных социальных сетей с помощью непрерывных преобразований.....	250
Использование Painless для расширенных конфигураций преобразования.....	253
Знакомство с Painless.....	254
Переменные, операторы и управление выполнением.....	255
Функции.....	260
Совместное использование Python и Elasticsearch.....	263
Краткий обзор клиентов Python Elasticsearch.....	264
Зачем нам нужен Eland?.....	266
Знакомство с Eland.....	267
Заключение.....	269
Дополнительная литература.....	270

Глава 10. Обнаружение выбросов..... 272

Технические требования.....	273
Принцип работы механизма обнаружения выбросов.....	273
Обзор четырех методов обнаружения выбросов.....	274
Методы, основанные на расстоянии.....	274
Методы, основанные на плотности.....	275
Оценка влияния характеристики.....	276
Как рассчитывается оценка влияния характеристик для каждой точки?.....	277
Чем обнаружение выбросов отличается от обнаружения аномалий?.....	278
Сравнение вероятностных моделей и экземпляров.....	278
Подсчет оценок.....	279

Характеристики данных.....	279
Потоковая и пакетная обработка.....	279
Применение обнаружения выбросов на практике.....	280
Оценка качества обнаружения выбросов с помощью API Evaluate.....	285
Настройка гиперпараметров для обнаружения выбросов.....	290
Заключение.....	293
Глава 11. Классификационный анализ.....	294
Технические требования.....	295
Классификация: от данных к обученной модели.....	295
Классифицирующие модели учатся на данных.....	296
Конструирование признаков.....	298
Оценка модели.....	299
Простой пример классификации.....	300
Деревья решений с градиентным усилением.....	307
Введение в деревья решений.....	308
Градиентное усиление.....	309
Гиперпараметры.....	309
Интерпретация результатов.....	313
Вероятность класса.....	314
Оценка класса.....	314
Важность признака.....	314
Заключение.....	316
Дополнительная литература.....	317
Глава 12. Регрессия.....	318
Технические требования.....	318
Использование регрессионного анализа для прогнозирования цен на жилье.....	319
Использование деревьев решений в регрессионном анализе.....	326
Заключение.....	329
Дополнительная литература.....	329
Глава 13. Логический вывод моделей.....	330
Технические требования.....	330
Поиск, импорт и экспорт обученных моделей с помощью API.....	331
Обзор API обученных моделей.....	331
Экспорт и импорт обученных моделей с помощью API и Python.....	333
Обработчики логического вывода и конвейеры данных.....	336
Обработка отсутствующих или поврежденных данных в конвейерах.....	345
Получение развернутой информации о прогнозах.....	347
Импорт внешних моделей с помощью eLand.....	348
Кратко о поддержке внешних моделей в eLand.....	349
Обучение DecisionTreeClassifier и импорт в Elasticsearch с помощью eLand.....	349
Заключение.....	353

Приложение. Советы по обнаружению аномалий	354
Технические требования.....	354
Роль факторов влияния в разделенных и неразделенных заданиях	354
Использование односторонних функций	361
Исключение определенных интервалов времени	363
Исключение наступающего (известного) интервала времени	364
Создание события календаря	364
Остановка и запуск потока данных в нужное время	365
Исключение интервала времени постфактум.....	366
Клонирование задания и повторный запуск исторических данных.....	366
Возврат задания к предыдущему снимку модели.....	366
Использование настраиваемых правил и фильтров	368
Создание собственных правил	369
Использование настраиваемых правил для оповещения «сверху вниз».....	370
Соображения относительно пропускной способности заданий.....	371
О вреде излишней сложности сценариев.....	372
Обнаружение аномалий в вычисляемых полях	373
Заключение	376
Предметный указатель	377



От издательства



Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Ракет очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторах



Рич Кольер (Rich Collier) – архитектор решений в Elastic. Он присоединился к команде Elastic после приобретения Prekert. Рич имеет более чем 20-летний опыт работы в качестве архитектора решений и системного инженера предпродажной подготовки программного обеспечения, оборудования и сервисных решений. Профессиональные интересы Рича включают аналитику больших данных, машинное обучение, обнаружение аномалий, обнаружение угроз, обеспечение безопасности, управление производительностью приложений, веб-приложения и технологии контакт-центров. Рич проживает в Бостоне, штат Массачусетс.

Камилла Монтонен (Camilla Montonen) – старший инженер по машинному обучению в Elastic.

Бахаалдин Азарми (Bahaaldine Azarmi), или коротко Баха, – архитектор решений в Elastic. До этого Баха был соучредителем ReachFive, платформы маркетинговых данных, ориентированной на поведение пользователей и социальную аналитику. Баха также сотрудничал с различными поставщиками программного обеспечения, такими как Talend и Oracle, где он занимал должности архитектора решений и системного архитектора. Баха является автором нескольких книг, в том числе *Learning Kibana 5.0*, *Scalable Big Data Architecture* и *Talend for Big Data*. Живет в Париже и имеет степень магистра компьютерных наук Парижского технологического института.



О рецензентах

Апурва Джоши (Apoorva Joshi) в настоящее время работает специалистом по безопасности данных в Elastic (ранее Elasticsearch), где она занимается применением машинного обучения для обнаружения вредоносных программ в конечных точках системы. До Elastic работала научным сотрудником в FireEye, где исследовала применение машинного обучения в задачах безопасности электронной почты. У нее разностороннее инженерное образование: бакалавр электротехники и магистр информационных технологий (с акцентом на машинное обучение).

Лицзюань Чжун (Lijuan Zhong) – опытный инженер в области технологий Elastic и облачных вычислений. У нее степень магистра информационных технологий и почти 20-летний опыт работы в сфере информационных технологий и телекоммуникаций. Сейчас сотрудничает с Netnordic – основным партнером Elastic в Швеции. Она начала свой путь в Elastic в 2019 году и стала сертифицированным инженером Elastic, также прошла курс машинного обучения Стэнфордского университета. Возглавляет множество образовательных программ и проектов, связанных с Elastic и машинным обучением, и клиенты всегда очень довольны результатом. Она была организатором конференции Elastic Stockholm в 2020 г., приняла участие в конференции сообщества Elastic в 2021 г. и выступила с докладом о машинном обучении с помощью Elastic Stack. В 2021 году была удостоена бронзовой медали за вклад в развитие Elastic.



Предисловие



Elastic Stack, ранее известный как ELK Stack, представляет собой комплексное решение для анализа журналов, которое помогает пользователям эффективно получать, обрабатывать и анализировать данные поиска. Благодаря применению машинного обучения – ключевой особенности решения – Elastic Stack делает этот процесс еще более эффективным. Эта книга содержит всесторонний обзор функций машинного обучения Elastic Stack как для анализа данных временных рядов, так и для классификации, регрессии и обнаружения выбросов.

Знакомство с экосистемой Elastic Stack начинается с интуитивно понятного объяснения концепций машинного обучения. Затем под руководством авторов вы выполните анализ временных рядов для различных типов данных, таких как файлы журналов, сетевые потоки, показатели приложений и финансовые данные. По мере прочтения глав вы научитесь использовать машинное обучение в Elastic Stack для ведения журнала, обеспечения безопасности и отслеживания показателей. Наконец, вы узнаете, как анализ фреймворков открывает доступ к совершенно новым сценариям использования данных, в которых вам поможет машинное обучение.

После прочтения этой книги вы приобретете практический опыт совместного использования технологии машинного обучения и Elastic Stack, а также знания, необходимые для включения машинного обучения в вашу платформу распределенного поиска и анализа данных.

Для кого эта книга

Если вы профессионал в области данных и хотите получить представление о технологиях Elasticsearch, не прибегая к помощи специалиста по машинному обучению и не разрабатывая собственные решения, то эта книга про совместное применение машинного обучения и Elastic Stack для вас. Вы также найдете эту книгу полезной, если хотите интегрировать машинное обучение с вашими приложениями для мониторинга, обеспечения безопасности и аналитики. Чтобы извлечь из данной книги максимальную пользу, необходимо знать и уметь применять на практике Elastic Stack.



Какие темы охватывает эта книга

Глава 1 служит введением в тему и справочным пособием по историческим проблемам ручного анализа данных в IT и технологиях безопасности. В этой главе также представлен всесторонний обзор базовых принципов работы

машинного обучения Elastic (Elastic ML), чтобы читатель получил полное представление о том, что происходит «за кулисами».

В *главе 2* объясняется, каким образом применяются возможности машинного обучения в Elastic Stack, а также подробно описывается теория работы алгоритмов Elastic ML. Кроме того, в этой главе дается подробное объяснение логики операций машинного обучения применительно к Elastic.

В *главе 3* подробно рассматриваются методы автоматического обнаружения аномалий с обучением без учителя, которые лежат в основе анализа временных рядов.

В *главе 4* показано, что сложные модели временных рядов Elastic ML можно использовать не только для обнаружения аномалий. Возможности прогнозирования позволяют пользователям экстраполировать тенденции и поведение в будущем, чтобы помочь в решении таких задач, как планирование мощности.

В *главе 5* рассказано, как детально истолковать результаты обнаружения и прогнозирования аномалий и использовать их в своих целях в визуализации данных, информационных панелях и инфографике.

В *главе 6* рассмотрены различные методы интеграции возможностей упреждающего уведомления Elastic и данных, полученных с помощью машинного обучения, чтобы сделать обнаружение аномалий еще более эффективным.

В *главе 7* показано, как использование Elastic ML для проверки целостности и анализа данных из разрозненных источников в коррелированных представлениях дает аналитику преимущество с точки зрения наследования подходов.

В *главе 8* объясняется, как обнаружение аномалий используется другими приложениями в Elastic Stack для повышения эффективности анализа данных.

В *главе 9* рассмотрен анализ фрейма данных, его отличие от обнаружения аномалий временных рядов и рассказано, какие инструменты доступны пользователю для загрузки, подготовки, преобразования и анализа данных с помощью Elastic ML.

В *главе 10* описано применение анализа фреймов в сочетании с Elastic ML в аналитике данных.

В *главе 11* говорится о возможности классификационного анализа фреймов данных в сочетании с Elastic ML.

В *главе 12* рассмотрено использование регрессионного анализа фреймов данных в сочетании с Elastic ML.

Глава 13 описывает использование обученных моделей машинного обучения для логического вывода – прогнозирования выходных значений во время реальной работы системы.

Приложение включает в себя множество практических советов, которые отчасти выходят за рамки других глав. Эти полезные советы помогут вам максимально эффективно использовать Elastic ML.

КАК ПОЛУЧИТЬ МАКСИМАЛЬНУЮ ОТДАЧУ ОТ ЭТОЙ КНИГИ



Чтобы получить максимальную отдачу от этой книги, вам понадобится компьютер с хорошим подключением к интернету и учетной записью Elastic.

СКАЧИВАНИЕ ИСХОДНОГО КОДА ПРИМЕРОВ

Вы можете скачать файлы примеров кода для этой книги с GitHub по адресу <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition>. Если выйдет обновление кода, оно появится в репозитории GitHub.

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ И СОГЛАШЕНИЯ, ПРИНЯТЫЕ В КНИГЕ

В книге используются следующие типографские соглашения.

Курсив – используется для смыслового выделения важных положений, новых терминов, имен команд и утилит, а также слов и предложений на естественном языке.

Моноширинный шрифт – применяется для листингов программ, а также в обычном тексте для обозначения имен переменных, функций, типов, объектов, баз данных, переменных среды, операторов, ключевых слов и других программных конструкций и элементов исходного кода.

Моноширинный полужирный шрифт – используется для обозначения команд или фрагментов текста, которые пользователь должен ввести дословно без изменений, а также в листингах программ, если необходимо обратить особое внимание на фрагмент кода.

Моноширинный курсив – применяется для обозначения в исходном коде или в командах шаблонных меток-заполнителей, которые должны быть заменены соответствующими контексту реальными значениями.

-  Советы оформлены так.
-  Примечания оформлены так.
-  Важные примечания оформлены так.

Часть I

.....

ЗНАКОМСТВО С МАШИНЫМ ОБУЧЕНИЕМ И ELASTIC STACK

В этой части представлено обобщенное описание Elastic ML – не только с точки зрения алгоритмов, но и с точки зрения организации работы программного обеспечения в Elastic Stack.

Эта часть книги состоит из следующих глав:

- главы 1 «Машинное обучение в информационных технологиях»;
- главы 2 «Подготовка и использование Elastic ML».



Глава 1

.....

Машинное обучение в информационных технологиях



https://t.me/it_books

Десять лет назад идея использования технологий машинного обучения (ML) в IT-структурах или IT-безопасности казалась чем-то вроде научной фантастики. Однако сегодня это одно из самых популярных модных словечек, используемых поставщиками программного обеспечения. Очевидно, что за минувшее десятилетие произошел серьезный сдвиг как в осознании потребности в технологии ML, так и в возможностях, которые она предоставляет. Эта эволюция важна для понимания того, как появился инструмент Elastic ML и для решения каких проблем он был разработан.

Эта глава посвящена обзору истории и концепций, лежащих в основе работы Elastic ML. В ней также представлены различные виды анализа, которые можно провести, и задачи, которые можно решить с помощью Elastic ML. В частности, мы рассмотрим следующие темы:

- преодоление исторических вызовов в IT;
- что нам делать с потоком данных;
- причины появления автоматического обнаружения аномалий;
- машинное обучение без учителя и с учителем;
- использование машинного обучения без учителя для обнаружения аномалий;
- применение машинного обучения с учителем в аналитике фреймов данных.



ПРЕОДОЛЕНИЕ ИСТОРИЧЕСКИХ ВЫЗОВОВ В IT

К специалистам по поддержке IT-структур и архитекторам решений предъявляют высокие требования. Их роль не ограничивается внедрением новых передовых проектов и технологий для бизнеса; они также должны поддерживать безопасную и бесперебойную работу развернутых в настоящее время приложений. Сегодняшние приложения значительно сложнее, чем

когда-либо прежде, – они разбиты на множество компонентов, распределены и, возможно, виртуализированы/контейнеризированы. Они могут быть разработаны с использованием Agile-методики или аутсорсинговой командой. Вдобавок они, скорее всего, постоянно меняются. Некоторые команды DevOps заявляют, что обычно они вносят более 100 изменений в день в действующую производственную систему. Попытаться понять состояние и поведение современного приложения уровня предприятия – все равно что механику попытаться отремонтировать автомобиль, пока он едет по шоссе.

Аналитики по безопасности в области IT тоже с трудом справляются с повседневной работой, но, очевидно, у них другой фокус внимания – обеспечение безопасности предприятия и устранение возникающих угроз. Хакеры, вредоносные программы и инсайдеры-мошенники стали настолько распространенными и изощренными, что по мнению большинства специалистов по безопасности сегодня вопрос заключается не в том, будет ли взломана организация, а в том, насколько быстро она узнает об этом (если вообще узнает). Очевидно, что узнать о взломе как можно раньше (до того, как будет нанесен слишком большой ущерб) предпочтительнее, чем услышать об этом впервые от правоохранительных органов или из вечерних новостей.

Но что же нам делать? Возможно, проблема в том, что экспертам по приложениям и аналитикам службы безопасности не хватает данных, которые помогли бы им эффективно выполнять свою работу? На самом деле в большинстве случаев ситуация противоположная. Многие IT-специалисты и организации тонут в данных.

Что нам делать с потоком данных?

IT-отделы десятилетиями вкладывали силы и средства в инструменты мониторинга, и нередко в их распоряжении есть дюжина или более инструментов, активно собирающих и архивирующих данные, объем которых измеряется в терабайтах или даже петабайтах в день. Источники этих данных чрезвычайно вариативны – от элементарной статистики на уровне инфраструктуры и сети до результатов глубокой диагностики и/или файлов журналов системы и приложений.

Ключевые показатели эффективности (key performance indicators, KPI) на уровне бизнеса также можно отслеживать, иногда включая данные об опыте конечного пользователя. Глубина и широта охвата доступных данных сегодня больше, чем когда-либо. Для обнаружения возникающих проблем или угроз, скрытых в этих данных, традиционно использовалось несколько основных подходов к преобразованию «сырых» данных в информационные объекты:

- **фильтрация/поиск:** некоторые инструменты предоставляют пользователю возможность фильтрации или поиска, чтобы сократить данные до более удобоваримого ограниченного набора. Хотя эта возможность чрезвычайно полезна, она чаще всего используется бессистемно, в основном когда возникает подозрение в наличии проблемы. Даже в этом

случае успех обычно зависит от способности пользователя понять, что он ищет, и от его уровня опыта – как от знания предыдущих ситуаций, так и от навыков использования самой технологии поиска;

- **визуализация:** панели мониторинга, диаграммы и виджеты также чрезвычайно полезны для понимания того, что происходит с данными, и выявления тенденций. Однако визуализации пассивны по своей сути и требуют постоянного наблюдения на предмет обнаружения значимых отклонений. Когда количество собираемых и отображаемых на экране показателей превышает возможности человеческого восприятия (или даже площадь экрана для их отображения), полезность визуализации резко снижается;
- **пороговые значения/правила:** чтобы обойти физические ограничения визуализации и внести в наблюдение за данными активный компонент (т. е. реакцию на изменение данных), многие инструменты позволяют пользователю определять правила или действия, которые запускаются при соблюдении определенных условий или возникновении определенных зависимостей между элементами данных. Однако маловероятно, что вы сможете реалистично определить все подходящие рабочие диапазоны или смоделировать все возможные зависимости в современных сложных и распределенных приложениях. Кроме того, количество и скорость изменений в приложении или среде могут быстро сделать любой статический набор правил бесполезным.

Аналитики обнаружили, что погрязли в ложных срабатываниях предупреждающих систем – широко известная проблема мальчика, который кричал о несуществующих волках, – что приводит к возмущению пользователей по поводу инструментов, генерирующих предупреждения, и скептицизму по поводу ценности, которой обладает такое предупреждение.

В конечном счете стало ясно, что нужен другой подход – такой, который не обязательно был бы полным отказом от прошлых методов, но мог бы обеспечить уровень автоматизации и значимого увеличения эмпирической ценности данных. Посмотрим правде в глаза: люди несовершенны – у нас есть скрытые предубеждения и ограничения способности запоминать информацию, и мы легко отвлекаемся и утомляемся. Алгоритмы машинного обучения при правильном использовании могут легко восполнить эти недостатки.

ПРИЧИНЫ ПОЯВЛЕНИЯ АВТОМАТИЧЕСКОГО ОБНАРУЖЕНИЯ АНОМАЛИЙ

Машинное обучение, будучи очень обширной темой, охватывающей многие области, от беспилотных автомобилей до компьютерных программ, приносящих выигрыш в играх, стало естественным кандидатом на роль эффективного решения. Если вы понимаете, что большинство задач мониторинга приложений или поиска угроз безопасности представляют собой всего лишь вариации на тему *поиска отличий от обычного хода событий*, тогда дисципли-

на обнаружения аномалий становится естественным местом для применения методов машинного обучения IT-специалистами.

Однако в науке об обнаружении аномалий, безусловно, нет ничего нового. Многие очень умные люди в течение долгих лет исследовали и применяли различные алгоритмы и методы. Но на практике обнаружение аномалий в IT-данных сопровождается некоторыми специфическими ограничениями, которые делают интересные с академической точки зрения алгоритмы непригодными для работы. Речь о следующих ограничениях:

- **своевременность.** Уведомление об отключении, нарушении или другой существенной аномальной ситуации должно стать известно как можно быстрее, чтобы смягчить последствия. Стоимость простоя или риск продолжения нарушения безопасности сводятся к минимуму, если быстро устранить проблему. Алгоритмы, которые не успевают отслеживать сегодняшние IT-данные в реальном времени, имеют ограниченную ценность;
- **масштабируемость.** Как упоминалось ранее, в современных IT-средах объем, скорость и вариативность IT-данных продолжают стремительно расти. Алгоритмы, которые занимаются мониторингом и анализом огромных данных, должны иметь возможность линейного масштабирования сообразно с данными, иначе спустя какое-то время они утрачат применимость;
- **эффективность.** Бюджеты IT-подразделений часто подвергаются тщательной проверке на предмет нерациональных расходов, и многие организации постоянно пытаются их урезать. Закупка дополнительного парка суперкомпьютеров для запуска неэффективных алгоритмов вряд ли будет утверждена руководством компании. Скорее, в качестве частного решения придется использовать скромное стандартное оборудование с типичными характеристиками;
- **обобщаемость.** Хотя узкоспециализированная наука о данных часто является лучшим способом решения конкретной информационной проблемы, разнообразие данных в IT сформировало потребность в подходе, который применим в большинстве случаев. Повторное использование одних и тех же методов намного более рентабельно в долгосрочной перспективе;
- **адаптивность.** Постоянно меняющаяся IT-среда быстро сделает жесткий алгоритм бесполезным. Обучение и переподготовка модели ML превращаются в бесконечное занятие и фактически в пустую трату времени, чего мы не можем себе позволить;
- **точность.** Мы уже говорили, что усталость от ложных предупреждений из-за устаревших пороговых и основанных на правилах систем является реальной проблемой. Замена одного генератора ложных тревог на другой никого не впечатлит;
- **простота использования.** Даже если все ранее упомянутые ограничения могут быть удовлетворены, любое решение, для реализации которого потребуются армия специалистов по данным, окажется слишком дорогостоящим и будет немедленно отвергнуто.

Итак, мы подошли к сути задачи – созданию быстрого, масштабируемого, точного и недорогого решения для обнаружения аномалий, которое все будут охотно использовать, потому что оно работает безупречно. Без проблем!

Как бы пугающе это ни звучало на самом деле, основатель и технический директор Prekert Стив Додсон принял этот вызов еще в 2010 году. Хотя Додсон, несомненно, заложил в фундамент компании свои академические знания, технология, которая в конечном итоге превратилась в Elastic ML, зародилась в муках попыток устранения реальных сбоев приложений уровня предприятия. Первая из них – досадный периодический сбой в работе торговой платформы в крупной лондонской финансовой компании. Додсон и несколько инженеров, присоединившихся к этому предприятию, помогли команде банка использовать технологию обнаружения аномалий для автоматического поиска «иголки в стоге сена», что позволило аналитикам сосредоточиться на небольшом наборе соответствующих показателей и записей в журналах событий, которые вызывали подозрения. Выявление первопричины (отказ службы, восстановление которой вызывало каскадный сбой других служб и причиняло ущерб) в конечном итоге обеспечило стабильную работу приложения и избавило банк от необходимости тратить много денег на другое решение – дорогостоящее обновление сетевой инфраструктуры.

Однако со временем стало ясно, что даже этот показательный успех был только началом. Спустя несколько лет и несколько тысяч примеров использования в реальном мире возник союз Prekert и Elastic – сочетание платформы, делающей большие данные легкодоступными, и технологий, которые помогли преодолеть ограничения традиционных методов анализа.

Перенесемся в 2021 год, спустя полные 5 лет после объединения усилий, когда Elastic ML прошел долгий путь в развитии и расширении возможностей платформы ML. Это второе издание книги включает в себя обновления, внесенные в Elastic ML за прошедшие годы, в том числе интеграцию с некоторыми решениями Elastic, касающимися наблюдаемости и безопасности. Во второе издание мы добавили введение в аналитику фреймов данных, которая подробно обсуждается в третьей части книги. Чтобы получить обновленное и глубокое понимание того, как работает Elastic ML, нам сначала нужно рассмотреть терминологию и идеи, а потом двигаться дальше.

МАШИННОЕ ОБУЧЕНИЕ БЕЗ УЧИТЕЛЯ И С УЧИТЕЛЕМ

Хотя существует множество подтипов машинного обучения, два самых известных (и относящихся к Elastic ML) – это *обучение без учителя* и *обучение с учителем*.

В машинном обучении без учителя нет внешнего руководства или указаний со стороны людей. Другими словами, алгоритмы должны изучать (и моделировать) шаблоны данных исключительно самостоятельно. В целом самая большая проблема здесь состоит в том, чтобы алгоритмы точно выявляли отклонения от нормальных шаблонов входных данных, дабы обеспечить

вывод модели, значимый для пользователя. Если алгоритм не может этого сделать, то он бесполезен и непригоден для использования. Следовательно, алгоритмы должны быть достаточно надежными и способны учитывать все тонкости поведения входных данных.

В машинном обучении с учителем для моделирования желаемого результата используются размеченные входные данные (часто многомерные). Ключевое отличие от машинного обучения без учителя состоит в том, что человек априори решает, какие переменные использовать в качестве входных данных, а также предоставляет «достоверные» примеры ожидаемой целевой переменной – *обучающие данные*. Затем алгоритмы машинного обучения изучают, как входные переменные взаимодействуют и влияют на известную выходную цель. Чтобы точно получить желаемый результат (например, прогноз), алгоритм должен иметь набор «правильных данных», которые не только отражают зависимость выхода от входа, но и достаточно разнообразны, чтобы модель смогла изучить максимально широкий спектр сочетаний переменных на входе.

Таким образом, в обоих случаях требуются качественные входные данные, хорошие алгоритмические подходы и хороший механизм, позволяющий ML как изучать поведение данных, так и применять это обучение для оценки последующих наблюдений за этими данными. Давайте посмотрим, как Elastic ML использует машинное обучение без учителя и с учителем.

ИСПОЛЬЗОВАНИЕ МАШИННОГО ОБУЧЕНИЯ БЕЗ УЧИТЕЛЯ ДЛЯ ОБНАРУЖЕНИЯ АНОМАЛИЙ

Чтобы сформировать у вас более полное представление о том, как работает обнаружение аномалий Elastic ML с использованием машинного обучения без учителя, мы рассмотрим следующие темы:

- строгое определение необычности в контексте технологии;
- наглядный пример обучения без учителя;
- описание того, как технология ML моделирует, устраняет тенденции и оценивает данные.

Что такое необычность?

Обнаружение аномалий – это то, чем мы регулярно занимаемся в повседневной жизни, поэтому имеем интуитивное представление о сути процесса. Люди довольно хорошо работают с визуальной информацией, поэтому не удивительно, что если бы я спросил у 100 человек на улице, что необычного в графике на рис. 1.1, подавляющее большинство (включая далеких от техники людей) указало бы на всплеск линии.

Аналогично мы можем спросить у людей, что необычного на следующей фотографии (рис. 1.2).

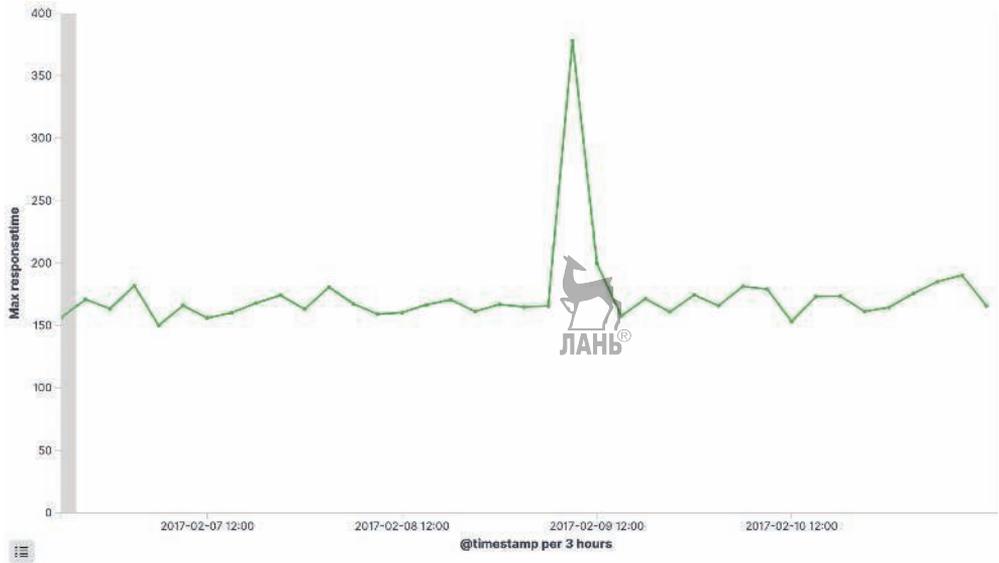


Рис. 1.1 ❖ Этот график содержит визуально заметную аномалию



Рис. 1.2 ❖ На этой фотографии запечатлен тюлень среди пингвинов

Большинство опрошенных наверняка ответят, что тюлень в окружении пингвинов – явление весьма необычное. Но людям бывает сложно сформулировать в явных терминах эвристику, которая лежит в основе таких выводов.

Есть две разные эвристики, которые мы могли бы использовать для определения различных видов аномалий, показанных на этих изображениях:

- сущность необычна, если ее поведение значительно отклоняется от установленного шаблона или диапазона, основанного на исторических данных;

- сущность необычна, если некоторые характеристики этой сущности значительно отличаются от тех же характеристик других членов группы или популяции.

Эти ключевые определения будут иметь прямое отношение к обнаружению аномалий Elastic ML, поскольку они образуют два основных фундаментальных режима работы алгоритмов обнаружения аномалий (временной и популяционный анализ, о которых пойдет речь в главе 3). Как вы увидите, пользователь будет решать, какой режим работы использовать для определенной задачи.

Изучение того, что является нормой

Как мы уже говорили, механизм обнаружения аномалий в Elastic ML основан на обучении без учителя, так как обучение происходит без привлечения специальных обучающих данных. Человек не помогает модели обучаться; она делает это самостоятельно, путем изучения представленных данных. Это немного похоже на изучение иностранного языка путем погружения, вместо того чтобы сидеть за словарями и учебниками грамматики.

Чтобы перейти от совершенно наивного состояния, когда о ситуации ничего неизвестно, к состоянию, в котором можно делать уверенные прогнозы, необходимо построить модель ситуации. Способ создания этой модели чрезвычайно важен, поскольку эффективность всех последующих действий, предпринятых на основе этой модели, будет сильно зависеть от точности модели. Модель должна быть гибкой и постоянно обновляться на основе новой информации, и это главное, что она должна делать в соответствии с парадигмой обучения без учителя.

Вероятностные модели

Вышеупомянутой цели вполне могут служить распределения вероятностей. Существует много фундаментальных типов распределений (в Elastic ML используются различные типы распределений, такие как пуассоновское, гауссово, логнормальное или даже сочетание моделей), но распределение Пуассона лучше всего подходит для обсуждения в первую очередь, потому что оно уместно в ситуациях, когда есть дискретные события («отсчеты») относительно времени (рис. 1.3).

Здесь показаны три различных варианта распределения, каждый с различным средним (λ) и максимальным ожидаемым значением k . Мы можем провести аналогию, которая гласит, что эти распределения моделируют ожидаемое количество почтовых отправлений, которые ежедневно приносят человеку домой, представленное знаком k на оси x :

- при $\lambda = 1$ вероятность того, что ежедневно будут доставлять ноль или одно почтовое отправление, составляет около 37 %. Возможно, это справедливо для студента колледжа, который не получает много писем;

- при $\lambda = 4$ вероятность получения трех или четырех писем составляет около 20 %. Это может быть хорошей моделью для молодого специалиста;
- при $\lambda = 10$ вероятность того, что в день будут доставлять 10 писем, составляет около 13 %, что может соответствовать молодой семье или домашнему хозяйству, которое каким-то образом оказалось во многих списках рассылки.

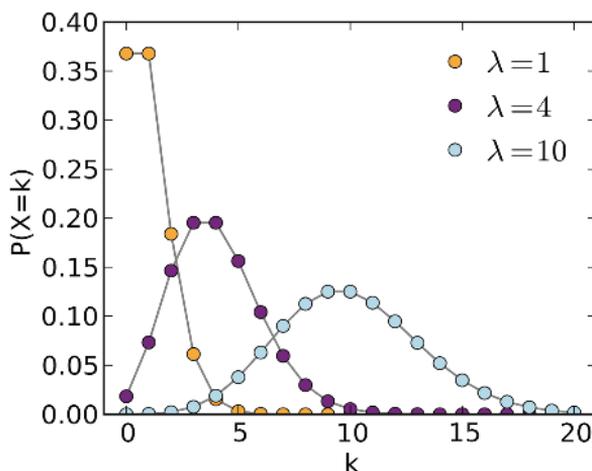


Рис. 1.3 ❖ График, демонстрирующий распределения Пуассона
(источник: https://en.wikipedia.org/wiki/Poisson_distribution#/media/File:Poisson_pmf.svg)

Дискретные точки на каждой кривой также дают вероятность других значений k . Поэтому модель может быть информативной и отвечать на такие вопросы, как «Вероятно ли получение 15 писем?». Как видите, это маловероятно для студента ($\lambda = 1$) или молодого специалиста ($\lambda = 4$), но более вероятно для большой семьи ($\lambda = 10$). В данном случае мы просто заявили, что представленные модели подходят для определенных категорий людей, — но совершенно очевидно, что в реальной жизни мы должны опираться на механизм обучения модели конкретным ситуациям, а не ограничиваться голословными утверждениями. Процесс обучения интуитивно понятен.

Обучение моделей

Если продолжить аналогию с доставкой писем, то становится ясно, что наилучшую модель для конкретного получателя можно определить путем ежедневного дежурства у почтового ящика и записи данных о том, сколько писем почтальон бросил в ящик. Также достаточно очевидно, что чем больше наблюдений сделано, тем выше будет ваша уверенность в точности модели. Другими словами, если провести у почтового ящика всего 3 дня, это даст менее полную информацию и уверенность, чем 30 дней или даже 300.

Фактически мы можем разработать алгоритм выбора соответствующей модели на основе наблюдений. Частью этого процесса самоотбора должна быть тщательная проверка выбора алгоритмом как самого типа модели (то есть распределение Пуассона, Гаусса, логнормальное и т. д.), так и конкретных коэффициентов этого типа модели (как в предыдущем примере λ). Для этого проводится постоянная оценка соответствия модели. Для оценки вероятных значений параметров модели по набору данных в целом также используются байесовские методы, но с учетом того, сколько информации было просмотрено до определенного момента времени. Алгоритмы машинного обучения делают это автоматически.

! Для тех, кто хочет более глубоко погрузиться в некоторые типичные математические процессы, происходящие за кулисами, рекомендуем научную статью по адресу <http://www.ijmlc.org/papers/398-LC018.pdf>.

Самое главное, что процесс уточнения модели является непрерывным, так что новая информация рассматривается вместе со старой, с экспоненциальным взвешиванием более свежей информации. Такая модель после 60 наблюдений могла бы напоминать график, представленный на рис. 1.4.

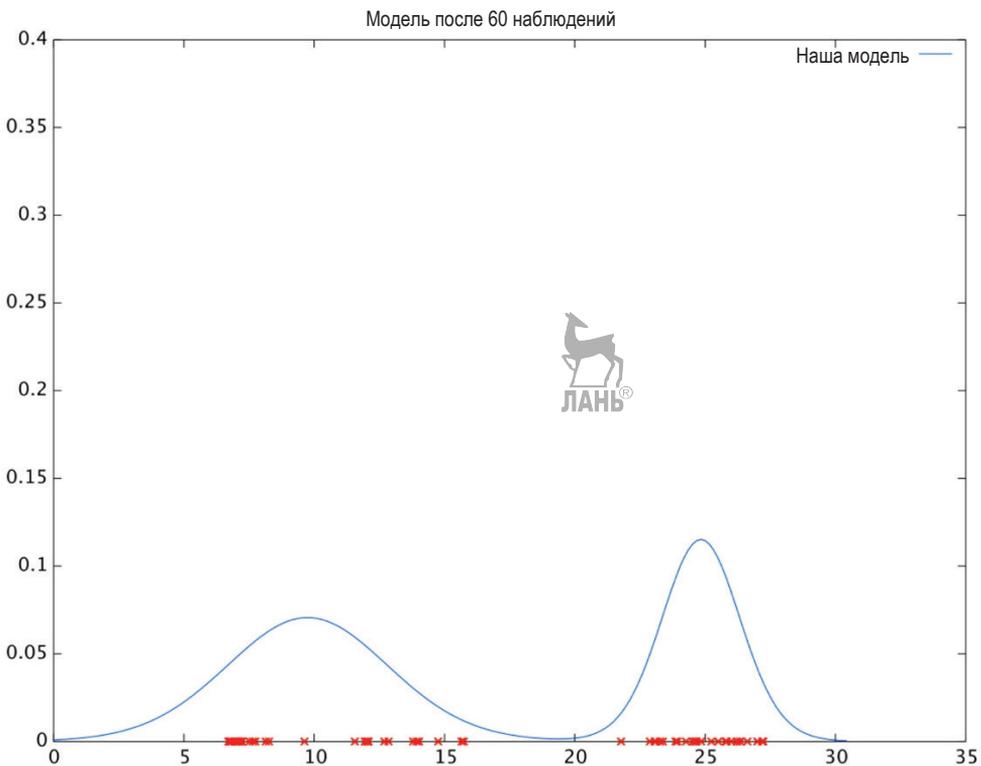


Рис. 1.4 ❖ Пример модели после 60 наблюдений

После 400 наблюдений модель будет выглядеть совсем иначе, поскольку данные представляют собой множество новых наблюдений со значениями от 5 до 10.

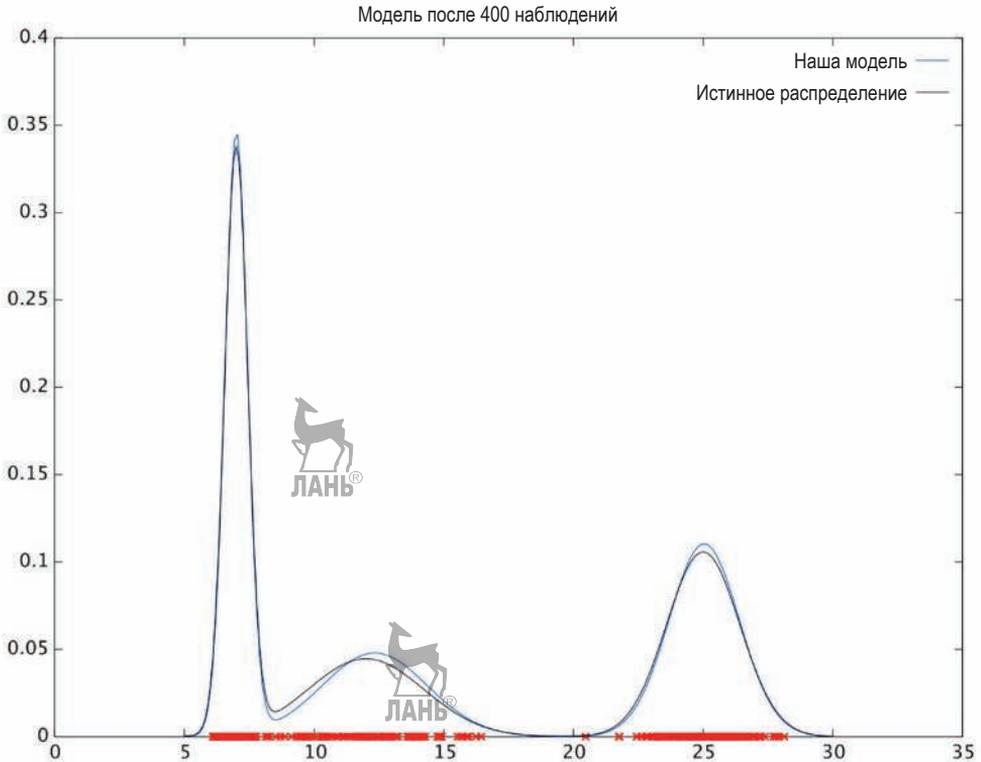


Рис. 1.5 ❖ Пример модели после 400 наблюдений

Также обратите внимание, что модель может иметь несколько модов, или областей/кластеров, с более высокой вероятностью. Сложность и точность совпадения изученной модели (показана синей кривой) с теоретической идеальной моделью (выделена черным цветом) имеет большое значение. Чем точнее модель, тем лучше представление состояния нормы для этого набора данных и, таким образом, в конечном итоге тем точнее прогноз того, как будущие значения совпадают с этой моделью.

Непрерывный характер моделирования также требует, чтобы эту модель можно было сериализовать и поместить в долгосрочное хранилище, а затем восстановить и возобновить обучение или использование позже. Как мы увидим, реализация этого процесса создания, хранения и использования модели представляет собой сложную оркестровку, которая, к счастью, автоматически выполняется средствами Elastic ML.

Выявление и устранение тенденций

Еще одним важным аспектом точного моделирования реальных данных является учет явных тенденций и скрытых закономерностей, которые возникают естественным образом. Например, наблюдаются ли у вас приливы и спады объема данных ежечасно и/или ежедневно при большей активности в рабочие часы или в рабочие дни? Если да, то это необходимо учитывать. Elastic ML автоматически выявляет основные тенденции в данных (линейный рост, периодические гармоника и т. д.) и учитывает их. Рассмотрим следующий график (рис. 1.6).

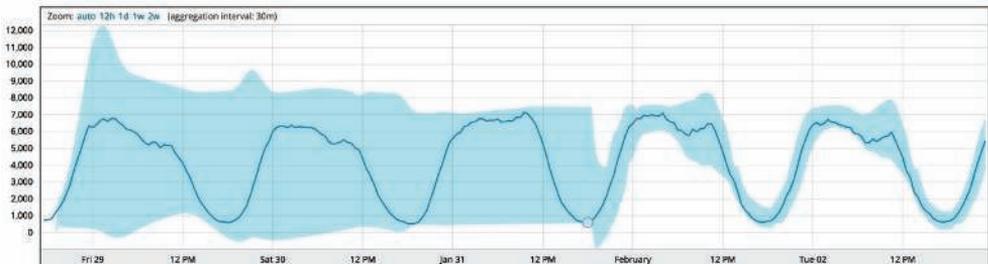


Рис. 1.6 ❖ Пример обнаружения периодичности

На этом графике показано, как мы изучаем, а затем выводим суточный цикл. Границы прогноза модели (представленные в виде голубой оболочки вокруг темно-синего сигнала) резко изменяются после автоматического обнаружения трех последовательных итераций этого цикла.

Таким образом, по мере того как с течением времени набирается больше данных, модели становятся более точными как с точки зрения более зрелой функции распределения вероятностей, так и за счет автоматического распознавания и устранения тенденций других стандартных шаблонов (например, рабочих дней, выходных и т. д.), которые могут не появляться в течение нескольких дней или недель. В следующем примере со временем обнаруживаются несколько тенденций, включая ежедневный и еженедельный цикл и общий линейный наклон (рис. 1.7).

Эти изменения модели записываются в виде системных аннотаций. Аннотации как общая концепция будут рассмотрены в следующих главах.

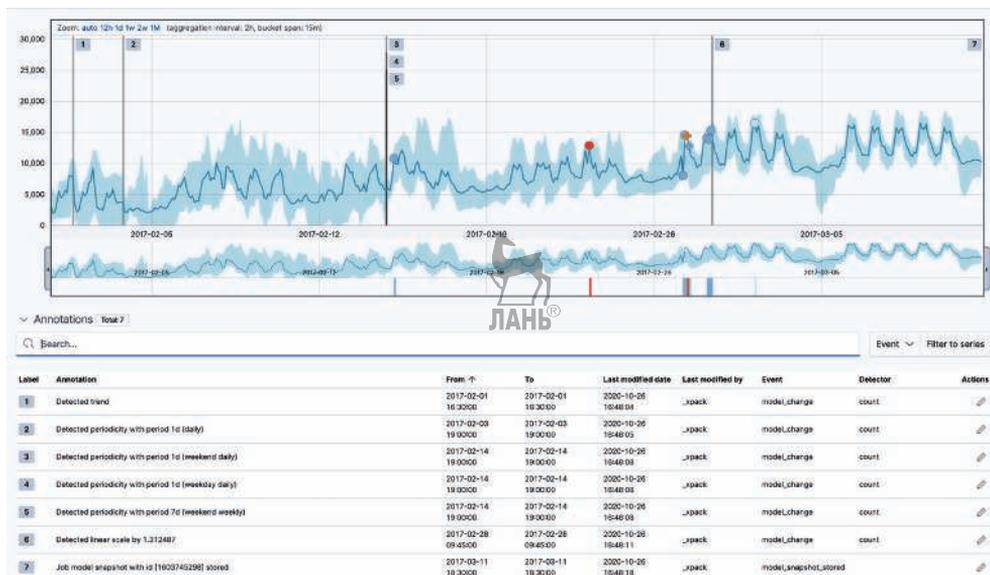


Рис. 1.7 ❖ Обнаружение нескольких тенденций

Оценка степени необычности

После построения модели вероятность любого будущего наблюдаемого значения может быть найдена в имеющемся распределении вероятностей. Ранее мы задавали вопрос: «Вероятно ли получение 15 писем?» На этот вопрос теперь можно дать эмпирический ответ в зависимости от модели, с числом от 0 (невозможно) до 1 (абсолютно достоверно). Elastic ML будет использовать модель для вычисления этого дробного значения с точностью примерно до 300 значащих цифр (что может быть полезно при работе с очень низкими вероятностями). Рассмотрим график, представленный на рис. 1.8.

В этом примере вероятность наблюдения значения 921 вычислена как $1,444e - 9$ (или, как чаще записывают, всего лишь 0,0000001444 %). Это очень маленькое значение, которое люди с трудом воспринимают и оценивают. ML будет использовать вычисленное значение вероятности и с помощью процесса квантильной нормализации отображать его на шкалу значимости от 0 до 100, где 100 – это наивысший уровень необычности, возможный для этого конкретного набора данных. В данном случае результат расчета вероятности $1,444e - 9$ нормализован до оценки 94. Эта нормализованная оценка пригодится позже в качестве средства для оценки серьезности аномалии в целях предупреждения и/или сортировки.

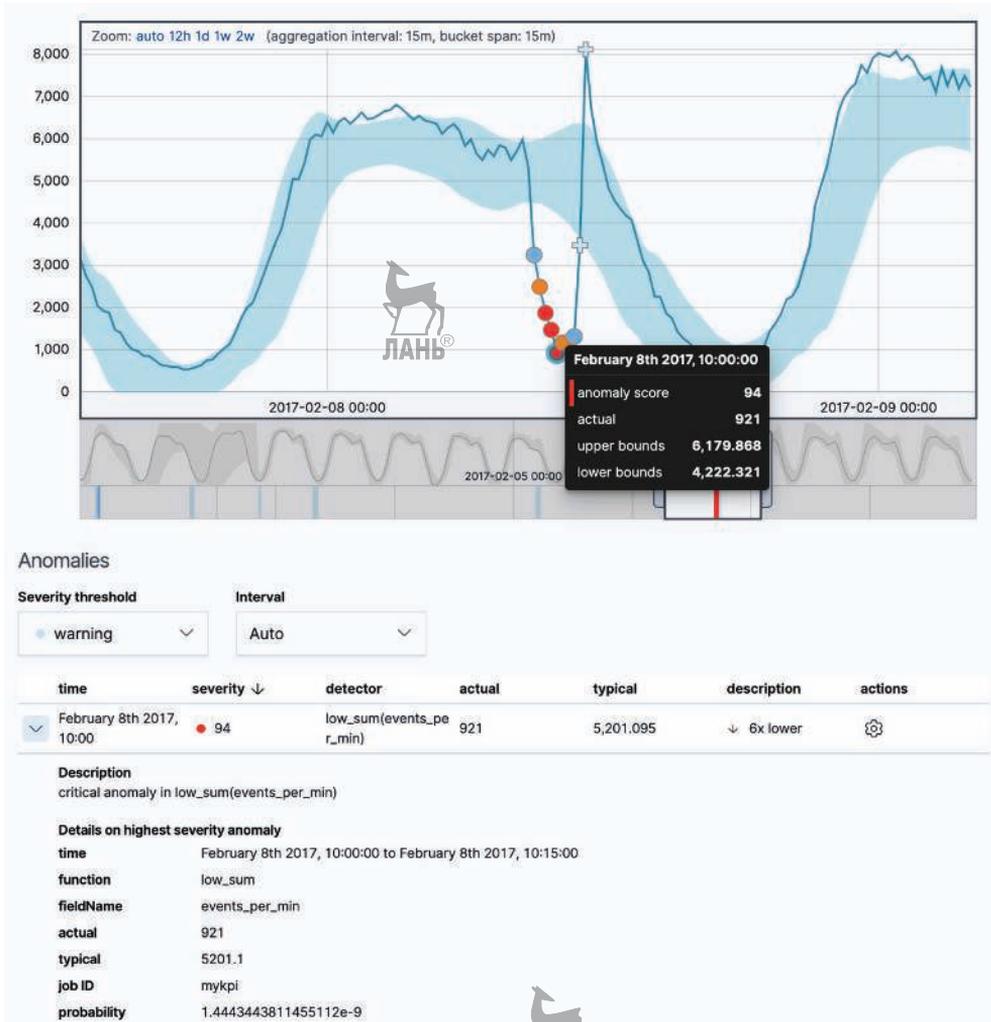


Рис. 1.8 ❖ Оценка аномалий

Роль времени

В Elastic ML все действия по обнаружению аномалий, которые мы будем обсуждать в оставшейся части книги, будут учитывать фактор времени, связанный с данными и анализом. Другими словами, при поиске аномалий Elastic ML полагает, что входные данные являются данными временных рядов, и они будут проанализированы с приращениями времени. Это ключевой момент, который помогает различать обнаружение аномалий и аналитику фреймов данных в дополнение к парадигме «обучение с учителем/без учителя».

Вы увидите, что есть небольшое, но важное различие между анализом совокупности (глава 3) и обнаружением выбросов (глава 10). Хотя оба они эффек-

тивно выявляют сущности, которые заметно отличаются от своих сородичей, анализ совокупности при обнаружении аномалий выполняется в рамках временного ряда, тогда как анализ с целью обнаружения выбросов не ограничен временем. Более подробно мы поговорим об этом в следующих главах.

ПРИМЕНЕНИЕ МАШИННОГО ОБУЧЕНИЯ С УЧИТЕЛЕМ В АНАЛИТИКЕ ФРЕЙМОВ ДАННЫХ

За исключением обнаружения выбросов (описанного в главе 10), которое основано на обучении без учителя, остальная часть аналитики фреймов данных использует обучение с учителем. В частности, есть два основных типа задач, которые позволяет решить аналитика фреймов данных Elastic ML:

- **регрессия.** Используется для прогнозирования непрерывного числового значения (цены, продолжительности, температуры и т. д.);
- **классификация.** Используется для прогнозирования того, относится ли что-то к определенному классу (например, является ли транзакция мошеннической и т. д.).

В обоих случаях модели строятся с использованием обучающих данных для сопоставления входных переменных (которые могут быть числовыми или категориальными) с выходными прогнозами посредством обучающих деревьев решений. Конкретная реализация, используемая Elastic ML, представляет собой особый вариант XGBoost, фреймворка дерева решений с градиентным усилением с открытым исходным кодом, который недавно приобрел определенную известность среди специалистов по данным благодаря своей способности выигрывать соревнования Kaggle.

Процесс обучения с учителем

Процесс машинного обучения с учителем значительно отличается от такового без учителя. При обучении с учителем вы четко отделяете этап обучения от этапа прогнозирования. Очень упрощенная версия процесса выглядит, как показано на рис. 1.9.

Здесь мы видим, что на этапе обучения признаки извлекаются из необработанных обучающих данных для создания *матрицы признаков* (также называемой *фреймом данных*) для передачи в алгоритм машинного обучения и создания модели. Модель может быть проверена на части данных (валидация модели), чтобы увидеть, насколько хорошо она работает, и могут быть предприняты последующие уточняющие шаги, чтобы скорректировать извлечение признаков или уточнить параметры алгоритма машинного обучения, используемого для повышения точности прогнозов модели.

Как только пользователь решает, что модель достаточно эффективна, эта модель «перемещается» в рабочий процесс прогнозирования, где она при-

меняется к новым данным. На основе модели поочередно *выводится* один новый вектор признаков, чтобы сформировать прогноз.

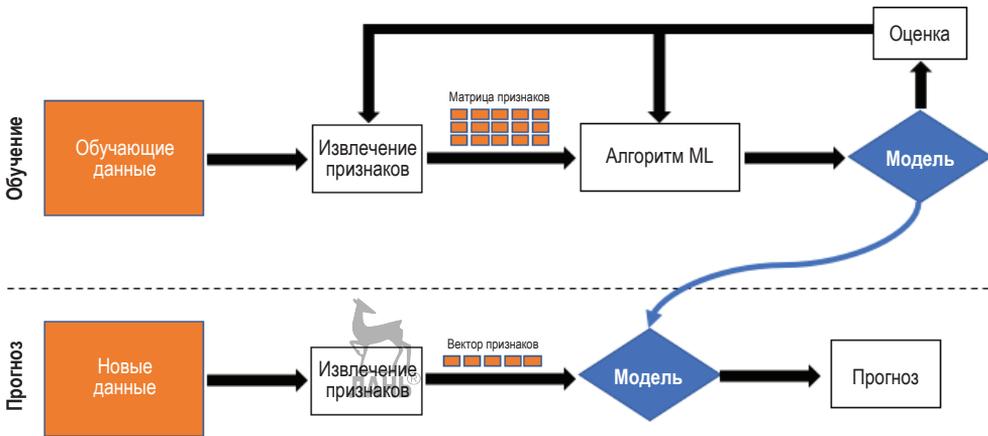


Рис. 1.9 ❖ Машинное обучение с учителем

Чтобы получить интуитивное представление о том, как это работает, представьте сценарий, в котором вы хотите продать свой дом, но не знаете, по какой цене его выставить. Вы изучаете предыдущие продажи в вашем районе и отмечаете для себя разницу в стоимости домов в зависимости от различных факторов (количество спален, количество ванных комнат, площадь в квадратных метрах, близость к школам/магазинам, возраст дома и т. д.). Эти факторы являются «признаками», которые учитываются вместе (а не по отдельности) для каждой предыдущей продажи.

Этот корпус исторических данных продаж составляет ваш набор обучающих данных. Он полезен, потому что вы точно знаете, сколько стоит каждый дом (и это параметр, значение которого вы в конечном итоге хотите спрогнозировать для своего дома). Если вы достаточно хорошо изучите этот набор, то можете интуитивно понять, что цены на дома сильно зависят от некоторых признаков (например, количества спален) и что другие признаки (возможно, возраст дома) не так сильно влияют на ценообразование. Это понятие, так называемую *важность признака* (feature importance), мы еще раз обсудим в следующей главе.

Обладая достаточным количеством обучающих данных, вы можете иметь хорошее представление о том, какова должна быть стоимость вашего дома, учитывая, что это дом 30-летней давности с тремя спальнями, двумя ванными комнатами и площадью 1700 квадратных футов. Другими словами, вы построили в уме модель на основе исследования сопоставимых домов, проданных за последний год или около того. Если прошлые продажи – это обучающие данные, то признаки вашего дома (спальни, ванные комнаты и т. д.) – это *векторы признаков* (feature vector), которые будут определять ожидаемую цену с учетом модели, которую вы «обучили» у себя в голове.

Ваша простая ментальная модель, очевидно, не настолько строгая, как та,

которую можно было бы построить на основе регрессионного анализа с использованием машинного обучения и принимая во внимание десятки соответствующих входных признаков, но эта простая аналогия должна сформировать у вас представление о процессе обучения на предыдущих, ранее известных ситуациях и затем применить это знание к настоящей, новой ситуации.



ЗАКЛЮЧЕНИЕ

В этой главе мы рассмотрели историю применения машинного обучения в прикладных задачах IT в связи с необходимостью автоматизировать анализ огромного и постоянно растущего объема данных в корпоративных средах. Вы получили более глубокое понимание различных типов машинного обучения, задействованных в Elastic ML, включая как обнаружение аномалий (обучение без учителя), так и анализ фреймов данных (обучение с учителем).

По мере продвижения по оставшимся главам мы будем часто сопоставлять варианты прикладных задач, которые пытаемся решить, с различными режимами работы Elastic ML.

Если ваши данные представлены временными рядами, т. е. они появляются регулярно с течением времени (данные показателей/производительности, файлы журналов, транзакции и т. д.), вполне возможно, что вы сможете обойтись только механизмом обнаружения аномалий Elastic ML. Как вы увидите дальше, он невероятно гибкий и простой в использовании и позволяет выполнять множество сценариев использования с широким спектром данных. Это что-то вроде швейцарского армейского ножа! Большая часть этой книги (главы с 3 по 8) будет посвящена тому, как использовать обнаружение аномалий и дополнительные возможности прогнозирования, чтобы получить максимальную отдачу от данных временных рядов, обработанных в Elastic Stack.

Если вас больше интересует поиск необычных объектов в популяции/когорте (необычное поведение пользователей/объектов), у вас может возникнуть непростой выбор между использованием анализа популяции для обнаружения аномалий или обнаружением выбросов в аналитике фрейма данных. Решение в первую очередь зависит от того, нужно ли вам делать это почти в реальном времени, – если да, то вы наверняка выберете анализ популяции. Если работа в реальном времени не требуется и/или если вам требуется одновременно рассматривать несколько признаков, вы должны предпочесть обнаружение выбросов. В главе 10 более подробно говорится о сравнении и преимуществах каждого подхода.

Остается много других вариантов использования, требующих многопланового подхода к моделированию. Кроме вышеупомянутого примера ценообразования на недвижимость, к ним относятся такие задачи, как распознавание языка, анализ оттока клиентов, обнаружения вредоносных программ и т. д. Все они относятся к области машинного обучения с учителем и аналитике фреймов данных и будут рассмотрены в главах с 11 по 13.

В следующей главе вы узнаете, как настроить Elastic ML и применять его на практике. Садитесь поудобнее и приготовьтесь наслаждаться новыми знаниями!

Глава 2

.....

Подготовка и использование Elastic ML



В предыдущей главе вы узнали в общих чертах, каким образом Elastic ML использует машинное обучение без учителя для автоматического обнаружения аномалий и машинное обучение с учителем для анализа фреймов данных. Дальше мы подробно расскажем о том, как Elastic ML работает в составе Elastic Stack (Elasticsearch и Kibana)¹.

В этой главе основное внимание будет уделено как установке (на самом деле включению) функций Elastic ML, так и подробному обсуждению порядка действий, особенно в отношении обнаружения аномалий. В частности, мы рассмотрим следующие темы:

- включение функций Elastic ML;
- обзор процессов.



ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

В этой главе мы будем использовать Elastic Stack в том виде, в котором он представлен в версии 7.10, и рабочий процесс Elasticsearch Service для Elastic Cloud по состоянию на ноябрь 2020 года.

ВКЛЮЧЕНИЕ ФУНКЦИЙ ELASTIC ML

Процесс включения функций Elastic ML внутри Elastic Stack в собственном кластере немного отличается от использования Elasticsearch Service (ESS) для Elastic Cloud. Если коротко, в собственном кластере функции машинного обучения активируются с помощью лицензионного ключа (коммерческого или пробного). В варианте ESS для использования Elastic ML необходимо

¹ Если точнее, то Elastic Stack – обширная экосистема компонентов, которые служат для поиска и обработки данных. Основные компоненты Elastic Stack – это Elasticsearch, Kibana, Logstash и Beats. Неразбериха с постоянно меняющимися названиями и торговыми марками Elastic, особенно в последние 5 лет, способна запутать кого угодно. – *Прим. перев.*

подготовить в облачном кластере выделенный узел машинного обучения. Далее мы подробно объясним, как это сделать в обоих случаях.

Включение машинного обучения в собственном кластере

Если у вас есть собственный (управляемый вами) кластер, созданный в результате загрузки дистрибутивов Elasticsearch и Kibana по умолчанию (доступно на <https://www.elastic.co/downloads/>), включить функции Elastic ML с помощью лицензионного ключа очень просто. Убедитесь, что вы не используете лицензионные дистрибутивы Apache 2.0 с открытым исходным кодом, которые не содержат кодовую базу X-Pack.

Elastic ML, в отличие от большинства функций Elastic Stack, не является бесплатным – для него требуется коммерческая лицензия (в частности, уровня Platinum). Однако это открытый исходный код, поскольку он находится в открытом доступе на GitHub (<https://github.com/elastic/ml-cpp>), и пользователи могут просматривать код, комментировать его или даже скачивать. Но *использование* Elastic ML регулируется коммерческим соглашением с компанией Elastic.

Когда Elastic ML был впервые выпущен (еще во времена версий v5.x), он был частью пакета с закрытым исходным кодом, известного как X-Pack, требующего отдельной процедуры установки. Однако начиная с версии 6.3 код X-Pack стал открытым (<https://www.elastic.co/what-is/open-x-pack>) и помещен в стандартный дистрибутив Elasticsearch и Kibana. Следовательно, отдельный этап установки X-Pack больше не нужен, достаточно лишь «включить» новую функцию через коммерческую или пробную лицензию.

Процедура установки Elasticsearch и Kibana выходит за рамки этой книги, но ее легко выполнить, следуя онлайн-документации на веб-сайте Elastic (доступном по адресу <https://www.elastic.co/guide/>).

После запуска Elasticsearch и Kibana перейдите к параметру **Stack** (Стек) в меню навигации слева и выберите **License Management** (Управление лицензиями). Вы увидите экран как на рис. 2.1.

Обратите внимание, что по умолчанию применяется бесплатный уровень лицензии **Base** (Базовый). Он позволяет использовать некоторые расширенные функции, отсутствующие в лицензированном дистрибутиве Apache 2.0 с открытым исходным кодом или в сторонних сервисах (например, Amazon Elasticsearch Service). Удобное руководство по сравнению функций, доступных на разных уровнях лицензии, можно найти на веб-сайте Elastic по адресу <https://www.elastic.co/subscriptions>.

Как было сказано ранее, для Elastic ML требуется лицензия уровня Platinum. Если вы приобрели лицензию Platinum в Elastic, вы можете применить эту лицензию, нажав кнопку **Update license** (Обновить лицензию), как показано на рис. 2.1. Если у вас нет лицензии Platinum, вы можете запустить бесплатную 30-дневную пробную версию, нажав кнопку **Start my trial** (Начать использовать пробную версию), чтобы включить Elastic ML и другие функции Platinum (при условии что вы согласны с условиями лицензии), как показано на рис. 2.2.

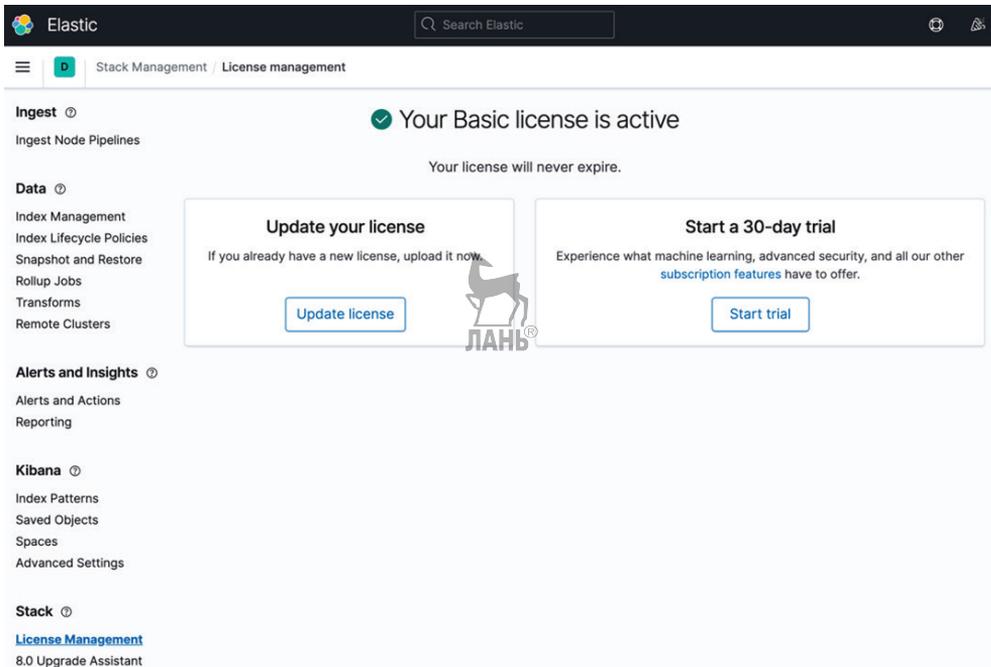


Рис. 2.1 ❖ Экран управления лицензиями в Kibana

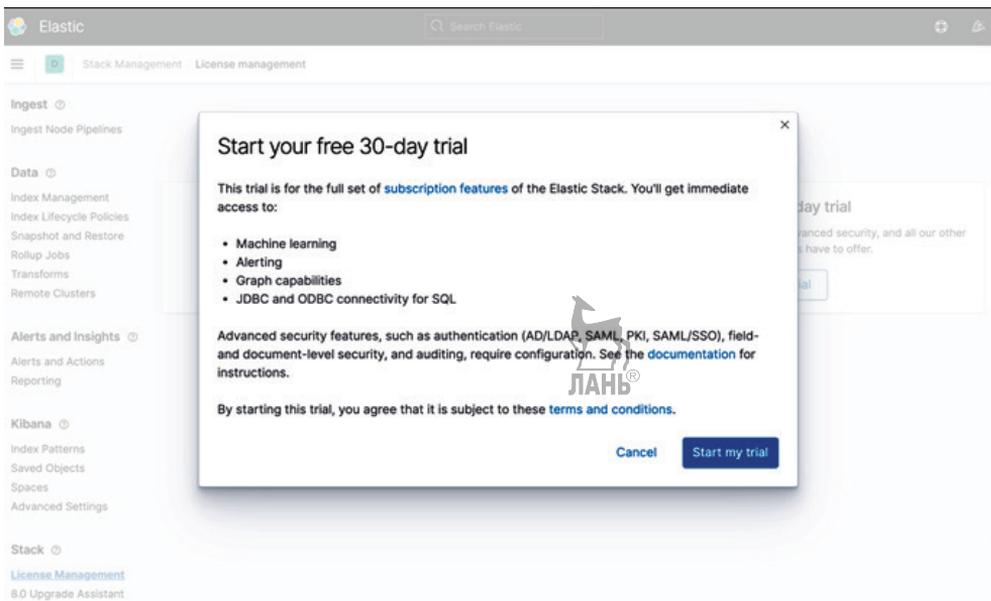


Рис. 2.2 ❖ Запуск бесплатной 30-дневной пробной версии

Как только завершится активация пробной версии, на экране лицензирования появится сообщение, что вы сейчас находитесь в активной пробной версии Platinum функций Elastic Stack (рис. 2.3).

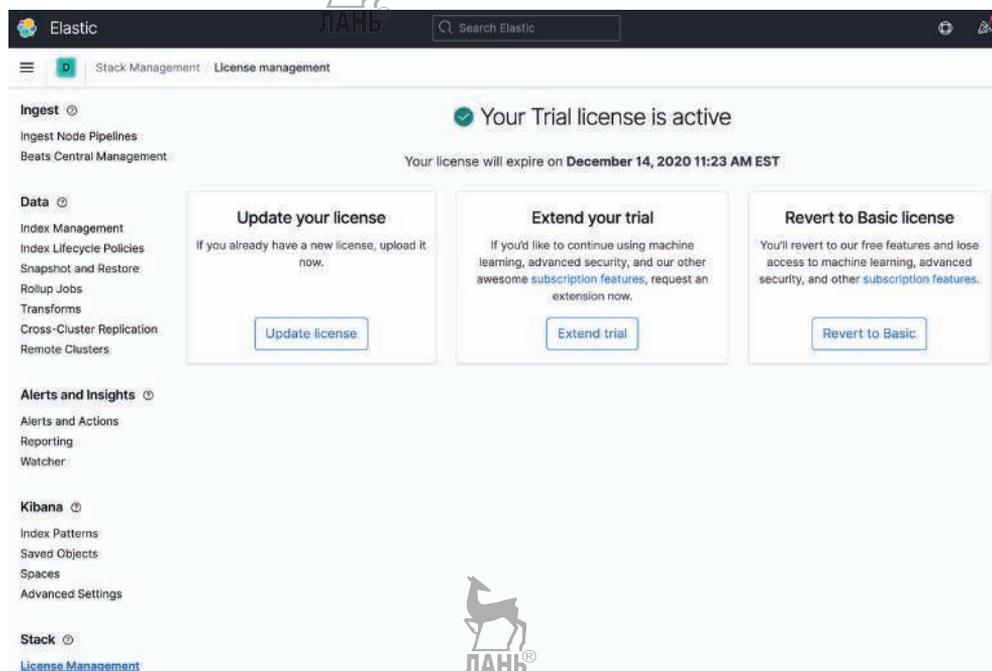


Рис. 2.3 ❖ Пробная лицензия активирована

Сразу же после активации пробной версии вы можете начать использовать Elastic ML. Чтобы воспользоваться возможностями других функций Platinum, необходимы дополнительные шаги по настройке, но эти шаги выходят за рамки данной книги. Обратитесь к документации Elastic для получения рекомендаций по настройке этих функций.

Включение машинного обучения в облаке – Elasticsearch Service

Если загрузка, установка и создание собственного кластера Elastic Stack вас привлекают меньше, чем использование платформы Elastic Stack в качестве услуги, перейдите в Elastic Cloud (<https://www.elastic.co/cloud/>) и подпишитесь на бесплатную пробную версию, используя только вашу электронную почту (рис. 2.4).

Затем вы можете выполнить следующие шаги.

1. Оказавшись в интерфейсе Elastic Cloud после входа в систему, вы сможете запустить бесплатную пробную версию, нажав кнопку **Start**

your free trial (Начать бесплатную пробную версию), как показано на рис. 2.5.

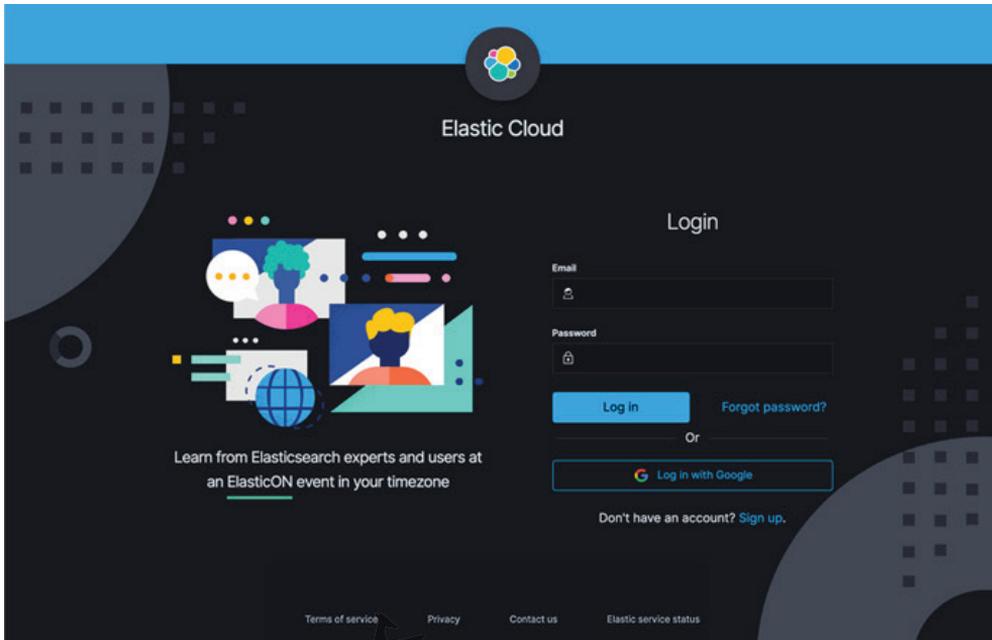


Рис. 2.4 ❖ Экран приветствия Elastic Cloud

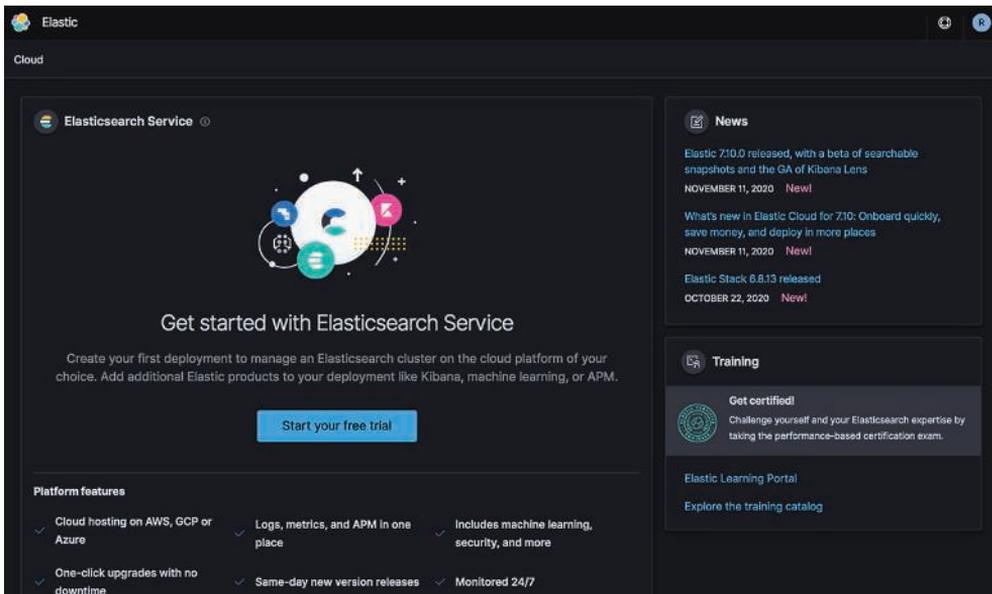


Рис. 2.5 ❖ Главный экран Elastic Cloud

После нажатия кнопки вы увидите, что ваша 14-дневная бесплатная пробная версия ESS активирована (рис. 2.6).

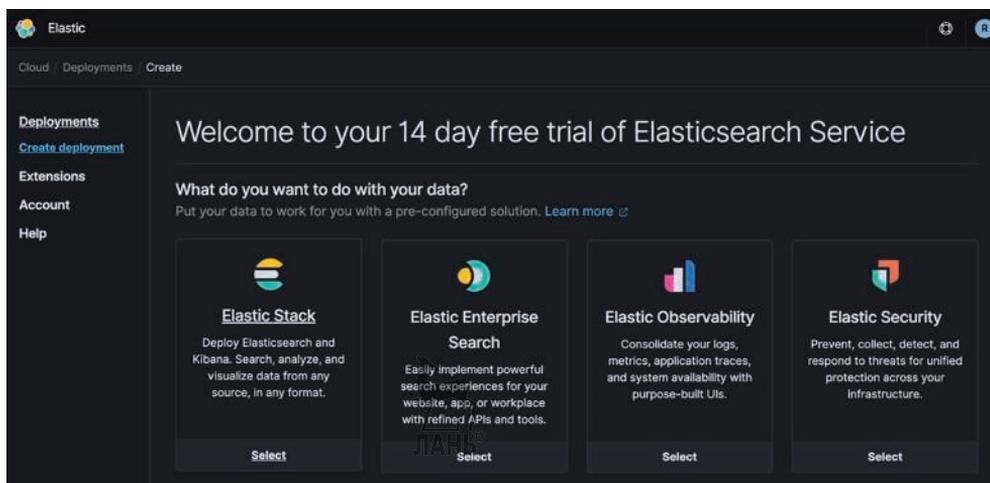


Рис. 2.6 ❖ Пробная версия службы Elasticsearch включена

- Конечно, чтобы опробовать Elastic ML, вам сначала понадобится подготовленный кластер Elastic Stack. Существует несколько вариантов создания того, что ESS называет *deployments* (развертывания), причем некоторые из них предназначены для конкретных случаев использования. В этом примере мы воспользуемся шаблоном **Elastic Stack** слева на рис. 2.6 и выберем профиль оборудования **I/O Optimized** (Оптимизированный для ввода-вывода), но не бойтесь во время ознакомления экспериментировать с другими параметрами (рис. 2.7).
- Вы также можете выбрать, у какого облачного провайдера и в каком регионе запускать кластер, но, что наиболее важно, если вы хотите использовать функции машинного обучения, то должны включить узел машинного обучения, сначала нажав кнопку **Customize** (Настроить) в правом нижнем углу.
- После нажатия кнопки **Customize** вы увидите новый экран, позволяющий добавить узел машинного обучения (рис. 2.8).
- Внизу рис. 2.8 есть ссылка **Add Machine Learning nodes** (Добавить узлы машинного обучения). При нажатии на нее откроется интерфейс конфигурации узла ML (рис. 2.9).

❗ В течение 14-дневного бесплатного пробного периода ESS вы можете добавить только один узел ML объемом 1 Гб (в одной или двух доступных зонах). Если вы перейдете от бесплатной пробной версии к платной подписке, то, разумеется, сможете создать больше узлов машинного обучения и занять больший объем.

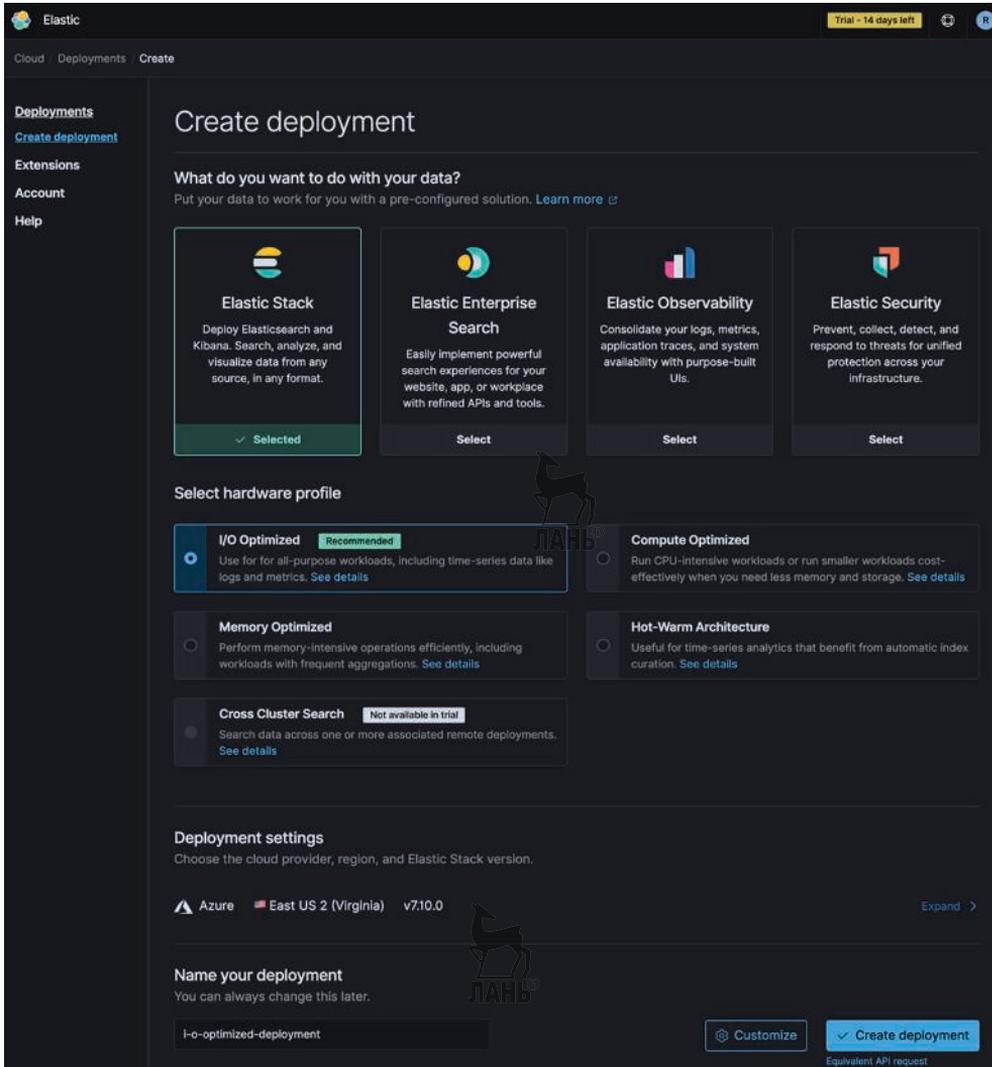


Рис. 2.7 ❖ Создание развертывания ESS

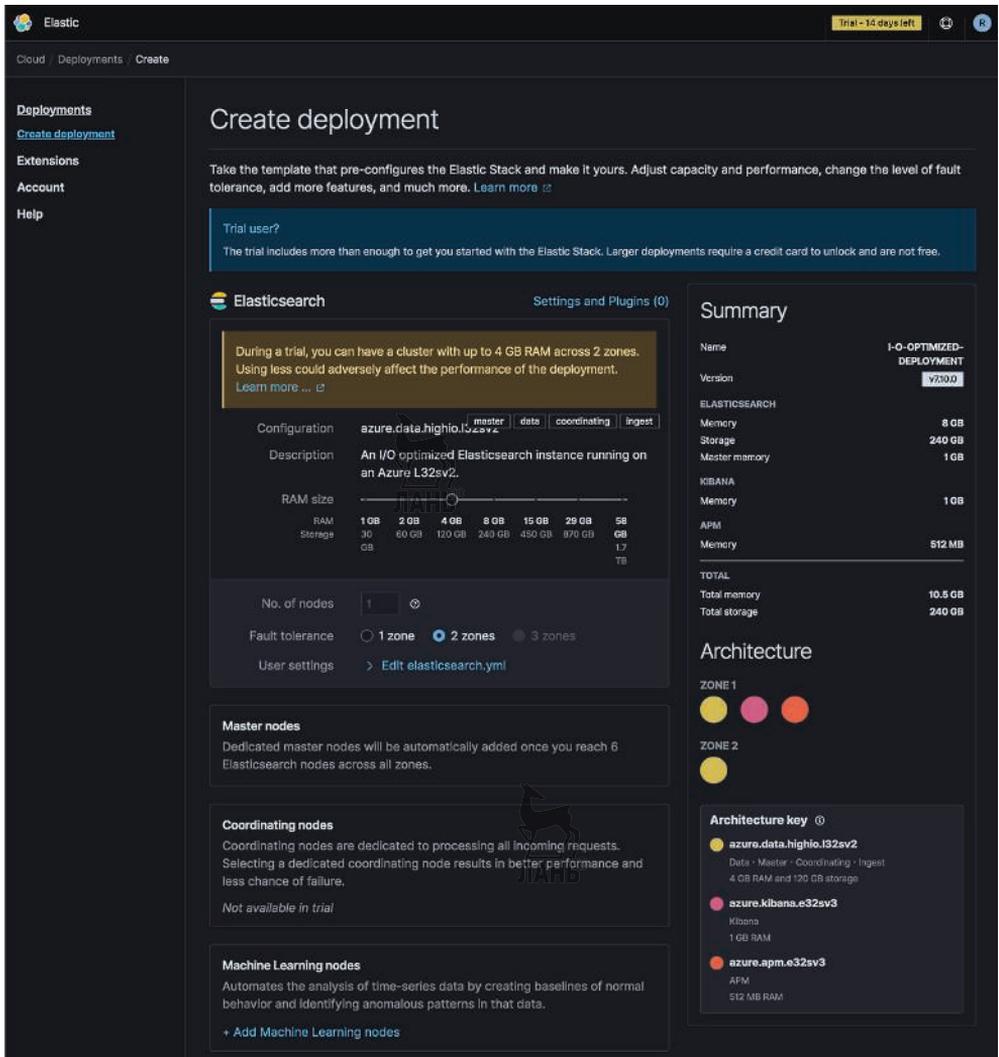


Рис. 2.8 ❖ Настройка развертывания для добавления узла машинного обучения

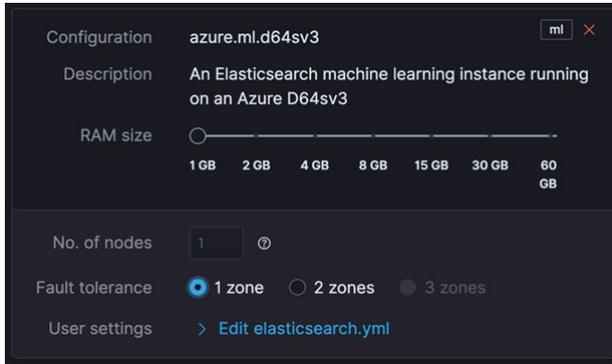


Рис. 2.9 ❖ Добавление узлов ML

- После добавления узла машинного обучения в конфигурацию нажмите кнопку **Create Deployment** (Создать развертывание), чтобы запустить процесс создания кластера для ESS, который займет несколько минут. Тем временем вам будут показаны учетные данные по умолчанию, которые вы будете использовать для доступа к кластеру (рис. 2.10).

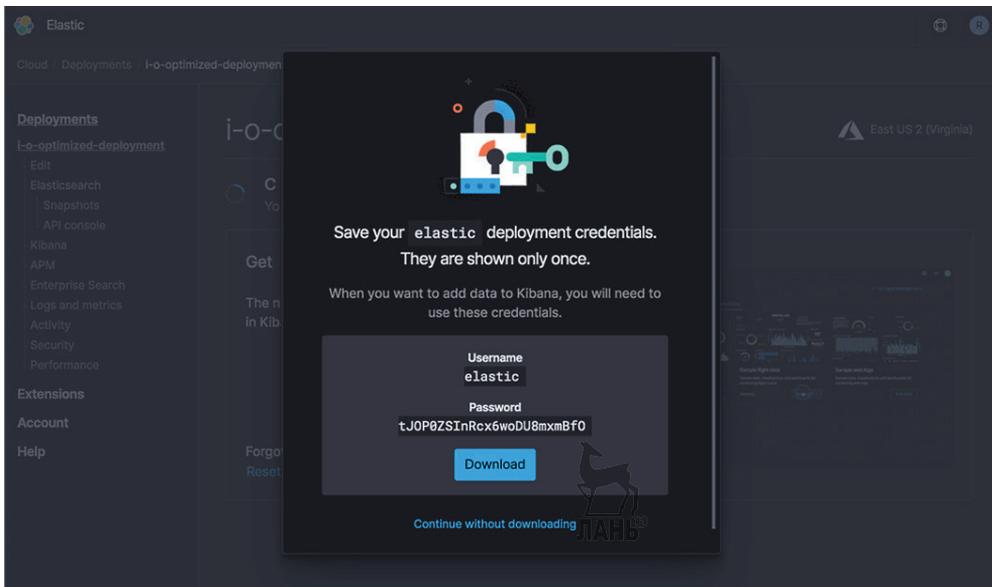


Рис. 2.10 ❖ Учетные данные, назначенные по умолчанию

Вы можете загрузить эти учетные данные для использования позже. Не волнуйтесь, если вы забыли их загрузить, – вы всегда можете сбросить пароль позже, если потребуется.

- После того как кластер будет создан и запущен, как показано на рис. 2.11 (обычно спустя несколько минут), вы увидите следующее представле-

ние вашего развертывания с кнопкой **Open Kibana** (Открыть Kibana), которая позволит вам запустить ваше развертывание (рис. 2.11).

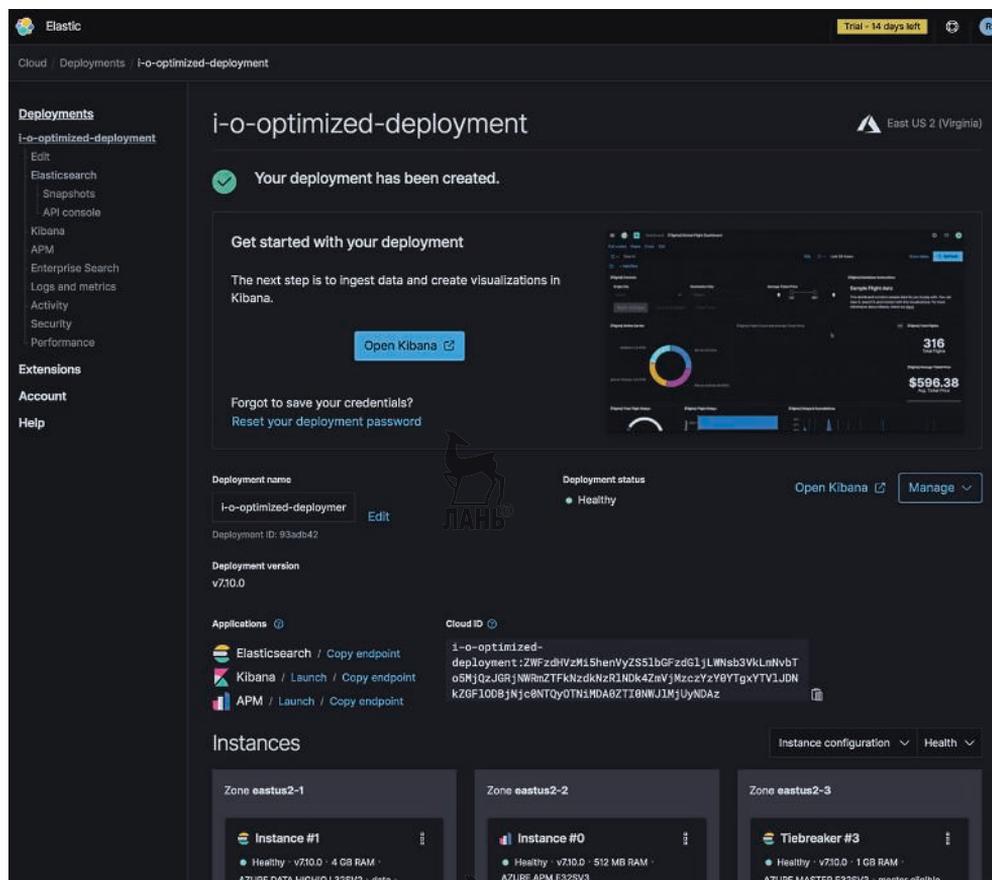


Рис. 2.11 ❖ Развертывание успешно создано

После нажатия кнопки **Open Kibana** вы автоматически авторизуетесь в Kibana, где сразу будете готовы к использованию ML – никаких дополнительных действий по настройке не требуется.

На данный момент, с точки зрения пользователя, который хочет использовать Elastic ML, разница между конфигурацией собственного кластера, показанной ранее, и конфигурацией, созданной в ESS, невелика. Однако одним из основных отличий является то, что в конфигурации ESS Elastic ML всегда изолирован в *выделенный узел ML*. В конфигурации собственного кластера узлы машинного обучения могут быть выделены или иметь общую роль (например, такие как *data*, *ingest* и *ml* на одном узле). Мы обсудим эту концепцию позже в данной главе.

Теперь, когда у нас есть работающий Elastic Stack с включенным машинным обучением, мы приблизились к возможности начать анализ данных,

о котором пойдет речь в главе 3. Но сначала давайте разберемся с *операционализацией* (внутренней структурой, взаимосвязями и принципом работы составных частей) Elastic ML.

ОБЗОР ОПЕРАЦИОНАЛИЗАЦИИ ELASTIC ML

Во время подготовки к использованию Elastic ML вам будет полезно понять ряд ключевых концепций, касающихся того, как Elastic ML работает в Elastic Stack. Сюда входит понимание того, как аналитика выполняется на узлах кластера и как данные, которые должны быть проанализированы с помощью машинного обучения, извлекаются и обрабатываются.

! Некоторые концепции, представленные в этом разделе, могут казаться вам непонятными до тех пор, пока вы не начнете использовать Elastic ML на реальных примерах. Это нормально, если вы чувствуете, что вам удобнее бегло пролистать (или даже пропустить) этот раздел сейчас и вернуться к нему позже, приобретя некоторый реальный опыт использования Elastic ML.

Узлы ML

Прежде всего, поскольку Elasticsearch по своей природе является распределенным решением со многими узлами, вполне естественно, что функция машинного обучения Elastic Stack работает как встроенный плагин, который подчиняется многим из тех же операционных концепций. Как описано в документации (<https://www.elastic.co/guide/en/elasticsearch/reference/current/ml-settings.html>), ML можно включить на любом или всех узлах, но в производственной системе рекомендуется иметь выделенные узлы ML. Вы видели, что этот (безусловно, правильный) подход фактически навязан пользователю в Elastic Cloud ESS – если пользователь хочет использовать машинное обучение, ему придется создать выделенные узлы.

Наличие выделенных узлов машинного обучения также помогает оптимизировать типы ресурсов, необходимых только для машинного обучения. В отличие от узлов данных, которые подвергают значительной нагрузке дисковый ввод-вывод из-за индексации и поиска, узлы ML более требовательны к вычислениям и памяти. Обладая этими знаниями, вы можете выбрать оборудование, которое лучше подходит для выделенных узлов машинного обучения.

Следует отметить один ключевой момент: алгоритмы машинного обучения не работают в виртуальной машине Java (JVM). Это исполняемые файлы на C++, которые будут использовать оперативную память, оставшуюся от всего, что выделено для кучи JVM. При выполнении заданий машинного обучения процесс, вызывающий анализ (он называется `autodetect` для обнаружения аномалии и `data_frame_analyzer` для анализа фреймов данных), можно увидеть в списке процессов (например, выполнив команду `ps` в Linux). Каждому активно выполняемому заданию машинного обучения соответствует один

процесс. В многоузловых системах диспетчер ML распределяет задания по каждому из узлов с поддержкой ML, чтобы сбалансировать рабочую нагрузку.

Elastic ML руководствуется параметром `xpack.ml.max_machine_memory_percent`, который определяет, сколько системной памяти может быть отведено заданиям ML. Значение этого параметра по умолчанию – 30 %. Это значение относится к полной памяти машины, а не к свободной на данный момент части. Не забывайте, что JVM Elasticsearch может занимать около 50 % машинной памяти системы, поэтому оставить 30 % для ML, а оставшиеся 20 % для операционной системы и других вспомогательных процессов – это разумно, хотя и консервативно. Задания не будут назначены узлу, если это приведет к тому, что расчетное использование памяти заданиями ML превысит предел, определенный этим параметром.

Хотя не существует эмпирической формулы для определения размера и количества выделенных узлов машинного обучения, есть несколько хороших практических правил:

- иметь один выделенный узел машинного обучения (два для систем с повышенной доступностью/отказоустойчивостью, если один узел вдруг станет недоступным) на каждый кластер размером до 10 узлов данных;
- иметь как минимум по два узла ML на каждый кластер размером около 20 узлов;
- добавлять дополнительный узел машинного обучения при развертывании дополнительных 10 узлов данных.

Этот общий подход, заключающийся в том, чтобы зарезервировать около 10–20 % емкости вашего кластера для выделенных узлов машинного обучения, безусловно, является разумным ориентиром, но он не избавляет вас от необходимости проводить собственный выбор размера, тестирование характеристик и мониторинг ресурсов. Как мы увидим в нескольких последующих главах, потребности в ресурсах для ваших задач машинного обучения будут во многом зависеть от того, какие виды анализа выполняются, а также от плотности и объема анализируемых данных.

Задания

В Elastic ML *задание* (job) – это единица работы. Существуют как *задания по обнаружению аномалий*, так и *задания по анализу фреймов данных*. Те и другие принимают какие-либо данные на входе и производят новую информацию на выходе. Задания можно создавать с помощью пользовательского интерфейса ML в Kibana или программно через API. Им также требуются узлы с поддержкой ML.

Задания по обнаружению аномалий можно запускать как однократный пакетный анализ (по ряду исторических данных) или как непрерывный процесс в реальном времени на данных временных рядов – данных, которые постоянно индексируются вашим Elastic Stack (или на самом деле одновременно происходит и то, и другое).

Возможен вариант, когда задания анализа фреймов данных не являются непрерывными, а представляют собой однократные прогоны, которые производят выходные результаты и/или выходную модель для последующего вывода. Мы рассмотрим этот вариант более подробно в главах с 9 по 13.

Следовательно, с точки зрения ввода в эксплуатацию, задания по обнаружению аномалий немного сложнее, поскольку несколько таких заданий могут выполняться одновременно, выполняя независимые действия и анализируя данные из разных хранилищ. Другими словами, в типичном кластере, вероятно, постоянно будут выполняться задания по обнаружению аномалий.

Конфигурация задания по обнаружению аномалий состоит из следующих элементов, которые мы позже рассмотрим более детально:

- название/идентификатор задания;
- окно сегментации анализа (период сегмента);
- определение и настройки запроса на получение необработанных данных для анализа (поток данных);
- объект применения конфигурации обнаружения аномалий (детектор).

Далее мы сосредоточимся на сегментировании данных временных рядов, которое является важным понятием в анализе данных реального времени.

Сегментирование данных в анализе временных рядов

Сегментирование входных данных – важная концепция механизма обнаружения аномалий Elastic ML. Сегментирование задается с помощью ключевого параметра `bucket_span` на уровне задания, в соответствии с которым входные данные из потока данных собираются в мини-пакеты для обработки. *Период сегмента* (`bucket span`) можно рассматривать как интервал агрегации перед анализом – временное окно, в течение которого часть данных агрегируется для целей анализа. Чем меньше значение `bucket_span`, тем более детализирован анализ, но также выше вероятность появления шумовых артефактов в данных.

Для иллюстрации этого соображения на следующем графике показан один и тот же набор данных, агрегированный за три разных интервала (рис. 2.12).

Обратите внимание на то, что выраженный аномальный выброс, наблюдаемый в версии, агрегированной с 5-минутным интервалом, становится почти полностью потерянным, если данные агрегированы с 60-минутным интервалом из-за малой продолжительности всплеска (< 2 минут). Фактически на этом 60-минутном интервале выброс даже не выглядит таким уж аномальным.

В этом и заключается практическая целесообразность при выборе `bucket_span`. С одной стороны, более короткий период агрегации полезен, потому что он увеличит частоту анализа (и, таким образом, сократит интервал оповещения, если обнаружено что-то аномальное), но с другой – если сделать его слишком коротким, это может выделить в данных аномалии, которые не имеют для вас значения.

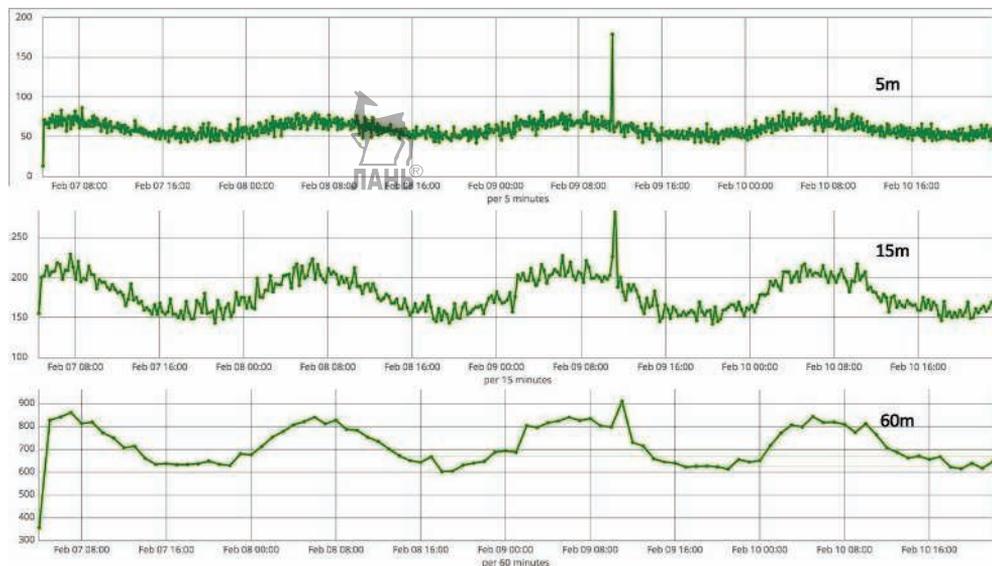


Рис. 2.12 ❖ Агрегирование одних и тех же данных на разных промежутках времени

Если кратковременный всплеск, показанный в предыдущих данных, является значимой аномалией для вас, то 5-минутного просмотра данных будет достаточно. Если, однако, кратковременные аномалии данных кажутся вам ненужной суетой, избегайте низкого значения `bucket_span`.



Некоторые дополнительные практические соображения можно найти в блоге Elastic: <https://www.elastic.co/blog/explaining-the-bucket-span-in-machine-learning-for-elasticsearch>.

Загрузка данных в Elastic ML

Очевидно, что задания по обнаружению аномалий нуждаются в данных для анализа (а также для построения и совершенствования статистических моделей). Эти данные поступают из ваших хранилищ временных рядов в Elasticsearch. *Поток данных (datafeed)* – это механизм, с помощью которого эти данные извлекаются (ищутся) на регулярной основе и поставляются алгоритмам машинного обучения. Его конфигурация в основном скрыта от пользователя, за исключением случая создания расширенного задания в пользовательском интерфейсе (или с помощью API обнаружения аномалий). Тем не менее важно понимать, как работает поток данных.

Поток данных регулярно запрашивает данные по имени *хранилища*, которое содержит данные для анализа. Как часто поток данных запрашивает данные (и сколько данных получает за раз), зависит от ряда факторов:

- `query`: фактический запрос (выраженный в Elasticsearch DSL), который будет использоваться для извлечения данных из исходного хранилища для анализа. Пользователь может запросить все документы, разме-

щенные в исходном хранилище, или выборочно фильтровать и/или агрегировать данные;

- `bucket_span`: вы уже знаете, что `bucket_span` управляет шириной временного окна текущего анализа. Следовательно, задача потока данных – убедиться, что сегменты заполнены хронологически упорядоченными данными. Поэтому вы можете видеть, что поток данных запрашивает диапазон дат в Elasticsearch;
- `frequency`: параметр, который определяет, как часто запрашиваются необработанные данные с физической точки зрения. Если это интервал от 2 до 20 мин, частота запросов будет равна `bucket_span` (например, каждые 5 мин запрашивать данные за последние 5 минут). Если `bucket_span` больше, `frequency` по умолчанию будет меньшим числом (более частые запросы), поэтому подразумевается, что длинный интервал будет запрашиваться по частям. Это полезно, если набор данных довольно объемный. Другими словами, интервал длинного `bucket_span` будет разделен на более мелкие интервалы просто для целей запросов;
- `query_delay`: этот параметр определяет количество времени «после текущего момента», по истечении которого поток данных должен запрашивать данные для заданного диапазона `bucket_span`. По умолчанию это 60 с, если задание настраивается через API, или случайное значение от 60 до 120 с, если задание настраивается через пользовательский интерфейс. Например, если установлено значение `bucket_span`, равное 5 мин, и значение `query_delay`, равное 60 с, в момент 12:01 поток данных запросит данные, поступившие в диапазоне с 11:55 до 12:00. Эта дополнительная небольшая задержка сглаживает задержки в конвейере приема, чтобы гарантировать, что никакие данные не будут исключены из анализа, если их прием задерживается по какой-либо внешней причине. Если система обнаруживает, что в задании по обнаружению аномалий отсутствуют данные из-за возможных задержек приема, будет создана сгенерированная системой аннотация (пометка-уведомление), чтобы предупредить пользователя о том, что происходит пропуск данных и что, возможно, потребуется увеличить значение `query_delay` для исправления этого;
- `scroll_size`: в большинстве случаев тип поиска, который поток данных выполняет для Elasticsearch, использует API *прокрутки* (`scroll`) (<https://www.elastic.co/guide/en/elasticsearch/reference/current/scroll-api.html>). Размер прокрутки определяет, сколько данных запрашивает поток у Elasticsearch за раз. Например, если настроен запрос данных журнала каждые 5 мин, но в типичном 5-минутном окне умещается 1 млн событий, прокрутка этих данных означает, что не весь 1 млн событий будет получен с помощью одного гигантского запроса. Напротив, это будет сделано при помощи нескольких запросов с приращением `scroll_size`. По умолчанию размер прокрутки настроен на скромное значение 1000. В этом случае, чтобы получить 1 млн записей, возвращенных в ML, поток данных будет запрашивать у Elasticsearch по 1000 строк 1000 раз. Увеличение `scroll_size` до 10 000 уменьшит количество прокруток до 100. Смысл этой настройки в том, что более мощные кластеры должны

иметь возможность обрабатывать большой `scroll_size` и, таким образом, быть более эффективными в общем процессе.

Однако есть исключение в случае задания с одной метрикой. Задание с одной метрикой (более подробно описанное в главе 3) – это простое задание машинного обучения, которое позволяет анализировать только показатели однократного ряда. В этом случае API прокрутки не используется для получения необработанных данных – поток данных автоматически создает агрегацию запросов (используя агрегацию `date_histogram`). Этот метод агрегирования также можно использовать для любого задания по обнаружению аномалий, но в настоящее время он требует прямого редактирования конфигурации JSON задания и рассчитан на опытных пользователей.

С точки зрения подачи данных в Elastic ML для заданий анализа фреймов, эта парадигма отличается от обнаружения аномалий, поскольку данные не поступают на анализ непрерывно, в реальном времени. Подробное описание передачи данных в задание анализа фрейма данных будет рассмотрено в главах 9–13.

Теперь, когда у вас есть более глубокое понимание того, как данные поступают в Elastic ML для анализа, давайте рассмотрим некоторые служебные хранилища, которые используются для организации работы Elastic ML.

Служебные хранилища

В работе Elastic ML применяется несколько служебных хранилищ:

- `.ml-config`;
- `.ml-state-*`;
- `.ml-notifications-*`;
- `.ml-annotations-*`;
- `.ml-stats-*`;
- `.ml-anomalies-*`.

Все эти хранилища являются *системными* (и большинство из них являются *скрытыми*), что означает, что они не предназначены для записи или иных манипуляций со стороны конечного пользователя. Однако часто бывает полезно понять их функцию/роль, поэтому давайте рассмотрим каждую из них по очереди.

.ml-config

Хранилище `.ml-config` содержит информацию о конфигурации всех заданий машинного обучения, которые в настоящее время определены в системе. Информация, содержащаяся в этом хранилище, доступна для чтения и интерпретации среднестатистическому пользователю.

.ml-state-*

Хранилище `.ml-state` – это место, где Elastic ML хранит внутреннюю информацию о ходе выполнения заданий аналитики фреймов данных и ста-

тистических моделях обнаружения аномалий, которые были изучены для определенного набора данных, а также дополнительную логистическую информацию. Это хранилище не предназначено для пользователей – его используют внутренние алгоритмы машинного обучения, которые будут читать и записывать в него информацию.

.ml-notifications-*

В этом хранилище Elastic ML хранит сообщения аудита, которые появляются в разделе **Job Messages** (Сообщения заданий) на странице **Job Management** (Управление заданиями) пользовательского интерфейса обнаружения аномалий. Эти сообщения содержат основную информацию о создании и деятельности заданий. Кроме того, здесь можно найти основные ошибки выполнения. Однако более подробную информацию о выполнении заданий машинного обучения можно найти в файле `elasticsearch.log`.

.ml-annotations-*

В этом хранилище хранятся записи аннотаций, связанных с заданиями по обнаружению аномалий. Сюда входят аннотации, созданные пользователями, которые можно определить с помощью пользовательского интерфейса обнаружения аномалий, а также аннотации, созданные системой, такие как предупреждения о задержке получения данных и уведомления о снимках модели.

.ml-stats-*

Это хранилище содержит информацию о ходе и производительности заданий анализа фреймов данных.

.ml-anomalies-*

Хранилища `.ml-anomalies-*` содержат подробные результаты заданий машинного обучения. Они играют важную роль в использовании результатов алгоритмов машинного обучения. Вся информация, отображаемая в пользовательском интерфейсе машинного обучения, будет основываться на этих данных результатов. Кроме того, упреждающее оповещение об аномалиях будет осуществляться за счет настройки запросов по этим хранилищам. Более подробная информация об этом будет представлена в главе 6.

Теперь, когда мы знаем названия и назначение системных хранилищ, принадлежащих Elastic ML и управляемых им, давайте более детально рассмотрим хранилища `.ml-state` и `.ml-anomalies` и их вклад в оркестровку во время выполнения заданий по обнаружению аномалий.

Оркестровка обнаружения аномалий

Поскольку задания по обнаружению аномалий могут выполняться непрерывно на данных временных рядов, поступающих в реальном времени, для них

требуется довольно сложная оркестровка. Упрощенная схема этого процесса показана на рис. 2.13.



Рис. 2.13 ❖ Упрощенная схема оркестровки задания по обнаружению аномалий

Процесс `autodetect`, который является физическим воплощением задания по обнаружению аномалий, – это то, что представлено этапом «анализ/модель» на рис. 2.13. Хранилище `.ml-state` иногда считывается и записывается процессом `autodetect`, как описано в следующем разделе. Выходные данные процесса `autodetect` (результаты анализа) хранятся в хранилищах `.ml-anomalies-*`.

Как правило, вышеупомянутые процедуры выполняются один раз для каждого `bucket_span` (за исключением фактического чтения/записи `.ml-state`). Ключевой вывод заключается в том, что такая оркестровка позволяет задаче обнаружения аномалий работать в оперативном режиме (то есть не в автономном/пакетном режиме) и постоянно учиться на вновь полученных данных. Этот процесс также автоматически обрабатывается Elastic ML, поэтому пользователю не нужно беспокоиться о сложной последовательности выполнения операций.

Снимки модели обнаружения аномалий

Как упоминалось в предыдущем разделе, состояние модели обнаружения аномалий хранится в `.ml-state`. Однако на самом деле оно не читается и не записывается в каждом периоде `bucket_span`. Вместо этого состояние модели в основном сохраняется в памяти процесса `autodetect` и только периодически сериализуется в `.ml-state`. Если задание по обнаружению аномалий вызвано для обработки большого количества исторических данных или выполняется в режиме реального времени, то модель сериализуется следующими способами:

- периодически, по расписанию примерно каждые 3–4 ч (или с интервалом, определяемым `background_persist_interval`, если он явно задан);
- когда задание на обнаружение аномалий переводится в состояние `closed` (закрыто).

Из-за этой периодической сериализации модели старые снимки автоматически удаляются при выполнении ежесуточного ночного задания по обслуживанию системы. По умолчанию, если в хранилище `.ml-state` есть снимки более чем на 1 день старше самого нового снимка, они удаляются, за исключением первого снимка каждый день. Кроме того, удаляются все снимки, которые старше самого нового снимка более чем на 10 дней. Если вы хотите исключить конкретный снимок из списка на удаление и сохранить его на неопределенный срок, используйте пользовательский интерфейс в Kibana или API обновленных снимков модели, чтобы установить для параметра `retain` (сохранение) значение `true`.

Сохранение моментальных снимков модели позволяет пользователю отменить задание, чтобы использовать один из ранее сделанных снимков модели в случае, если что-то пойдет не так в работе или возникнет непредвиденная ситуация. В приложении мы рассмотрим пример, демонстрирующий, как возвращать исходное состояние до запуска задания при помощи снимка модели.

ЗАКЛЮЧЕНИЕ

В этой главе мы разобрали процедуры включения функций Elastic ML как в собственном локальном кластере Elastic Stack, так и в Elasticsearch Service для Elastic Cloud. Кроме того, мы заглянули за кулисы системы и увидели там точки глубокой интеграции с остальной частью Elastic Stack и то, как Elastic ML работает с функциональной точки зрения.

В следующих главах фокус нашего внимания смещается с базовой теории в область практического использования. Начиная со следующей главы мы перейдем к обширным возможностям обнаружения аномалий Elastic ML, и вы узнаете, как настраивать задания для решения некоторых практических задач анализа журналов, показателей и поведения пользователей.

АНАЛИЗ ВРЕМЕННЫХ РЯДОВ – ОБНАРУЖЕНИЕ И ПРОГНОЗИРОВАНИЕ АНОМАЛИЙ

В этой части мы обсудим методику анализа временных рядов, полученную после приобретения компании Prelert, и ее последующую интеграцию в Elastic Stack.

Эта часть книги состоит из следующих глав:

- главы 3 «Обнаружение аномалий»;
- главы 4 «Прогнозирование»;
- главы 5 «Интерпретация результатов»;
- главы 6 «Создание и использование оповещений»;
- главы 7 «Выявление истинных причин аномалий»;
- главы 8 «Другие приложения Elastic Stack для обнаружения аномалий».

Глава 3

Обнаружение аномалий



https://t.me/it_books

Обнаружение аномалий было изначально заложено в набор функций Elastic ML и является наиболее зрелой технологией, уходящей своими корнями во времена PreAlert (до приобретения Elastic в 2016 году). Эта технология зарекомендовала себя как надежная, простая в использовании, мощная и применимая во всех видах сценариев использования данных временных рядов.

В этой объемной и насыщенной новыми знаниями главе мы уделим основное внимание использованию Elastic ML для обнаружения аномалий в частоте появления документов/событий, редких событий и числовых значений, выходящих за рамки ожидаемой нормальной работы. Мы разберем несколько простых, но впечатляющих примеров, которые иллюстрируют как эффективность Elastic ML, так и простоту его использования.

В частности, рассмотрим следующие темы:

- типы заданий Elastic ML;
- устройство детектора;
- обнаружение изменений частотности событий;
- обнаружение изменений значений показателей;
- обзор расширенных функций детектора;
- разделение анализа по категориальным признакам;
- обзор временного и популяционного анализа;
- категоризация в анализе неструктурированных сообщений;
- управление Elastic ML через API.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ



Информация в этой главе основана на Elastic Stack, существующем в версии 7.10. Как и во всех главах, весь пример кода можно найти на GitHub: <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition>.

ТИПЫ ЗАДАНИЙ ELASTIC ML

На экране пользовательского интерфейса Elastic ML для настройки заданий по обнаружению аномалий отображаются пять разных мастеров настройки заданий (рис. 3.1).

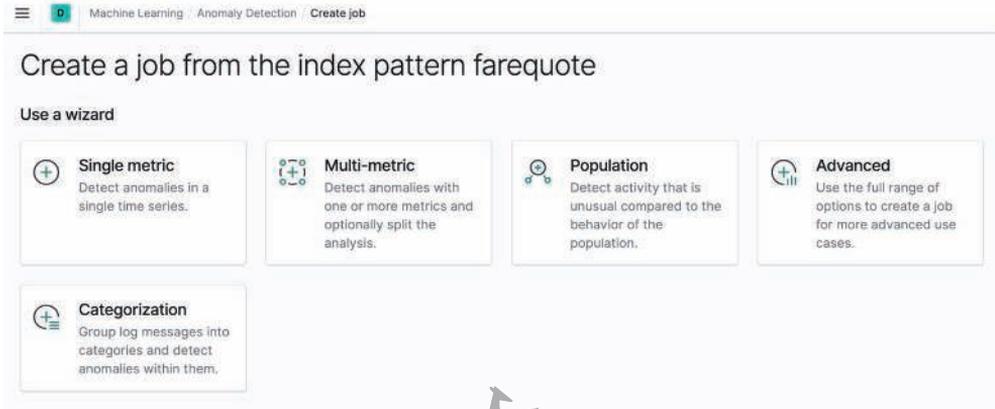


Рис. 3.1 ❖ Экран пользовательского интерфейса настройки заданий, предлагающий различные мастера настройки

Наличие разных мастеров настройки наводит на мысль, что существуют разные «типы» заданий. На самом деле существует только один тип задания, но задание по обнаружению аномалий имеет множество опций, и различные мастера упрощают определенные аспекты настройки конфигурации. С помощью мастера **Advanced** (или API) вы можете настроить любую опцию, какую пожелаете. Фактически, когда Elastic ML впервые выпустили в виде бета-версии в версии 5.4, это был единственный способ настройки. С тех пор в Elastic ML добавили другие мастера для простоты и удобства настройки под определенные сценарии использования.

Задание по обнаружению аномалий имеет множество параметров конфигурации, но двумя наиболее важными из них являются конфигурация анализа и поток данных.

Конфигурация анализа – это описание того, какие аномалии обнаружит задание. Она, в свою очередь, содержит конфигурацию обнаружения (называемую *детектором*), а также несколько других настроек, таких как диапазон сегмента. *Поток данных* – это конфигурация запроса, который будет выполнять Elasticsearch для извлечения данных, подлежащих последующему анализу детектором.

Особенности и назначение различных мастеров заданий можно определить так:

- задания, созданные мастером *Single metric*, имеют только один детектор. Их потоки данных содержат запросы и агрегации, поэтому они отправляют в алгоритмы ML только обобщенные данные. Агрегации создаются автоматически на основе параметров конфигурации в мастере. В задании также используется флаг `summary_count_field_name` (установленный со значением `doc_count`), чтобы сигнализировать о том, что следует ожидать агрегированные данные (а не исходные данные из хранилища-источника);
- задания, созданные мастером *Multi-metric*, могут иметь один или несколько детекторов. Анализ также можно разделить по категори-

альным полям, установив `partition_field_name` (о нем сказано далее в этой главе). Их потоки данных не содержат агрегаций (поскольку код машинного обучения должен видеть все документы для каждого возможного экземпляра значения поля и агрегировать его самостоятельно), поэтому алгоритмам ML передаются полные документы Elasticsearch;

- *задания, созданные мастером Population, могут иметь один или несколько детекторов.* Мастер также устанавливает флаг `over_field_name` (описанный далее в этой главе), сигнализирующий о том, что необходимо использовать анализ популяции. Анализ еще можно разделить по категориальным полям, установив `by_field_name` (о нем сказано далее в этой главе). Их потоки данных не содержат агрегаций, поэтому алгоритмам ML передаются полные документы Elasticsearch;
- *задания, созданные мастером Categorization, имеют только один детектор.* Мастер также устанавливает флаг `categorization_field_name` (описанный далее в этой главе), который сигнализирует о том, что следует использовать анализ категоризации. Анализ категоризации присваивает `by_field_name` (описан далее в этой главе) значение `mlcategory`. Анализ еще можно разделить по категориальным полям, установив `partition_field_name` (о нем сказано далее в этой главе). Их потоки данных не содержат агрегаций, поэтому алгоритмам ML передаются полные документы Elasticsearch;
- *задания, созданные мастером Advanced, могут использовать все доступные параметры.* Обязанность пользователя – знать, что он делает, и правильно настроить работу. Однако пользовательский интерфейс не позволяет пользователю совершать большинство ошибок. Опытный пользователь может обойтись использованием только мастера Advanced для создания любого задания по обнаружению аномалий.

Количество вариантов создания заданий может показаться устрашающим, особенно если учесть, что про них написано выше. Но не волнуйтесь – как только мы познакомимся с терминологией и рассмотрим несколько примеров, вы обнаружите, что конфигурации заданий очень разумны; по мере накопления опыта вы начнете конфигурировать задания, не задумываясь. Давайте сделаем следующий шаг и изучим устройство детектора.

УСТРОЙСТВО ДЕТЕКТОРА



В основе работы по обнаружению аномалий лежат конфигурация анализа и детектор. Детектор состоит из нескольких ключевых компонентов:

- *функция* (function);
- *поле* (field);
- *поле раздела* (partition field);
- *поле by* (by field);
- *поле over* (over field).

Мы рассмотрим каждый из них по очереди. Обратите внимание, что в следующих нескольких разделах мы часто будем ссылаться на фактические имена настроек в конфигурации задания, как если бы использовали расширенный редактор заданий или API. Хотя полезно понимать систему наименований, по мере чтения этой главы вы также заметите, что многие детали конфигурации задания абстрагируются от пользователя или имеют более удобные для пользовательского интерфейса метки, чем настоящие имена настроек.

Функция

Функция детектора описывает, как данные будут агрегированы или измерены в пределах интервала анализа (промежуток времени). Функций много, но их можно разделить на следующие категории, представленные в табл. 3.1.

Таблица 3.1. Категории функций детектора

Функции подсчета	Функции метрик	Прочие функции
count*	min	info_content*
non_zero_count*	max	rare
distinct_count*	metric	freq_rare
	mean*	lat_long
	median*	time_of_day
	sum*	time_of_week
	non_nuli_sum*	
	varp*	

Функции, отмеченные звездочкой (*), также имеют односторонние варианты с высоким или низким значением (например, `low_independent_count`), которые позволяют обнаруживать аномалии только в одном направлении.

Поле

Некоторые функции детектора для своей работы требуют наличия поля в данных. Вот несколько примеров функций с полями:

- `max(bytes);`
- `mean(products.price);`
- `high_distinct_count(destination.port).`

Имя поля, с которым функция напрямую работает, называется просто `field_name`.

Поле partition



Часто бывают случаи, когда анализ обнаружения необходимо разделить по категориальному полю, чтобы анализ можно было провести отдельно для

всех уникальных экземпляров этого поля. В этом случае поле раздела (параметр называется `partition_field_name`) определяет поле для разделения по категориям. Например, в электронной коммерции вам может потребоваться увидеть средний доход по категории (мужская одежда, женские аксессуары и т. д.). В этом случае поле `category` будет полем `partition`. Мы рассмотрим разделение анализа позже в этой главе.

Поле by

Подобно полю раздела, поле `by` (параметр называется `by_field_name`) – это еще один механизм разделения анализа, но он ведет себя по-разному в отношении того, как моделируются и оцениваются результаты. Кроме того, поле `by` является обязательным, если используются функции `gare` или `freq_gare`. Различия в использовании поля `by` для разделения по сравнению с полем `partition` мы подробнее обсудим позже в этой главе.

Поле over



Поле `over` (параметр называется `over_field_name`) сигнализирует алгоритмам обнаружения аномалий, что желателен *популяционный анализ*, где объекты сравниваются со своими сверстниками (а не с их собственным прошлым состоянием). Популяционный анализ подробно обсуждается далее в этой главе.

Формула детектора

Если бы мы задокументировали все возможные варианты конфигурации для детектора, а затем создали бы карту в виде блок-схемы, она выглядела бы как на рис. 3.2.

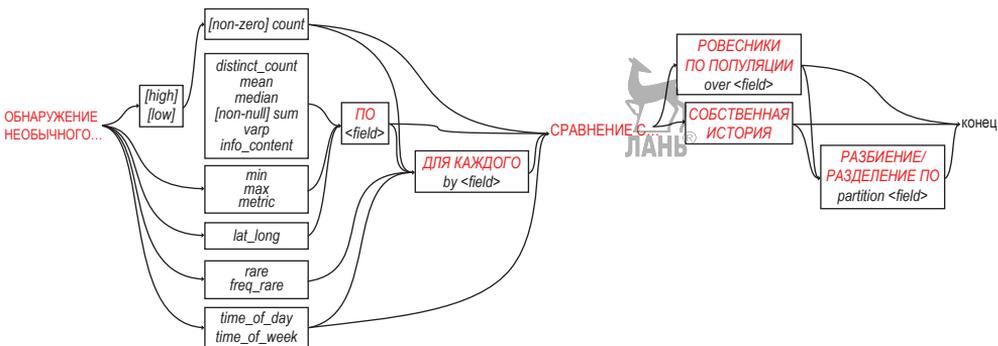


Рис. 3.2 ❖ Формула построения детектора с нуля

В диаграмме, показанной на рис. 3.2, применяются следующие обозначения:

- текст с прописной буквы – это объяснение, а курсивный текст – это настройки конфигурации детектора (`by_field_name`, `partition_field_name` и `over_field_name` сокращены до `partition` и `over`);
- пункты в квадратных скобках являются необязательными (`high`, `low`, `non-zero`, `non-null`);
- выберите только одну выходную ветвь (обратите внимание, что из `gag/freq_gag` выходит только одна ветвь, потому что параметр `by` является обязательным).

Сравнение какого-либо объекта с его собственной историей достигается просто за счет отказа от поля `over`.

Рассмотрев в деталях устройство детектора, мы переходим к практическим примерам различных сценариев использования детекторов. Мы начнем с применения функций подсчета `count`, которые позволяют нам обнаруживать изменения частотности событий с течением времени.

ОБНАРУЖЕНИЕ ИЗМЕНЕНИЙ ЧАСТОТНОСТИ СОБЫТИЙ

Есть много важных вариантов использования, которые основаны на идее обнаружения изменения частотности событий. К ним относятся, например, следующие сценарии:

- обнаружение потока сообщений об ошибках, внезапно появляющихся в файле журнала;
- обнаружение внезапного падения количества заказов, обрабатываемых онлайн-системой;
- определение внезапного чрезмерного количества попыток доступа (например, внезапного увеличения количества попыток входа в систему для определенного идентификатора пользователя).

Чтобы найти аномалию, мы сначала должны иметь механизм, позволяющий понять *нормальную* частоту событий. Но полагаться на наше ошибочное человеческое наблюдение и интуицию – не всегда самый простой (и не самый надежный) подход.

Подробнее о функциях `count`

Как упоминалось в главе 2, у заданий Elastic ML есть «рецепт» обнаружения аномалий, известный как *детектор*. Детектор – это ключ к определению того, какие аномалии пользователь хочет обнаружить. Внутри детектора есть *функция*, которая выбирает характерный признак того, что должно быть обнаружено. В случае функций `count` признаком является частота появления чего-либо с течением времени. Мы рассмотрим три основных варианта функции подсчета частотности:

- `count`: подсчитывает количество документов в сегменте, полученных в результате запроса к хранилищу необработанных данных `raw_data`;

- `high_count`: то же самое, что и `count`, но помечает аномалию только в том случае, если число больше ожидаемого;
- `low_count`: то же самое, что и `count`, но помечает аномалию только в том случае, если число меньше ожидаемого.

Позже вы увидите, что в Elastic ML есть множество *односторонних функций* (только для обнаружения аномалий в определенном направлении). Кроме того, важно знать, что функции подсчета не подсчитывают значение поля или даже наличие полей в документе; они просто подсчитывают количество документов в хранилище с течением времени.

Чтобы получить более полное представление о том, что делают функции подсчета, давайте перейдем к простому примеру, использующему образцы данных в Kibana.

1. Чтобы добавить образцы данных из любого источника, на главном экране Kibana нажмите кнопку **Add data** (Добавить данные), как показано на рис. 3.3:

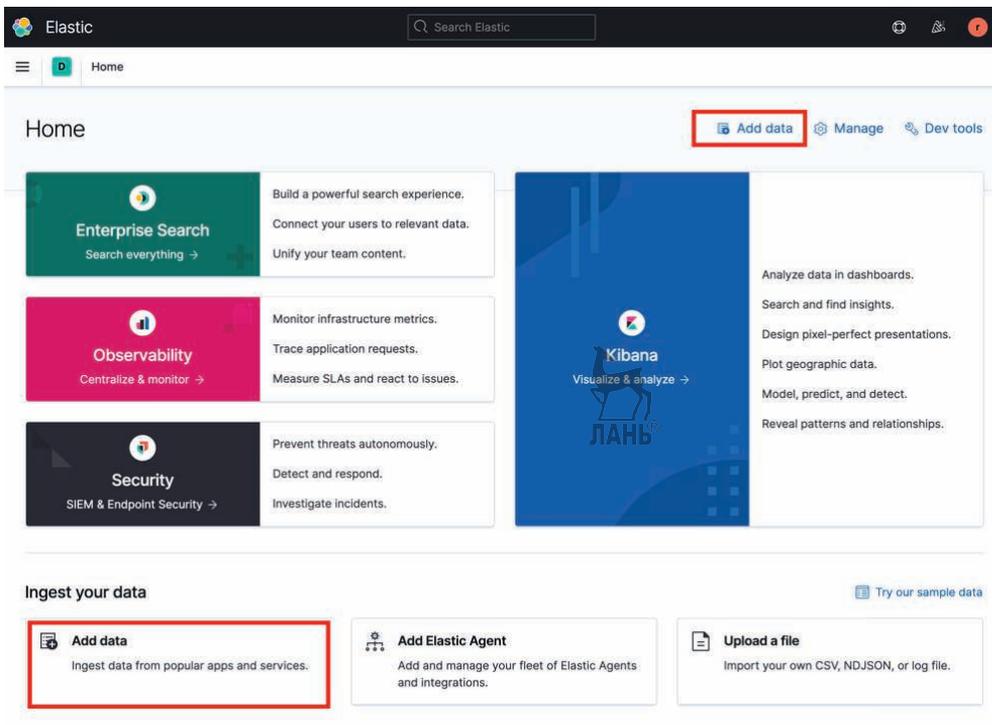


Рис. 3.3 ❖ Главный экран Kibana с опциями добавления данных

2. После нажатия кнопки **Add data** выберите **Sample data** (Образец данных), чтобы отобразить три набора данных (рис. 3.4).

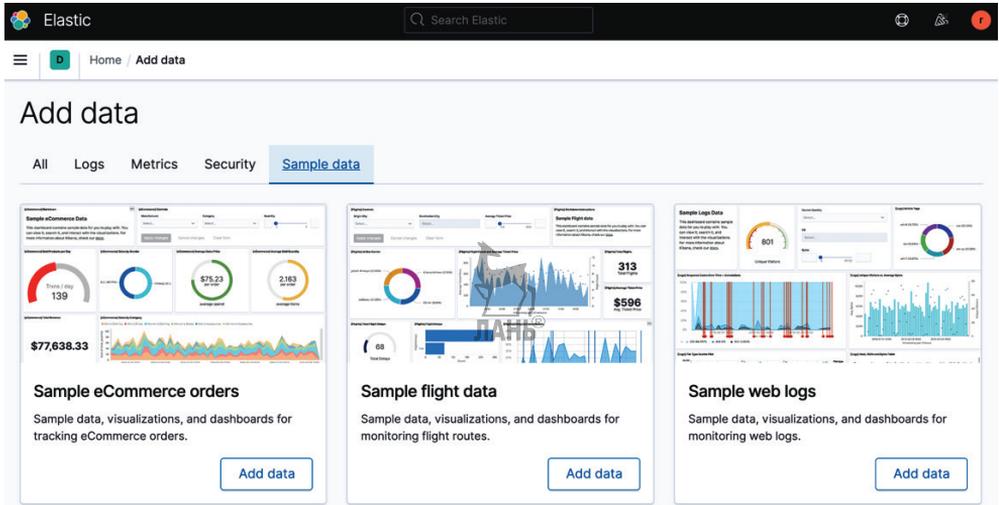


Рис. 3.4 ❖ Добавление образцов данных

3. Нажмите каждую из трех кнопок **Add data** в каждом разделе, чтобы загрузить этот образец набора данных в свой Elastic Stack. После завершения загрузки мы перейдем непосредственно к машинному обучению, выбрав значок меню с тремя горизонтальными линиями (☰) в верхнем левом углу Kibana, чтобы открыть список приложений, а затем выберите **Machine Learning** (Машинное обучение), как показано на рис. 3.5.

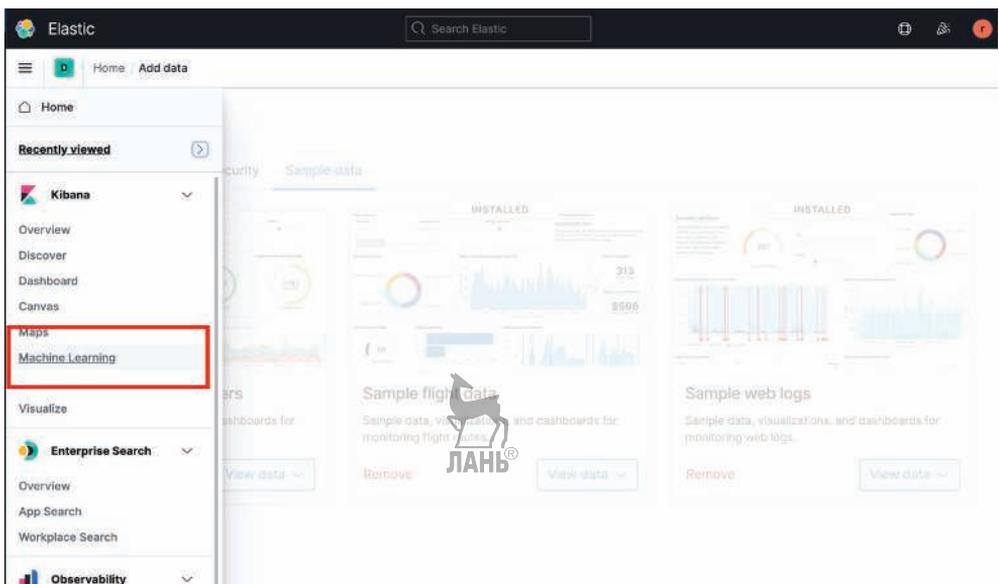


Рис. 3.5 ❖ Выбор машинного обучения в меню приложений Kibana

- После перехода по ссылке мы окажемся на странице обзора машинного обучения, где сразу можем создать наше первое задание по обнаружению аномалий. Нажмите кнопку **Create job** (Создать задание), как показано на рис. 3.6.

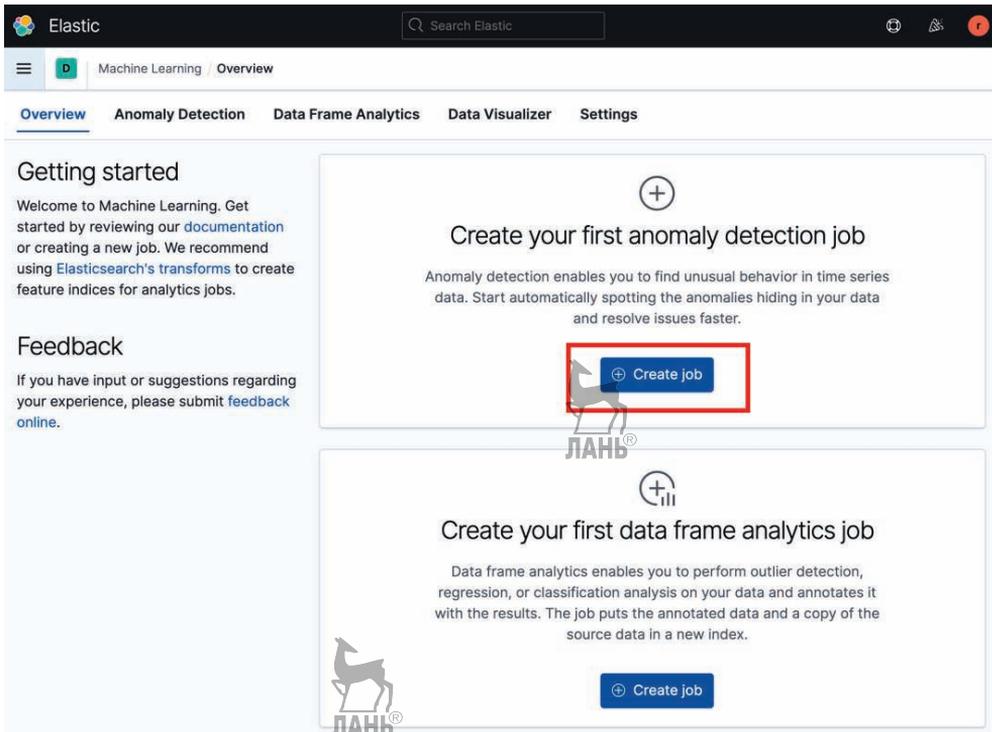


Рис. 3.6 ❖ Экран приветствия Elastic Cloud

- Наша следующая задача – выбрать шаблон хранилища (отмечен значком с изображением осколков) или сохраненный поиск (отмечен значком увеличительного стекла), содержащий данные, которые мы хотели бы проанализировать. Если выбран сохраненный поиск, то основой вашей работы по машинному обучению будет отфильтрованный запрос, который был ранее создан и сохранен в приложении Kibana Discover. Однако в этом примере мы выберем хранилище `kibana_sample_data_logs` (рис. 3.7), так как хотим передать каждый документ в этом хранилище через Elastic ML.

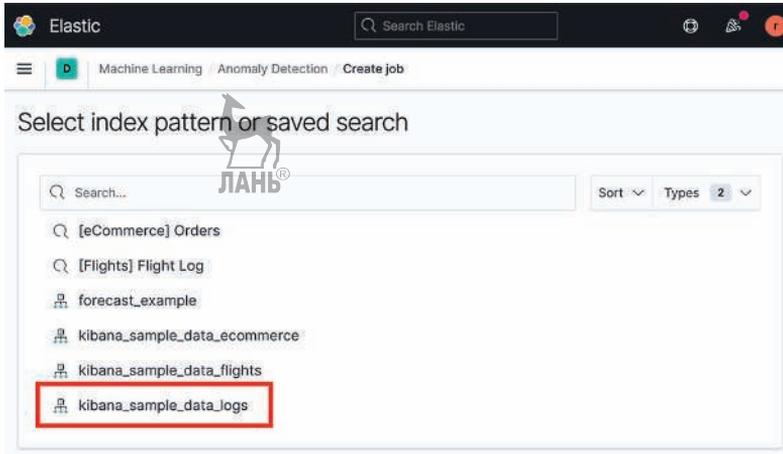


Рис. 3.7 ❖ Выбор хранилища `kibana_sample_data_logs` для анализа

6. На следующем экране (рис. 3.8) мы выберем мастер задания одиночной метрики **Single metric**, потому что на данном этапе мы заинтересованы в анализе только одного аспекта данных: их подсчета с течением времени.

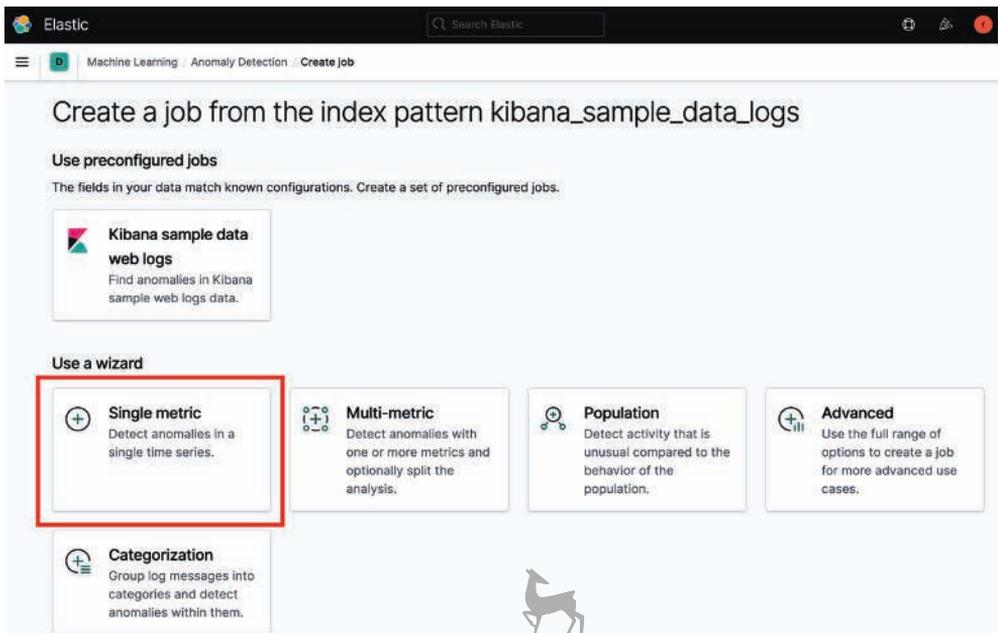


Рис. 3.8 ❖ Выбор задания с одной метрикой **Single metric**

7. На следующем экране, в соответствии с нашим примером, вы *должны (это важно!)* выбрать кнопку **Use full kibana_sample_logs_data** (Использовать полное хранилище `kibana_sample_logs_data`), чтобы включить аномальную выборку в этот набор данных (рис. 3.9).

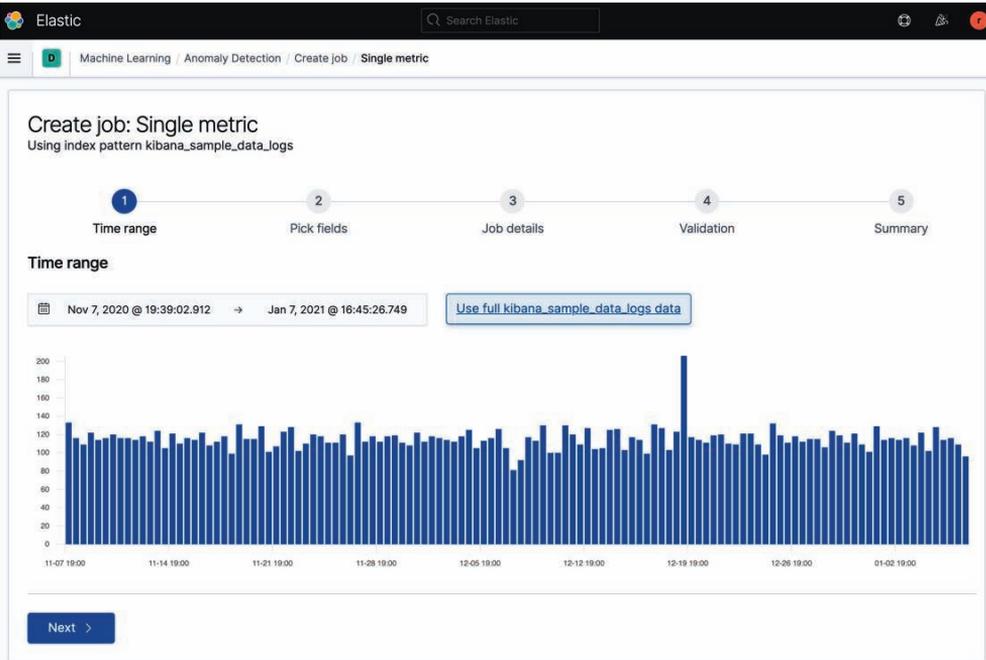


Рис. 3.9 ❖ Использование всех данных в хранилище

! Эти демонстрационные данные, когда они установлены, фактически помещают около половины данных в «прошлое» и половину в «будущее» (путем динамического изменения меток времени при захвате). Это сделано для того, чтобы статические данные могли выглядеть как «в реальном времени», например при просмотре панелей мониторинга данных за «последний час». В результате мы попросим Elastic ML проанализировать данные из прошлого и будущего, хотя в реальной жизни невозможно получить данные из будущего. Ради простоты примера пока не обращайтесь на это внимание, поскольку аномалия, которую мы хотели бы продемонстрировать, находится во второй половине набора данных (т. е. в «будущем»).

Теперь нажмите кнопку **Next** (Далее), чтобы перейти к следующему шагу мастера настройки.

8. После нажатия кнопки **Next** нам нужно будет выбрать то, что мы хотим проанализировать, в раскрывающемся списке **Pick fields** (Выбрать поля). Мы выберем опцию **Count (Event rate)** (Подсчет/частотность события), чтобы сосредоточиться на нашей исходной цели, которая заключается в обнаружении изменений в частоте событий в этом хранилище с течением времени (рис. 3.10).

The screenshot displays the 'Create job: Single metric' configuration page in the Elastic UI. The page is titled 'Using index pattern kibana_sample_data_logs'. A progress indicator at the top shows five steps: 1. Time range (completed), 2. Pick fields (current step), 3. Job details, 4. Validation, and 5. Summary. The 'Pick fields' section features a dropdown menu with the following options: 'Count(Event rate)' (selected), 'High count(Event rate)', 'Low count(Event rate)', 'agent.keyword' (with sub-option 'Distinct count(agent.keyword)'), and 'bytes'. Below the dropdown is a line chart showing the event rate over time, with a significant spike around 12-19 19:00. At the bottom of the configuration area, there are settings for 'Bucket span' (15m) and 'Sparse data' (selected). Navigation buttons for 'Previous' and 'Next' are located at the bottom left.

Рис. 3.10 ❖ Выбор количества событий с течением времени в качестве объекта обнаружения

Обратите внимание, что в выпадающем списке можно выбрать и другие варианты анализа в зависимости от типа данных поля. Мы рассмотрим некоторые из этих вариантов позже, в следующих примерах. Нажмите кнопку **Next**, чтобы продолжить, оставив на данный момент другие параметры по умолчанию.

9. Теперь нам нужно дать имя нашей задаче по обнаружению аномалий. В поле **Job ID** (Идентификатор задания) введите интуитивно понятное имя. В этом примере мы используем имя `web_logs_rate` (рис. 3.11). Снова оставьте другие параметры по умолчанию и нажмите кнопку **Next**.
10. Далее следует шаг проверки задания, чтобы убедиться, что все настроено правильно и задание работоспособно (рис. 3.12). Нажмите кнопку **Next**, чтобы продолжить.

Elastic

Machine Learning / Anomaly Detection / Create job / Single metric

Create job: Single metric

Using index pattern kibana_sample_data_logs

Time range Pick fields **3** Job details Validation Summary

Job details

Job ID
A unique identifier for the job. Spaces and the characters / ? , " < > | * are not allowed

Job ID

Job description
Optional descriptive text

Job description

Groups
Optional grouping for jobs. New groups can be created or picked from the list of existing groups.

Groups

> Additional settings

> Advanced

< Previous

Рис. 3.11 ❖ Присвоение имени заданию по обнаружению аномалий

Elastic

Machine Learning / Anomaly Detection / Create job / Single metric

Create job: Single metric

Using index pattern kibana_sample_data_logs

Time range Pick fields Job details **4** Validation Summary

Validation

✓ Time range
Valid and long enough to model patterns in the data.

✓ Model memory limit
Valid and within the estimated model memory limit. [Learn more](#)

< Previous

Рис. 3.12 ❖ Шаг проверки задания

11. На этом этапе задание готово к созданию. Обратите внимание на рис. 3.13 – для вас были выбраны некоторые разумные параметры по умолчанию, такие как **Model memory limit** (Ограничение памяти модели) и **Enable model plot** (Включить построение графика модели).

The screenshot shows the 'Create job: Single metric' configuration page in the Elastic Machine Learning console. The page is titled 'Using index pattern kibana_sample_data_logs'. A progress bar at the top indicates the current step is 'Summary' (5), with previous steps 'Time range', 'Pick fields', 'Job details', and 'Validation' completed. Below the progress bar, a line graph titled 'New job from index pattern kibana_sample_data_logs' shows a periodic signal with a prominent spike. The x-axis represents time from 11:07:19.000 to 01:02:19.000. Below the graph, the job configuration is displayed in a grid:

Job ID web_logs_rate	Bucket span 15m	Enable model plot True	Start Nov 7, 2020 @ 19:39:02.912
Job description No description provided	Influencers No influencers selected	Use dedicated index False	End Jan 7, 2021 @ 18:45:28.749
Groups No groups selected		Model memory limit 11MB	

At the bottom, there is a 'Start immediately' toggle (checked) and a note: 'If unselected, job can be started later from the jobs list.' Navigation buttons include 'Previous', 'Create job', 'Preview JSON', and 'Convert to advanced job'.

Рис. 3.13 ❖ Задание на обнаружение аномалий готово к созданию

12. После нажатия кнопки **Create job** (Создать задание) вы увидите анимированный предварительный просмотр результатов, наложенных поверх данных, как показано на рис. 3.14. Теперь нажмите кнопку **View results** (Просмотр результатов), чтобы подробно изучить, что было обнаружено заданием по обнаружению аномалий в данных.

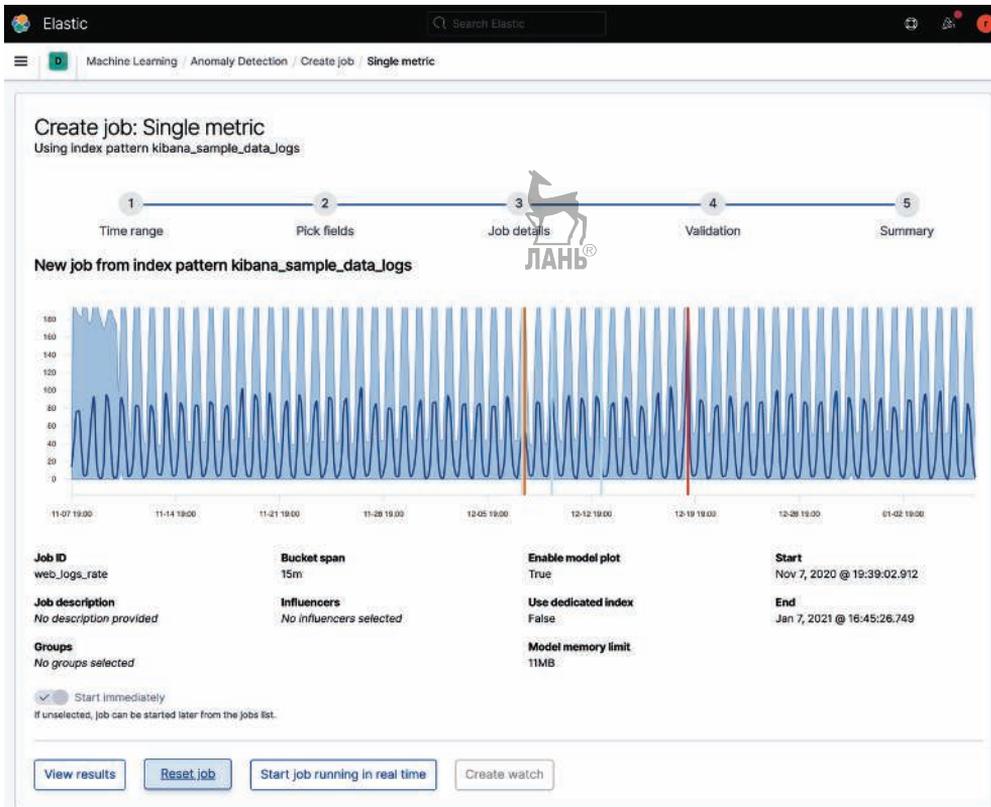


Рис. 3.14 ❖ Предварительный просмотр результатов выполнения задания

- Используя ползунок под основным графиком, отрегулируйте положение и ширину области просмотра, чтобы увеличить большой пик (рис. 3.15).

! При увеличении, уменьшении и изменении масштаба просто помните об интервале агрегирования графика по сравнению с диапазоном периода задания (обведено на рис. 3.15). Если вы увеличили масштаб до более широкого представления, интервал агрегирования графика может оказаться больше, чем диапазон сегмента задания, что делает положение аномалии на графике менее точным.

На рис. 3.15 мы можем видеть, что очень большой всплеск событий был отмечен как две отдельные аномалии, потому что фактическое количество веб-запросов, замеченных в журналах, было примерно в 11 раз выше ожидаемого (с учетом изученной модели данных до момента аномалии). График показывает две аномалии рядом друг с другом, потому что очевидно, что событие-всплеск охватило более одного 15-минутного временного интервала. Вы также можете заметить, что по умолчанию в таблице под графиком отображается только одна аномалия. Это связано с тем, что для параметра

Interval (Интервал) по умолчанию установлено значение **Auto** (Авто), а аномалии, смежные по времени, суммируются вместе, и отображается только наивысший балл. Если для параметра **Interval** установлено значение **Show all** (Показать все), в таблице будут перечислены обе записи об аномалиях (рис. 3.16).

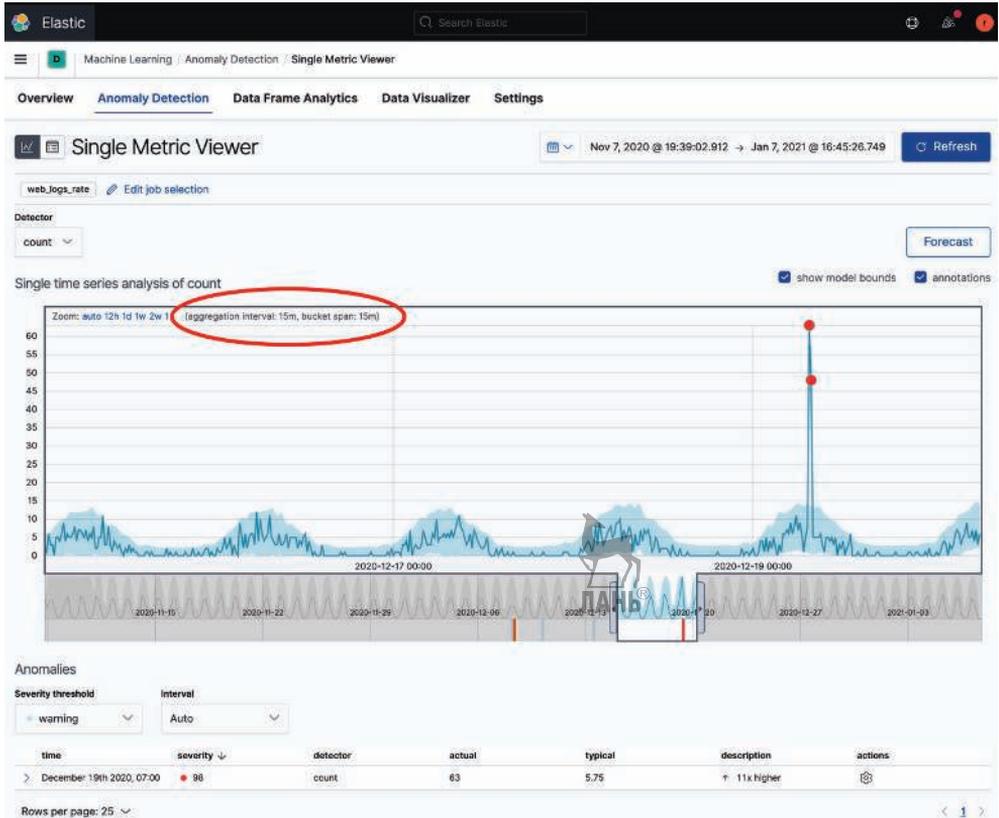


Рис. 3.15 ❖ Результаты свидетельствуют о наличии критической аномалии

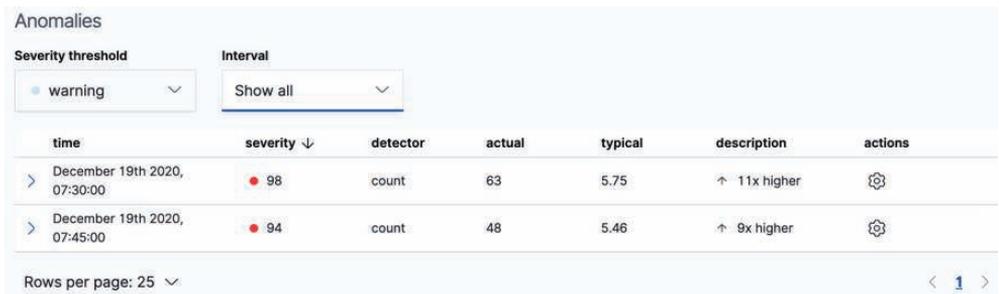


Рис. 3.16 ❖ Настройка интервала для отображения всех аномалий

В этом примере есть еще одна вещь, на которую следует обратить внимание, а именно другие, менее выраженные аномалии, встреченные ранее в наборе данных (рис. 3.17).

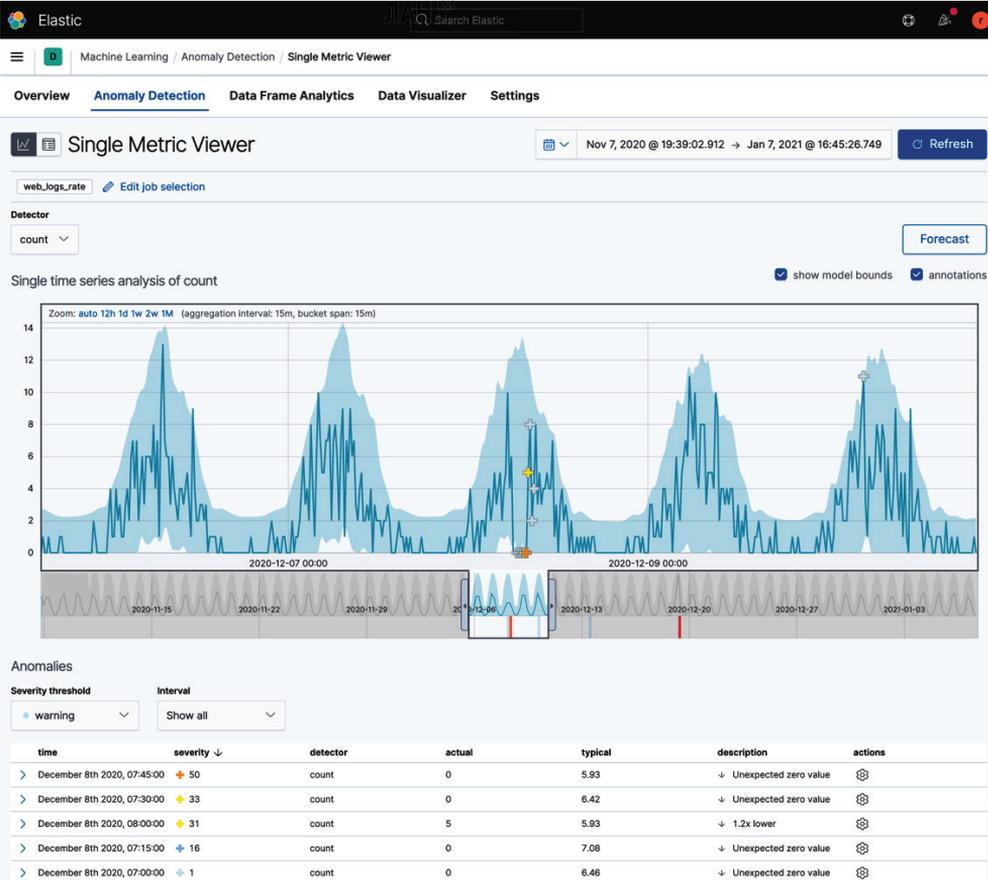


Рис. 3.17 ❖ Аномалии нескольких сегментов

В отношении этих менее очевидных аномалий следует признать несколько ключевых моментов:

- их оценка ниже, чем у массивного всплеска, который мы только что исследовали, поэтому они, условно говоря, не такие уж аномальные, но весьма интересны и заслуживают внимания;
- аномалия здесь – «отсутствие» ожидаемых значений. Другими словами, функция count интерпретирует отсутствие данных (событий) как 0, и это может быть аномалией, если ожидается, что события должны были произойти;
- эти аномалии не являются аномалиями одного сегмента, а, напротив, представляют собой аномалии в нескольких сегментах (multi-bucket

anomalies). Такие аномалии обозначаются другим символом в пользовательском интерфейсе (крестик вместо точки). Они обозначают случаи, в которых фактическое сингулярное значение не обязательно может быть аномальным, но есть тенденция, которая возникает в скользящем окне из 12 последовательных сегментов. Здесь вы можете увидеть заметный спад, охватывающий несколько соседних сегментов.

! Дополнительные пояснения об интерпретации аномалий, представленных в нескольких сегментах, вы можете найти в подробной статье по адресу <https://www.elastic.co/blog/interpreting-multi-bucket-impact-anomalies-using-elastic-machine-learning-features>.

На этом примере мы видели, как функция подсчета позволяет нам легко обнаруживать очевидный (и даже не столь очевидный) набор аномалий, связанных с общей частотой появления событий (документов) в хранилище с течением времени. Давайте продолжим наше исследование, рассмотрев другие функции подсчета и вхождения.

Другие функции подсчета

В дополнение к функциям, которые мы уже рассмотрели, есть несколько других функций подсчета, которые предоставляют нам более широкий выбор вариантов использования.

Ненулевой подсчет

Функции ненулевого подсчета (`non_zero_count`, `low_non_zero_count` и `high_non_zero_count`) позволяют обрабатывать анализ на основе подсчета, а также поддерживают точное моделирование в тех случаях, когда данные могут быть разреженными и вы хотели бы обработать отсутствие данных не как ноль, а как `null` – другими словами, набор данных временного ряда, который выглядит следующим образом:

4,3,0,0,2,0,5,3,2,0,2,0,0,1,0,4

функцией `non_zero_count` будет интерпретирован так:

4,3,2,5,3,2,2,1,4

Обработка нулей как `null` может быть полезна в случаях, когда ожидается отсутствие регулярных измерений через равные промежутки времени. Вот некоторые практические примеры таких данных:

- количество авиабилетов, приобретенных в месяц физическим лицом;
- количество перезагрузок сервера за сутки;
- количество попыток входа в систему в час.

Чтобы выбрать ненулевую версию функций подсчета в мастерах заданий, просто включите опцию **Sparse data** (Разреженные данные) во время настройки (рис. 3.18).

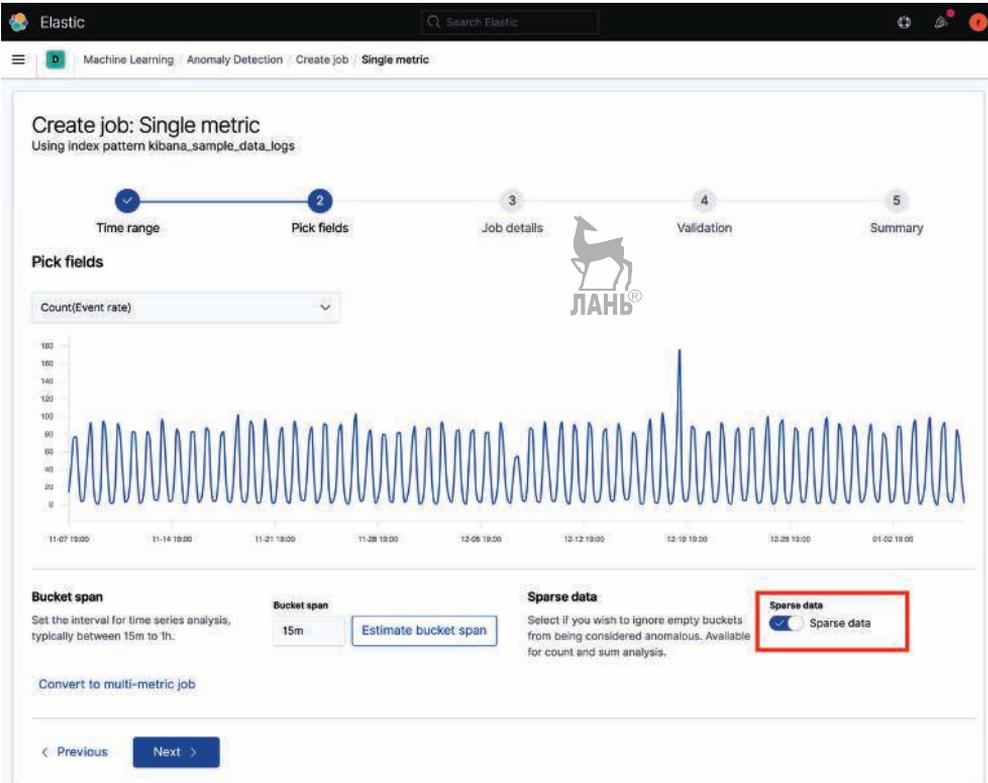


Рис. 3.18 ❖ Добавление опции **Sparse data** для выбора ненулевой функции подсчета

Позже в этой главе, во время настройки задания в расширенном мастере заданий или через API, мы будем явно использовать имена функций (например, `high_non_zero_count`) вместо переключения параметров с более общими описаниями.

Раздельный подсчет

Функции раздельного подсчета (`unique_count`, `low_distinct_count` и `high_independent_count`) измеряют *уникальность* (мощность, *cardinality*) значений для конкретного поля. Есть много возможных применений этой функции, особенно когда она используется в контексте анализа популяции (см. далее в этой главе) для обнаружения объектов, которые содержат чрезмерно разнообразный набор значений полей. Хороший классический пример – поиск IP-адресов, которые выполняют сканирование портов, получают доступ к необычно большому количеству различных номеров портов назначения на удаленных машинах или отправляют множество разных URL-адресов на веб-сервер – другими словами, действуют как автоматизированный бот, а не типичный пользователь-человек. Например, если бы в качестве конфигурации детектора в последнем примере по хранилищу `kibana_sample_data_logs`

мы выбрали `distinct_count(url.keyword)`, мы бы выявили тот же аномальный фрейм, но по другой причине – не только аномально высокий общий объем запросов, как показано на рис. 3.15, но и аномально большое разнообразие запрашиваемых URL (рис. 3.19).

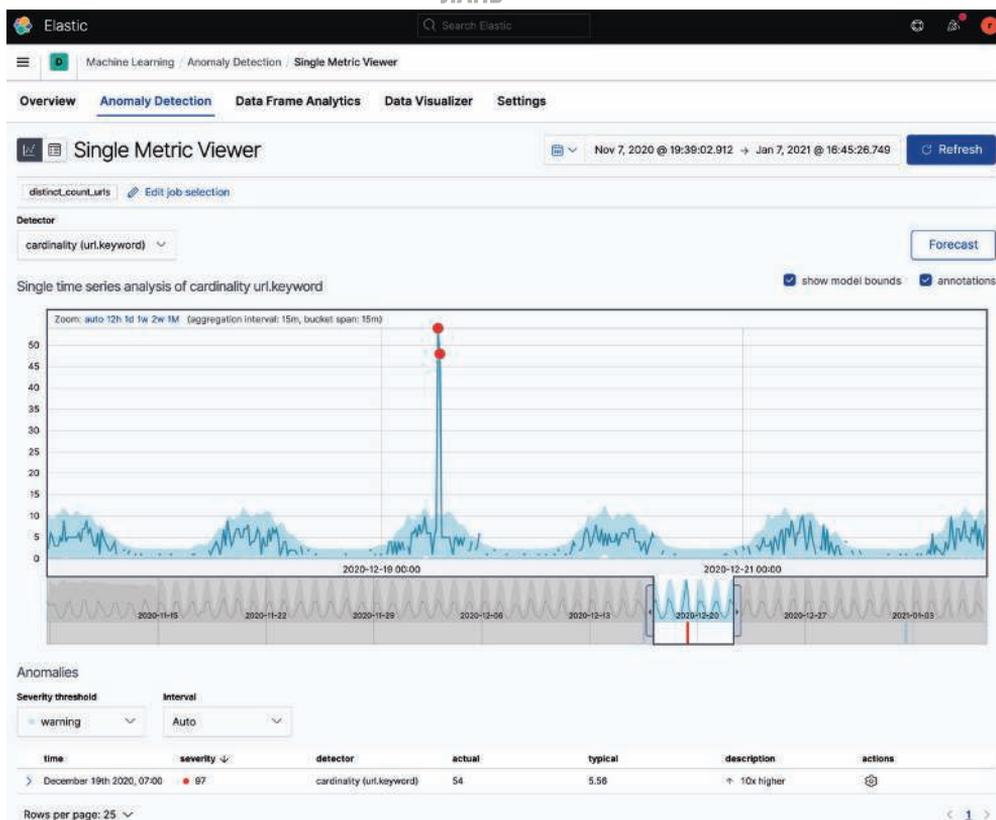


Рис. 3.19 ❖ Пример детектора раздельного подсчета

На этом мы закончим рассматривать функции на основе подсчета и обратимся к функциям на основе показателей, которые позволяют нам анализировать числовые поля в данных.

ОБНАРУЖЕНИЕ ИЗМЕНЕНИЙ ЗНАЧЕНИЙ ПОКАЗАТЕЛЕЙ

Очевидно, что не все данные, передаваемые из систем, будут текстовыми или категориальными по своей природе – большая часть их будет числовой. Обнаружение изменений значений показателей с течением времени идеально подходит для обнаружения аномалий, потому что, как упоминалось в главе 1, историческая парадигма предупреждения об аномальных числовых

показателях с помощью статических пороговых значений десятилетиями создавала проблемы. Давайте посмотрим, что Elastic ML может предложить в отношении функций, помогающих обнаруживать аномальные изменения в числовых полях ваших данных.

Метрические функции

Метрические функции (metric functions) работают с числовыми полями и возвращают числовые значения. Это, пожалуй, самые простые для понимания функции детектора.

min, max, mean, median u metric

Эти функции работают именно так, как можно ожидать из их названия: они возвращают минимум, максимум, среднее и медианное значение всех числовых наблюдений для интересующего поля в заданном промежутке времени.

Функция `metric` немного уникальна тем, что на самом деле это просто сокращенный способ указать, что минимальное, максимальное и среднее значение должны использоваться вместе.

Следует отметить, что если частота данных (например, данных, поступающих из источника выборки, такого как Metricbeat) в точности совпадает с диапазоном сегмента, то существует только одна выборка на диапазон сегмента. Это означает, что минимум, максимум, среднее значение и медиана интересующей области – это одно и то же значение (значение отдельного наблюдения как такового). Поэтому, если возможно, обычно лучше иметь несколько числовых выборок на интервал сегмента, если вы хотите реализовать дискриминацию с помощью этих функций.

Еще один факт, который следует отметить, заключается в том, что эти метрические функции обрабатывают отсутствие данных как `null`. Другими словами, если ваши данные немногочисленны и есть интервалы сегментов, в которых отсутствуют наблюдения, эти отсутствующие данные не будут «притягивать к нулю» статистику интересующей области. Вот почему эти метрические функции не имеют «ненулевых» (или «не-null») аналогов.

varp

Функция `varp` измеряет общую дисперсию показателя во времени – его волатильность. Использование этой функции может быть применимо для поиска случаев, когда числовое значение поля обычно является довольно равномерным, но вы хотели бы определить, было ли изменение.

sum u non-null sum

Функция `sum` вернет сумму всех числовых наблюдений для интересующего поля в интервале времени сегмента. Используйте версию `non-null sum`, если у вас есть разреженные данные и вы не хотите, чтобы их отсутствие считалось нулем, что неизбежно «притянет к нулю» значение суммы.

Если бы мы выбрали `sum(bytes)` в качестве конфигурации детектора в последнем примере по хранилищу `kibana_sample_data_logs`, мы бы обнаружили ту же аномалию, но по другой причине – мы видим, что сделанные запросы также привели к увеличению количества байтов, переданных с веб-сервера (рис. 3.20).

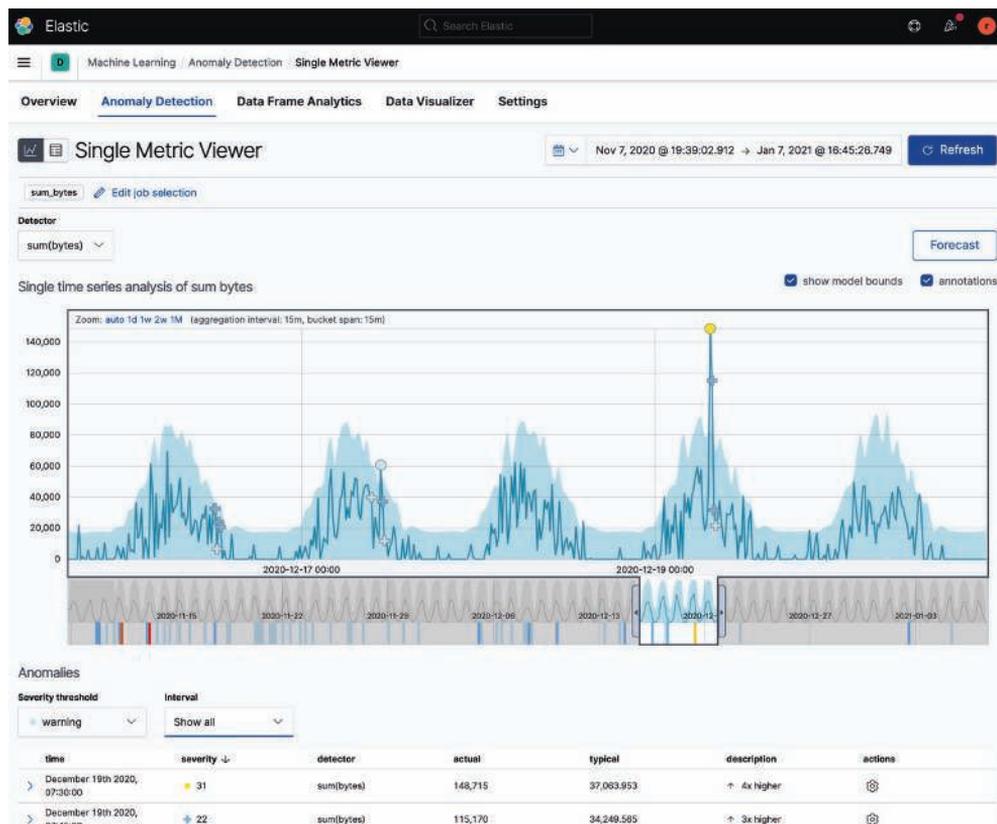


Рис. 3.20 ❖ Пример детектора `sum`

Это совершенно разумно, учитывая, что увеличение количества запросов к веб-серверу будет коррелировать с увеличением количества передаваемых байтов.

Теперь, когда мы оценили относительно простые функции детектора, давайте перейдем к более сложным и продвинутым функциям.

ОБЗОР РАСШИРЕННЫХ ФУНКЦИЙ ДЕТЕКТОРА

В дополнение к упомянутым выше базовым функциям детектора есть также несколько других, более продвинутых функций, которые обеспечивают некоторые очень уникальные возможности. Иные из этих функций доступ-

ны только в том случае, если задание машинного обучения настроено с помощью мастера расширенных заданий или через API.

Функция rare



В контексте потока временной информации (например, файла журнала) представление о чем-то статистически редком (происходящем с низкой частотой) парадоксальным образом является одновременно очевидным и трудным для понимания. Если бы меня попросили, например, пролистать файл журнала и найти редкое сообщение, у меня мог возникнуть соблазн пометить первое новое сообщение, которое я увидел, как редкое. Но что, если практически каждое сообщение было новым? Все они являются редкими? Или наоборот, в этом случае нет редких сообщений?

Чтобы определить редкость как полезное понятие в контексте потока событий во времени, мы должны согласиться с тем, что объявление чего-либо редким должно учитывать контекст, в котором оно существует. Если есть много других рутинных объектов и мало уникальных, то мы можем считать уникальные объекты редкими. Если есть много уникальных объектов, то мы будем считать, что ни один из них не является редким.

При применении функции rare в задании машинного обучения необходимо объявить, на каком поле эта функция фокусируется. Это поле затем определяется как `by_field_name`. Конфигурация функции rare не имеет собственного мастера в пользовательском интерфейсе Elastic ML, поэтому вам нужно будет определить ее с помощью мастера расширенных заданий. Например, чтобы найти записи журнала, в которых упоминается редкое название страны, структурируйте свой детектор, как показано на рис. 3.21.

Create detector

Function Analysis functions to be performed e.g. sum, count.	Function <input type="text" value="rare"/>	Field Required for functions: sum, mean, median, max, min, info_content, distinct_count.	Field <input type="text"/>
By field Required for individual analysis where anomalies are detected compared to an entity's own past behavior.	By field <input type="text" value="geo.src"/>	Over field Required for population analysis where anomalies are detected compared to the behavior of the population.	Over field <input type="text"/>
Partition field Allows segmentation of modeling into logical groups.	Partition field <input type="text"/>	Exclude frequent If set, it will automatically identify and exclude frequently occurring entities which may otherwise have dominated results.	Exclude frequent <input type="text"/>
Description Override the default detector description with a meaningful description of what the detector is analyzing.	Description <input type="text" value="rare by 'geo.src'"/>		
			<input type="button" value="Cancel"/> <input type="button" value="Save"/>

Рис. 3.21 ❖ Пример детектора rare

Это может быть удобно для поиска доступа из неожиданного географического региона (например, «Наши администраторы обычно почти ежедневно входят в систему из офисов в Нью-Йорке и Лондоне, но никогда из Москвы!»).

Функция `freq_gage`

Функция `freq_gage` – это специализированная версия функции `gage`, поскольку она ищет членов популяции, которые вызывают частое появление редких значений `by_field_name`. Например, вы можете найти конкретный IP-адрес, с которого идут запросы на доступ ко многим редким URL-адресам, которые обычно не характерны для всей совокупности клиентских IP-адресов. Можно предположить, что с этого IP-адреса кто-то пытается незаконно получить доступ к скрытым разделам веб-сайта или предпринимает попытки атак, таких как SQL-инъекция.

Функция `info_content`

Функция `info_content` – пожалуй, самая специализированная функция детектора в арсенале Elastic ML. Первоначально она была разработана как средство измерения количества энтропии в текстовых строках (как много в ней символов, и насколько они разнообразны). Это связано с тем, что во вредоносных программах используются хорошо известные методы шифрования инструкций и/или данных полезной нагрузки для передачи управления и контроля (`command and control`, C2) и действий по краже данных. Обнаружение подобной активности по этой функции данных более надежно, чем просмотр других функций (таких как количество отправленных байтов или подсчет отдельных объектов).

Используемый алгоритм по существу выполняет следующие шаги:

- 1) сортирует уникальные строки в алфавитном порядке;
- 2) объединяет эти уникальные строки в одну длинную строку;
- 3) применяет алгоритм `gzip` для сжатия этой длинной строки. Информационное содержание – это длина сжатых данных.

Некоторые из заданий машинного обучения в Elastic SIEM используют функцию `info_content` – мы расскажем о ней в главе 8.

Функции геолокации

Если вы ищете географическое положение, которое необычно для изученного ранее региона местоположения на Земле, вам пригодится функция `lat_long`, в которой аргумент `field_name` представляет собой пару чисел, разделенных запятыми, в диапазоне от -180 до 180 (например, `40.75, -73.99` – это координаты Таймс-сквер в Нью-Йорке). Функция `lat_long` также может работать с полем `geo_point`, полем `geo_shape`, содержащим значения точек, или агрегацией `geo_centroid`. Примером использования функции может быть от-

метка ненормального (и потенциально мошеннического или вредоносного) местоположения для конкретного пользователя, транзакции и т. д.

Функции времени

Не все события происходят случайно во времени, особенно когда речь идет о человеческом поведении. Мы можем принимать пищу, ездить на работу или входить в определенные системы в предсказуемое время дня или недели. Используя функции `time_of_day` и `time_of_week`, вы можете обнаруживать изменения поведения в рамках изученной временной процедуры. Если поведение предсказуемо на 24-часовом отрезке времени, то больше подходит `time_of_day`. Если обычное поведение зависит от дня недели, то более логичным выбором будет `time_of_week`.



- ❗ Не путайте использование этих функций времени с обучением детекторов на естественных последовательностях в заданиях по обнаружению аномалий. Как было сказано в главе 1, для устранения тенденций модель изучает время, в которое что-то происходит. Эти функции просто моделируют временную метку события в течение дня или недели. Например, если что-то обычно случается каждые сутки в 2 часа ночи, функция научится тому, что обычно это происходит на 7200-й секунде дня.

Теперь, когда мы изучили весь каталог функций детекторов, давайте пойдем дальше и посмотрим, как мы можем расширить возможности нашего анализа, разделив моделирование по объектам, которые представлены категориальными полями.



РАЗДЕЛЕНИЕ АНАЛИЗА ПО КАТЕГОРИАЛЬНЫМ ПРИЗНАКАМ

Вы уже смогли оценить мощь заданий по обнаружению интересных аномалий в одном наборе данных временного ряда. Однако есть несколько механизмов, с помощью которых анализ можно разделить по категориальному полю, чтобы запустить параллельный анализ по десяткам, сотням и даже нескольким тысячам уникальных объектов.

Настройка поля разделения

При использовании некоторых мастеров заданий (например, мастеров `Multi-metric` и `Population`) вы увидите возможность разделить анализ (рис. 3.22).

Здесь, на рис. 3.22, демонстрирующем использование мастера `Multi-metric` для создания задания по хранилищу `kibana_sample_data_ecommerce`, мы видим, что функция `high_sum` в поле `taxful_total_price` разделяется на экземпляры в поле с именем `category.keyword` (плюс включен параметр **Sparse data**).

Другими словами, анализ будет проводиться для каждой категории товаров в этом магазине электронной коммерции (мужская одежда, женские аксессуары и т. д.). Если анализ выполняется и результаты проверяются с помощью пользовательского интерфейса Anomaly Explorer, результат может выглядеть приблизительно так, как показано на рис. 3.23.

The screenshot shows the 'Create job: Multi-metric' configuration page in the Elastic Anomaly Explorer. The progress bar indicates the 'Pick fields' step is active. The main area displays a line chart for 'High sum(taxful_total_price)' with a significant spike for 'Men's Clothing' around 11-08 19:00. Below the chart, configuration options are set: 'Split field' is 'category.keyword', 'Influencers' is 'category.keyword', 'Bucket span' is '15m', and 'Sparse data' is checked.

Рис. 3.22 ❖ Разделение по категориальному полю

Обратите внимание на рис. 3.23: представление Anomaly Explorer отличается от того, что мы видели до сих пор в Single Metric Viewer. Anomaly Explorer показывает 10 самых аномальных категорий (поле, по которому выполняется разделение) временного ряда. Заметим, что показаны не все категории, а только те, которые имеют аномалии, – и очевидно, что категория «Мужская одежда» была самой необычной с выручкой 9 ноября в размере 2250 долларов США (в этой версии набора данных). Вы узнаете больше об интерпретации результатов заданий с несколькими критериями (multi-metric) и научитесь широко использовать Anomaly Explorer в главе 5.

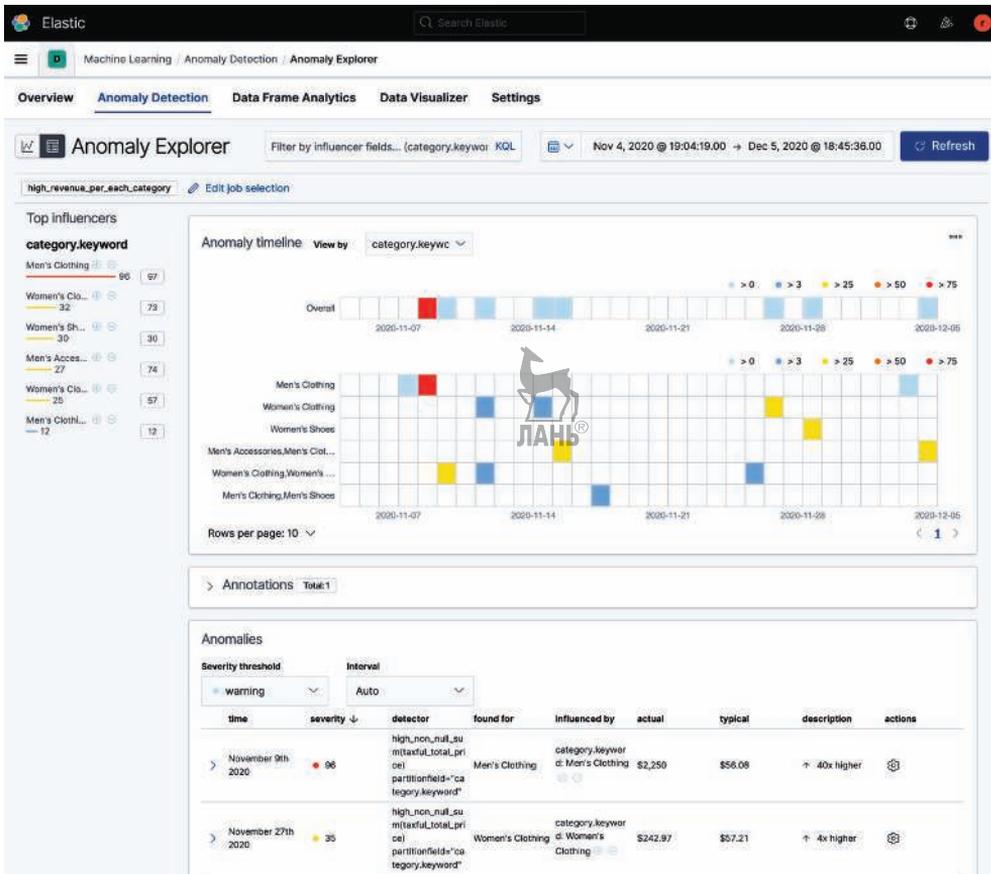


Рис. 3.23 ❖ Результаты раздельного анализа по категориям

Разница между разделением с использованием `partition` и `by_field`

Как было сказано ранее, когда задействованы мастер `Multi-metric` и разделение, параметру `partition_field_name` присваивается значение поля, выбранного в пользовательском интерфейсе.

Однако когда выбрано разделение в мастере `Population`, для разделения анализа выбирается `by_field_name`. Если используется мастер `Advanced`, то можно определить `partition_field_name` и/или `by_field_name` (если оба, то это фактически двойное разделение). Поэтому было бы полезно знать, чем эти две настройки, которые эффективно разделяют анализ, отличаются друг от друга.

Если вы хотите реализовать «жесткое» разделение анализа, используйте `partition_field_name`:

- выбранное поле, как правило, должно иметь < 10 000 различных значений на задание, поскольку для разделения требуется больше памяти;
- каждый экземпляр поля подобен независимой переменной.

Оценка аномалий в одном разделе более независима от других разделов. Если вы хотите реализовать «мягкое» разделение, используйте `by_field_name`:

- выбранное поле, как правило, должно иметь < 100 000 различных значений на задание;
- больше подходит для атрибутов объекта (зависимые переменные);
- при подсчете результата учитывается история других полей `by`.

Давайте углубимся в этот последний из перечисленных пунктов, относящийся к «истории» других полей `by`. Что именно это означает?

В общем случае в заданиях анализа по обнаружению аномалий существует концепция, связанная с моментом первого появления объекта; мы назовем этот момент *началом времени* (down of time). Когда что-то происходит в начале времени (то есть когда задание впервые видит данные для `host:X` или `eggo_code:Y`), может возникнуть одна из двух ситуаций:

- этот новый объект рассматривается как «новинка» и сам по себе примечателен и потенциально заслуживает того, чтобы быть отмеченным как аномалия. Для этого вам нужно, чтобы «начало времени» наступило одновременно с запуском задания;
- этот новый объект – всего лишь часть обычного «расширения» данных – возможно, в сеть был добавлен новый сервер или в каталог товаров был добавлен новый `product_id`. В этом случае просто начните моделировать этот новый объект и не беспокойтесь о его появлении. Для этого вам нужно, чтобы «начало времени» наступило, когда этот объект впервые появится.

При разделении с использованием `by_field_name` «начало времени» – это когда было запущено задание ML, а при разделении с использованием `partition_field_name` «начало времени» – это когда упомянутый раздел впервые появился в данных. Таким образом, в ситуации, когда появляется что-то «новое», вы получите разные результаты в зависимости от разделения.

Является ли двойное разделение пределом возможного?

Как уже упоминалось, используя в мастере заданий Advanced как `partition_field_name`, так и `by_field_name`, вы можете получить двойное разделение. Но если вам нужно выполнить больше разделений, вам придется полагаться на другие методы. А именно вам нужно будет создать *скриптовое поле*, которое представляет собой конкатенацию двух (или более) полей. Использование скриптовых полей рассматривается в одном из примеров в приложении.

Теперь, когда вы узнали о концепции разделения анализа, давайте сосредоточимся на различиях между временным и популяционным анализами при обнаружении аномалий.

ОБЗОР ВРЕМЕННОГО И ПОПУЛЯЦИОННОГО АНАЛИЗОВ

Еще в главе 1 вы узнали, что есть два подхода к определению аномальности объекта:

- произошло ли кардинальное изменение поведения объекта по сравнению с его собственным поведением в пределах известного временного ряда;
- наблюдается ли кардинальное отличие объекта от других членов однородной популяции.

По умолчанию используется первый режим (который мы будем просто называть *временным анализом*), если только в конфигурации детектора явным образом не указана опция `over_field_name`.

Популяционный анализ может быть очень полезен при обнаружении аномалий во множестве важных прикладных сценариев. Например, мы хотим найти машины, которые регистрируют больше (или меньше) событий, чем машины с аналогичной конфигурацией, в силу следующих причин:

- неправильные изменения конфигурации, которые привели к внезапному появлению большего количества ошибок в файле журнала системы или приложения;
- системе, которая может быть скомпрометирована вредоносным ПО, дано указание подавлять ведение журнала в определенных ситуациях, что резко уменьшает объем журнала;
- система потеряла соединение или вышла из строя, что привело к уменьшению объема журнала;
- в остальном безобидное изменение настройки уровня ведения журнала (отладка вместо обычного журналирования), теперь заставляющее ваши журналы занимать раздражающе много места на диске.

Другой способ, которым часто используется популяционный анализ, – это *анализ поведения пользователей/сущностей* (user/entity behavioral analysis, EUBA), когда сравнение действий объекта или человека в сопоставлении с их аналогами может выявить следующее:

- **автоматизированные пользователи:** вместо типичного человеческого поведения или модели использования автоматизированный сценарий может демонстрировать поведенческие модели, которые выглядят совершенно иначе с точки зрения скорости, продолжительности и разнообразия создаваемых ими событий. Автоматическая идентификация автоматизированных пользователей может оказаться полезной, будь то поиск веб-сканера, пытающегося собрать информацию о товарах и ценах из онлайн-каталога, или обнаружение бота, который может быть вовлечен в распространение дезинформации в социальных сетях;

- **пользователи-разведчики:** будь то настоящий человек, определяющий границы того, что ему сойдет с рук, или интеллектуальная вредоносная программа, ведущая разведку сетевой структуры, такой пользователь-разведчик может выполнять самые разные действия, надеясь найти нужную информацию или уязвимость (например, путем сканирования портов). Часто использование функции `independent_count` помогает найти разведчика;
- **злоумышленники и агрессивные пользователи:** после фазы разведки злоумышленник или вредоносное ПО переходит к активным действиям, таким как отказ в обслуживании, перебор паролей или кража ценной информации. Опять же, злонамеренные и агрессивные пользователи резко выделяются на общем фоне своим поведением в отношении объема, разнообразия и интенсивности активности в единицу времени.

Практическим примером может быть поиск клиента, который тратит намного больше денег, чем его соседи по популяции. Независимо от того, делаете ли вы это в контексте упреждающего расследования потенциального мошенничества или заинтересованы в расширении услуг для своих самых состоятельных клиентов, вам все равно необходимо найти эти выбросы. Если бы мы использовали хранилище `kibana_sample_data_ecommerce`, которое добавили ранее в примере задания в этой главе, то могли бы создать задание по анализу популяции, выбрав мастер `Population`, а затем выбрав поле `customer_full_name.keyword` для **Population field** (чтобы сравнивать каждого пользователя со всеми остальными). Для детектора мы выберем поле `taxful_total_price`, которое представляет собой общий доход от каждого заказа, размещенного физическим лицом (рис. 3.24).

После выполнения этого задания вы должны увидеть результаты, показанные на рис. 3.25.

Здесь, на рис. 3.25, мы видим, что в списке самых необычных пользователей (в данном случае – крупнейших покупателей за период времени) доминирует пользователь по имени Вагди Шоу (`Wagdi Show`), который, очевидно, разместил заказ на общую сумму 2250 долл. Проницательные читатели узнают эту аномалию из более раннего примера – за исключением того, что на этот раз наш анализ нацелен на пользователя, а не на категорию товара.

Как видите, популяционный анализ может быть очень мощным и широко используется в тех случаях, когда нужно выявить отдельные объекты. Поэтому он очень полезен в случаях использования аналитики в целях безопасности. Теперь давайте обратимся к одной дополнительной, но мощной возможности обнаружения аномалий `Elastic ML` – способности эффективно анализировать неструктурированные сообщения журнала с помощью процесса, называемого категоризацией.

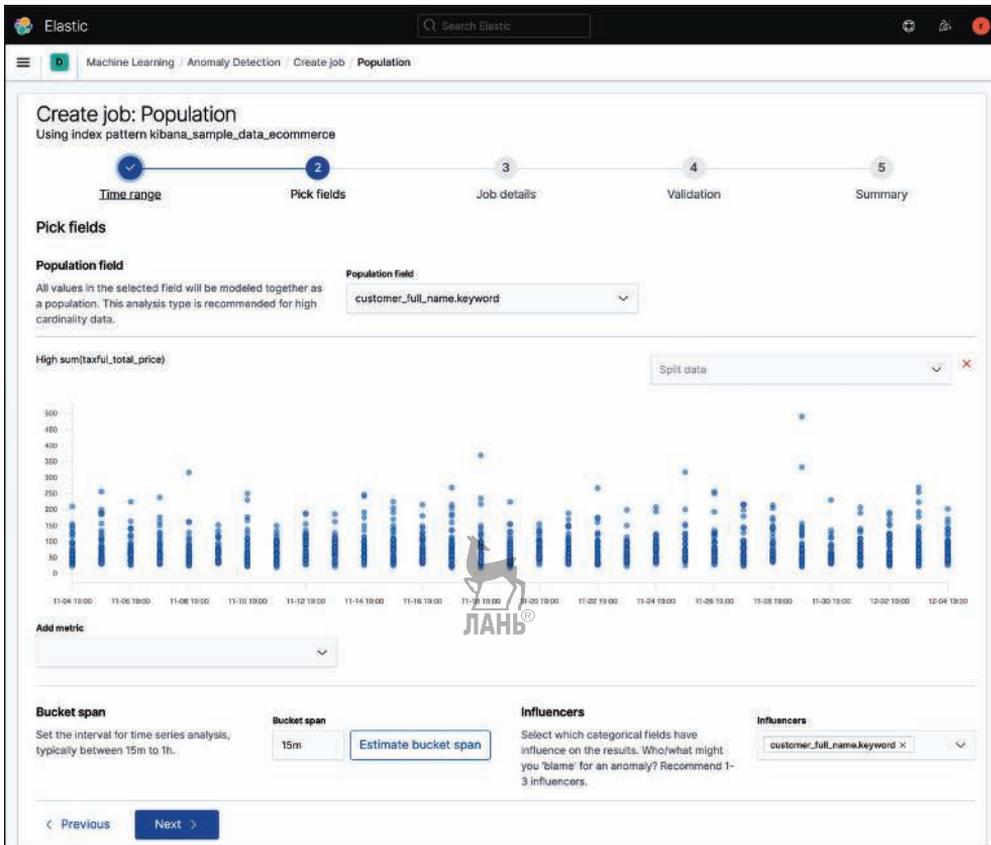


Рис. 3.24 ❖ Популяционный анализ доходов по пользователям

КАТЕГОРИЗАЦИЯ В АНАЛИЗЕ НЕСТРУКТУРИРОВАННЫХ СООБЩЕНИЙ

Представьте, что вы устраняете проблему, просматривая определенный файл журнала. Вы видите в журнале строку, которая сообщает, что не обновлена главная таблица базы данных:

```
18/05/2020 15:16:00 DB Not Updated [Master] Table
```

Если у вас нет глубоких знаний о внутренней работе приложения, создавшего этот журнал, вы можете не знать, важно ли сообщение. Тот факт, что не обновляется главная таблица базы данных, на первый взгляд выглядит как тревожная ситуация. Однако если вы знаете, что приложение регулярно записывает это сообщение, изо дня в день, несколько сотен раз в час, то наверняка решите, что наличие такого сообщения следует игнорировать, потому что приложение работает нормально день за днем, несмотря на запись тревожного сообщения в файл журнала.

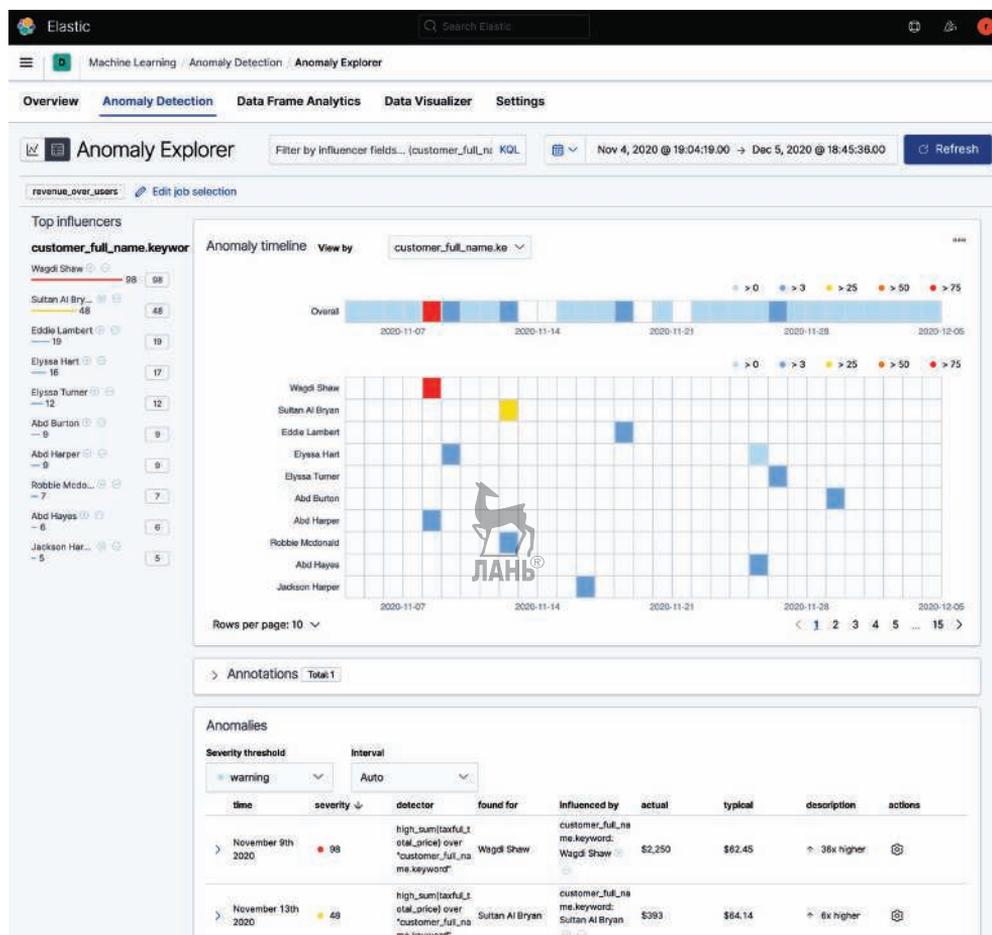


Рис. 3.25 ❖ Результаты анализа популяции крупнейших покупателей

Проблема, очевидно, заключается в человеческом толковании. Проверка записей журнала и прочтение негативной фразы относительно важного компонента системы потенциально склоняют человека к мысли, что сообщение заслуживает внимания из-за возможной проблемы. Однако частота сообщения (это происходит регулярно) информирует человека о том, что сообщение не настолько важное, потому что приложение работает (то есть нет сообщений о сбоях), несмотря на то что эти сообщения записываются в журнал.

Человеку может быть сложно обработать данную информацию (оценить содержание и актуальность сообщения, а также частоту в рамках временного ряда) даже для нескольких типов сообщений в файле журнала. А теперь представьте, что существуют тысячи уникальных типов сообщений, а система генерирует миллионы строк журнала в день. Даже самый опытный знаток приложения и эксперт в области поиска/визуализации сочтет экспертный анализ непрактичным, если не невозможным.

На помощь приходит Elastic ML с возможностями, которые позволяют эмпирически оценить как уникальность содержания сообщений, так и относительную частоту появления.

Типы сообщений, подходящие для категоризации

Нам следует быть достаточно строгими и избирательными при выборе типов строк журнала на основе сообщений, которые подходят для такого рода анализа. Что мы *не* рассматриваем, так это строки журнала/события/документы, которые являются полностью произвольными и, вероятно, выступают результатом человеческого творчества (электронные письма, твиты, комментарии и т. д.). Подобные сообщения слишком произвольны и разнообразны по своему построению и содержанию.

Вместо этого мы сосредоточимся на сгенерированных компьютером сообщениях, которые, очевидно, возникают, когда приложение сталкивается с различными ситуациями или исключениями, тем самым ограничивая их структуру и содержимое относительно дискретным набором возможностей (понимая, что все равно могут присутствовать некоторые переменные аспекты сообщения). Например, давайте посмотрим на следующие несколько строк журнала приложения:

```
18/05/2016 15:16:00 S ACME6 DB Not Updated [Master] Table
18/05/2016 15:16:00 S ACME6 REC Not INSERTED [DB TRAN] Table
18/05/2016 15:16:07 S ACME6 Using: 10.16.1.63!svc_
prod#uid=demo;pwd=demo
18/05/2016 15:16:07 S ACME6 Opening Database = DRIVER={SQL
Server};SERVER=10.16.1.63;network=dbmssocn;
address=10.16.1.63,1433;DATABASE=svc_
prod;uid=demo;pwd=demo;AnsiNPW=No
18/05/2016 15:16:29 S ACME6 DBMS ERROR : db=10.16.1.63!svc_
prod#uid=demo;pwd=demo Err=-11 [Microsoft][ODBC SQL Server
Driver][TCP/IP Sockets]General network error. Check your
network documentation.
```

Здесь мы видим, что существует множество сообщений с разным текстом, но все равно просматривается некоторая общая структура. После отметки даты/времени и имени сервера, с которого было отправлено сообщение (здесь ACME6), идет содержание сообщения, где приложение информирует внешний мир о происходящем в данный момент – все работает как надо или возникают ошибки.

Предварительная категоризация

Чтобы навести порядок в беспорядочном потоке сообщений в файле журнала, Elastic ML будет использовать метод группировки похожих сообщений вместе с использованием алгоритма кластеризации схожих строк. Упрощенная логика действий этого алгоритма такова:

- неизменяемым словарным (английским) словам уделяется больше внимания, чем изменяемым (то есть `network` и `address` являются словарными словами, но `dbmssocn`, скорее всего, является изменяемой/переменной строкой);
- неизменяемые словарные слова пропускают через алгоритм подбора строк (аналогичный *расстоянию Левенштейна*¹), чтобы определить, насколько похожа текущая строка журнала на предыдущие строки;
- если разница между текущей строкой журнала и существующей категорией небольшая, ее добавляют в эту категорию. В противном случае создают новую категорию для текущей строки журнала.

В качестве простого примера рассмотрим эти три сообщения:

```
Error writing file "foo" on host "acme6"
Error writing file "bar" on host "acme5"
Opening database on host "acme7"
```

Алгоритм объединяет первые два сообщения в одну категорию, так как они относятся к ошибке записи файла (`Error writing file`), тогда как третьему сообщению будет присвоена собственная (новая) категория.

Система имен этих категорий очень проста: ML будет называть их просто `mlcategory N`, где `N` – увеличивающееся целое число. Следовательно, в этом примере первые две строки будут связаны с `mlcategory 1`, а третья строка будет связана с `mlcategory 2`. В реалистичном машинном журнале могут быть тысячи (или даже десятки тысяч) категорий, которые возникают из-за разнообразия сообщений журнала, но все равно набор возможных категорий должен быть конечным. Однако если количество категорий начнет исчисляться сотнями тысяч, его трудно считать ограниченным набором возможных типов сообщений, и такой набор данных уже не будет подходящим кандидатом для анализа путем категоризации.

Анализ категорий

Теперь, когда сообщения классифицированы с помощью алгоритма, описанного ранее, следующий шаг – провести анализ (с использованием функций `count` или `gate`). В этом случае мы не будем считать строки журнала (и, следовательно, документы хранилища `Elasticsearch`) сами по себе; вместо этого мы собираемся подсчитать частоту появления различных категорий, которые являются выходными данными алгоритма. Строки журнала из примера выше, если они возникли в пределах одного сегмента, дадут следующий результат алгоритма категоризации:

```
mlcategory 1: 2
mlcategory 2: 1
```

¹ Расстояние Левенштейна (также *редакционное расстояние*, или *дистанция редактирования*) между двумя строками в теории информации и компьютерной лингвистике – это минимальное количество операций с одним символом (вставки, удаления или замены), необходимых для превращения одной строки в другую. – *Прим. перев.*

Другими словами, было два появления ошибки типа `Error writing file` и одно появление ошибки типа `Opening database on host` в последнем интервале сегмента. Именно эта информация будет в конечном итоге обработана моделью машинного обучения, чтобы определить, является ли она необычной.

Пример задания по категоризации

Благодаря мастеру заданий категоризации в пользовательском интерфейсе процесс настройки этого типа заданий чрезвычайно прост. Давайте сначала предположим, что у нас есть загруженный неструктурированный файл журнала (например, файл `secure.log` в папке `example_data` на GitHub):

❗ Дополнительные сведения о том, как принимать данные с помощью визуализатора файлов, представлены в подробной статье по адресу <https://www.elastic.co/blog/importing-csv-and-log-data-into-elasticsearch-with-file-data-visualizer>.

1. Выбрав интересующее вас хранилище и мастер `Categorization`, а затем указав соответствующий временной диапазон для анализа, вы перейдете к выбору детектора категоризации (детектор **Count** или детектор **Rare**), а также нужного поля категоризации (**Categorization field**). В данном случае наши данные содержат только два поля (`@timestamp` и `message`). Следовательно, мы будем классифицировать в Elastic ML поле `message`. В этом примере мы также выберем детектор **Count** (рис. 3.26).

Обратите внимание, что вы можете просмотреть поле выбранной категории, чтобы убедиться, что оно дает разумные результаты. Также заметим, что в разделе **Examples** (Примеры) представлено визуальное подтверждение того, что Elastic ML сфокусирован на неизменяемом тексте сообщений журнала.

2. После подтверждения конфигурации и запуска задания вы увидите в окне мастера предварительный просмотр результатов по мере их обнаружения и анализа (рис. 3.27).



Elastic Search Elastic

Machine Learning / Anomaly Detection / Create job / Categorization

Create job: Categorization

Using index pattern secure_log

1 Time range 2 Pick fields 3 Job details 4 Validation 5 Summary

Pick fields

Categorization detector

Count

Look for anomalies in the event rate of a particular category.

✓ Selected

Rare

Look for categories that occur rarely in time.

Select

Categorization field

Specifies which field will be categorized. Using text data types is recommended. Categorization works best on machine written log messages, typically logging written by a developer for the purpose of system troubleshooting.

Categorization field:

Selected category field is valid

1000 field values analyzed, 100% contain 3 or more tokens.

Analyzer used: ml_classic

[Edit categorization analyzer](#)

> View all checks performed

Examples

```
Oct 22 21:46:05 localhost sudo: ec2-user : TTY=pts/0 : PWD=/opt/elastic/ml/elasticsearch-7.0.0-alpha1-SNAPSHOT/config : USER=root : COMMAND=/etc/init.d/elasticsearch start
Oct 22 22:27:27 localhost ssh[9571]: Received disconnect from 178.33.169.154 port 57556:11: Bye Bye [preauth]
Oct 23 00:30:09 localhost ssh[9874]: input.userauth.request: invalid user admin [preauth]
Oct 23 02:45:56 localhost ssh[10203]: Received disconnect from 114.207.154.2 port 46893:11: Bye Bye [preauth]
Oct 23 08:51:26 localhost ssh[10964]: Disconnected from 61.188.189.7 port 46115 [preauth]
```

Enable per-partition categorization

If per-partition categorization is enabled then categories are determined independently for each value of the partition field.

Enable per-partition categorization Enable per-partition categorization

Bucket span

Set the interval for time series analysis, typically between 15m to 1h.

Bucket span:

[< Previous](#) [Next >](#)

Рис. 3.26 ❖ Настройка задания категоризации

Create job: Categorization
Using index pattern secure_log

1 Time range 2 Pick fields 3 Job details 4 Validation 5 Summary

New job from index pattern secure_log

Total categories: 23

Example

```
Oct 22 21:35:06 localhost sudo: ec2-user : TTY=pts/0 ; PWD=/opt/elastic/ml ; USER=root ; COMMAND=/etc/init.d/elasticsearch start
Oct 22 21:35:06 localhost runuser: pam_unix(runuser-1:session): session opened for user ec2-user by ec2-user(uid=0)
Oct 23 02:46:09 localhost sshd[10215]: input_userauth_request: invalid user localhost [preauth]
Oct 23 09:08:53 localhost sshd[11206]: Bad protocol version identification 'GET /login.html HTTP/1.1' from 107.170.209.239 port 54052
Oct 23 12:10:37 localhost sshd[11572]: Unable to negotiate with 52.10.104.241 port 55780: no matching host key type found. Their offer: ecdsa-sha2-nistp384 [preauth]
```

Job ID secure_log	Bucket span 15m	Enable model plot False	Start Oct 22, 2020 @ 15:02:19.000
Job description No description provided	Influencers micategory	Use dedicated index False	End Oct 25, 2020 @ 16:43:40.000
Groups No groups selected		Model memory limit 26MB	

Start immediately
If unselected, job can be started later from the jobs list.

[View results](#) [Reset job](#) [Start job running in real time](#) [Create watch](#)

Рис. 3.27 ❖ Предварительный просмотр результатов задания категоризации

В этом простом примере в данных было обнаружено 23 категории. Когда результаты просматриваются в Anomaly Explorer, мы видим, что главная аномалия здесь – это категория номер 7.

3. Если нажать мышкой на столбец примеров категорий, представление расширяется, демонстрируя примеры строк журнала, которые попали в эту категорию (рис. 3.28).

The screenshot displays the 'Anomalies' section of a search interface. At the top, there are filters for 'Severity threshold' (warning) and 'Interval' (Auto). Below this is a table of anomalies. The first row is selected, showing a severity of 72 and a description 'More than 100x higher'. A dropdown menu is open over the 'category examples' column, showing options for 'View examples' and 'Configure rules'. Below the table, there is a 'Details' section with 'Terms', 'Regex', and 'Examples'.

time	severity	detector	found for	influenced by	actual	typical	description	category examples	actions
October 24th 2020	72	count by micategory	micategory 7	micategory: 7	7	0.0115	More than 100x higher	Oct 22 20:58:03 loc... Oct 23 13:35:14 loc... Oct 24 20:02:45 loc... Oct 24 20:02:54	View examples Configure rules

Details [Category examples](#)

Terms
localhost sshd Received disconnect from port disconnected by user

Regex
.*localhost.*?sshd.*?Received.*?disconnect.*?from.*?port.*?disconnected.*?by.*?user.*

Examples

Oct 22 20:58:03 localhost sshd[2074]: Received disconnect from 72.93.85.203 port 53552:11: disconnected by user

Oct 23 13:35:14 localhost sshd[8946]: Received disconnect from 72.93.85.203 port 59933:11: disconnected by user

Oct 24 20:02:45 localhost sshd[15462]: Received disconnect from 188.255.78.23 port 60164:11: disconnected by user [preauth]

Oct 24 20:02:54 localhost sshd[15467]: Received disconnect from 188.255.78.23 port 60170:11: disconnected by user [preauth]

Рис. 3.28 ❖ Результаты задания по категоризации

Здесь мы видим, что количество полученных сообщений Received disconnect резко возросло.

4. Нажав на значок шестеренки, как показано на рис. 3.28 справа вверху, мы можем выбрать **View examples** (Просмотр примеров), чтобы перейти в пользовательский интерфейс Kibana Discover, но с отфильтрованным соответствующим сообщением и растянутым временным интервалом (рис. 3.29).

Панель запроса Discover была автоматически заполнена соответствующим запросом KQL, чтобы ограничить наше представление типом сообщений, который признан аномальным.

5. Если мы удалим этот фильтр запроса, то получим все сообщения в файле журнала во время этой аномалии и увидим более обширную картину происходящего, которая заключается в том, что кто-то предпринимал много попыток аутентификации (рис. 3.30).

Судя по всему, наблюдается шквал попыток аутентификации с использованием хорошо известных имен пользователей (user, test и т. д.). Похоже, мы обнаружили попытку взлома системы методом грубой силы (брутфорс, brute-force), просто используя категоризацию!

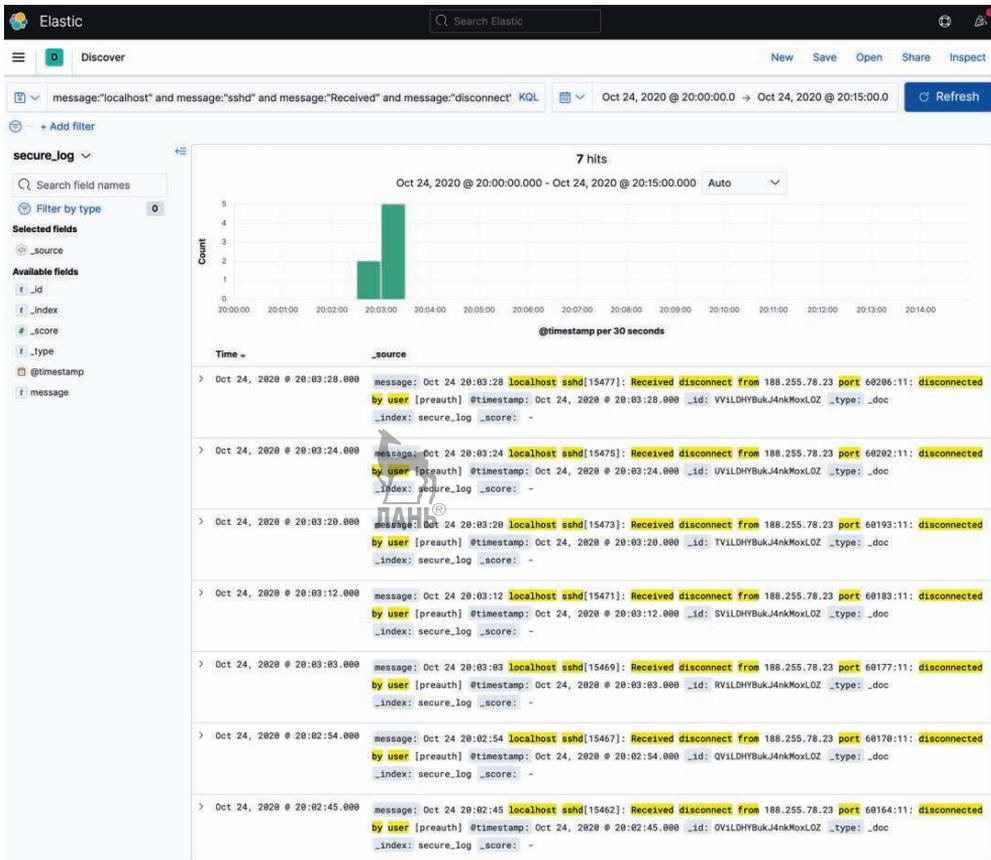


Рис. 3.29 ❖ Проверка необработанных строк журнала из результатов задания категоризации

Когда не следует использовать категоризацию

Несмотря на то что категоризация весьма полезна, ей присущи свои ограничения. В частности, можно назвать такие варианты данных, когда попытка использовать категоризацию, скорее всего, даст плохие результаты:

- поля текста произвольной формы, вероятно, созданные людьми, – твиты, комментарии, электронные письма и заметки;
- строки журнала, которые действительно содержат правильные пары имя/значение, такие как журнал доступа к сети;
- документы, содержащие много многострочного текста, такого как трассировка стека, XML и т. д.

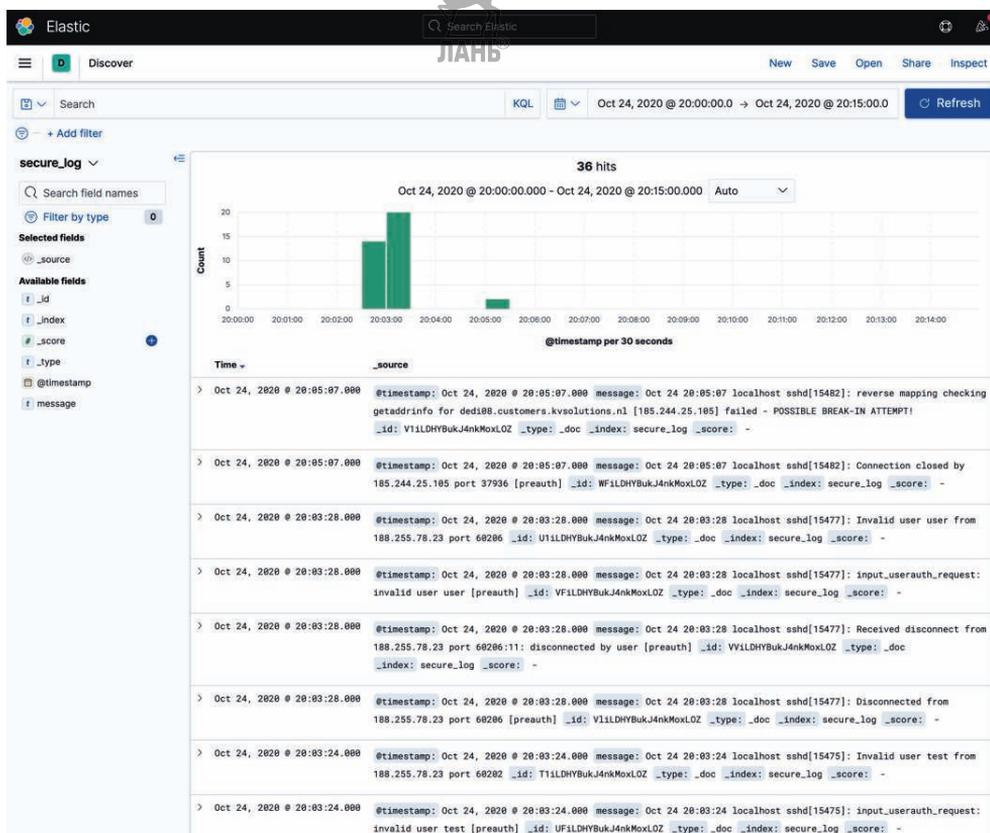


Рис. 3.30 ❖ Просмотр всех линий журнала в период аномалии

С учетом вышесказанного можно сделать вывод, что категоризация бывает чрезвычайно полезной в тех случаях, когда анализ неструктурированного текста в противном случае потребовал бы значительных усилий от человека-аналитика.

УПРАВЛЕНИЕ ELASTIC ML ЧЕРЕЗ API

Как и практически все компоненты Elastic Stack, модуль ML также можно полностью автоматизировать с помощью вызовов API, включая настройку заданий, выполнение и сбор результатов. Фактически все ваши взаимодействия в пользовательском интерфейсе Kibana за кулисами используют API Elastic ML. Вы можете, например, полностью разработать свой собственный пользовательский интерфейс, если вам нужны определенные рабочие процессы или визуализации.

! Более подробную информацию об API обнаружения аномалий вы найдете на сайте <https://www.elastic.co/guide/en/machine-learning/current/ml-api-quickref.html>. Часть аналитики фреймов данных в Elastic ML имеет совершенно отдельный API, который будет обсуждаться в главах с 9 по 13.

Мы не будем вдаваться в подробности каждого вызова API, но хотели бы выделить некоторые компоненты, на которые стоит обратить внимание.

Первым очевидным API, который следует упомянуть, является интерфейс создания заданий, который позволяет создавать конфигурацию заданий Elastic ML. Например, если вы хотите создать заново задание анализа популяции, показанное на рис. 3.24, следующий вызов создаст такое задание с названием `revenue_over_users_api`:



```
PUT _ml/anomaly_detectors/revenue_over_users_api
{
  "job_id": "revenue_over_users_api",
  "analysis_config": {
    "bucket_span": "15m",
    "detectors": [
      {
        "detector_description": "high_sum(taxful_total_price)
over customer_full_name.keyword",
        "function": "high_sum",
        "field_name": "taxful_total_price",
        "over_field_name": "customer_full_name.keyword"
      }
    ],
    "influencers": [
      "customer_full_name.keyword"
    ]
  },
  "data_description": {
    "time_field": "order_date",
    "time_format": "epoch_ms"
  }
}
```

Обратите внимание, что поле `job_id` должно быть уникальным при создании задания.

Чтобы создать конфигурацию сопутствующего потока данных для этого задания, мы должны выполнить следующий отдельный вызов API:



```
PUT _ml/datafeeds/datafeed-revenue_over_users_api
{
  "datafeed_id": "datafeed-revenue_over_users_api",
  "job_id": "revenue_over_users_api",
  "query_delay": "60s",
  "indices": [
    "kibana_sample_data_ecommerce"
  ],
  "query": {
```

```

    "bool": {
      "must": [
        {
          "match_all": {}
        }
      ]
    },
    "scroll_size": 1000,
    "chunking_config": {
      "mode": "auto"
    }
  }
}

```

Обратите внимание, что запрос по умолчанию к хранилищу – `match_all`, что означает отсутствие фильтрации. Конечно, мы могли бы вставить любой допустимый Elasticsearch DSL в блок запроса для выполнения настраиваемых фильтров или агрегаций. Этот подход мы рассмотрим позже.

Существуют и другие API, которые можно использовать для извлечения результатов или изменения других аспектов выполнения задания машинного обучения. Обратитесь к онлайн-документации для получения дополнительной информации.

ЗАКЛЮЧЕНИЕ

Вы видели, что Elastic ML может выделять различия в объеме, разнообразии и уникальности метрик и сообщений журнала, в том числе тех, которые сначала нуждаются в некоторой категоризации. Кроме того, мы показали, что популяционный анализ может быть чрезвычайно интересной альтернативой обнаружению временных аномалий, когда основное внимание уделяется поиску самых необычных сущностей. Эти методы помогают решить проблему ограниченных возможностей человека, который во многих случаях не способен определить, что действительно необычно и заслуживает внимания и исследования.

Навыки, приобретенные в данной главе, пригодятся в последующих главах, где вы увидите, как ML помогает в поиске первопричин сложных IT-проблем, выявлении снижения производительности приложений или обнаружении деятельности вредоносных программ и злонамеренной внешней активности.

В следующей главе вы увидите, как модели временных рядов, построенные с помощью заданий по обнаружению аномалий, могут быть использованы для прогнозирования тенденций ваших данных в будущем.

Глава 4

.....

Прогнозирование



Прогнозирование – это естественное продолжение моделирования временных рядов в Elastic ML. Поскольку показательные модели описывают историческое поведение данных и строятся на их основе, можно спроецировать эту информацию вперед во времени и предсказать, как что-то должно вести себя в будущем.

На этом этапе мы потратим некоторое время на изучение концепций прогнозирования, а также на рассмотрение некоторых практических примеров.

В частности, в этой главе будут рассмотрены следующие темы:

- ключевое различие между предсказаниями и пророчествами;
- для чего применяется прогнозирование;
- как работает прогнозирование;
- прогнозирование одиночного временного ряда;
- просмотр результатов прогнозирования;
- прогнозирование нескольких временных рядов.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Информация и примеры, представленные в этой главе, актуальны для версии 7.11 Elastic Stack.

КЛЮЧЕВОЕ РАЗЛИЧИЕ МЕЖДУ ПРЕДСКАЗАНИЯМИ И ПРОРОЧЕСТВАМИ

Прошлые показатели не свидетельствуют о будущих результатах. Эту форму отказа от ответственности используют финансовые компании, когда они ссылаются на эффективность таких продуктов, как паевые инвестиционные фонды. Но данный отказ от ответственности представляет собой немного странное противоречие, потому что прошлое – это все, с чем мы можем и должны работать. Если компании, входящие в паевой инвестиционный фонд, имели стабильно положительные квартальные результаты в течение последних восьми кварталов подряд, гарантирует ли это, что они так же будут



иметь положительные результаты в следующие восемь кварталов и что их капитализация будет продолжать расти? Вероятность свидетельствует в пользу этого факта, но это еще не все. И прежде чем впасть в опасное заблуждение, будто способность Elastic ML к прогнозированию поможет нам заработать состояние на фондовом рынке, мы должны четко осознать ключевое ограничение – всегда есть неконтролируемые факторы.

Причина, по которой финансовые компании используют вышеупомянутый отказ от ответственности, заключается в том, что часто возникают неизвестные, неконтролируемые факторы, которые могут значительно повлиять на траекторию любого процесса. Например, правительство может внести в нормативные акты или торговую политику изменения, которые помогут или помешают компании работать и приносить прибыль, или же внезапно может вскрыться мошенничество в бухгалтерском учете и выяснится, что руководители сговорились фальсифицировать корпоративную отчетность, которая не отражает реальное положение дел, и в конечном итоге компания обанкротится.

Эти факторы считаются *неизвестными и внешними* по следующим причинам:

- они находятся вне контроля самой организации (как в примере, когда правительство определяет экономическую политику независимо от деятельности компании);
- они не входят в доступную информацию о системе (внешний инвестор в режиме реального времени имеет доступ только к общедоступным отчетам, а не к сведениям о мошеннических действиях по фабрикации этих отчетов).

Следовательно, ваши экономические прогнозы хороши ровно настолько, насколько хороши имеющаяся у вас информация и ваша способность устранить или смягчить влияние внешних неизвестных факторов, которые повлияют на ваш прогноз. Это справедливо и в мире информационных технологий. Не всегда можно предсказать тенденцию или сбой, если существенное влияние оказывает неизвестный внешний фактор (некорректное изменение конфигурации, отказ оборудования и т. д.). Однако мы можем использовать вероятностный анализ, чтобы составить наилучшее возможное представление о будущем, максимально абстрагируясь от непредсказуемых внешних факторов. Наличие этого представления позволяет нам реализовать некоторые сценарии использования прогнозирования, не зацикливаясь на пророчествах¹. Давайте теперь поговорим о том, как мы можем использовать прогнозирование на практике.

¹ Здесь авторы оставили между строк мысль о том, что формализованное использование доступной информации временных рядов – это *прогнозирование*, а разговоры о потенциально неизвестных и непредсказуемых факторах – это *пророчество*. Мошенничество руководителей инвестиционного фонда невозможно спрогнозировать, но можно напрогнозировать, а потом сидеть в ожидании, пока пророчество сбудется (я же говорил!) или не сбудется. Но доходность инвестиционного фонда можно спрогнозировать на основании исторической информации о динамике доходности и стоимости акций предприятий, в которые фонд инвестирует средства. – *Прим. перев.*

Для чего применяется прогнозирование?

В контексте Elastic ML на самом деле есть только два в некотором роде похожих сценария, в которых можно использовать прогнозирование:

- **приоритет значения:** вы экстраполируете временной ряд в будущее, чтобы спрогнозировать вероятную будущую ценность. Это, например, ответы на вопросы типа: «Сколько виджетов я буду продавать в день через 2 месяца?»;
- **приоритет времени:** в этом сценарии вы оцениваете вероятное время, в течение которого будет достигнуто ожидаемое значение. Это ответы на вопросы типа: «Достигнет ли нагрузка на сервер значения 80 % в течение следующей недели?»

Различия между этими двумя вариантами использования могут заключаться не только в том, как поставлен вопрос (как выполняется поиск данных), но и в том, как вы интерпретируете вывод. Однако, прежде чем мы углубимся в несколько примеров использования функции прогнозирования, давайте уделим немного времени обсуждению того, как она работает с логической точки зрения.

Как работает прогнозирование?

Первое, что нужно понять, – получение прогноза на основе имеющихся данных является расширением известного задания по обнаружению аномалий. Другими словами, вам необходимо настроить задание обнаружения аномалий, и это задание должно проанализировать исторические данные, прежде чем вы сможете делать прогнозы на основе этих данных. Это связано с тем, что в процессе прогнозирования используются модели, созданные заданием «Обнаружение аномалий». Чтобы спрогнозировать данные, вам необходимо выполнить те же шаги, которые использовались для создания задания на обнаружение аномалий, как описано в предыдущих главах. Если при выполнении этого задания обнаружены аномалии, вы можете не обращать на них внимания, если ваша единственная цель – выполнить прогноз. После того как задание изучило некоторые исторические данные, модель или модели (если задание настроено для анализа данных из более чем одного временного ряда), связанные с этим заданием, становятся текущими и актуальными, как показано на рис. 4.1.

Мы будем рассматривать период времени, предшествующий текущему моменту, как *историческое обучение* (historical learning) – период, в течение которого модели учились на реальных данных. Когда в определенный момент пользователь желает получить прогноз, создается копия модели (моделей), и запускается отдельный процесс, который берет эти модели и экстраполирует их в «будущее». Этот процесс выполняется параллельно, чтобы не мешать исходным моделям и их развитию, как показано на рис. 4.2.

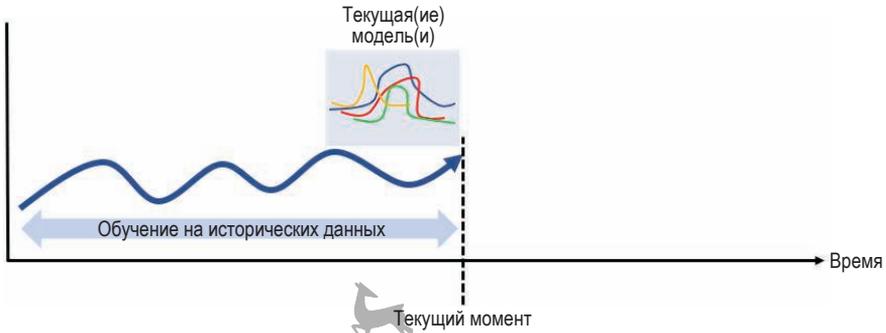


Рис. 4.1 ❖ Условное представление процесса создания моделей на исторических данных

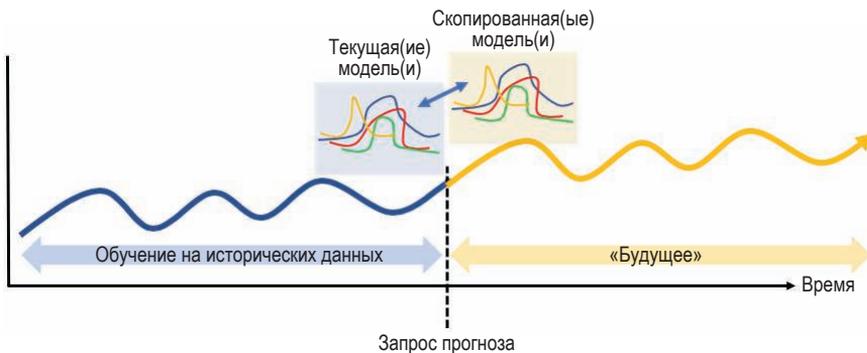


Рис. 4.2 ❖ Символьное представление моделей, скопированных для прогнозирования будущего

Значения прогноза записываются в то же хранилище результатов, что и обнаружение аномалий, но как результат особого типа (подробнее об этом позже), и они доступны для просмотра в пользовательском интерфейсе или через API.

Важно отметить, что нормальное выполнение задания машинного обучения, анализирующего реальные данные, будет продолжаться (если оно выполняется в реальном времени), и, следовательно, по прошествии некоторого времени может возникнуть разница между спрогнозированным значением (сделанным во время выполнения задания прогноза) и фактическим значением на тот момент, когда наступит время прогноза (рис. 4.3).

Наличие ошибки прогноза следовало ожидать, но мы всегда надеемся, что она будет минимальной. Разница между прогнозом и реальностью в выбранный момент в настоящее время не используется в Elastic ML, но, возможно, в будущем она поможет модели формировать более точные последующие прогнозы. Конечно, также возможно, что причиной ошибки прогноза послужит неизвестный внешний фактор (как мы говорили ранее).

Другой (возможно, более простой) способ понять неопределенность прогнозов – это представить предсказание результата подбрасывания монеты.

Вы можете наблюдать последовательность предыдущих подбрасываний монеты, но если вы не принимаете во внимание физику подбрасывания монеты (скорость, высоту, вращение и т. д.) и полагаетесь только на результаты прошлых наблюдений, то вы не добьетесь лучшей точности прогноза, чем 50/50. Кроме того, вполне возможно, что в течение периода обучения модели Elastic ML не обнаружит данных, согласованных с чьим-либо поведением. Поэтому шум в данных также вносит в прогноз некоторое количество вариаций или неопределенности.

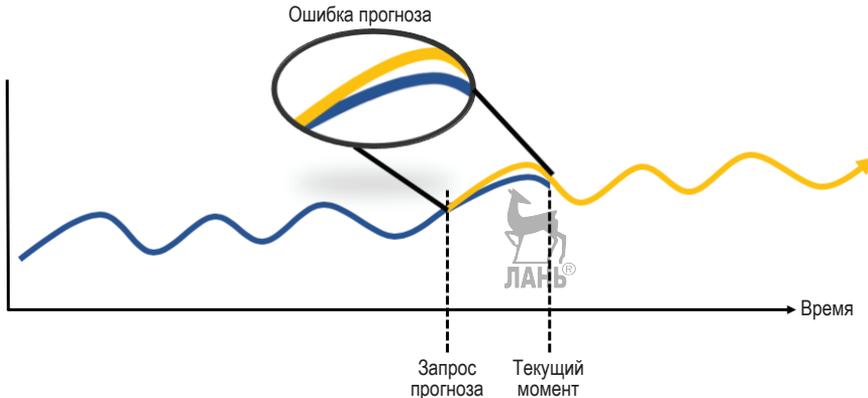


Рис. 4.3 ❖ Ошибка прогноза

Пользователь может делать несколько прогнозов в разное время. Они будут храниться отдельно, как показано на рис. 4.4.

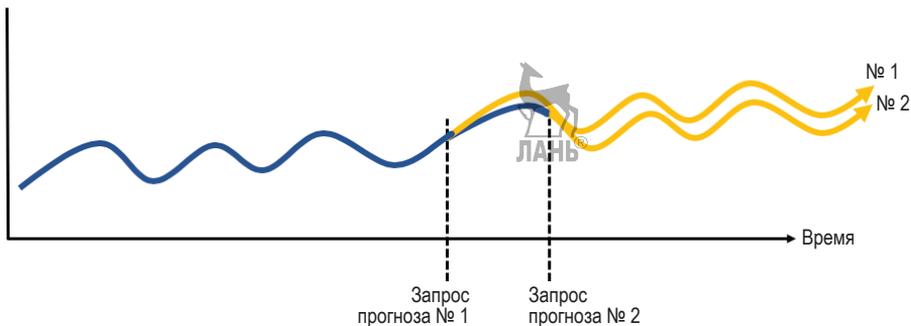


Рис. 4.4 ❖ Прогнозы, сделанные в разное время

Прогнозы №1 и №2 различаются внутренним уникальным идентификатором экземпляра прогноза. Это станет видно позже, когда мы посмотрим, как результаты прогнозов размещаются в хранилище.

Теперь, когда у вас есть базовое представление о процессе прогнозирования, давайте рассмотрим пример использования Elastic ML для прогнозирования одиночного временного ряда.

ПРОГНОЗИРОВАНИЕ ОДИНОЧНОГО ВРЕМЕННОГО РЯДА

Чтобы проиллюстрировать процедуру прогнозирования, мы начнем с набора данных, который представляет собой одиночный временной ряд. Хотя этот набор данных является общим для разных примеров, вы можете представить, что он отражает метрику производительности системы, количество транзакций, обработанных системой, или даже данные о доходах от продаж. Важной особенностью этого набора данных является то, что он содержит несколько различных трендов, зависящих от времени: дневной тренд, недельный тренд и общий возрастающий тренд. Elastic ML обнаружит все эти тренды и будет эффективно предсказывать их в будущем. Стоит отметить, что набор данных также содержит некоторые аномалии, но будущие аномалии невозможно предсказать, поскольку они являются непредсказуемыми событиями по определению. Так как здесь мы говорим исключительно о прогнозировании, при построении моделей для прогнозирования мы будем игнорировать наличие любых аномалий, обнаруженных в нашем наборе данных.

С учетом сказанного, давайте перейдем к примеру, используя набор данных `forecast_example` из репозитория GitHub. После загрузки данные можно легко импортировать в Kibana с помощью визуализатора данных Elastic ML. Последовательно выполните следующие действия.

1. Чтобы загрузить образцы данных, на главном экране Kibana нажмите кнопку **Upload a file** (Загрузить файл), как показано на рис. 4.5.

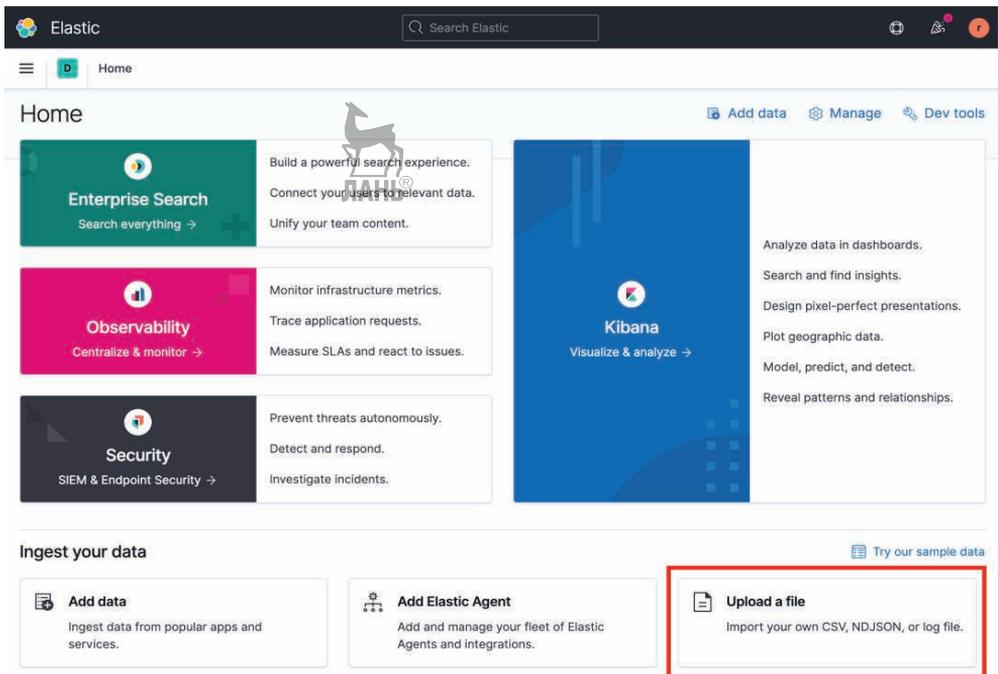


Рис. 4.5 ❖ Загрузка файла в Kibana

- Выберите файл `forecast_example.json` на локальном компьютере. Визуализатор данных отобразит первые 1000 строк файла, чтобы вы могли предварительно увидеть его содержимое, а также разбить различные поля (рис. 4.6).

The screenshot shows the Elastic Data Visualizer interface. At the top, there is a search bar and navigation tabs for Overview, Anomaly Detection, Data Frame Analytics, Data Visualizer (selected), and Settings. The main content area is titled 'forecast_example.json' and is divided into three sections:

- File contents:** Shows the first 1000 lines of the JSON file. The first 12 lines are visible, each representing a document with an 'index' field (containing a unique ID) and an 'amount' field (containing a numerical value).
- Summary:** Provides a high-level overview of the data. It indicates that 1000 lines were analyzed in ndjson format. There are buttons for 'Override settings' and 'Analysis explanation'.
- File stats:** Displays statistical information for two fields: '@timestamp' and 'amount'.
 - @timestamp:** 500 documents (50%), 500 distinct values. A horizontal bar chart shows the top values, with the most frequent being 1485907200000, 1485911700000, 1485914400000, 1485917100000, 1485933300000, and 1485936000000, each representing 0.2% of the documents.
 - amount:** 500 documents (50%), 488 distinct values. A table shows the minimum (1786), median (6758), and maximum (14277) values. A horizontal bar chart shows the top values: 2289 (0.4%), 2871 (0.4%), and 5456 (0.4%).

At the bottom of the interface, there are 'Import' and 'Cancel' buttons.

Рис. 4.6 ❖ Предварительный просмотр содержимого файла для загрузки

3. Нажмите кнопку **Import** (Импорт), затем присвойте имя конечному хранилищу для данных, которые будут загружены, как показано на рис. 4.7.

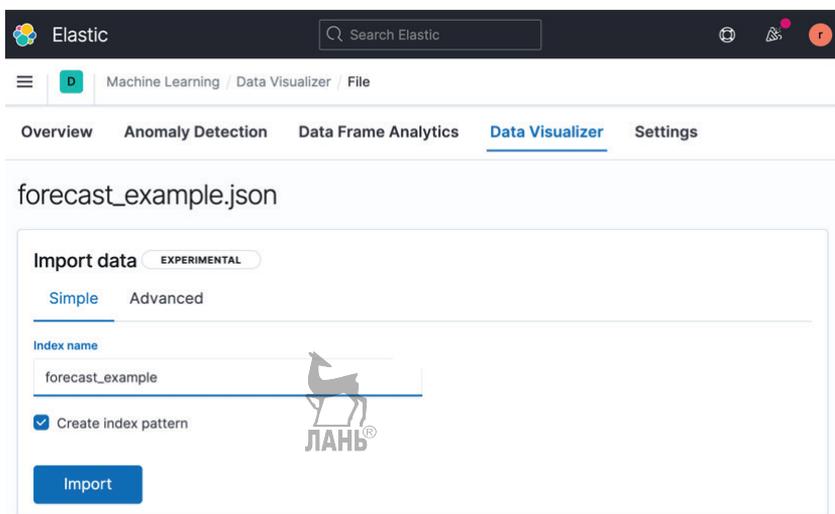


Рис. 4.7 ❖ Присвоение имени целевому хранилищу для загрузки

4. Введите имя для целевого хранилища, убедитесь, что опция **Create index pattern** (Создать шаблон хранилища) отмечена, если вы загружаете эти данные впервые, и снова нажмите кнопку **Import**, чтобы завершить загрузку. Вы должны увидеть успешное завершение загрузки, как показано на рис. 4.8.
5. После загрузки данных перейдите в раздел **Machine Learning** (Машинное обучение) и создайте задание **Anomaly Detection** (Обнаружение аномалий), выбрав шаблон хранилища с названием, которое ввели на предыдущем шаге (рис. 4.9).
6. Этот конкретный набор данных представляет собой только одну метрику временного ряда (поле с названием `amount`), поэтому мы просто воспользуемся мастером **Single metric** (Единственная метрика) для создания задания, как показано на рис. 4.10.
7. На следующем снимке экрана (рис. 4.11) показано, что мы будем анализировать данные только до 1 марта 2017 г. @ 00:00:00.000, чтобы оставить возможность сравнения нашего прогноза с реальными последующими данными. Для этого сначала нажмите кнопку **Use full forecast example data** (Использовать полные данные `forecast_example`), а затем вручную измените дату окончания, чтобы она соответствовала тому, что показано на рис. 4.11.

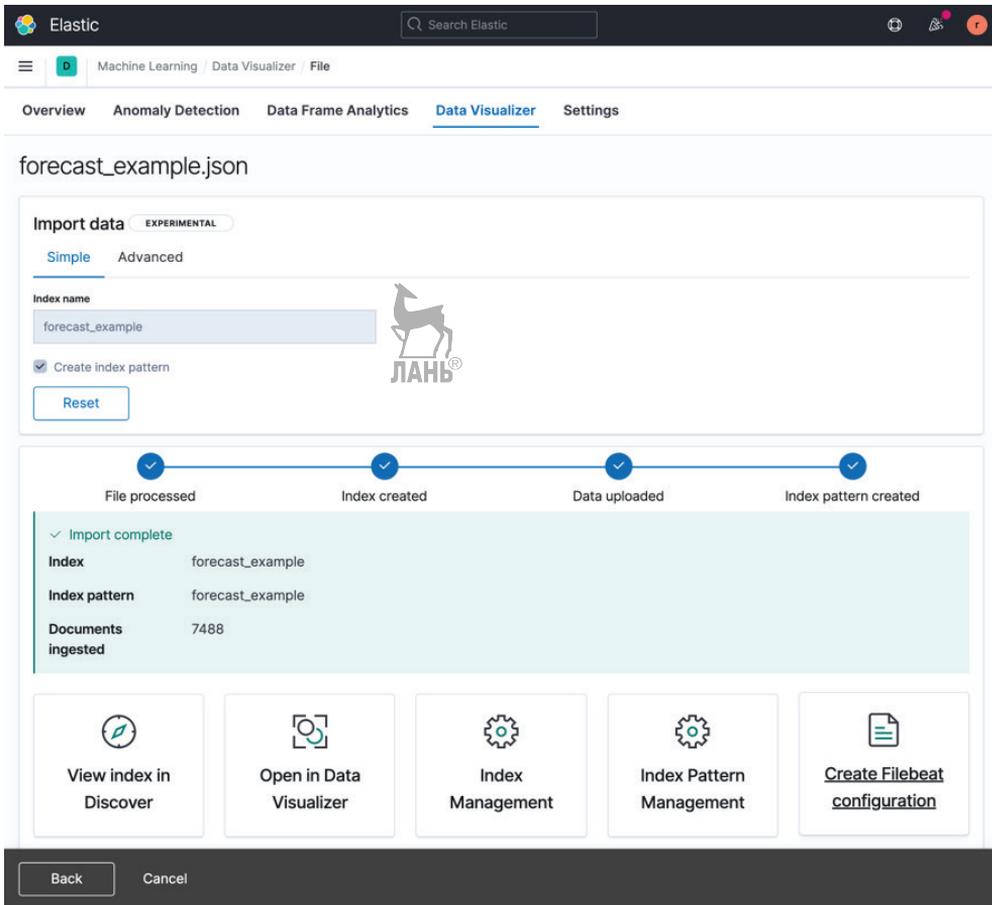


Рис. 4.8 ❖ Загрузка завершена

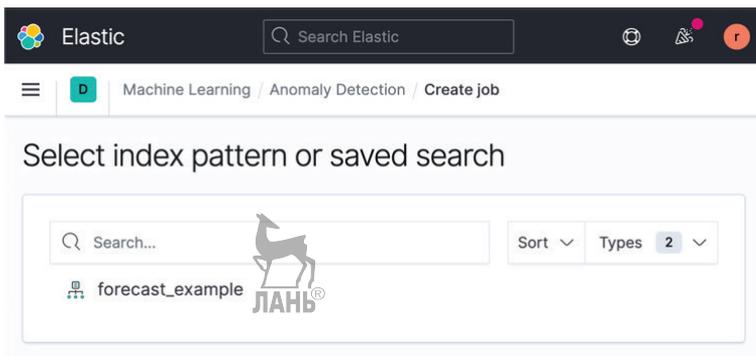


Рис. 4.9 ❖ Сначала нужно создать задание на обнаружение аномалий

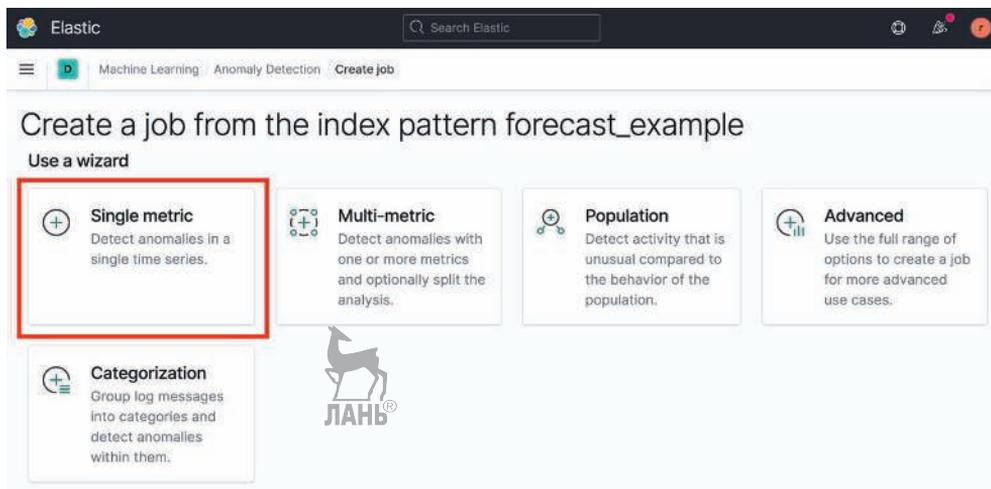


Рис. 4.10 ❖ Выбор мастера **Single metric** для текущих данных

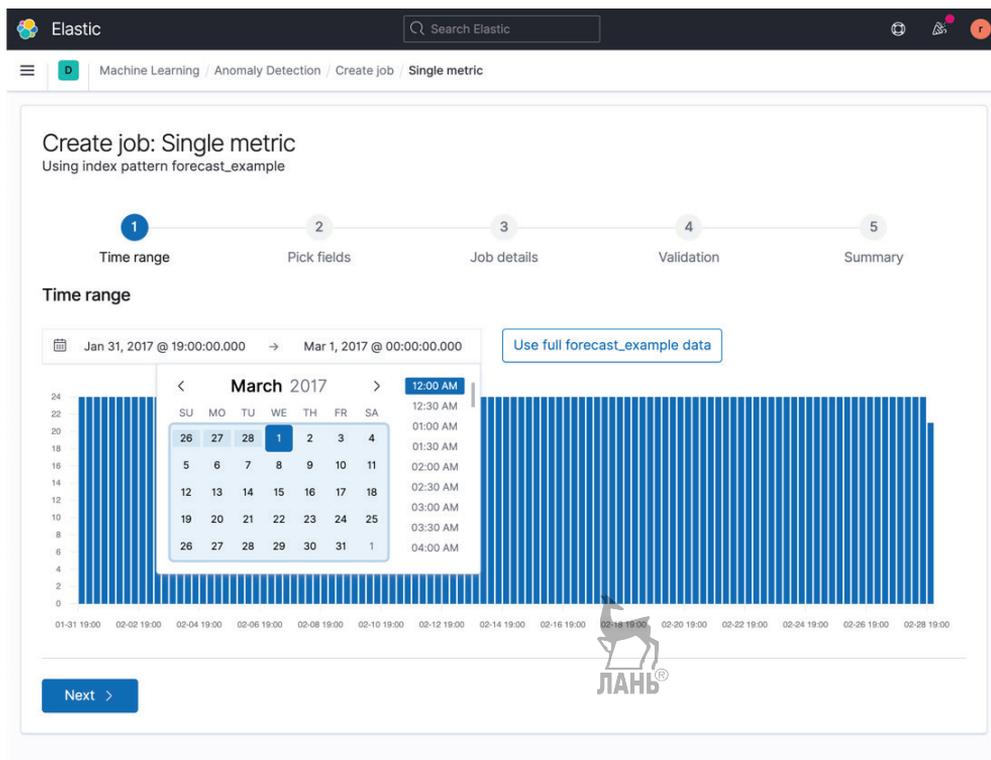


Рис. 4.11 ❖ Использование данных только до определенной даты

! В этом примере представлены данные за период с 31 января 2017 года по 1 марта 2017 года. Несмотря на то что сейчас они все оказались в прошлом, мы можем сделать вид, что находимся в этом временном интервале, и заявить, что сегодняшняя дата – 1 марта 2017 г. Поэтому нам нужно, чтобы задание машинного обучения анализировало данные между 31 января и «сегодня», а затем использовало машинное обучение для прогнозирования этой метрики на 10 дней вперед. Позже мы увидим, насколько точен наш прогноз относительно остальных данных. Если ваш часовой пояс Kibana настроен на ваше местное время, даты в этой главе могут выглядеть немного иначе, поскольку снимки экрана были сделаны с версией Kibana, настроенной на восточный часовой пояс США (US).

Теперь нажмите кнопку **Next** (Далее), чтобы перейти к следующему шагу мастера настройки.

- После нажатия кнопки **Next** нам нужно будет выбрать предмет анализа в раскрывающемся списке **Pick fields** (Выбрать поля). Мы выберем поле **Sum (amount)**, потому что поле суммы представляет собой просто числовое значение, меняющееся с течением времени (рис. 4.12).

The screenshot shows the 'Create job: Single metric' configuration page in Elastic Kibana. The page is titled 'Create job: Single metric' and 'Using index pattern forecast_example'. A progress bar at the top indicates the current step is 'Pick fields' (step 2 of 5). Below the progress bar, the 'Pick fields' section shows a dropdown menu with 'Sum(amount)' selected. A line chart displays a time series of data points from 01-31 19:00 to 02-28 19:00. The y-axis represents the sum of amounts, ranging from 10,000 to 130,000. The chart shows a fluctuating pattern with several peaks. Below the chart, there are settings for 'Bucket span' (set to 15m) and 'Sparse data' (set to 'Sparse data'). A 'Next' button is visible at the bottom right.

Рис. 4.12 ❖ Выбор поля **Sum (amount)** в качестве предмета анализа

Нажмите кнопку **Next**, чтобы продолжить, оставив на данный момент другие параметры без изменения (значения по умолчанию).

- Теперь нам нужно дать название нашему заданию по обнаружению аномалий – в поле **Job ID** (Идентификатор задания) введите интуитивно понятное имя. На следующем снимке экрана было использовано имя `forecast_example` (рис. 4.13).

The screenshot shows the 'Create job: Single metric' configuration page in the Elastic ML console. At the top, there's a search bar and navigation links for 'Machine Learning', 'Anomaly Detection', 'Create job', and 'Single metric'. A progress bar indicates five steps: 1. Time range, 2. Pick fields, 3. Job details (current step), 4. Validation, and 5. Summary. The 'Job details' section includes:

- Job ID:** A text input field containing 'forecast_example'. A tooltip explains that spaces and characters like /, ?, " < > | * are not allowed.
- Job description:** A text area for optional descriptive text, currently empty.
- Groups:** A dropdown menu with the option 'Select or create groups'.
- Additional settings:** A link to expand more options.
- Advanced:** A link to expand more options.

 At the bottom, there are 'Previous' and 'Next' navigation buttons, with 'Next' being the active button. A watermark logo for 'ЛАНЬ' is visible in the bottom center of the interface.

Рис. 4.13 ❖ Назначение имени заданию по обнаружению аномалий

Вновь оставьте другие параметры по умолчанию и нажмите кнопку **Next**.

- Далее следует этап проверки, чтобы убедиться, что все правильно настроено для задания анализа, как показано на рис. 4.14. Нажмите кнопку **Next**, чтобы продолжить.
- На этом этапе работа готова к созданию, как показано на следующем снимке экрана:

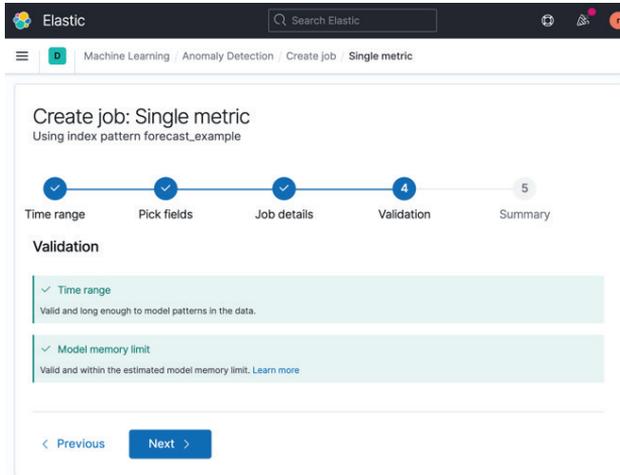


Рис. 4.14 ❖ Этап проверки задания

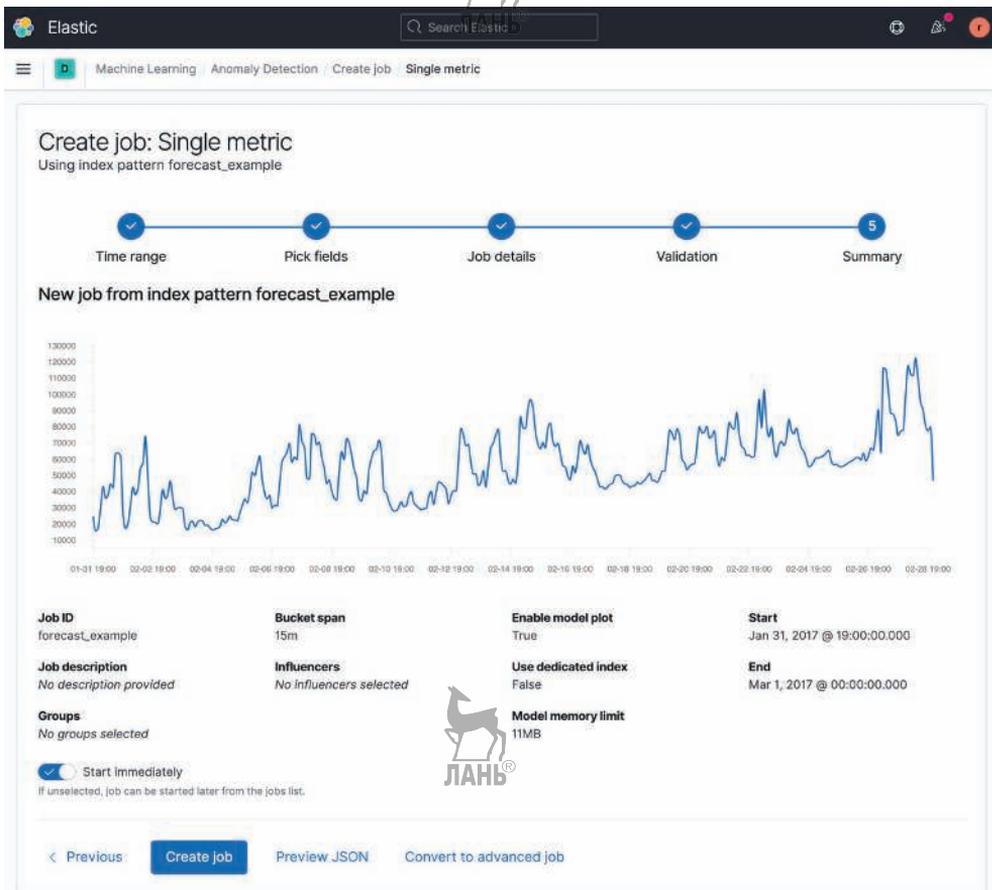


Рис. 4.15 ❖ Задание на обнаружение аномалий готово к созданию

12. После нажатия кнопки **Create job** (Создать задание) вы увидите анимированный предварительный просмотр результатов, наложенных поверх данных, как показано на следующем снимке экрана (рис. 4.16).

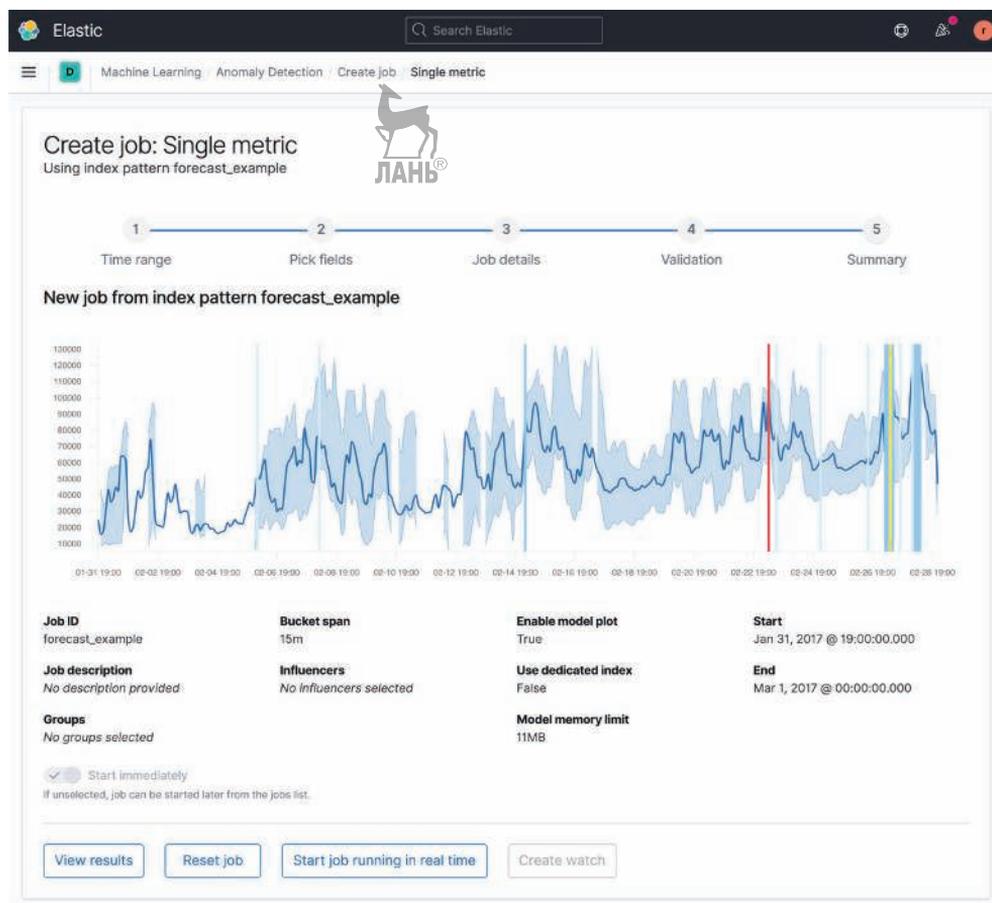


Рис. 4.16 ❖ Предварительный просмотр результатов выполнения задания

Чтобы получить доступ к функции прогнозирования, нам нужно нажать кнопку **View Results** (Просмотр результатов), которая приведет нас к средству просмотра единственной метрики **Single Metric Viewer**. Здесь мы можем увидеть весь набор данных и оценить форму и сложность поведения этих данных; есть как дневные, так и еженедельные периодические компоненты, а также постоянный положительный наклон (тренд), который вызывает дрейф данных с течением времени, как изображено на рис. 4.17.

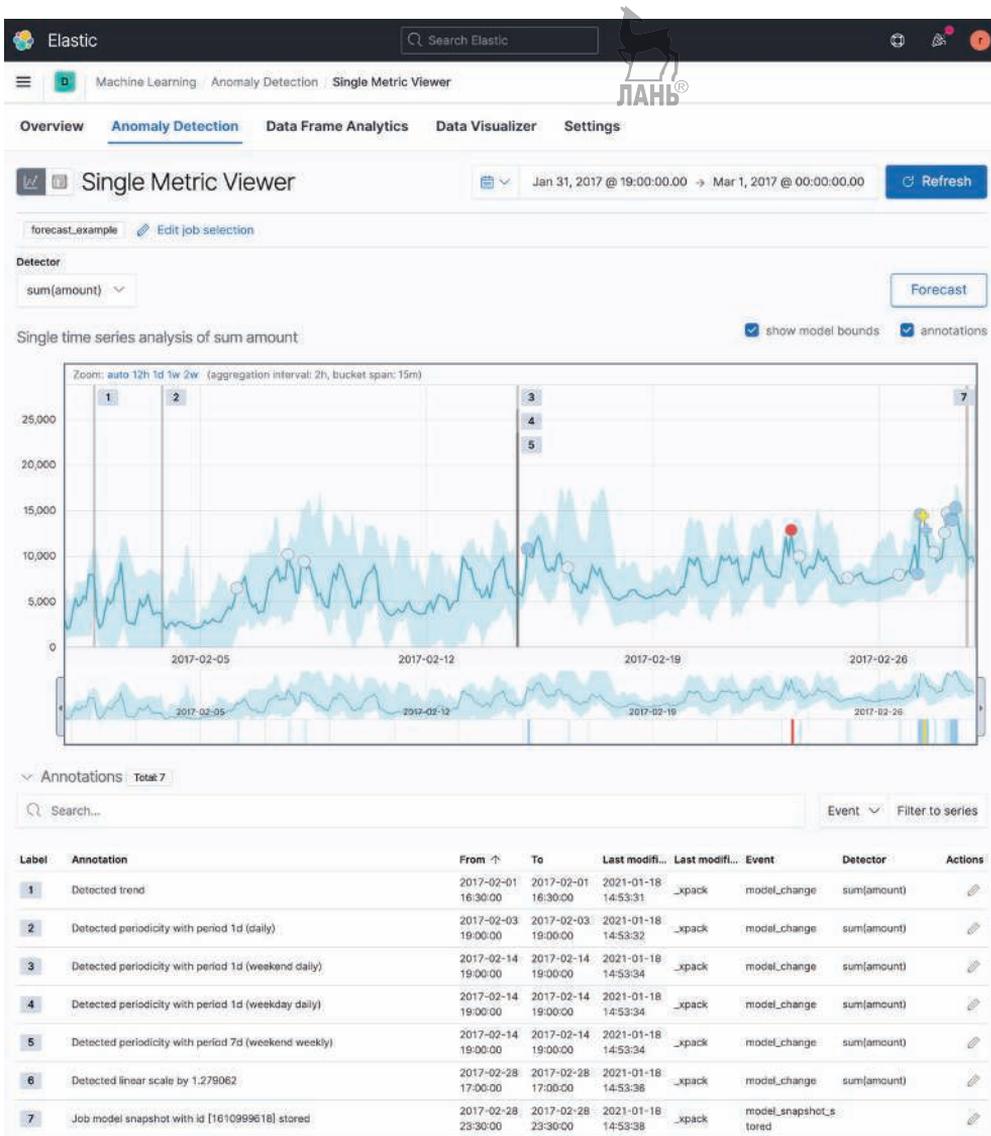


Рис. 4.17 ❖ Результаты с добавленными отображениями аннотаций

Если мы раскроем и изучим аннотации, то увидим, где Elastic ML обнаружил различные тенденции в данных. Несмотря на то что сейчас нас интересует только прогнозирование на основе этих данных, задание все равно будет указывать на аномалии в истории данных. Мы можем просто игнорировать их, если захотим.

13. Чтобы вызвать прогноз по этим данным, нажмите кнопку **Forecast** (Прогноз) и в диалоговом окне введите продолжительность 10 дней (10d), как показано на следующем снимке экрана (рис. 4.18).

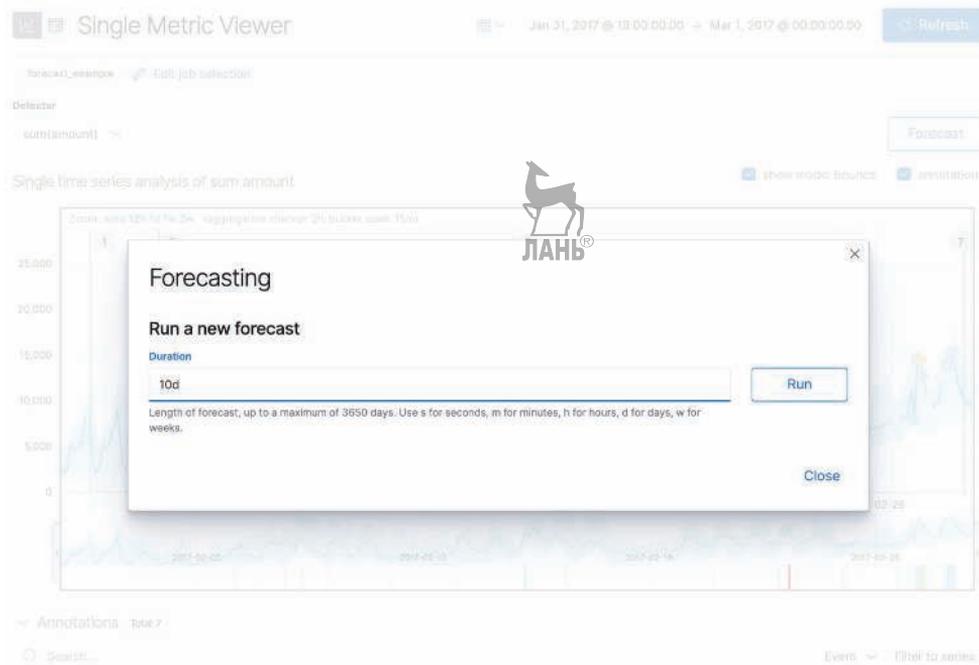


Рис. 4.18 ❖ Создание нового 10-дневного прогноза

- ! Не стоит запрашивать продолжительность прогноза, превышающую продолжительность данных, проанализированных заданием машинного обучения. Другими словами, не запрашивайте двухнедельный прогноз, если для машинного обучения применялись данные только за одну неделю. У вас должна быть как минимум равная продолжительность обучающего и прогнозируемого периодов (в идеале период исторических данных должен быть больше, чем период прогноза). Наконец, предоставьте модели достаточно большую последовательных данных, чтобы из нее можно было выявить закономерности. Например, для получения качественных прогнозов нужны минимум три цикла периодического шаблона.

После нажатия кнопки **Run** (Выполнить), показанной на рис. 4.18, прогноз будет запущен в фоновом режиме. Мы можем увидеть результаты нашего прогноза практически сразу, как показано на рис. 4.19.



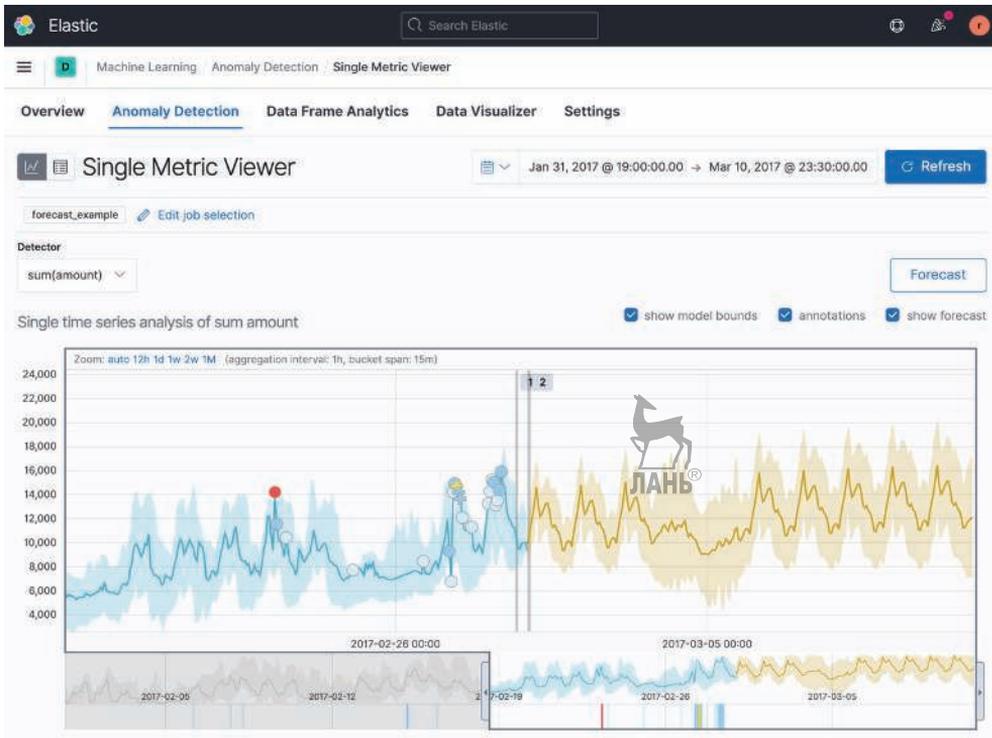


Рис. 4.19 ❖ Результаты прогноза

Заштрихованная область вокруг прогнозируемой зоны – это 95-й процентиль *доверительного интервала*. Другими словами, Elastic ML рассчитал, что существует 95%-ная вероятность того, что будущие значения попадут в этот диапазон (и, аналогично, только 2,5%-ная вероятность того, что будущие значения будут либо выше, либо ниже доверительного интервала). 95-й процентильный диапазон в настоящее время является фиксированным значением и не может быть установлен пользователем.

Теперь, когда у нас есть возможность создавать простые прогнозы из пользовательского интерфейса, давайте рассмотрим результаты прогноза подробнее, а затем перейдем к более сложному примеру.

ПРОСМОТР РЕЗУЛЬТАТОВ ПРОГНОЗИРОВАНИЯ

Теперь, когда мы составили прогноз, мы можем более подробно изучить результаты, полученные в процессе прогнозирования. Мы можем в любое время просмотреть результаты ранее созданного прогноза в пользовательском интерфейсе одним из двух способов. Первый способ – нажать кнопку **Forecast** в **Single Metric Viewer**, чтобы отобразить список предыдущих прогнозов (рис. 4.20).

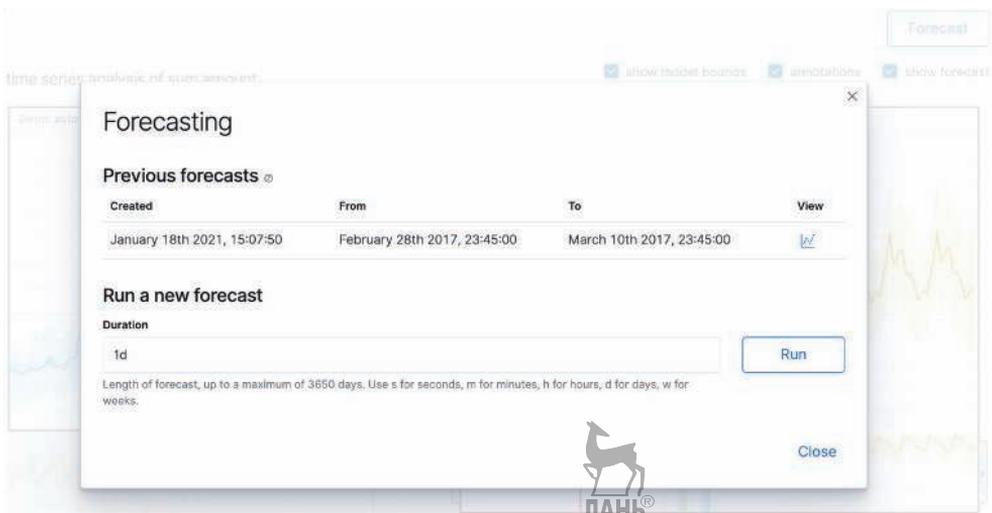


Рис. 4.20 ❖ Просмотр ранее созданных прогнозов в Single Metric Viewer

Кроме того, вы можете просмотреть их на странице **Job Management** (Управление заданиями) на вкладке **Forecasts** для этого задания, как показано на следующем снимке экрана (рис. 4.21).

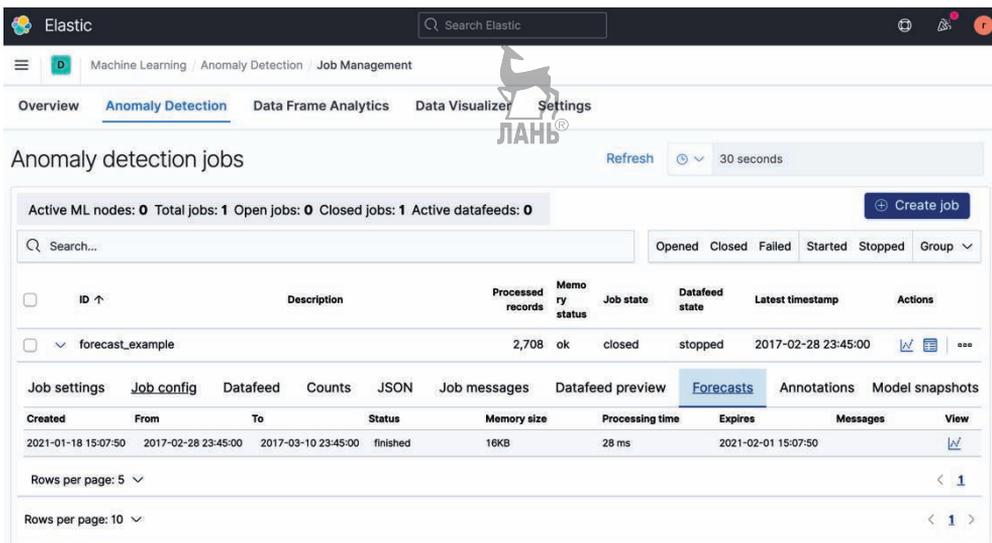


Рис. 4.21 ❖ Просмотр ранее созданных прогнозов на странице управления заданиями

❗ Срок хранения прогнозов, построенных в Kibana, по умолчанию составляет 14 дней. После этого результаты прогноза удаляются безвозвратно. Если требуется другой срок хранения, прогноз необходимо будет вызвать через конечную точку API `_forecast`, что мы обсудим позже; документацию по этому вопросу можно найти по адресу <https://www.elastic.co/guide/en/elasticsearch/reference/current/ml-forecast.html>.

При просмотре результатов прогноза в **Single Metric Viewer** обратите внимание на то, что при наведении указателя мыши на точки данных прогноза во всплывающем окне будут перечислены три ключевых элемента информации о точке данных – значение прогноза, значение верхней границы и значение нижней границы, как показано на рис. 4.22.

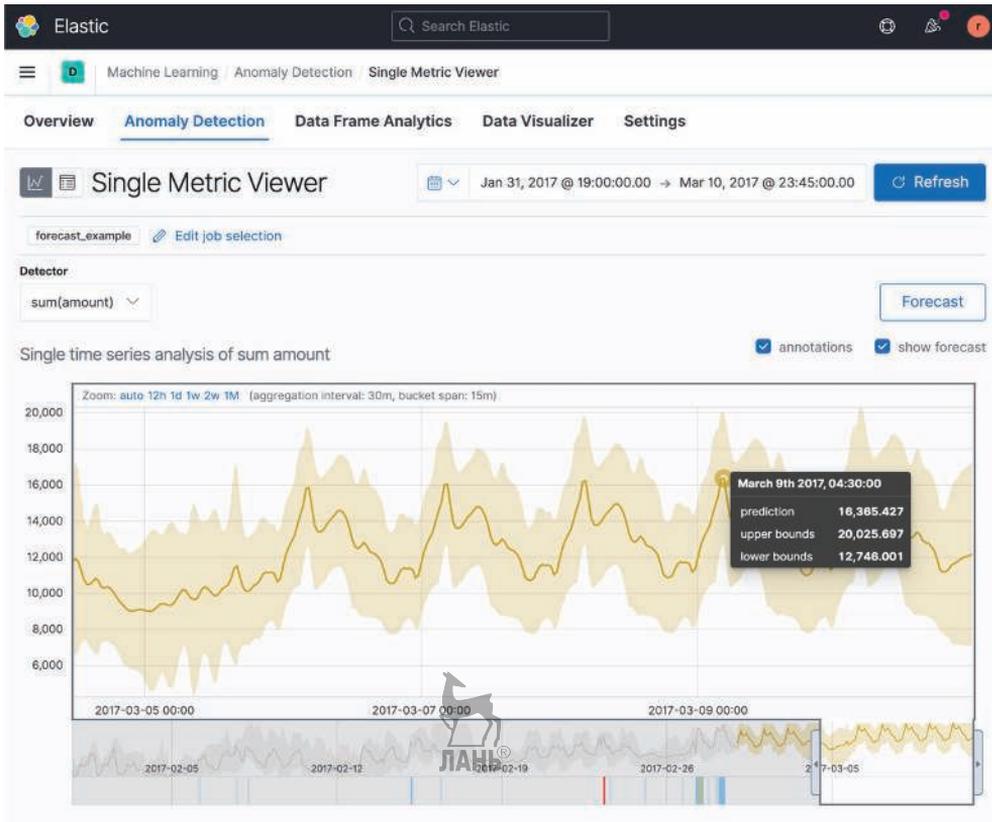


Рис. 4.22 ❖ Информация, отображаемая во всплывающем окне прогноза

Напомним, что верхняя и нижняя границы определяют диапазон доверительной вероятности 95-го перцентиля. Значение прогноза – это значение с наибольшим правдоподобием (вероятностью). Эти три ключевых значения хранятся в хранилищах результатов `.ml-anomalies-*` со следующими именами:

- `forecast_prediction;`
- `forecast_upper;`
- `forecast_lower.`

В главе 5 вы узнаете, как обратиться к хранилищам `.ml-anomalies-*`, чтобы найти информацию, относящуюся к прогнозам, и как использовать эту информацию для других целей, например для информационных панелей или предупреждений.

Если вы хотите увидеть, насколько хорошо прогноз Elastic ML совпадает с фактическими данными на следующие 10 дней (модели задания ML еще не видели те дни), вы можете вернуться на страницу управления заданиями, загрузить задание, чтобы продолжить его, и проанализировать оставшиеся данные. Для этого щелкните ссылку **Start datafeed** (Запустить поток данных) в меню с правой стороны, как показано на рис. 4.23.

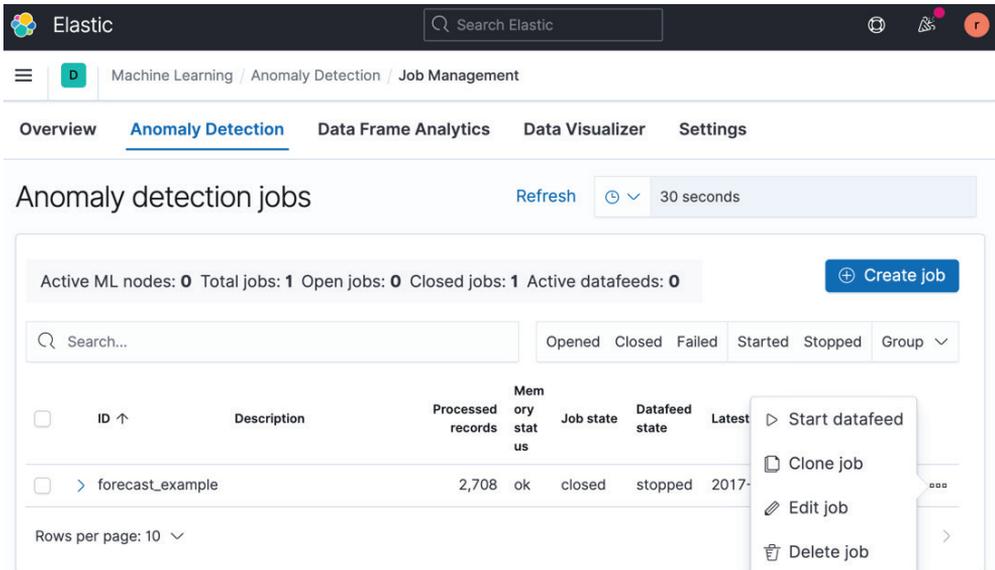


Рис. 4.23 ❖ Запуск потока данных со страницы управления заданиями

После появления диалогового окна установите в поле **Search start time** (Время начала поиска) значение **Continue from 2017-03-01 00:00:00** (Продолжить с 01.03.2017 00:00:00, или со времени, соответствующего вашему местному часовому поясу) и укажите в поле **Search end time** (Время окончания поиска) 11 марта 2017 в 12:00, как показано на рис. 4.24.

После этого вернитесь в **Single Metric Viewer** для задания, убедитесь, что вы просматриваете правильный диапазон времени с помощью средства выбора времени Kibana, и нажмите кнопку **Forecast**, чтобы просмотреть ранее созданный прогноз, как описано выше в этой главе. Теперь вы сможете увидеть значения прогноза, наложенные на фактические значения данных, как показано на рис. 4.25.

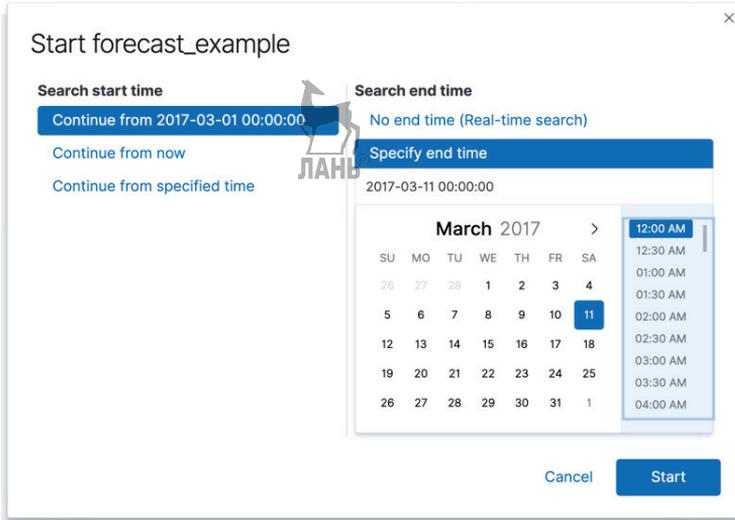


Рис. 4.24 ❖ Продолжение анализа потока данных с места остановки

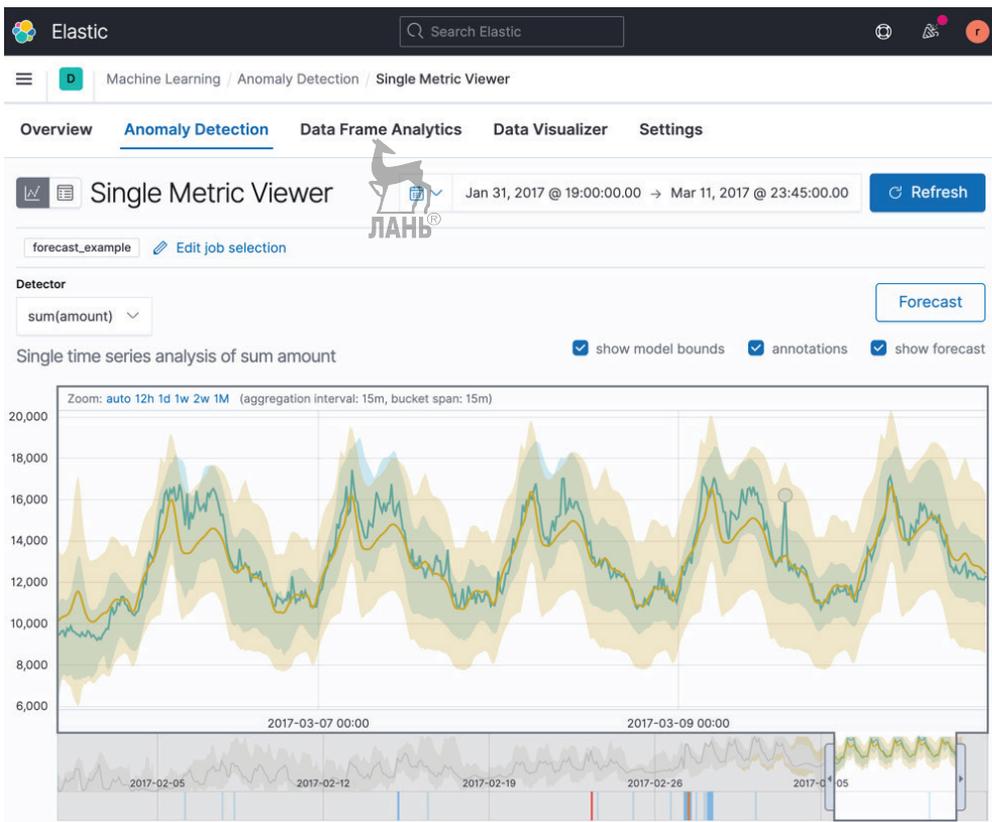


Рис. 4.25 ❖ Сравнение прогноза с фактическими данными

Как мы и говорили в начале этой главы, наблюдается небольшое расхождение между прогнозом Elastic ML и фактическими значениями данных. Это связано с тем, что прогнозы являются вероятностными, а вероятность влечет за собой определенный уровень неопределенности. Однако это не умаляет полезности прогнозов. Благодаря возможности упреждающего уведомления (глава 6) мы могли бы быть предупреждены об опасности взлома. Упреждающее уведомление особенно полезно, когда пользователи не могут отслеживать сотни или тысячи объектов по отдельности. В следующем разделе вы увидите, как прогнозирование нескольких метрик позволяет нам автоматически отслеживать эти объекты.

ПРОГНОЗИРОВАНИЕ НЕСКОЛЬКИХ ВРЕМЕННЫХ РЯДОВ

Чтобы выполнить прогнозирование для нескольких временных рядов, вам просто понадобится задание машинного обучения, которое моделирует несколько временных рядов. Предположим, что у нас есть задание машинного обучения, которое проанализировало веб-запросы по странам. Используя встроенные учебные образцы веб-журналов (`kibana_sample_data_logs`), которые мы применяли в главе 3, вы можете легко создать задание с несколькими метриками, которое подсчитывает события, разделенные по коду страны, из которой поступил запрос (поле называется `geo.src`), как показано на рис. 4.26.

В этом наборе данных 183 уникальные страны. После создания и выполнения задания по обнаружению аномалий, которое построит базовые модели для всех 183 стран, вы можете вызвать прогноз. Если мы применим к вызову прогноза тот же подход, что и раньше (через Single Metric Viewer), мы можем ошибочно подумать, что прогноз будет выполнен только для отображаемого ряда, как показано на следующем снимке экрана (рис. 4.27).

На предыдущем снимке экрана (рис. 4.26) мы видим, что код страны, выбранный для `geo.src`, – это CN (Китай). Но, как указано в предупреждающем сообщении всплывающего окна **Forecasting** (Прогнозирование), нажатие кнопки **Run** запустит прогноз, который будет выполняться для всех разделов, присутствующих в задании (в данном случае мы знаем, что существует более 100 разделов).



Create job: Multi-metric
Using index pattern kibana_sample_data_logs

1 Time range — 2 **Pick fields** — 3 Job details — 4 Validation — 5 Summary

Pick fields

Data split by geo.src

Count(Event rate)

Split field: geo.src

Bucket span: 15m [Estimate bucket span](#)

Influencers: geo.src

Sparse data: Sparse data

Рис. 4.26 ❖ Создание задания с несколькими метриками для прогнозирования

Forecasting

⚠ Note that this data contains more than 100 partitions so running a forecast may take a long time and consume a high amount of resource

Run a new forecast

Duration: 1d [Run](#)

Length of forecast, up to a maximum of 3650 days. Use s for seconds, m for minutes, h for hours, d for days, w for weeks.

[Close](#)

Рис. 4.27 ❖ Вызов многомерного прогноза из Single Metric Viewer

В качестве альтернативы для вызова прогноза мы можем использовать конечную точку `API_forecast`. Для этого в консоли **Dev Tools** (Инструменты разработчика) можно отправить следующий запрос:

```
POST _ml/anomaly_detectors/web_traffic_per_country/_forecast
{
  "duration": "1d"
}
```

! Задание обнаружения состояний должно быть в «открытом» состоянии, прежде чем вы сможете вызывать прогноз через API.

Ответ на вызов API выглядит следующим образом:

```
{
  "acknowledged" : true,
  "forecast_id" : " sm7AF3cBpc7Wt6MbTWYg"
}
```

Результаты запрошенного нами прогноза будут доступны для просмотра либо в **Single Metric Viewer**, либо программно путем запроса к хранилищу результатов, как это описано в главе 5. С помощью **Single Metric Viewer** мы можем увидеть прогноз для любой страны, нажав кнопку **Forecast**, выбрав наш предыдущий прогноз, а затем выбрав код страны, как показано на рис. 4.28.

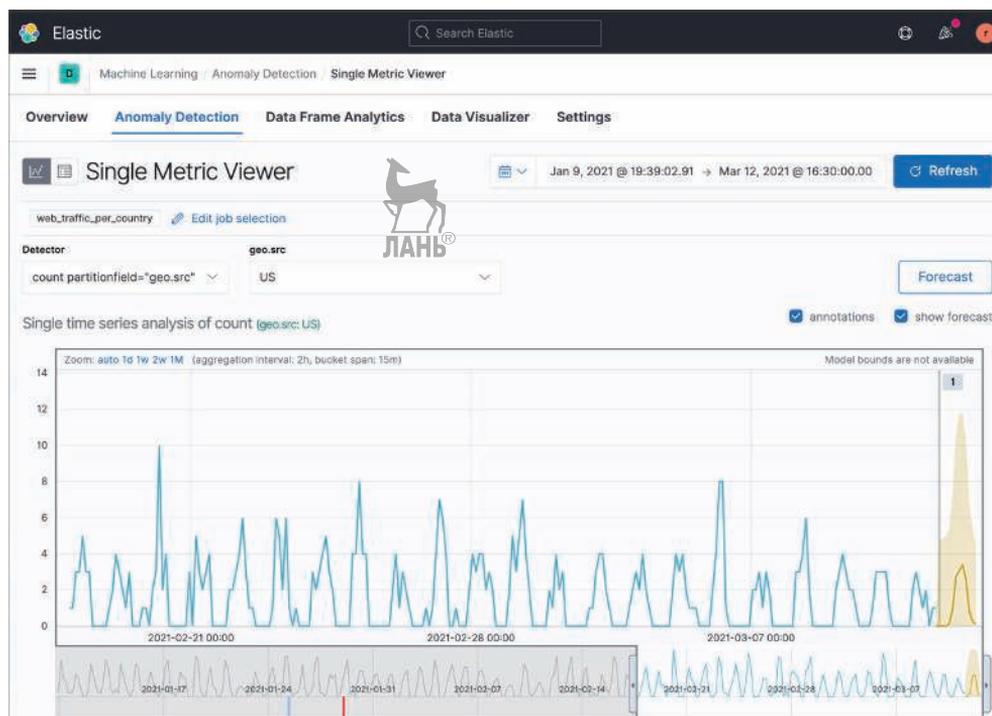


Рис. 4.28 ❖ Просмотр прогноза для отдельного раздела в составе общего прогноза по нескольким временным рядам



- ! Просмотр результатов многомерного прогноза в **Single Metric Viewer** не вполне совершенен, если речь идет о версии 7.10. Это связано с тем, что в представлении отображаются только те разделы, для которых записаны аномалии. Эта функция была добавлена в версии 7.11.

Прогнозирование по нескольким временным рядам может быть чрезвычайно полезным в сценариях планирования мощности, когда необходимо проанализировать сотни или, возможно, тысячи объектов и составить прогноз, чтобы увидеть, возможны ли какие-либо проблемы в ближайшем будущем.

ЗАКЛЮЧЕНИЕ

Помимо обнаружения аномалий, у Elastic ML есть дополнительная функция: способность экстраполировать модели временных рядов в будущее для целей прогнозирования. Если применить эту функцию в сценариях, подразумевающих расширенное обнаружение вторжений и планирование нагрузочной емкости, она снимает с нас бремя ручного построения графиков, отслеживания и прогнозирования будущих событий в зависимости от того, как объекты наблюдения вели себя в прошлом.

В следующей главе вы более детально изучите толкование результатов, полученных в результате обнаружения аномалий и прогнозирования, а также научитесь использовать эти результаты для информационных панелей и упреждающих предупреждений.



Глава 5

.....

Интерпретация результатов



Как вы видели в предыдущих главах, Elastic ML способен выполнять чрезвычайно полезный анализ как для обнаружения аномалий, так и для прогнозирования. Но тем не менее до этого момента мы лишь относительно поверхностно рассматривали результаты, полученные с помощью Elastic ML. В этой главе вы более детально исследуете получаемые результаты, узнаете, как они хранятся и как их можно использовать для получения дополнительной информации.

В частности, в этой главе будут рассмотрены следующие темы:

- просмотр хранилища результатов Elastic ML;
- оценка аномалий;
- подробнее о схеме хранилища результатов;
- аномалии в нескольких сегментах;
- результаты прогноза;
- API результатов Elastic ML;
- пользовательские панели мониторинга и рабочие панели Canvas.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Информация в этой главе основана на использовании Elastic Stack версии 7.10.

ПРОСМОТР ХРАНИЛИЩА РЕЗУЛЬТАТОВ ELASTIC ML

Поскольку значительную часть этой главы мы посвятим тому, как пользователи должны интерпретировать результаты заданий обнаружения аномалий Elastic ML, будет полезно изучить связь между передаваемой информацией и ее представлением для хранения во внутреннем хранилище результатов Elastic ML. Чтобы быстро получить представление об этом хранилище, вы можете либо обратиться к нему по шаблону напрямую, используя `API_search` в Elasticsearch, либо, что более интуитивно понятно, добавить шаблон хра-

нилица в Kibana и просмотреть его с помощью собственных инструментов Kibana. Но сначала вы должны выполнить небольшую процедуру, чтобы предоставить Kibana доступ к внутреннему хранилищу результатов Elastic ML.

1. В Kibana щелкните боковое меню и выберите в списке **Stack Management** (Управление стеком), как на рис. 5.1.

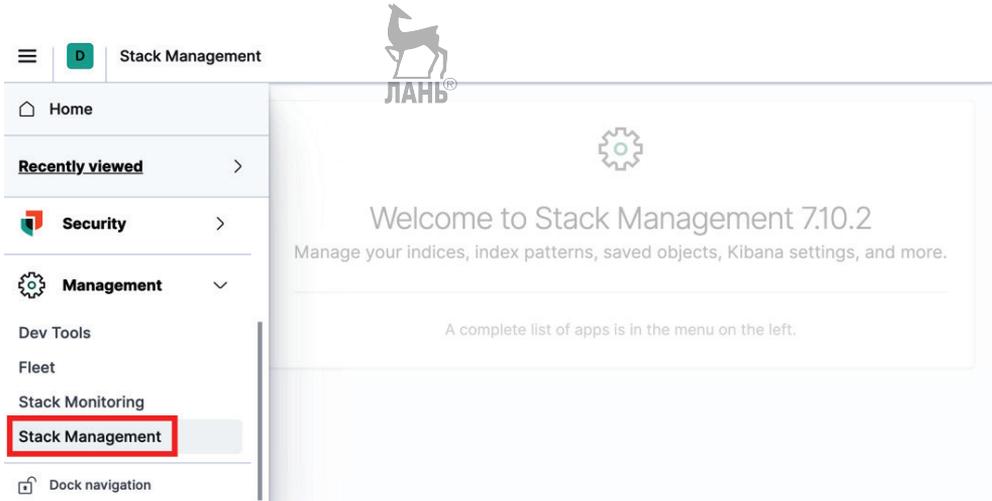


Рис. 5.1 ❖ Выбор опции **Управление стеком**

2. Выберите **Index Patterns** (Шаблоны хранилища) (рис. 5.2).

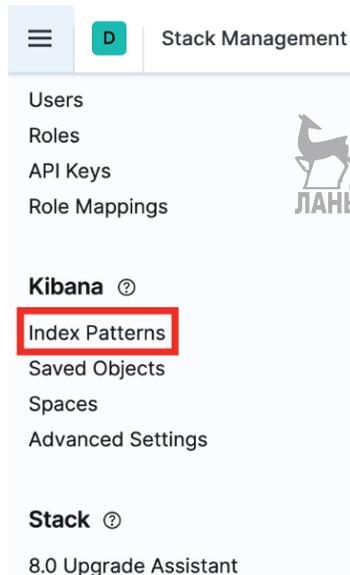


Рис. 5.2 ❖ Выбор опции **Шаблоны хранилища**

3. Выберите **Create index pattern** (Создать шаблон хранилища) (рис. 5.3).



Рис. 5.3 ❖ Выбор кнопки **Создать шаблон хранилища**

4. Введите `.ml-anomalies-*` в поле **Index pattern name** (Имя шаблона хранилища), а потом включите переключатель **Include system and hidden indices** (Включая системные и скрытые хранилища). Затем нажмите кнопку **Next step** (Следующий шаг) (рис. 5.4).

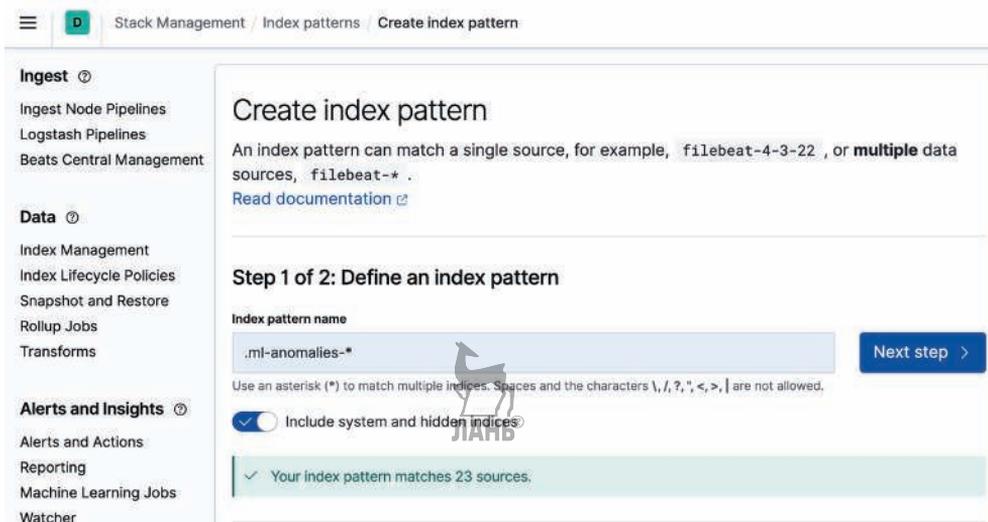


Рис. 5.4 ❖ Присвоение имени шаблону хранилища

5. Выберите опцию `timestamp` в поле **Time field** (Поле времени) и нажмите кнопку **Create index pattern** (Создать шаблон хранилища) (рис. 5.5).

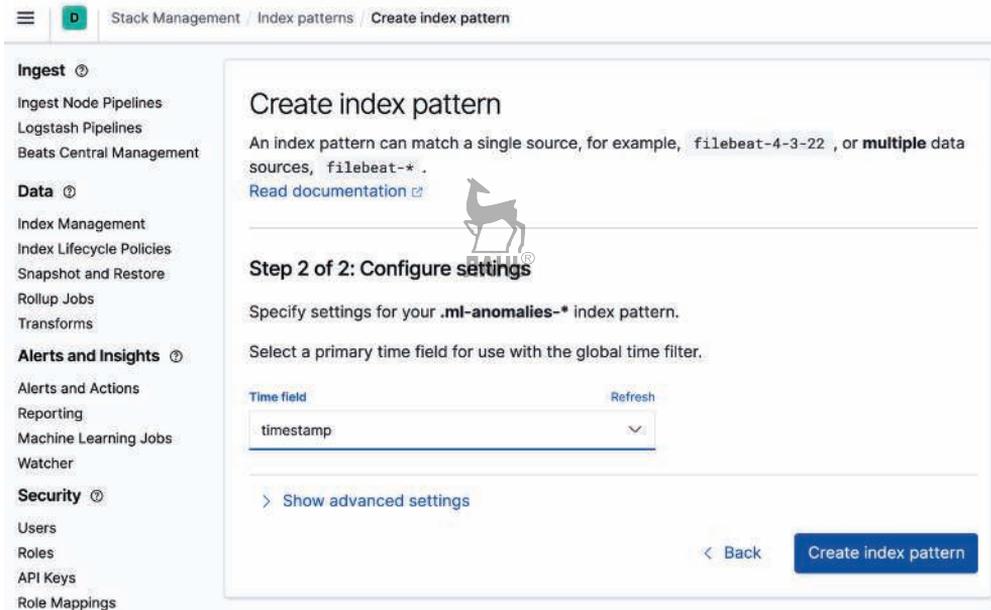


Рис. 5.5 ❖ Определение поля времени

6. Убедитесь, что шаблон хранилища определен (рис. 5.6).

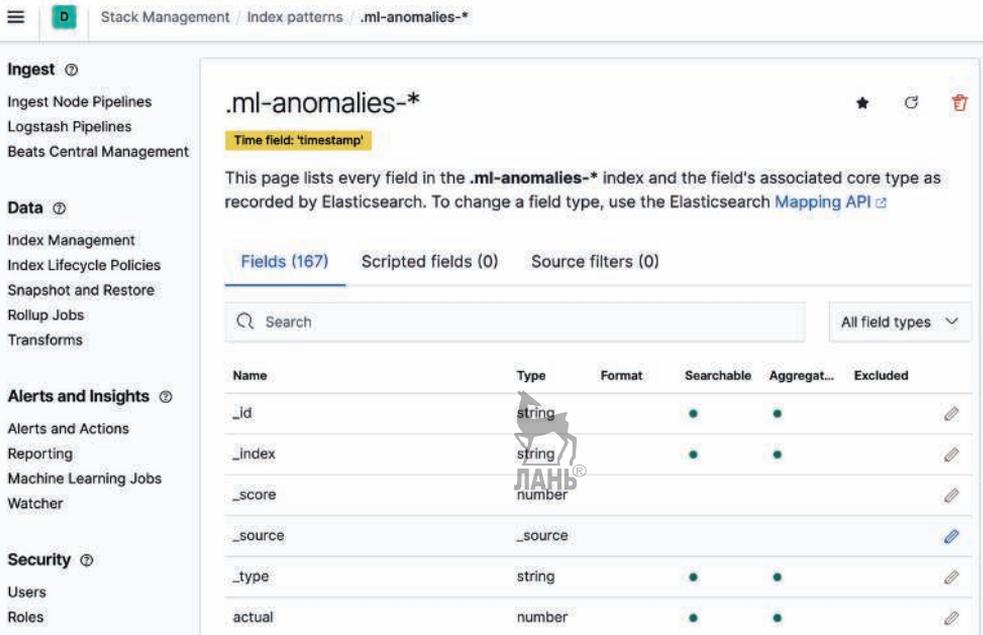


Рис. 5.6 ❖ Подтверждение того, что шаблон хранилища определен

Теперь, когда шаблон хранилища `.ml-anomalies-*` определен, вы можете использовать Kibana Discover для изучения содержимого хранилища результатов (выберите **Discover** в главном меню Kibana, как показано на рис. 5.7).

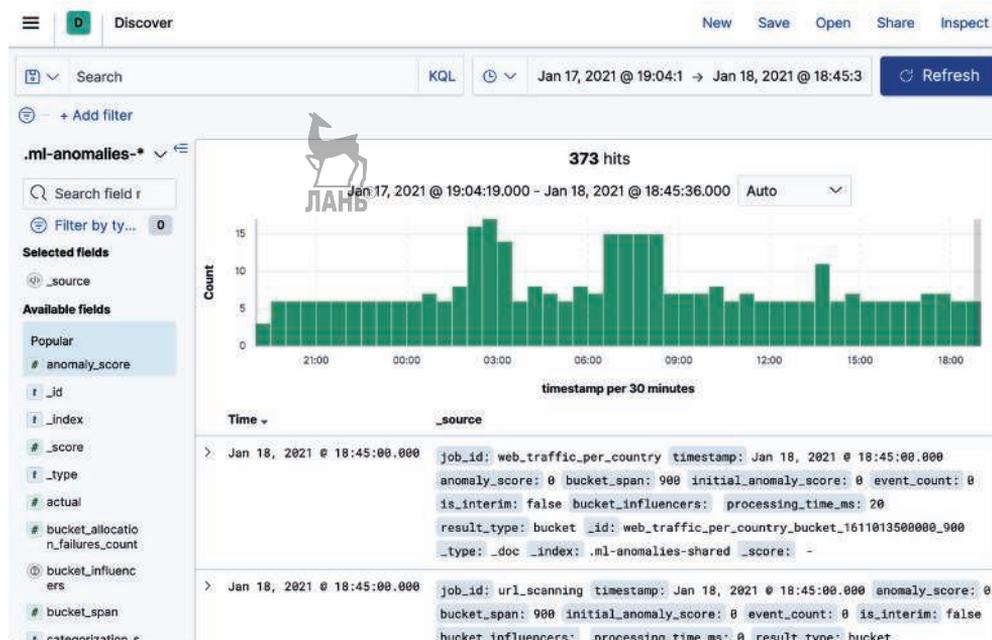


Рис. 5.7 ❖ Просмотр хранилища результатов в Kibana Discover

Получив доступ к просмотру хранилища результатов в Kibana Discover, вы можете использовать возможности поиска и фильтрации Discover, чтобы исследовать результаты. Например, вы можете запросить все аномалии на уровне записи для определенного имени задания по обнаружению аномалий, где оценка аномалии записи превышает определенное значение (рис. 5.8).

Синтаксис этого запроса на языке KQL (Kibana Query Language) выглядит следующим образом:

```
job_id:"web_traffic_per_country" and result_type:"record" and record_score>90
```

Здесь мы видим два конкретных вхождения, соответствующих нашему запросу. В хранилище результатов содержится множество информации, и вы будете последовательно учиться расшифровывать большую часть этой информации на протяжении всей главы.

❗ Хотя просматривать и запрашивать результаты в шаблоне хранилища `.ml-anomalies-*` безопасно, вы должны помнить, что хранилища, соответствующие этому шаблону, являются системными хранилищами, и не следует пытаться вручную изменить или удалить содержимое этих хранилищ.

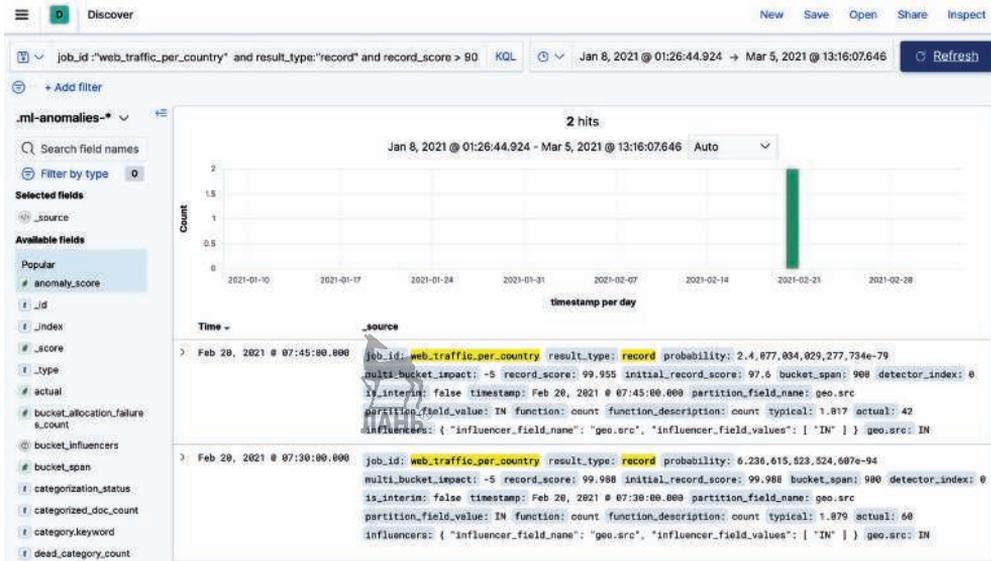


Рис. 5.8 ❖ Использование Kibana Discover для поиска и фильтрации аномалий

Первое, что необходимо понять, – это то, что существуют разные виды результатов (отсюда и поле `result_type`), а также разные виды оценок, которые отражают анализ с разных сторон или уровней. Поэтому мы начнем с более детального рассмотрения различных видов оценок и того, как эти оценки рассчитываются и сохраняются в хранилище результатов.

ОЦЕНКА АНОМАЛИЙ

Интерпретация результатов заданий по обнаружению аномалий в Elastic ML в первую очередь требует от вас понимания того факта, что существует несколько уровней оценок степени необычности, выраженных в результатах. Вот они:

- **уровень сегмента** (`result_type:bucket`): на этом уровне суммируются результаты всего задания по обнаружению аномалий за определенный сегмент времени. По сути это информация о том, насколько необычен этот сегмент времени с учетом конфигурации вашего задания;
- **уровень фактора влияния** (`result_type:influencer`): используется для лучшего понимания природы наиболее необычных объектов (факторов влияния) в течение определенного промежутка времени;
- **уровень записи** (`result_type:record`): это наиболее подробная информация о каждом аномальном происшествии или аномальном объекте в сегменте времени. В зависимости от конфигурации задания (несколько детекторов, разбиения и т. д.) за сегмент времени может накопиться много документов уровня записи.

Кроме того, чтобы разобраться, как выполняется оценка, вам также необходимо хорошо усвоить следующие понятия:

- **нормализация** – приведение оценки аномальности к фиксированной шкале от 0 до 100;
- **факторы влияния** – сущности, которые вызывают образование аномалий своим весомым вкладом в набор данных в момент возникновения аномалии.

Далее мы более подробно рассмотрим каждую из пяти вышеупомянутых концепций.

Оценка на уровне сегмента

Оценка аномалии на уровне сегмента сродни ответу на вопрос: «Насколько необычным был этот интервал времени по сравнению со всеми другими интервалами времени для этого задания?», где этот интервал определяется параметром `bucket_size` задания обнаружения аномалий. Если в вашем задании есть несколько детекторов или разбиений, результатом которых может быть несколько объектов одновременно, то каждый результат на уровне сегмента является агрегированным представлением всех этих вещей. Оценку аномалий на уровне сегмента можно просмотреть несколькими способами, первым из которых является *обзорная полоса (Overall)* вверху пользовательского интерфейса Anomaly Explorer (рис. 5.9).

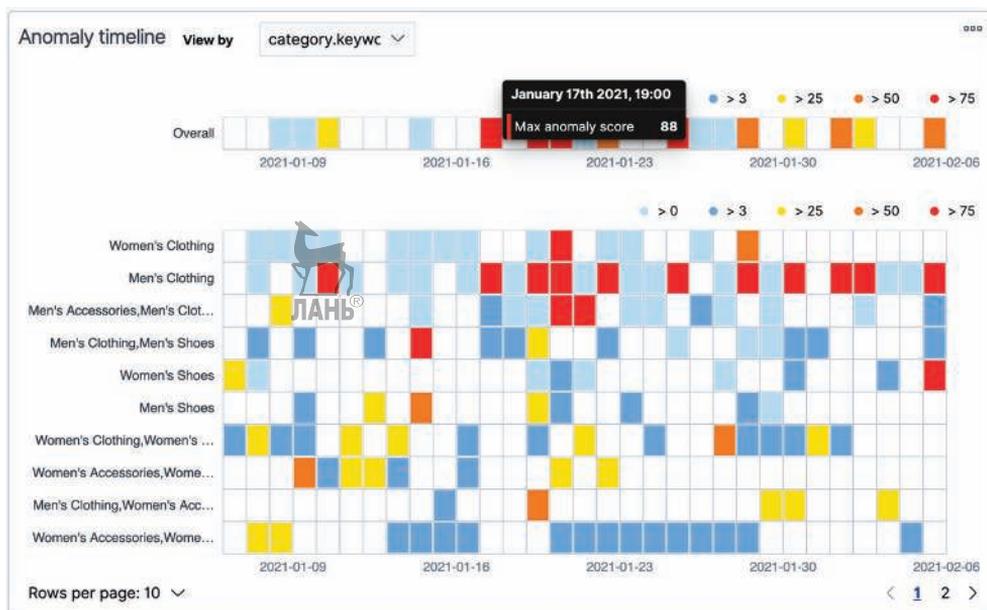


Рис. 5.9 ❖ Обзорная полоса **Overall** в Anomaly Explorer

На рисунке видно, что полоса **Overall** раскрашена в различные цвета – оценки значимости, причем конкретная выделенная плитка показывает максимальный балл аномалии (**Max anomaly score**), равный **88**. Здесь важно отметить, что временной диапазон, показанный в этом представлении, включает данные с 6 января по 5 февраля; поэтому на обзорной полосе по горизонтали расположено около 30 «плиток», каждая из которых представляет один день. Задание по обнаружению аномалий было настроено на 15 минут, поэтому на каждой плитке отображается максимальная оценка за весь день. Если вы увеличите масштаб отображения до одного дня, используя средство выбора времени Kibana (элемент управления диапазоном даты/времени в правом верхнем углу экрана), то увидите больше деталей, в частности аномалии на уровне сегмента времени, которые наблюдались между 02:00 и 02:30 (рис. 5.10).



Рис. 5.10 ❖ Обзорная полоса в Anomaly Explorer после увеличения детализации

Здесь также важно отметить, как общая обзорная полоса связана (или не связана) с сеткой полос под ней. Эта сетка полос показывает оценку на уровне фактора влияния (обсуждается в следующем разделе) и поэтому не имеет прямого отношения к оценке на уровне сегмента. Многие люди заблуждаются, когда думают, что общая обзорная полоса – это некая комбинация (например, максимальная оценка) элементов из сетки под ней. На рис. 5.10 вы можете видеть, что это явно не так, поскольку группа оценок на уровне фактора влияния во второй строке сетки (около 07:00) не имеет соответствующей оценки на общем уровне (сегменте). Почему так? Если коротко, дело в том, что общая полоса представляет собой сравнение сегментов времени друг с другом. Следовательно, самые необычные периоды времени получают самые высокие оценки, а периоды времени, которые намного менее необычны (из-за количества и серьезности отдельных аномалий в пределах этого периода времени), получают меньшие оценки или даже не получают оценок вообще. Процесс определения этой относительной оценки называется нор-

мализацией. Это важная часть оценки на всех уровнях, и она заслуживает отдельного объяснения.

Нормализация

Как впервые было сказано в главе 1, необработанные значения вероятности для конкретных аномалий нормализуются по шкале от 0 до 100. Этот процесс позволяет проводить относительное ранжирование аномалий, а также сводить значения к фиксированному интервалу значений, что удобно при оценке значимости в целях сортировки и/или оповещения.

Ключевым аспектом здесь является понятие *относительного ранжирования*. Другими словами, нормализованные значения учитывают ненормальности, которые уже были замечены заданием по обнаружению аномалий, и соответственно ранжируют их. Это также означает, что ранее присвоенные нормализованные оценки *могут изменяться* со временем по мере обнаружения новых аномалий. Поэтому вы можете заметить, что оценки в хранилище результатов имеют как «начальное», так и текущее значение, например:

- `initial_anomaly_score`: начальная оценка аномалии на уровне сегмента, которая была записана во время регистрации аномалии;
- `anomaly_score`: текущая нормализованная оценка аномалии на уровне сегмента.

Эти два значения могут быть одинаковыми, но могут и начать различаться со временем. Начальная оценка – это фиксированное значение, но текущая оценка может быть скорректирована по мере того, как со временем будут встречаться более выраженные аномалии. Процесс нормализации происходит каждые несколько часов во время работы в реальном времени или спонтанно, если аналитика обнаруживает резкие изменения в таблице нормализации. Это также происходит, если задание по обнаружению аномалий *закрыто* (переводится в закрытое состояние). Нормализация может заново оценить аномалии в прошлом, независимо от того, как настроен параметр `renormalization_window_days` (30 дней, или 100 сегментов времени, – это значение по умолчанию, и это значение можно изменить только в том случае, если задание создается с помощью API или мастера расширенных заданий путем прямого изменения файла JSON, содержащего конфигурацию задания).

Оценка на уровне фактора влияния

Оценка аномалии на уровне фактора влияния сродни ответу на вопрос «Какие сущности самые необычные в этот момент времени?», где мы теперь сравниваем эти сущности друг с другом. Оценки на уровне факторов влияния можно просмотреть двумя способами: первый – это основная сетка обзорных полос в центре пользовательского интерфейса Anomaly Explorer, а второй – это список **Top influencers** (Наиболее влиятельные факторы) в левой части (рис. 5.11).



Рис. 5.11 ❖ Представление факторов влияния в Anomaly Explorer

Здесь мы видим, что в основной сетке обзорных полос коды стран поля `geo.src` перечислены в порядке уменьшения общего балла фактора влияния. Обратите внимание, что, несмотря на то что сетка настроена на отображение 10 строк на странице, в списке указаны только шесть кодов стран (за этот период больше не нашлось стран, для которых есть значимые оценки факторов влияния). Кроме того, наиболее значимые факторы влияния перечислены в левой части, показывая для каждой сущности как максимальную оценку фактора влияния (99 для `geo.src:IN`), так и сумму всех оценок факторов влияния для этого временного диапазона (223 для `geo.src:IN`). В данном случае, поскольку для этого задания определен только один фактор влияния, эта информация может показаться избыточной. Однако для многих заданий определено более одного фактора влияния, поэтому в этом случае представление становится более разумным. Например, если мы посмотрим на задание анализа популяции по хранилищу `kibana_sample_data_logs`, в котором мы выбираем `distinct_count("url.keyword") over clientip` в качестве детектора и выбираем как `clientip`, так и `response.keyword` в качестве факторов влияния, представление Anomaly Explorer может выглядеть как на рис. 5.12.

Обратите внимание, что в поле **View by** (Просмотр по...) выбрано значение `clientip`, но в списке **Top influencers** слева показаны оба списка.

Anomaly Explorer является интерактивным, поэтому, если мы выберем плитку критической аномалии на общей обзорной строке на день 19 февраля, сетка и списки факторов влияния изменятся в связи с неявным применением фильтра для этого периода (рис. 5.13).

Теперь мы видим только соответствующие точки для выбранного дня. Получив некоторое представление о том, что такое факторы влияния, вы можете спросить: какие поля являются хорошими кандидатами на роль факторов влияния, если они могут быть представлены таким образом? Давайте сделаем небольшое отступление и рассмотрим понятие фактора влияния более подробно.

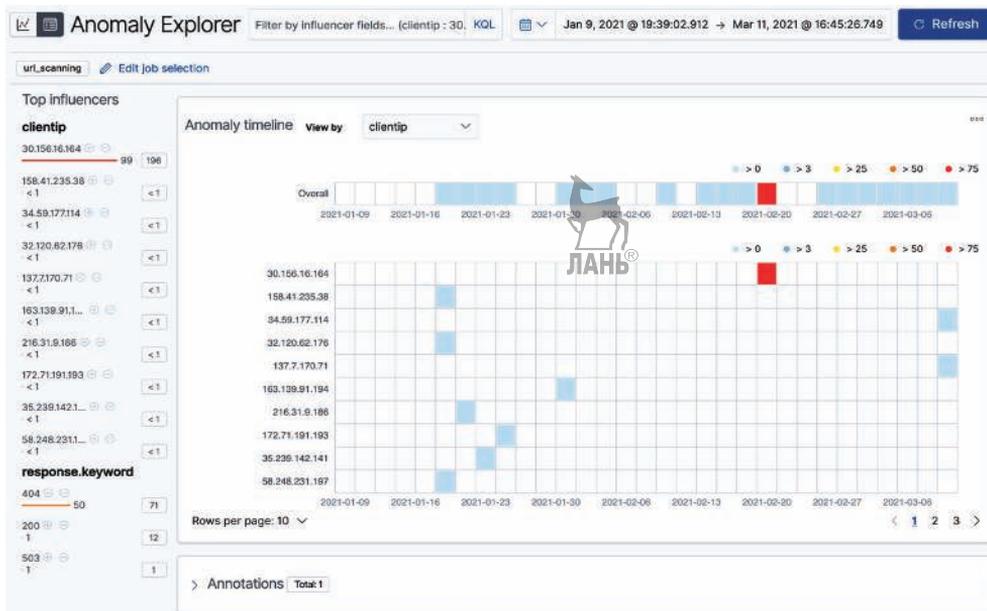


Рис. 5.12 ❖ Несколько факторов влияния в Anomaly Explorer

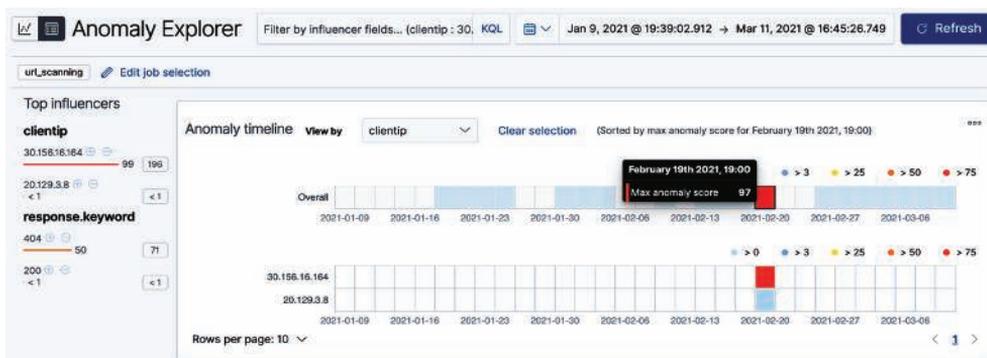


Рис. 5.13 ❖ Anomaly Explorer с фильтром за определенный день

Факторы влияния

В конфигурации задания обнаружения аномалий есть возможность объявить определенные поля факторами влияния. По сути, фактор влияния – это поле, описывающее некую сущность, относительно которой вы хотели бы знать, виновата ли она в существовании аномалии или, по крайней мере, внесла ли она значительный вклад. Обратите внимание, что любое поле, выбранное в качестве кандидата на роль фактора влияния, не обязательно должно быть частью логики детектора, хотя имеет смысл в качестве факторов влияния выбирать поля, которые используются как разбиения или популяции.

Если мы вернемся к примеру, показанному на рис. 5.13, то увидим, что поля `clientip` и `response.keyword` были объявлены как факторы влияния для задания (где `clientip` был частью конфигурации детектора, а `response.keyword` – нет). IP-адрес клиента `30.156.16.164` определен как главный фактор влияния. Это определение кажется несколько излишним, потому что данный IP-адрес и так является аномалией, но это ожидаемая ситуация, когда факторы влияния выбираются среди полей, которые определяют совокупность или являются полями разделения. Другой главный фактор влияния (`response.keyword`) имеет значение `404`. Эта конкретная информация чрезвычайно важна, поскольку дает пользователю прямую подсказку о том, что IP-адрес `30.156.16.164` делал во время аномалии. Если мы исследуем поведение IP-адреса во время аномалии, то увидим, что **100 %** сделанных запросов привели к ответу с кодом `404` (рис. 5.14).

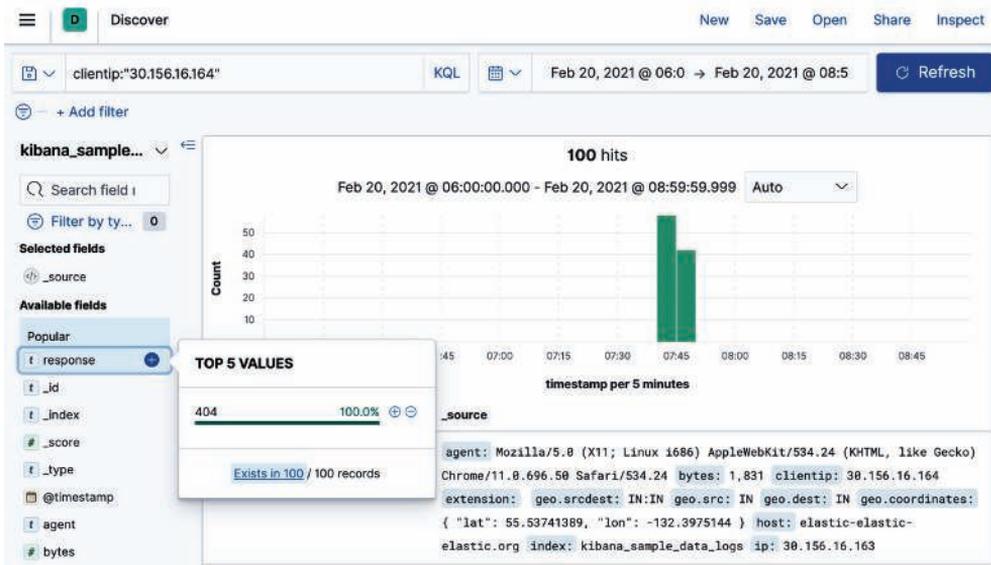


Рис. 5.14 ❖ Значение `404` поля фактора влияния значительно преобладает над другими результатами в период аномалии

Поэтому значение `404` имеет высокий балл в качестве фактора влияния (50, как показано на рис. 5.13). Вы можете подумать, что, поскольку 100 % запросов привели к ответу `404`, оценка фактора влияния также должна быть равна 100, но все не так просто. Оценки факторов влияния нормализованы относительно оценок других факторов влияния, и оценка также выражает, насколько необычным было значение `404` в остальное время. В этом конкретном примере набора данных встречаются еще сотни случаев ответа `404`, но большинство из них не были связаны с аномалиями. Наличие большого количества других таких же значений ограничивает оценку фактора влияния. Возможно, найдутся веские аргументы в пользу разделения концепций – одна оценка выражает необычность объекта на интервале времени, а другая оценка отражает

степень влияния значения поля на конкретную аномалию, – но на данный момент эти понятия формируют совместный рейтинг факторов влияния.

Также важно понимать, что процесс поиска потенциальных факторов влияния происходит уже после того, как Elastic ML обнаруживает аномалию. Другими словами, это не влияет на вычисления вероятности, выполняемые в рамках обнаружения. После определения аномалии ML будет систематически просматривать все экземпляры каждого поля, претендующего на роль фактора влияния, и удалять вклад этого экземпляра в данные в этом временном сегменте. Если после удаления оставшиеся данные больше не являются аномальными, то, исходя из контрфактических рассуждений, вклад этого экземпляра должен быть влиятельным и оценивается соответствующим образом (с помощью `influencer_score` в результатах).

Факторы влияния могут стать очень мощным инструментом, который можно использовать при просмотре результатов не только одного задания машинного обучения, но, возможно, нескольких связанных заданий. В главе 7 вы увидите, как эффективно использовать факторы влияния в анализе первопричин.

Оценка на уровне записи

Оценка аномалии на уровне записи – это самый низкий уровень абстракции результатов, содержащий наибольшее количество деталей. В пользовательском интерфейсе Anomaly Explorer результаты на уровне записи показаны в таблице внизу (рис. 5.15).

Anomalies								
Severity threshold		Interval						
warning		Auto						
time	severity ↓	detector	found for	influenced by	actual	typical	description	actions
February 20th 2021	99	distinct_count("url.keyword") over clientip	30.156.16.164	clientip: 30.156.16.164 response.keyword: 404	49	1.19	↑ 41x higher	
<p>Description critical anomaly in distinct_count("url.keyword") over clientip found for clientip 30.156.16.164</p> <p>Details on highest severity anomaly</p> <p>clientip 30.156.16.164</p> <p>time February 20th 2021, 07:30:00 to February 20th 2021, 07:45:00</p> <p>function distinct_count</p> <p>fieldName url.keyword</p> <p>actual 49</p> <p>typical 1.19</p> <p>job ID url_scanning</p> <p>probability 2.1100786126241776e-92</p> <p>Influencers</p> <p>clientip 30.156.16.164</p> <p>response.keyword 404</p>								

Рис. 5.15 ❖ Таблица аномалий, показывающая результаты на уровне записи

Обратите внимание, что если переключатель **Interval** (Интервал) установлен в положение **Auto** (Авто), то любые аномалии, которые следуют подряд, будут свернуты, и будет отображаться только аномалия с наивысшим баллом. Если в поле **Interval** установить значение **Show all** (Показать все), то при желании можно будет выявить каждую отдельную аномалию.

Распространенное заблуждение состоит в том, что оценка аномалии на уровне записи напрямую связана с отклонением, указанным в столбце **description** (описание) пользовательского интерфейса (здесь в 41 раз выше). Оценка определяется исключительно расчетом вероятности с использованием того же процесса нормализации, который был описан ранее. Поле **description** и даже значение `typical` — это упрощенные фрагменты контекстной информации, облегчающие понимание аномалии. Фактически, как вы увидите позже, поле **description** не сохраняется в хранилище результатов — оно вычисляется только на лету в Kibana.

Глядя на эти записи аномалий разного уровня в хранилище `.ml-anomalies-*`, мы видим, что многие поля предназначены для нашего использования. Некоторые могут быть очевидными, а некоторые нет. В следующем разделе мы тщательно рассмотрим схему хранилища результатов и опишем значение важных полей.

ОПИСАНИЕ СХЕМЫ ХРАНИЛИЩА РЕЗУЛЬТАТОВ

Как мы уже отмечали, внутри хранилища результатов есть множество различных документов, каждый из которых имеет свою полезность с точки зрения понимания результатов заданий по обнаружению аномалий. Здесь мы обсудим те из них, которые непосредственно относятся к трем уровням абстракции, представленным ранее в этой главе. Они вполне логично названы следующим образом:

- `result_type:bucket` — для получения результатов на уровне сегмента;
- `result_type:record` — для получения результатов на уровне записи;
- `result_type:influencer` — для получения результатов на уровне фактора влияния.

Распределение документов по типам будет зависеть от конфигурации задания машинного обучения и характеристик анализируемого набора данных. Принципы формирования документов по типам выглядят следующим образом:

- `result_type:bucket` — на каждый сегмент времени записывается один документ. Другими словами, если период сегментации составляет 15 минут, то один документ этого типа будет записываться каждые 15 минут. Его метка времени будет равна переднему краю сегмента. Например, для периода времени, охватывающего диапазон от 11:30 до 11:45, в итоговом документе этого типа будет метка времени 11:30;
- `result_type:record` — один документ записывается для каждого случая аномалии в пределах сегмента времени. Следовательно, если мы имеем дело с большим набором данных, охватывающим множество объектов (IP-адреса, имена хостов и т. д.), то во время крупного аномального со-

бытия или широко распространенного сбоя в некоторых сегментах времени могут быть сгенерированы сотни или даже тысячи записей об аномалиях. Этот документ также будет иметь метку времени, равную переднему краю сегмента;

- `result_type:influencer` – один документ для каждого фактора влияния, найденного для каждой записи об аномалии. Поскольку для каждой записи об аномалии потенциально может быть найдено несколько факторов влияния разных типов, эта разновидность документа может быть даже более объемной, чем результаты уровня записи. Этот документ также будет иметь метку времени, равную переднему краю сегмента.

Иметь представление о полях в этих документах будет особенно важно, когда мы перейдем к главе 6, рассказывающей об оповещениях, потому что неизбежно придется искать компромисс между детализацией оповещений (обычно чем больше, тем лучше) и количеством отдельных оповещений в единицу времени (обычно чем меньше, тем лучше). Мы вернемся к этому вопросу, когда начнем разрабатывать настоящие оповещения.

Результаты на уровне сегмента

На самом высоком уровне абстракции находятся результаты на уровне сегмента. Помните, что это агрегированные результаты для всего задания как функции времени, и, по сути, они отвечают на вопрос: «Насколько необычным был этот сегмент времени?»

Рассмотрим пример документа в хранилище `.ml-anomalies-*`, используя Kibana Discover и выполнив следующий запрос на языке KQL:

```
result_type : "bucket" and anomaly_score > 98
```

Ответ на запрос представлен на рис. 5.16.

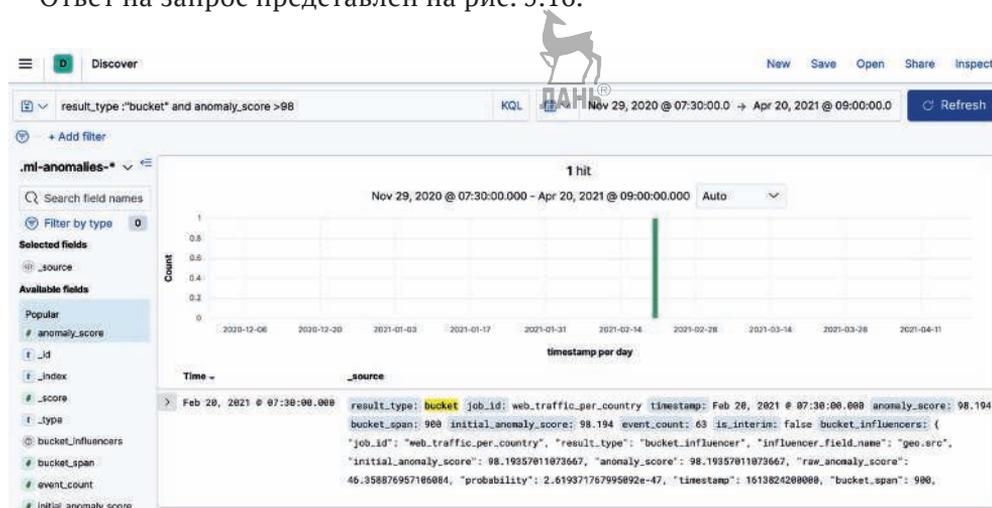


Рис. 5.16 ❖ Документ с результатами на уровне сегмента, показанный в Kibana Discover

Нажав на значок > рядом с меткой времени документа (рис. 5.16), вы развернете документ для просмотра во всех подробностях (рис. 5.17).

Expanded document

View surrounding documents View single document

Table	JSON
t _id	web_traffic_per_country_bucket_1613824200000_900
t _index	.ml-anomalies-shared
# _score	-
t _type	_doc
# anomaly_score	98.194
bucket_influencers	<pre>{ "job_id": "web_traffic_per_country", "result_type": "bucket_influencer", "influencer_field_name": "geo.src", "initial_anomaly_score": 98.19357011073667, "anomaly_score": 98.19357011073667, "raw_anomaly_score": 46.358876957106084, "probability": 2.619371767995892e-47, "timestamp": 1613824200000, "bucket_span": 900, "is_interim": false }, { "job_id": "web_traffic_per_country", "result_type": "bucket_influencer", "influencer_field_name": "bucket_time", "initial_anomaly_score": 98.19357011073667, "anomaly_score": 98.19357011073667, "raw_anomaly_score": 46.35887695710608, "probability": 2.6193717679951295e-47, "timestamp": 1613824200000, "bucket_span": 900, "is_interim": false } }</pre>
# bucket_span	900
# event_count	63
# initial_anomaly_score	98.194
is_interim	false
t job_id	web_traffic_per_country
# processing_time_ms	28
t result_type	bucket
timestamp	Feb 20, 2021 @ 07:30:00.000

Рис. 5.17 ❖ Детали документа на уровне сегмента в Kibana Discover

Вы можете видеть, что из нашего запроса на рис. 5.16 был возвращен только один документ уровня сегмента, соответствующий единственному аномальному периоду времени (с меткой времени 1613824200000 или в моем часовом поясе, 20 февраля 2021 года, 07:30:00 AM GMT-05:00), у которого anomaly_score превышает 98. Другими словами, не было других сегментов времени с такими большими аномалиями. Рассмотрим подробнее ключевые поля:

- timestamp – метка времени начала сегмента. В Kibana это поле будет по умолчанию отображаться в вашем местном часовом поясе (хотя оно сохраняется в хранилище в формате эпох с часовым поясом UTC);

- `anomaly_score` – текущая нормализованная оценка сегмента, основанная на диапазоне вероятностей, наблюдаемых для всего задания. Значение этой оценки может меняться со временем, поскольку новые данные обрабатываются заданием и могут быть обнаружены новые аномалии;
- `initial_anomaly_score` – начальная нормализованная оценка сегмента, то есть присвоенная, когда этот сегмент был впервые проанализирован. Эта оценка, в отличие от `anomaly_score`, не изменится по мере анализа большего количества данных;
- `event_count` – количество необработанных документов Elasticsearch, просмотренных алгоритмами машинного обучения в течение сегмента времени;
- `is_interim` – флаг, который показывает, завершен ли сегмент или он все еще ожидает получения всех данных в диапазоне сегмента. Это поле актуально для текущих заданий, выполняемых в режиме реального времени. Для некоторых типов анализа могут быть доступны промежуточные результаты, даже если не все данные сегмента были просмотрены;
- `job_id` – имя задания по обнаружению аномалий, создавшего этот результат;
- `processing_time_ms` – внутренний показатель производительности, показывающий, сколько времени (в миллисекундах) потребовалось для обработки данных этого сегмента;
- `bucket_influencers` – массив факторов влияния (и подробная информация о них), которые были найдены для этого текущего сегмента. Даже если факторы влияния не были выбраны как часть конфигурации задания или они не были выбраны как часть анализа, всегда будет существовать фактор влияния по умолчанию типа `influencer_field_name:bucket_time`, который в основном является внутренним устройством хранения записей, позволяющим упорядочивать аномалии на уровне сегмента в случаях, когда явные факторы влияния не могут быть определены.

Если у задания есть именованные и идентифицированные факторы влияния, то массив `bucket_influencers` может выглядеть так, как показано на рис. 5.17.

Обратите внимание, что в дополнение к записи по умолчанию для типа `influencer_field_name: bucket_time` в этом случае есть запись для имени поля `geo.src`, определенного анализом как фактор влияния. Это свидетельствует о том, что `geo.src` был релевантным фактором влияния, обнаруженным во время этой аномалии. Поскольку в конфигурации задания может быть выбрано несколько кандидатов на звание фактора влияния, следует отметить, что в данном случае `geo.src` является единственным полем фактора влияния, и никакие другие поля не были признаны влиятельными. Также следует отметить, что на этом уровне детализации конкретный экземпляр `geo.src` не раскрывается; эта информация будет раскрыта при запросах на более низких уровнях абстракции, которые мы обсудим далее.

Результаты на уровне записи

На более низком уровне абстракции располагаются результаты на уровне записи. Предоставляя как можно больше деталей, результаты на уровне записи показывают конкретные случаи аномалий и, по сути, отвечают на вопрос: «Какая сущность была необычной и насколько?»

Рассмотрим пример документа в хранилище `.ml-anomalies-*`, используя Kibana Discover и выполнив следующий запрос KQL:

```
result_type : "record" and record_score > 98
```

Результат будет выглядеть приблизительно так, как показано на рис. 5.18.

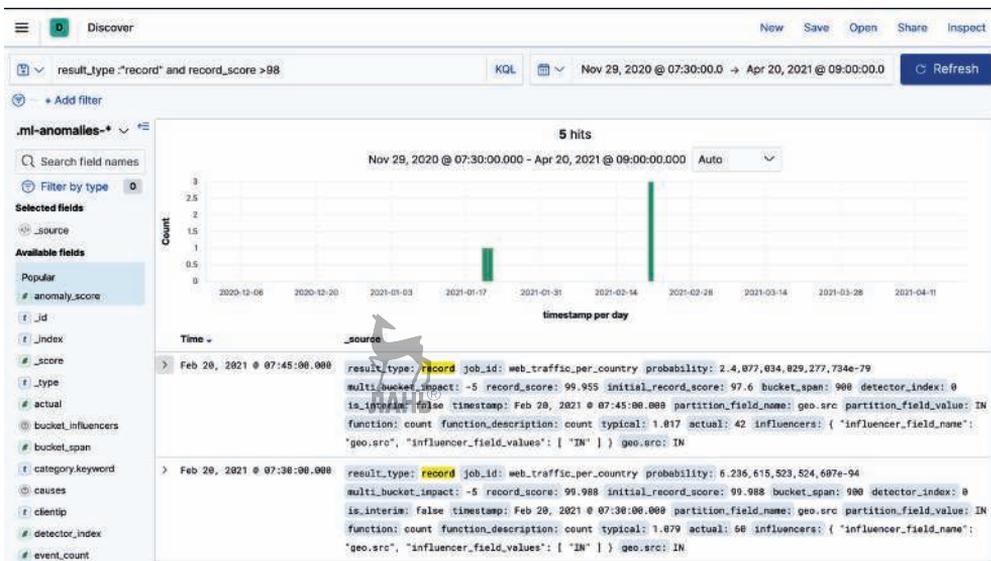


Рис. 5.18 ❖ Документ с результатами на уровне записи, показанный в Kibana Discover

Нажав на значок `>` рядом с меткой времени документа (рис. 5.18), вы развернете документ для просмотра во всех подробностях (рис. 5.19).

Вы можете видеть на рис. 5.18, что нашим запросом были возвращены несколько документов уровня записи. Рассмотрим подробнее ключевые поля:

- `timestamp` – метка времени начала сегмента, внутри которого произошла эта аномалия;
- `job_id` – имя задания по обнаружению аномалий, создавшего этот результат;
- `record_score` – текущая нормализованная оценка записи об аномалии, основанная на диапазоне вероятностей, наблюдаемых для всего задания. Значение этой оценки может меняться со временем, поскольку новые данные обрабатываются заданием и могут быть обнаружены новые аномалии;

Expanded document [View surrounding documents](#) [View single document](#)

Table JSON

t _id	web_traffic_per_country_record_1613825100000_900_0_-158853731021725837822058633927874012849_2
t _index	.ml-anomalies-shared
# _score	-
t _type	._doc
# actual	42
# bucket_span	900
# detector_index	0
t function	count
t function_description	count
t geo.src	IN
influencers	{ "influencer_field_name": "geo.src", "influencer_field_values": ["IN"] }
# initial_record_score	97.6
is_interim	false
t job_id	web_traffic_per_country
# multi_bucket_impact	-5
t partition_field_name	geo.src
t partition_field_value	IN
# probability	2.4,077,034,029,277,734e-79
# record_score	99.955
t result_type	record
timestamp	Feb 20, 2021 @ 07:45:00.000
# typical	1.017

Рис. 5.19 ❖ Детали документа на уровне записи в Kibana Discover

- `initial_record_score` – начальная нормализованная оценка записи об аномалии, то есть когда этот сегмент был впервые проанализирован. Эта оценка, в отличие от `record_score`, не изменится по мере анализа большего количества данных;
- `detector_index` – внутренний счетчик для отслеживания конфигурации детектора, к которой принадлежит эта аномалия. Очевидно, что для задания с одним детектором это значение будет равно нулю, но оно может быть ненулевым в заданиях с несколькими детекторами;
- `function` – ссылка для отслеживания того, какая функция детектора использовалась для создания этой аномалии;
- `is_interim` – флаг, который показывает, завершен ли сегмент или он все еще ожидает получения всех данных в пределах диапазона сегмента. Это поле актуально для текущих заданий, выполняемых в режиме реального времени. Для некоторых типов анализа могут быть доступны промежуточные результаты, даже если не все данные сегмента были просмотрены;

- `actual` – фактическое наблюдаемое значение проанализированных данных в этом сегменте. Например, если функцией является `count`, то она представляет количество документов, которые встречаются (и подсчитываются) в этом сегменте;
- `typical` – представление ожидаемого или прогнозируемого значения на основе модели машинного обучения для этого набора данных;
- `multi_bucket_impact` – значение по шкале от -5 до +5, которое определяет, насколько эта конкретная аномалия зависела от влияния вторичного анализа нескольких сегментов (поясняется позже в этой главе), от отсутствия влияния (-5) до полного влияния (+5);
- `influencers` – массив факторов влияния (и значения этих факторов), имеющих отношение к данной записи об аномалии.

Если в задании определены разделения (с именем `by_field_name` и/или `partition_field_name`) и найдены факторы влияния, то в документах результатов уровня записи будет больше информации, например как показано на рис. 5.19:

- `partition_field_name` – признак того, что было определено поле раздела и что для одного из значений поля раздела была обнаружена аномалия;
- `partition_field_value` – значение поля раздела, для которого возникла эта аномалия. Другими словами, имя сущности, для которой была обнаружена эта аномалия.

В дополнение к упомянутым здесь полям (которые имели бы вид `by_field_name` и `by_field_value`, если бы задание было настроено на использование поля `by`) мы также видим явный экземпляр поля `geo.src`. Это просто ярлык – каждый раздел `by` или `over_field_value` в результатах также будет иметь прямое имя поля.

Если ваше задание представляет собой популяционный анализ (с использованием `over_field_name`), то документ с результатами на уровне записи будет организован несколько иначе, так как отчет формируется с ориентацией на необычных членов совокупности. Например, если мы посмотрим на задание популяционного анализа по хранилищу `kibana_sample_data_logs`, в котором мы выбираем `distinct_count("url.keyword") over clientip` в качестве детектора, то пример документа с результатами на уровне записи также будет содержать массив причин (рис. 5.20).

Массив причин создан для компактного выражения всех аномальных действий, которые конкретный IP совершил в этом сегменте. Опять же, многие данные кажутся избыточными, но в первую очередь потому, что могут существовать разные способы агрегирования информации для представления результатов на информационных панелях или в оповещениях.

Кроме того, в случае этого популяционного анализа мы видим, что массив факторов влияния содержит как поле `clientip`, так и поле `response.keyword` (рис. 5.21).

Завершим наш обзор схемы хранилища результатов рассмотрением результатов на уровне факторов влияния.

```
causes {
  "probability": 2.1100766126241527e-92,
  "function": "distinct_count",
  "function_description": "distinct_count",
  "typical": [
    1.1901570565310753
  ],
  "actual": [
    49
  ],
  "field_name": "url.keyword",
  "over_field_name": "clientip",
  "over_field_value": "30.156.16.164"
}
```

Рис. 5.20 ❖ Документ на уровне записи, представляющий массив причин для задания популяционного анализа

```
influencers {
  {
    "influencer_field_name": "response.keyword",
    "influencer_field_values": [
      "404"
    ]
  },
  {
    "influencer_field_name": "clientip",
    "influencer_field_values": [
      "30.156.16.164"
    ]
  }
}
```

Рис. 5.21 ❖ Документ на уровне записи, представляющий массив факторов влияния для задания популяционного анализа

Результаты на уровне факторов влияния

Еще один объектив, через который можно взглянуть на результаты, – это факторы влияния. Просмотр результатов с этой точки зрения позволяет ответить на вопрос: «Какие сущности были самыми необычными в моем задании машинного обучения, и когда они были необычными?» Чтобы познакомиться со структурой и содержанием результатов на уровне факторов влияния, рассмотрим пример документа в хранилище `.ml-anomalies-*`, используя Kibana Discover и выполнив следующий запрос KQL:

```
result_type : "influencer" and response.keyword:404
```

Обратите внимание, что в этом случае мы запросили не оценку (`influencer_score`), а ожидаемое имя и значение объекта. Последний документ в списке (со значением `influencer_score`, равным 50,174) соответствует тому, что мы видели на рис. 5.13.

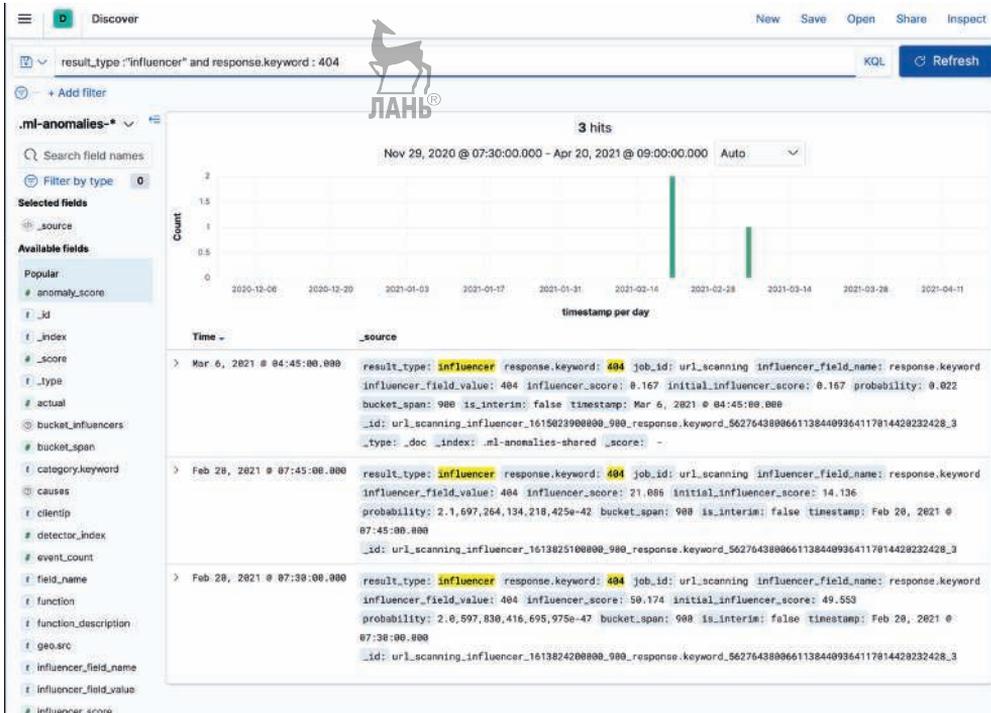


Рис. 5.22 ❖ Документ с результатами на уровне факторов влияния, показанный в Kibana Discover

Рассмотрим ключевые поля:

- `timestamp` – метка времени начала сегмента, внутри которого произошла anomальная активность, связанная с этим фактором влияния;
- `job_id` – имя задания по обнаружению аномалий, создавшего этот результат;
- `influencer_field_name` – имя поля, которое было объявлено как фактор влияния в конфигурации задания;
- `influencer_field_value` – значение поля фактора влияния, для которого актуален этот результат;
- `influencer_score` – текущая нормализованная оценка того, насколько необычным и значащим был фактор влияния для аномалии на данном этапе;
- `initial_influencer_score` – начальная нормализованная оценка фактора влияния, когда этот сегмент был впервые проанализирован. Эта оценка, в отличие от оценки `influencer_score`, не изменится по мере анализа большего количества данных;
- `is_interim` – флаг, который показывает, завершен ли сегмент или он все еще ожидает получения всех данных в диапазоне сегмента. Это поле

актуально для текущих заданий, выполняемых в режиме реального времени. Для некоторых типов анализа могут быть доступны промежуточные результаты, даже если не все данные сегмента были просмотрены.

Теперь, когда в деталях рассмотрели соответствующие поля, доступные пользователю, мы можем сохранить эту информацию в архиве для создания пользовательских панелей мониторинга, визуализаций и сложных оповещений, чем мы и займемся в следующих главах. Но прежде чем мы закончим эту главу, вам еще предстоит изучить несколько важных концепций. Далее следует обсуждение особого вида аномалии – аномалии в нескольких сегментах.

АНОМАЛИИ В НЕСКОЛЬКИХ СЕГМЕНТАХ

Почти все известные вам на данный момент аномалии, генерируемые заданиями по обнаружению аномалий Elastic ML, возникали в определенное время, квантованное с интервалом `bucket_span`. Тем не менее у вас, безусловно, могут возникнуть ситуации, в которых конкретное наблюдение в пределах одного сегмента может быть не таким уж необычным, но расширенное временное окно, охватывающее несколько сегментов, может оказаться значительно более необычным, чем любое отдельное наблюдение. Рассмотрим следующий пример.

Пример аномалии в нескольких сегментах

Мы впервые показали следующий график (рис. 5.23) под номером 3.17 в главе 3, а сейчас специально повторяем его, чтобы показать, как аномалии в нескольких сегментах отображаются в пользовательском интерфейсе Elastic ML.

Как мы обсуждали в главе 3, аномалии в нескольких сегментах обозначаются в пользовательском интерфейсе другим символом (крестиком вместо точки). Они соответствуют случаям, когда фактическое сингулярное значение не обязательно является аномальным, но есть тенденция, которая возникает в скользящем окне из 12 последовательных сегментов. Здесь вы можете увидеть заметный спад, охватывающий несколько соседних сегментов.

Обратите внимание, однако, что некоторые из маркеров аномалии в нескольких сегментах помещаются в точки времени, следующие после того, как данные «восстановлены». Это может несколько сбить с толку пользователей, пока вы не поймете, что, поскольку определение таких аномалий является вторичным анализом (в дополнение к посегментному анализу) и поскольку этот анализ представляет собой скользящее окно, наблюдающее аномалию в своей «хвостовой» части, передний край этого окна на момент обнаружения аномалии может оказаться в точке времени, когда ситуация уже восстановилась.

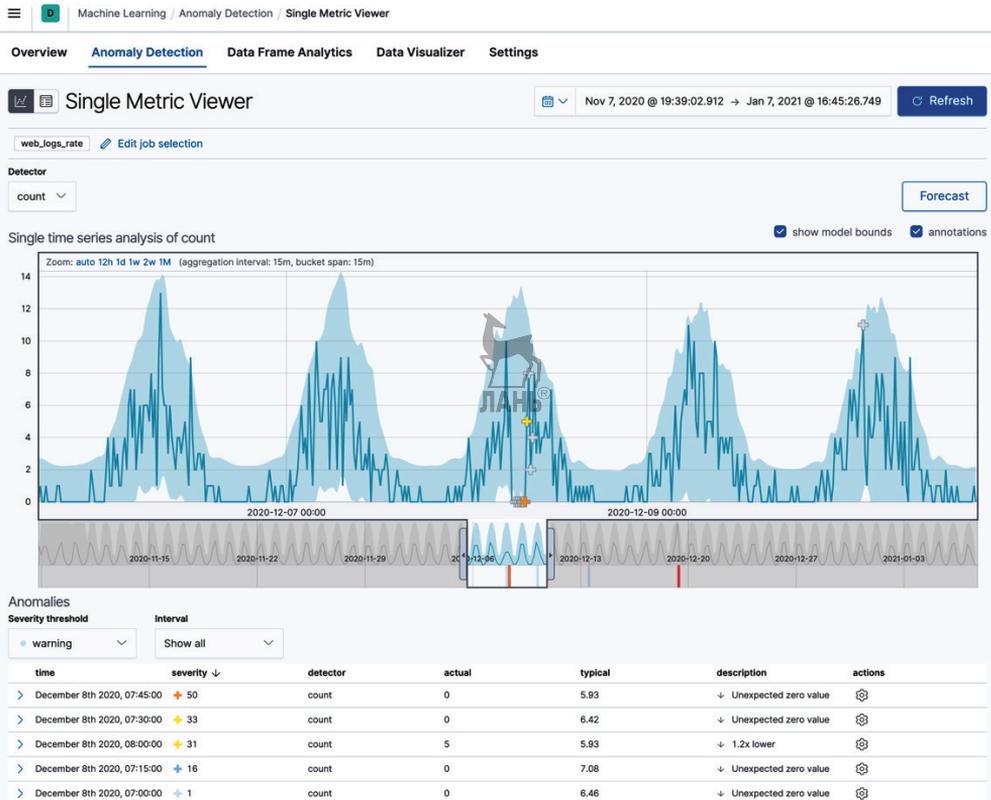


Рис. 5.23 ❖ Аномалии в нескольких сегментах, впервые показанные в главе 3

Оценка аномалии в нескольких сегментах

Как уже упоминалось выше, анализ нескольких сегментов – вторичный анализ. Поэтому для каждого диапазона сегментов вычисляются две вероятности – вероятность наблюдения, присутствующего в текущем сегменте, и вероятность наблюдения в нескольких сегментах – своего рода средневзвешенное значение текущего сегмента и 11 предыдущих. Если эти две вероятности представляют собой величины примерно одного порядка, тогда `multi_bucket_impact` будет низким (в отрицательной части шкалы от -5 до 5). Если, с другой стороны, вероятность наличия события в нескольких сегментах значительно ниже (а значит, событие более необычно), тогда `multi_bucket_impact` будет высоким.

В примере, показанном на рис. 5.24, пользовательский интерфейс покажет пользователю, что значение `multi_bucket_impact` велико (помечено словом `high` – высокий), но без указания фактических результатов.

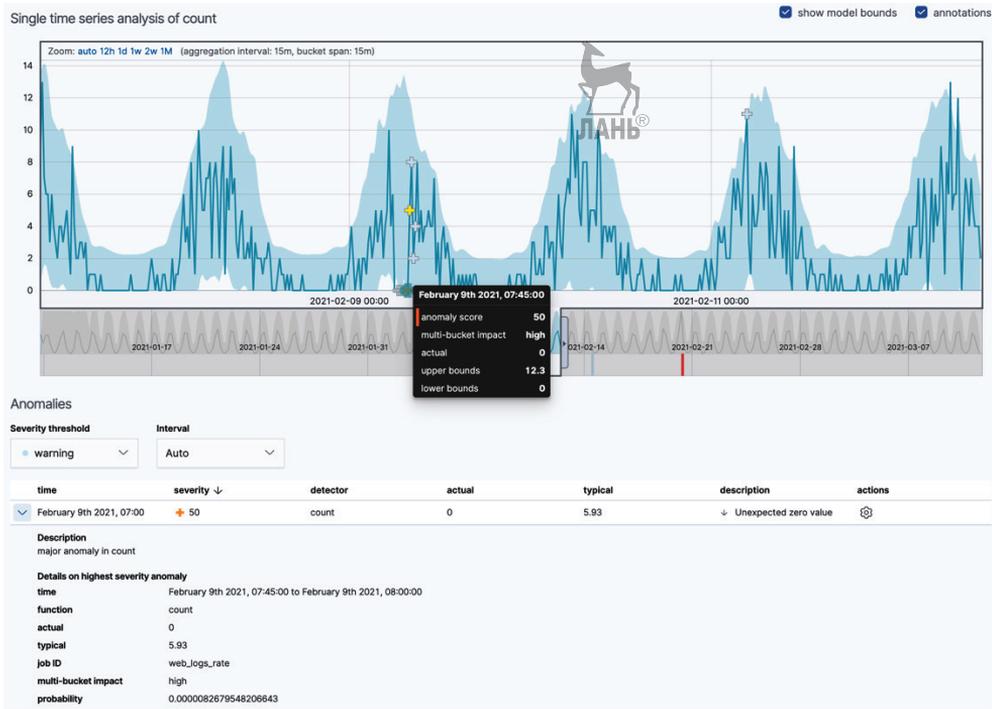


Рис. 5.24 ❖ Аномалии в нескольких сегментах с указанием оценки влияния

Однако если вы посмотрите на необработанный результат на уровне записи, то увидите, что `multi_bucket_impact` было присвоено значение 5 (рис. 5.25).



Рис. 5.25 ❖ Запись, соответствующая аномалии в нескольких сегментах, и оценка влияния в баллах по шкале от -5 до 5

Аномалии, наблюдаемые через окно в нескольких сегментах, дают вам другой взгляд на поведение ваших данных. Вам нужно помнить о том, как они обозначаются и оцениваются с помощью поля `multi_bucket_impact`, чтобы вы могли включать или исключать их, при необходимости, из вашей логики отчетов или оповещений.

Давайте теперь посмотрим на то, как результаты прогнозов представлены в хранилище результатов.



РЕЗУЛЬТАТЫ ПРОГНОЗА

Как подробно объясняется в главе 4, мы можем заставить Elastic ML экстраполировать в будущее тенденции анализируемых данных. Давайте заново рассмотрим рис. 5.26 с результатами прогноза, впервые показанный в главе 4 (рис. 4.21).

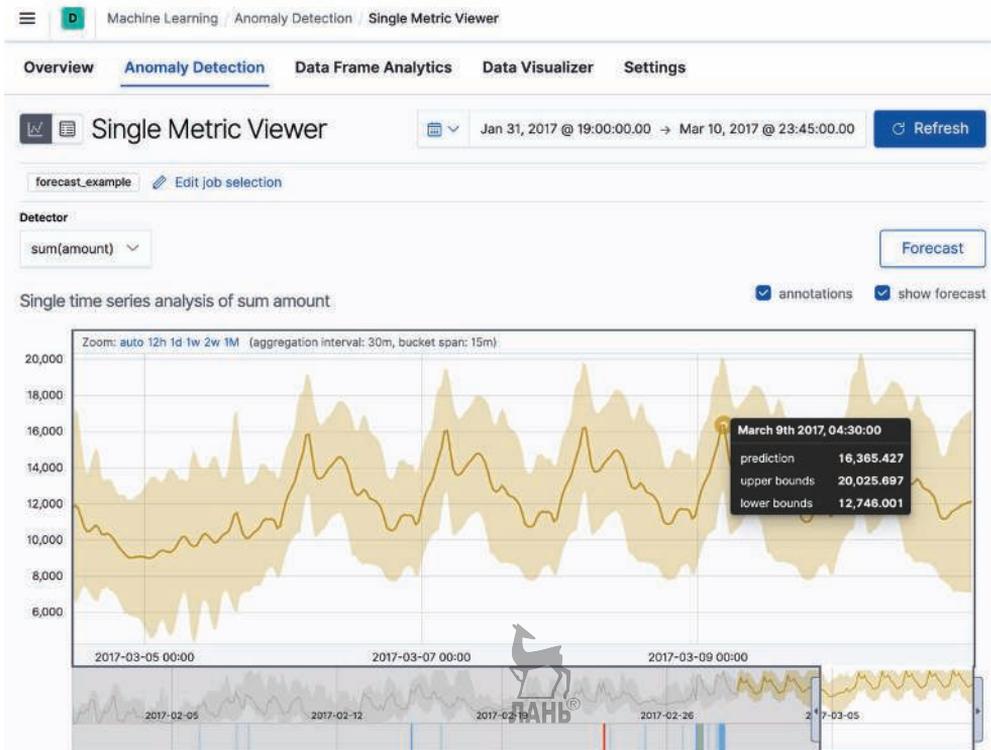


Рис. 5.26 ❖ Результаты прогноза, впервые показанные в главе 4

Напомним, что значение прогноза – это значение с наибольшим правдоподобием (вероятностью), а заштрихованная область – это диапазон 95-го перцентиля достоверности. Эти три ключевых значения хранятся в хранилищах результатов `.ml-anomalies-*` со следующими именами:

- `forecast_prediction;`
- `forecast_upper;`
- `forecast_lower.`

Запрос результатов прогноза

При запросе результатов прогноза в хранилищах результатов `.ml-anomalies-*` важно помнить, что результаты прогнозов являются временными – они имеют срок действия по умолчанию 14 дней после создания, особенно если они созданы из пользовательского интерфейса в Kibana. Если нужна другая продолжительность срока действия, тогда прогноз необходимо будет вызвать через конечную точку `API_forecast` и явно указать длительность `expires_in`.

Также следует помнить, что для одного и того же набора данных могут быть вызваны несколько прогнозов в разные моменты времени. Как было показано на рис. 4.4 и повторяется здесь (рис. 5.27), несколько вызовов дают несколько результатов прогнозов.

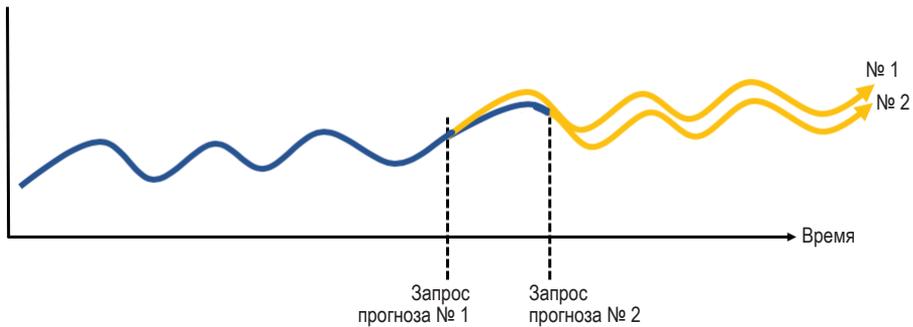


Рис. 5.27 ❖ Условная иллюстрация вызова нескольких прогнозов в разное время

Следовательно, нам нужен способ различать результаты. В пользовательском интерфейсе Kibana их можно различить, просто взглянув на дату создания (рис. 5.28).

Глядя на хранилище результатов, вы могли заметить, что каждый вызванный прогноз имеет уникальный `forecast_id` (идентификатор прогноза) (рис. 5.29).

Значение `forecast_id` видно только при вызове прогноза с использованием `API_forecast`, потому что это значение возвращается как часть полезной нагрузки вызова API.

Следовательно, если бы было создано несколько прогнозов, охватывающих общий период времени, было бы несколько результатов с разными идентификаторами.

Forecasting

Previous forecasts

Created	From	To	View
February 15th 2021, 14:41:05	March 10th 2017, 23:45:00	March 18th 2017, 00:45:00	View
February 9th 2021, 14:28:40	March 10th 2017, 23:45:00	March 25th 2017, 00:45:00	View

Run a new forecast

Duration

1d

Run

Length of forecast, up to a maximum of 3650 days. Use s for seconds, m for minutes, h for hours, d for days, w for weeks.

Close

Рис. 5.28 ❖ Просмотр нескольких ранее выполненных прогнозов

Discover

result_type:"model_forecast" and job_id:"forecast_example" KQL Mar 8, 2017 @ 06:51:49:255 → Apr 1, 2017 @ 13:56:30:101 Refresh

ml-anomalies-*

3,360 hits

Count

timestamp per 12 hours

TOP 5 VALUES

Source	Score
IOSp3cBpc7Wt6Mbvmdk	50.0%
Fm5E1Hc8pc7Wt6Mbvmdk	50.0%

job_id: forecast_example result_type: model_forecast forecast_id: Fm5E1Hc8pc7Wt6Mbvmdk bucket_span: 900 detector_index: 0 timestamp: Mar 25, 2017 @ 00:30:00.000 model_feature: bucket sum by person' forecast_lower: 0 forecast_upper: 16,896.55 forecast_prediction: 0,846.554 _id: forecast_example_model_forecast_IOSp3cBpc7Wt6Mbvmdk_1498416200886_988_0_0 _type: _doc _index: ml-anomalies-shared _score: -

Рис. 5.29 ❖ Просмотр результатов прогноза в Kibana Discover

При запросе результатов прогноза вы можете использовать одну из двух основных стратегий:

- **приоритет значения** – запрос предоставляет дату и время, и в результате возвращается конкретное значение для этого времени. Хорошим примером является вопрос: «Какой будет загруженность моего сервера через 5 дней?»;



- **приоритет времени** – запрос предоставляет значение, а результатом является время, в которое это значение реализуется. Хорошим примером является вопрос: «Когда загруженность моего сервера достигнет 80 %?»

Очевидно, что мы можем составить запросы того и другого типа. Например, чтобы реализовать запрос, ориентированный на время, нам нужно немного переформулировать запрос и попросить его вернуть дату (или даты), когда предсказанные значения соответствуют определенным критериям. Пользователь может запросить результаты прогноза, используя другие традиционные методы запроса (KQL, Elasticsearch DSL), но, чтобы продемонстрировать разнообразие методов, мы отправим запрос с помощью Elastic SQL в консоли Kibana Dev Tools:

```
POST _sql?format=txt
{
  "query": "SELECT forecast_prediction,timestamp FROM \".mlanomalies-*\"
WHERE job_id='forecast_example' AND forecast_
id='Fm5EiHcBpc7Wt6MbaGcw' AND result_type='model_forecast' AND
forecast_prediction>'16890' ORDER BY forecast_prediction DESC"
}
```

В этом запросе мы спрашиваем, есть ли моменты, в течение которых прогнозируемое значение превышает наше предельное значение 16 890. Ответ выглядит так:

```
forecast_prediction|      timestamp
-----+-----
16893.498325784924 | 2017-03-17T09:45:00.000Z
```

Другими словами, есть большая вероятность, что мы преодолеем порог 17 марта в 9:45 утра по GMT (хотя, как вы помните из главы 4, использованные образцы данных взяты из прошлого, и поэтому прогнозы тоже остались в прошлом). Теперь, когда у нас есть хорошее представление о том, как запрашивать результаты прогнозов, мы можем включать их в информационные панели и визуализации, о которых расскажем позже в этой главе, или даже в оповещения, как будет показано в главе 6.

Но прежде чем мы рассмотрим включение результатов в настраиваемые информационные панели и визуализации, давайте рассмотрим последнюю краткую тему – API результатов Elastic ML.

API РЕЗУЛЬТАТОВ ELASTIC ML



Если в дополнение к непосредственному запросу хранилищ результатов вам нужен доступ к результатам из своей программы, вы можете запросить API результатов Elastic ML. Некоторые части API повторяют то, что мы уже исследовали, а некоторые части уникальны. Далее мы рассмотрим API результатов более подробно.

Конечные точки API результатов

Вам доступны пять различных конечных точек API результатов:

- получение сегментов;
- получение факторов влияния;
- получение записей;
- получение обобщения сегментов;
- получение категорий.

Первые три конечные точки API дают результаты, которые являются избыточными в свете того, что мы уже рассмотрели в этой главе, путем прямого запроса хранилища результатов (через Kibana или с помощью API `Elasticsearch _search`), и доступ через API лишь обеспечивает большую гибкость, поэтому мы не будем их здесь обсуждать. Однако последние две конечные точки API являются новыми, и каждая заслуживает объяснения.

API обобщения сегментов



Вызов API обобщения сегментов – это средство, с помощью которого программным способом возвращаются сводные результаты множества заданий по обнаружению аномалий. Мы не станем исследовать каждый аргумент тела запроса и описывать каждое поле в теле ответа, поскольку вы всегда можете воспользоваться документацией. Но мы обсудим здесь важную функцию этого вызова API, который должен запросить результаты произвольного количества заданий и получить обобщенную оценку результата (называемую `overall_score`), включающую среднее значение `top_n` максимальной оценки `anomaly_score` сегмента для каждого запрошенного задания. В документации приводится пример вызова, который запрашивает два верхних задания (в наборе заданий, которые начинаются с имени `job-`), чья оценка аномалии сегмента при усреднении превышает `50,0`, начиная с определенной временной метки:

```
GET _ml/anomaly_detectors/job-*/results/overall_buckets
```

```
{
  "top_n": 2,
  "overall_score": 50.0,
  "start": "1403532000000"
}
```

Ответ на этот запрос выглядит так:

```
{
  "count": 1,
  "overall_buckets": [
    {
      "timestamp" : 1403532000000,
      "bucket_span" : 3600,
      "overall_score" : 55.0,
      "jobs" : [
        {
```



```

    "job_id" : "job-1",
    "max_anomaly_score" : 30.0
  },
  {
    "job_id" : "job-2",
    "max_anomaly_score" : 10.0
  },
  {
    "job_id" : "job-3",
    "max_anomaly_score" : 80.0
  }
],
"is_interim" : false,
"result_type" : "overall_bucket"
}
]
}

```



Обратите внимание, что в этом случае `overall_score` является средним из двух наивысших оценок (результат `overall_score 55,0` является средним значением для задания `job-3`, равного `80,0`, и для задания `job-1`, равного `30,0`), даже несмотря на то, что три задания по обнаружению аномалий соответствуют шаблону запроса `job-*`. Хотя это, безусловно, интересно, например для создания составного оповещения, вы должны осознавать ограничения такого отчета, особенно если вы можете получить доступ только к оценке аномалии на уровне сегмента, но не на уровне записи или фактора влияния. В главе 6 мы рассмотрим некоторые варианты составных оповещений.

API категорий

Вызов API категорий актуален только для заданий, использующих категоризацию, как подробно описано в главе 3. API категорий возвращает некоторые интересные внутренние определения категорий, обнаруженные в ходе текстового анализа документов. Если мы запустим API для задания категоризации, которое создали еще в главе 3 (сокращенное, чтобы для краткости возвращать только одну запись), результат будет следующим:

```

GET _ml/anomaly_detectors/secure_log/results/categories
{
  "page":{
    "size": 1
  }
}

```

Мы увидим такой ответ:

```

{
  "count" : 23,
  "categories" : [
    {

```



```

    "job_id" : "secure_log",
    "category_id" : 1,
    "terms" : "localhost sshd Received disconnect from port",
    "regex" :
".*?localhost.+?sshd.+?Received.+?disconnect.+?from.+?port.*",
    "max_matching_length" : 122,
    "examples" : [
      "Oct 22 15:02:19 localhost sshd[8860]: Received
disconnect from 58.218.92.41 port 26062:11: [preauth]",
      "Oct 22 22:27:20 localhost sshd[9563]: Received
disconnect from 178.33.169.154 port 53713:11: Bye [preauth]",
      "Oct 22 22:27:22 localhost sshd[9565]: Received
disconnect from 178.33.169.154 port 54877:11: Bye [preauth]",
      "Oct 22 22:27:24 localhost sshd[9567]: Received
disconnect from 178.33.169.154 port 55723:11: Bye [preauth]"
    ],
    "grok_pattern" : ".*? %{SYSLOGTIMESTAMP:timestamp}.+
?localhost.+?sshd.+? %{NUMBER:field}.+?Received.+?
disconnect.+?from.+? %{IP:ipaddress}.+?port.+? %{NUMBER:field2}.+
? %{NUMBER:field3}.*",
    "num_matches" : 595
  }
]
}

```

Ответ состоит из нескольких элементов:

- `category_id` – это номер категории сообщения (начинается с 1). Он соответствует значению поля `category` в хранилище результатов;
- `terms` – это список статических, неизменяемых слов, извлеченных из сообщения;
- `examples` – массив полных неизмененных строк журнала выборки, которые попадают в эту категорию. Они используются, чтобы показать пользователям, как выглядят некоторые из реальных строк журнала;
- `grok_pattern` – шаблон соответствия в стиле регулярного выражения, которое можно использовать для Logstash или входного конвейера, который можно использовать для сопоставления этой категории сообщений;
- `num_matches` – количество раз, когда эта категория сообщений была замечена в журналах на протяжении всего задания по обнаружению аномалий, выполняемого для этого набора данных.

Возможно, наиболее интересным использованием этого API является не обнаружение аномалий, а простое исследование уникального количества типов категорий и распределения этих типов в ваших неструктурированных журналах, чтобы ответить на такие вопросы, как «Какие типы сообщений находятся в моих журналах и сколько записей каждого типа?». Некоторые из этих возможностей пригодятся в будущем для создания процесса «подготовки данных», чтобы упростить пользователям загрузку неструктурированных журналов в Elasticsearch.

Давайте теперь исследуем, как результаты, полученные из заданий по обнаружению аномалий и прогнозированию в Elastic ML, могут быть использованы в настраиваемых панелях мониторинга, визуализациях и рабочих панелях Canvas.

ПОЛЬЗОВАТЕЛЬСКИЕ ПАНЕЛИ МОНИТОРИНГА И РАБОЧИЕ ПАНЕЛИ CANVAS

Очевидно, что теперь, когда вы знаете все тонкости устройства хранилища результатов, в котором хранятся все полезные вещи, полученные при обнаружении аномалий и прогнозной аналитике Elastic ML, ваши возможности ограничены только вашим воображением, и вы можете представить эти результаты в соответствии со своими целями. В этом разделе кратко представлены некоторые концепции и идеи, которые вы можете использовать, чтобы вывести результаты Elastic ML на большой экран.

Панель инструментов встраивания

Одним из недавних дополнений к возможностям Elastic ML является возможность встраивать временную шкалу Anomaly Explorer (обзорная полоса) в существующие настраиваемые информационные панели. Для этого просто нажмите на символ меню «три точки» в правом верхнем углу **Anomaly timeline** (Шкала времени аномалии) и выберите параметр **Add to dashboard** (Добавить на панель управления) (рис. 5.30).



Рис. 5.30 ❖ Добавление шкалы времени аномалии на другую панель мониторинга

На этом этапе выберите, какую часть видов обзорных полос вы хотите включить и на какие панели мониторинга вы хотите их добавить (рис. 5.31).

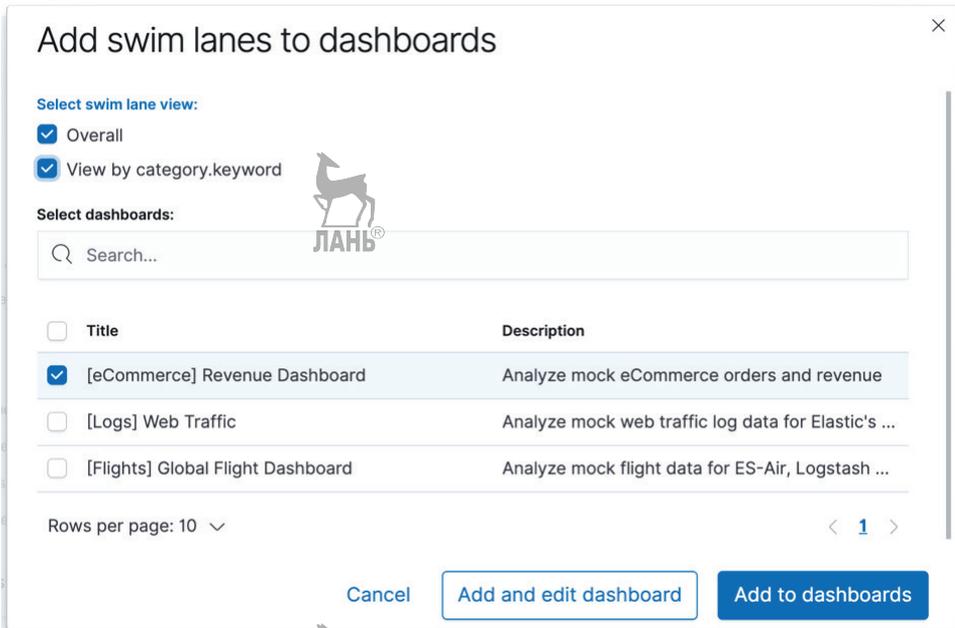


Рис. 5.31 ❖ Добавление шкалы времени аномалии на конкретную панель управления

Нажатие кнопки **Add and edit dashboard** (Добавить и редактировать панель мониторинга) перенесет вас на целевую панель мониторинга, где вы сможете перемещать и изменять размер встроенных панелей. Например, можно расположить аномалии рядом с другими визуализациями, как показано на рис. 5.32.

Аномалии как аннотации в TSVB

Компонент **Time Series Visual Builder** (TSVB, визуальный построитель временных рядов) в Kibana – это чрезвычайно гибкий конструктор визуализаций, который позволяет пользователям не только отображать данные своих временных рядов, но и аннотировать эти данные с помощью информации из других хранилищ. Это идеальный рецепт для отображения необработанных данных, а затем наложения аномалий из задания по обнаружению аномалий поверх этих данных. Например, вы можете создать TSVB для `kibana_sample_data_logs`, как показано на рис. 5.33.



Рис. 5.32 ❖ Новая панель управления, теперь содержащая визуализации аномалий в обзорных полосах

The screenshot shows the 'Panel options' configuration interface in Canvas. At the top, there is an 'Auto apply' toggle and a note: 'The changes will be automatically applied.' Below this, the 'Data' section is expanded, showing the following settings:

- Index pattern:** `kibana_sample_data_logs`
- Time field:** `@timestamp`
- Interval:** `15m` (Examples: auto, 1m, 1d, 7d, 1y, >=1m)
- Drop last bucket?:** Yes No

The 'Panel filter' section includes a search input field, a 'KQL' button, and an 'Ignore global filter?' option with Yes and No.

Рис. 5.33 ❖ Создание новой визуализации TSVB – параметры Panel

Кроме того, существует конфигурация на вкладке **Data** (Данные) для агрегирования терминов для пяти ведущих стран происхождения (`geo.src`).

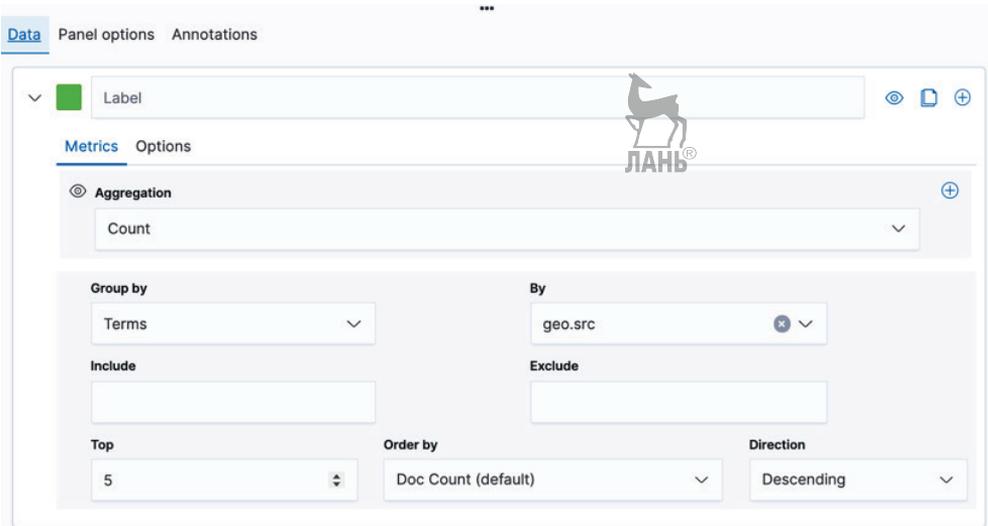


Рис. 5.34 ❖ Создание новой визуализации TSVB – параметры **Data**

Далее в нашем распоряжении конфигурация на вкладке **Annotations** (Аннотации), позволяющая наложить результаты ранее созданного задания по обнаружению аномалий с именем `web_traffic_per_country` и выбрать аномалии с рекордным количеством баллов более 90.



Рис. 5.35 ❖ Создание новой визуализации TSVB – параметры **Annotations**

Обратите внимание, что поле **Query string** (Строка запроса) выглядит довольно знакомым, учитывая то, о чем мы говорили в этой главе. TSVB также требует список разделенных запятыми полей для аннотаций (здесь мы вво-

дим `record_score` и `partition_field_value`) и **Row template** (Шаблон строки), который определяет форматирование аннотации (здесь мы вводим `Anomaly: {{record_score}} for {{partition_field_value}}`). Как только это будет сделано, вы получите окончательный результат (рис. 5.36).

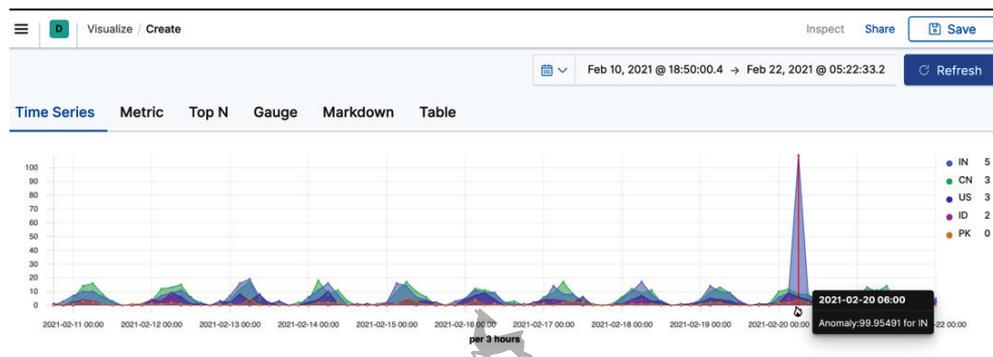


Рис. 5.36 ❖ Создание новой визуализации TSVB с аннотациями аномалий

Теперь у нас есть красивая панель визуализации с аномалиями, наложенными на исходные необработанные данные.

Настройка рабочих панелей Canvas

Kibana Canvas – это идеальный инструмент для создания идеально точной инфографики на основе данных из Elasticsearch. Вы можете создавать индивидуализированные отчеты с набором настраиваемых элементов. Работа с Canvas сильно отличается от стандартных информационных панелей Kibana. Canvas представляет собой рабочую область – так называемую *рабочую панель* (workpad), в которой вы можете создавать наборы слайдов, как в Microsoft PowerPoint.

Чтобы использовать обнаружение аномалий и/или результаты прогнозирования на рабочем столе Canvas, не нужно делать ничего особенного – вы уже изучили все, что нужно. Это связано с тем, что в Canvas очень легко использовать команду `essql` для запроса шаблона хранилища `.ml-anomalies-*` и извлечения важной информации.

Когда вы устанавливаете образцы данных Kibana, вместе с ними вы также получаете несколько образцов рабочих панелей Canvas (рис. 5.37).

Щелкнув по образцу рабочей панели **[Logs] Web Traffic** ([Журналы] Веб-трафик), вы откроете его для редактирования (рис. 5.38).

Если выбрать один из элементов на странице (допустим, счетчик **TOTAL VISITORS** (ВСЕГО ПОСЕТИТЕЛЕЙ) внизу, который в настоящее время показывает значение 324), а затем выбрать **Expression editor** (Редактор выражений), то в правой части экрана Canvas покажет детали элемента (рис. 5.39).

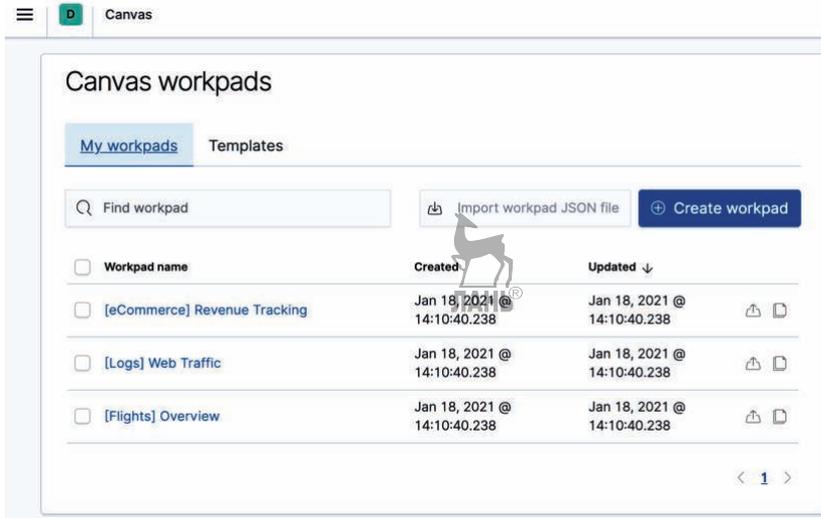


Рис. 5.37 ❖ Образцы рабочих панелей Canvas



Рис. 5.38 ❖ Пример рабочей панели веб-трафика

Настоящая «магия» получения данных в реальном времени заложена в команде `essql`, а остальная часть выражения – это просто форматирование. В качестве простого примера мы можем записать SQL следующим образом:

```
SELECT COUNT(timestamp) as critical_anomalies FROM \"m anomalies-*\"
WHERE job_id='web_logs_rate' AND result_
type='record' AND record_score>'75'
```

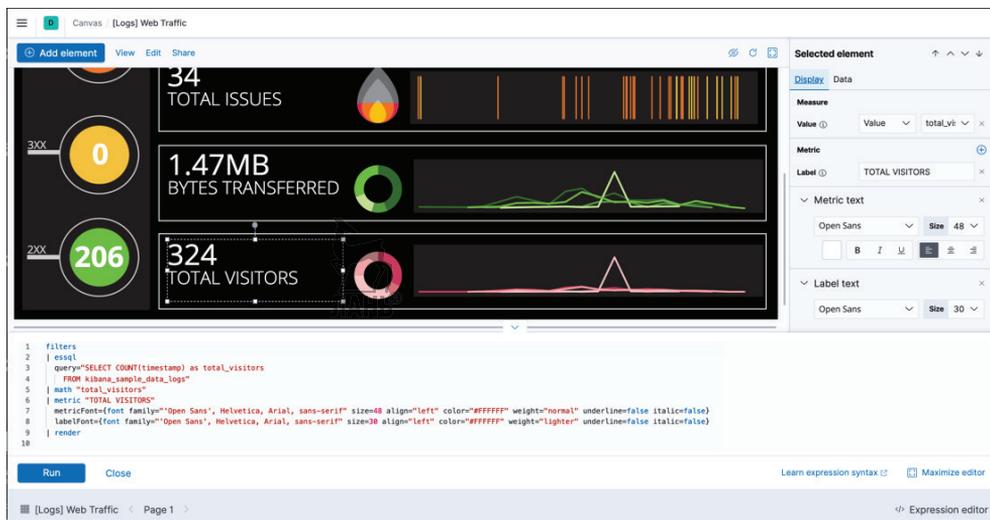


Рис. 5.39 ❖ Редактирование элемента Canvas в редакторе выражений

Следует отметить, что, поскольку имя шаблона хранилища `.ml-anomalies-*` начинается с неалфавитного символа, имя должно быть заключено в двойные кавычки, а эти двойные кавычки необходимо экранировать с помощью символа обратной косой черты.

Этот запрос вернет общее количество критических аномалий (тех, у которых `record_score` больше 75) для конкретного задания по обнаружению аномалий в этом наборе данных (рис. 5.40).

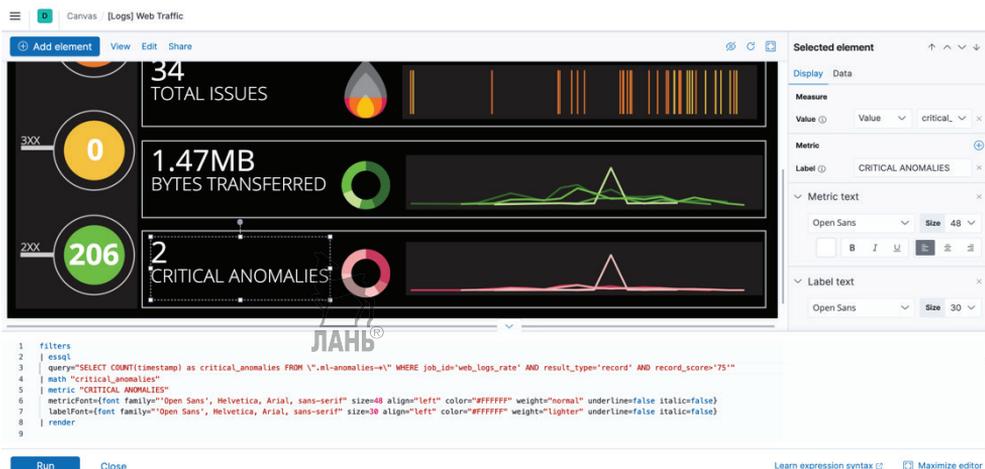


Рис. 5.40 ❖ Отображение количества критических аномалий

Короче говоря, вам не составит труда использовать Canvas для создания очень красивых и значимых визуализаций данных на основе информации либо из результатов обнаружения аномалий, либо из результатов прогноза.

ЗАКЛЮЧЕНИЕ

Механизмы обнаружения аномалий и прогнозирования Elastic ML создают впечатляющие и полезные результаты, которые можно исследовать с помощью богатого пользовательского интерфейса, предоставляемого в Kibana, или программно с помощью прямого обращения к хранилищам результатов через API. Понимание результатов ваших заданий по обнаружению и прогнозированию аномалий и возможность надлежащим образом использовать эту информацию для дальнейших настраиваемых визуализаций или оповещений делает эти настраиваемые ресурсы еще более мощными.

В следующей главе мы воспользуемся результатами для создания сложных и полезных упреждающих оповещений, чтобы еще больше повысить прикладную ценность Elastic ML.



Глава 6

.....

Создание и использование оповещений



В предыдущей главе говорилось о том, как результаты обнаружения и прогнозирования аномалий сохраняются в хранилищах Elasticsearch. В этой главе мы расскажем о создании информативных и прогнозных оповещений на основе полученных ранее результатов анализа.

На момент написания данной книги Elastic Stack находился на переломном этапе развития. В течение нескольких лет Elastic ML полагался на Watcher (компонент Elasticsearch), поскольку это был эксклюзивный механизм оповещения. Однако начиная с версии 7.11 введена новая платформа оповещения, работающая как часть Kibana, и в дальнейшем она будет основным механизмом оповещения.

Есть несколько интересных функций Watcher, которые пока не реализованы в механизме оповещений Kibana. Поэтому в данной главе мы продемонстрируем создание оповещений с использованием средств Kibana и Watcher. В зависимости от ваших потребностей вы можете решить, что из них лучше использовать.

В частности, в этой главе будут рассмотрены следующие темы:

- определение и принцип работы оповещений;
- создание оповещений из пользовательского интерфейса ML;
- создание оповещений с помощью Watcher.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

В этой главе мы будем использовать Elastic Stack в том виде, в котором он существует в версии 7.12.

ОПРЕДЕЛЕНИЕ И ПРИНЦИП РАБОТЫ ОПОВЕЩЕНИЙ

Надеюсь, мы не будем выглядеть слишком педантичными и скучными, если начнем главу с некоторых соображений, которые мы считаем чрезвычайно важными для понимания механизма оповещений, а затем перейдем к настройке и использованию этих оповещений.

Аномалии не обязательно нуждаются в оповещениях

Об этом нужно сказать прямо. Часто пользователи, которые научились обнаруживать аномалии, чувствуют себя обязанными предупреждать обо всем, что удалось заметить. Это стремление может создать проблемы, если обнаружение аномалий развернуто на сотнях, тысячах или даже десятках тысяч объектов. Инструменты обнаружения аномалий, безусловно, освобождают пользователей от необходимости определять конкретные, управляемые правилами исключения или жестко заданные пороговые значения оповещений, но при этом они способствуют раздуванию объема генерируемых данных при расширении масштаба использования. Вы должны понимать, что подробное оповещение о каждой маленькой аномалии может стать источником мешающего информационного шума, если не проявить разумную осторожность.

К счастью, в главе 5 вы познакомились с подходами, которые помогают нам исправить ситуацию:

- **обобщение:** вы узнали, что аномалии не только попадают в отчет по отдельности (на уровне записи), но также суммируются на уровне сегмента и на уровне фактора влияния. Эти итоговые оценки дают нам возможность генерировать оповещение на более высоком уровне абстракции, если мы того пожелаем;
- **нормализованная оценка:** поскольку каждое задание по обнаружению аномалий имеет настраиваемую шкалу нормализации, специально созданную для конкретной конфигурации детектора и анализируемого набора данных, это означает, что мы можем использовать нормализованную оценку, получаемую из Elastic ML, для ограничения частоты появления типичных оповещений. Возможно, для конкретного задания, которое вы создали, оповещение с минимальной оценкой аномалии 10 обычно дает около дюжины оповещений в день, оценка 50 дает около одного оповещения в день, а оценка 90 дает около одного оповещения в неделю. Другими словами, вы можете настроить оповещение в соответствии с вашими требованиями к количеству оповещений, которые вы предпочитаете получать за единицу времени (конечно, за исключением неожиданного сбоя в масштабах всей системы, который может создать больше оповещений, чем обычно);
- **корреляция/комбинация:** возможно, аномалия одной метрики (чрезмерно высокий уровень нагрузки на процессор) выглядит не столь убедительно, как группа связанных аномалий (нагрузка растет, объем

свободной памяти падает, время отклика увеличивается). В некоторых ситуациях оповещение о составных событиях или последовательностях событий может быть более значимым.

Суть в том, что, даже несмотря на отсутствие универсальной стратегии относительно наилучшего способа структурирования оповещений и повышения их эффективности, пользователю доступны варианты, позволяющие выбрать наиболее подходящее решение именно для него.

Точное время имеет значение

Еще в главе 2 вы узнали, что задания по обнаружению аномалий представляют собой относительно сложную оркестровку запроса необработанных данных, анализа этих данных и представления результатов в виде непрерывного процесса, который может выполняться почти в реальном времени. Поэтому существуют несколько ключевых аспектов конфигурации задания, которые определяют частоту этого процесса, а именно параметры `bucket_span`, `frequency` и `query_delay`. Эти параметры определяют, когда результаты «готовы» и какую метку времени получают значения. Это чрезвычайно важно, потому что оповещение о результатах обнаружения аномалий будет включать последующий запрос к хранилищам результатов (`.ml-anomalies-*`), и очевидно, что от того, когда этот запрос выполняется и какой временной диапазон используется, зависит, действительно ли вы обнаружите аномалии, которые ищете. Эти соображения проиллюстрированы на рис. 6.1.

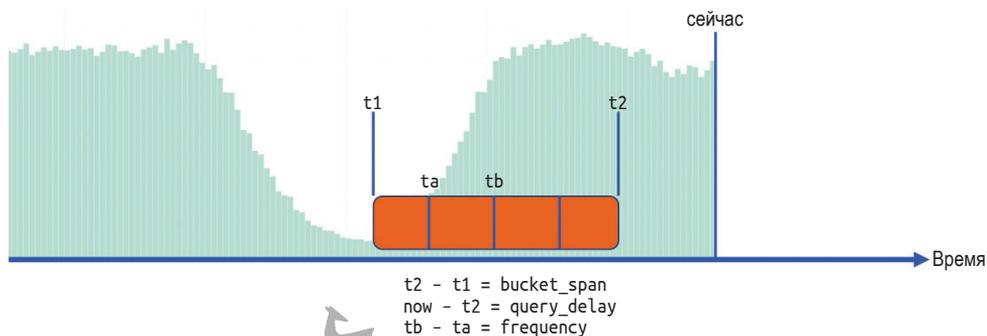


Рис. 6.1 ❖ Представление ширины сегмента, задержки запроса и частоты по отношению к текущему моменту

На рис. 6.1 мы видим, что конкретный период времени (представленный интервалом, равным $t_2 - t_1$) отстает от текущего системного времени (сейчас) на величину, равную `query_delay`. Внутри сегмента могут быть шаги времени, которые определены параметром `frequency`. Что касается того, как результаты этого сегмента записываются в хранилища результатов `.ml-anomalies-*`, вы должны помнить, что метка времени всех документов, написанных для этого сегмента, будет иметь значение метки времени, равное времени в момент t_1 – передний край сегмента.

В качестве практического примера рассмотрим следующие значения:

- `bucket_span` = 15 мин;
- `frequency` = 15 мин;
- `query_delay` = 2 мин.

Если сейчас 12:05, то сегмент, соответствующий интервалу 11:45–12:00, был запрошен и обработан Elastic ML примерно в 12:02 (из-за задержки `query_delay`), и вскоре после этого готовый документ был записан в `.ml-anomalies-*` (но записан с меткой времени, равной 11:45). Следовательно, если в 12:05 мы заглянем в `.ml-anomalies-*`, чтобы увидеть результаты на 11:45, мы можем быть уверены, что они существуют, и мы могли бы изучить содержимое. Однако если сейчас только 12:01, то готового документа для сегмента, соответствующего интервалу 11:45–12:00, пока не существует, и он появится только спустя еще минуту или около того. Как видите, выбор времени очень важен.

Если в нашем примере сценария мы уменьшим значение `frequency` до 7,5 или 5 мин, то у нас действительно «раньше» появится доступ к результатам сегмента, но результаты будут помечены как *промежуточные* (*interim*) и могут измениться после завершения обработки сегмента.

❗ Промежуточные результаты создаются в пределах сегмента, если значение `frequency` является кратным значению `bucket_span`, но не все детекторы дают промежуточные результаты. Например, если у вас есть детектор `max` или `high_count`, то промежуточный результат, который показывает значение выше ожидаемого по сравнению с типичным значением, возможен и разумен – вам не нужно видеть содержимое всей корзины, чтобы знать, что вы уже превысили ожидания. Однако если вы используете детектор `mean`, вам придется получить весь объем наблюдений в сегменте, прежде чем определять среднее значение, поэтому промежуточные результаты не имеют смысла и не выдаются.

Итак, с учетом сказанного, если мы теперь возьмем диаграмму на рис. 6.1 и немного увеличим время, но также нарисуем сегменты до и после упомянутого сегмента, новая диаграмма будет выглядеть, как показано на рис. 6.2.

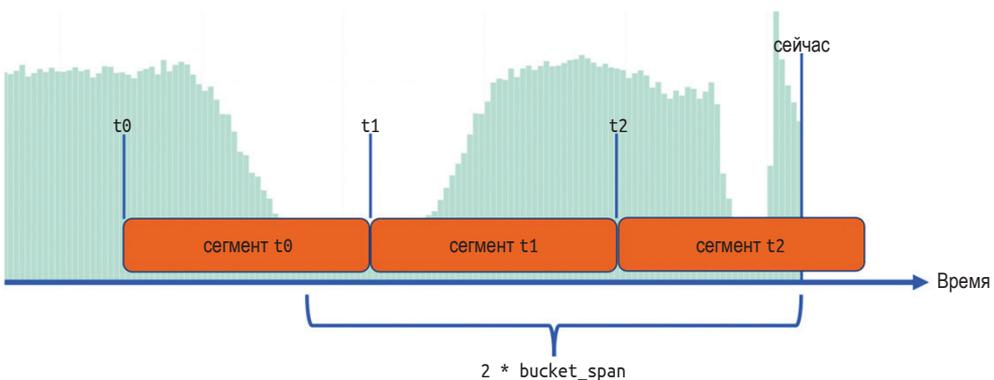


Рис. 6.2 ❖ Представление последовательных сегментов

На рис. 6.2 мы видим, что текущее системное время (обозначенное словом «сейчас») находится примерно в середине сегмента t_2 – следовательно, сегмент t_2 еще не завершен, и если есть какие-либо результаты, записанные для сегмента t_2 заданием по обнаружению аномалий, они будут отмечены флагом `is_interim:true`, как было показано в главе 5.

Если мы сделаем запрос оповещения, который отвечает на вопрос «Найдены ли какие-либо новые аномалии с тех пор, как мы проверяли результаты в последний раз?», и время этого запроса соответствует текущему системному времени «сейчас», как показано на рис. 6.2, то можно сделать следующие выводы:

- период «оглядки назад» должен быть примерно вдвое больше `bucket_span`. Это гарантирует, что мы увидим любые промежуточные результаты, которые могут быть опубликованы для текущего сегмента (здесь сегмент t_2), и любые окончательные результаты для предыдущего сегмента (здесь сегмент t_1). Результаты из сегмента t_0 не будут представлены, потому что метка времени для сегмента t_0 находится за пределами запрашиваемого окна времени – это нормально, если мы делаем запрос оповещения, который будет повторяться по правильно-му расписанию (как сказано в следующем пункте);
- время, выбранное для выполнения этого запроса, может попасть практически в любое место в пределах временного окна сегмента t_2 , и запрос все равно будет работать, как описано. Это важно, потому что расписание, по которому запускается запрос оповещения, скорее всего, не будет синхронизировано с расписанием, по которому выполняются задание обнаружения аномалий и запись результатов;
- скорее всего, вы запланируете поиск оповещений так, чтобы он срабатывал не реже, чем с интервалом, равным `bucket_span`, но он может выполняться и чаще, если мы заинтересованы в выявлении промежуточных аномалий в текущем, еще не завершенном сегменте;
- если вам не нужны промежуточные результаты, достаточно изменить запрос так, чтобы он включал условие `is_interim:false`.

У вас могло сложиться впечатление, что для правильной и надежной работы запроса оповещения не обойтись без навыков темной магии. К счастью, когда мы создаем оповещения с помощью Kibana из пользовательского интерфейса Elastic ML, вся необходимая работа будет сделана за вас.

Однако если вы чувствуете себя волшебником и полностью понимаете, как все это работает, то, возможно, вас не слишком пугает перспектива создания тонко настраиваемых условий оповещения с помощью Watcher, где у вас будет полный контроль.

В следующих разделах мы рассмотрим несколько примеров с использованием каждого метода, чтобы вы могли сравнить их работу.

СОЗДАНИЕ ОПОВЕЩЕНИЙ ИЗ ИНТЕРФЕЙСА МАШИННОГО ОБУЧЕНИЯ

С выпуском v7.12 Elastic ML изменил свой обработчик оповещений по умолчанию с Watcher на оповещения Kibana. До версии 7.12 у пользователя был выбор: принять опцию по умолчанию **watch** (экземпляр сценария для Watcher), если в пользовательском интерфейсе ML было выбрано оповещение, или самостоятельно создать объект **watch** с нуля. Сейчас мы уделим основное внимание новому рабочему процессу с использованием оповещений Kibana версии 7.12, который предлагает хороший баланс гибкости и простоты использования.

В качестве наглядного примера оповещения в реальном времени мы разработаем сценарий для набора данных веб-журналов Kibana, с которым познакомились в главе 3.

Мы должны пройти через три основных шага:

- 1) определяем несколько заданий по обнаружению аномалий для пробных данных;
- 2) определяем два оповещения для двух заданий по обнаружению аномалий;
- 3) запускаем моделирование аномального поведения, чтобы зафиксировать это поведение в оповещении.

Давайте сначала определим задания по обнаружению аномалий.

Определение заданий по обнаружению аномалий

Очевидно, что, прежде чем мы сможем создавать оповещения, нам нужны задания, выполняемые в режиме реального времени. Мы можем использовать примеры заданий машинного обучения, которые поставляются вместе с набором данных веб-журналов Kibana.

! Если у вас уже есть этот набор данных, загруженный в ваш кластер, вы должны удалить его и снова добавить. Это приведет к сбросу отметок времени в наборе данных, так что примерно половина данных будет в прошлом, а остальные – в будущем. Наличие некоторых данных в будущем позволит нам представить, будто данные появляются в реальном времени, и поэтому наши задания по обнаружению аномалий в реальном времени и наши оповещения об этих заданиях будут действовать так, как будто они действительно работают в режиме реального времени.

Для начала давайте перезагрузим образцы данных и создадим несколько пробных заданий.

1. На главном экране Kibana нажмите ссылку **Try our sample data** (Попробовать наши образцы данных) (рис. 6.3).

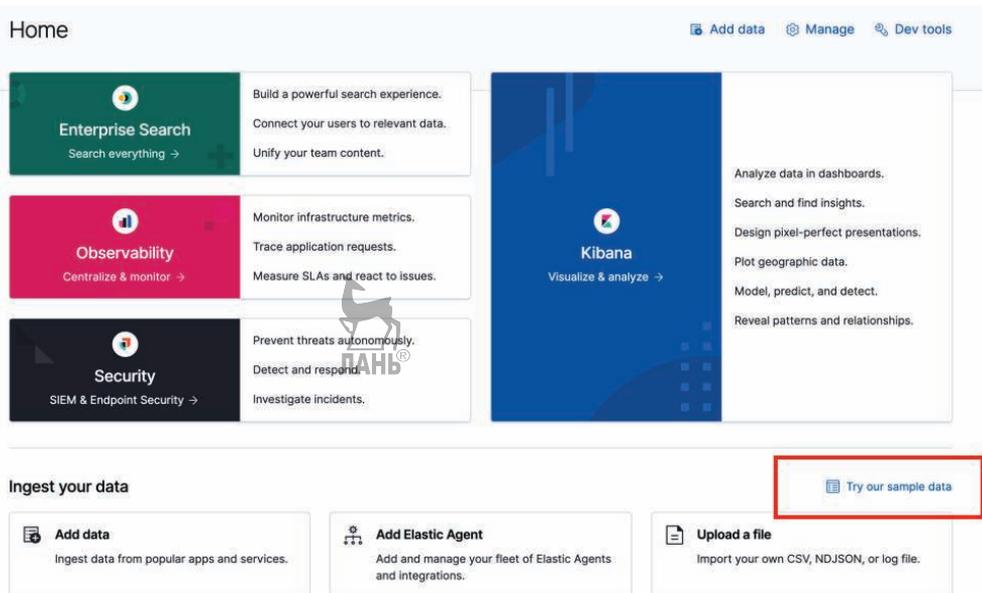


Рис. 6.3 ❖ Главный экран Kibana

- Щелкните **Index Patterns** (Шаблоны хранилища) в разделе **Sample web logs** (Образцы веб-журналов). Если они уже загружены, удалите и снова добавьте их (рис. 6.4).

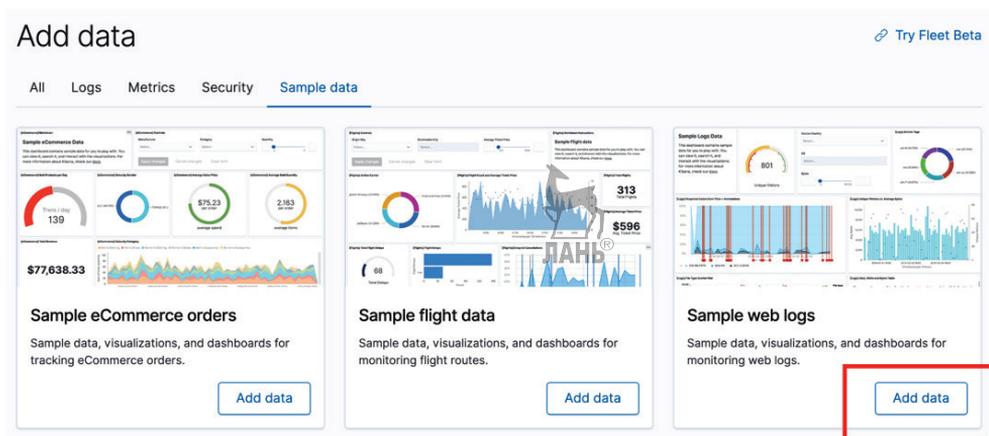


Рис. 6.4 ❖ Добавление примеров данных веб-журналов

- В меню **View data** (Просмотр данных) выберите **ML jobs** (Задания машинного обучения), чтобы создать несколько примеров заданий (рис. 6.5).

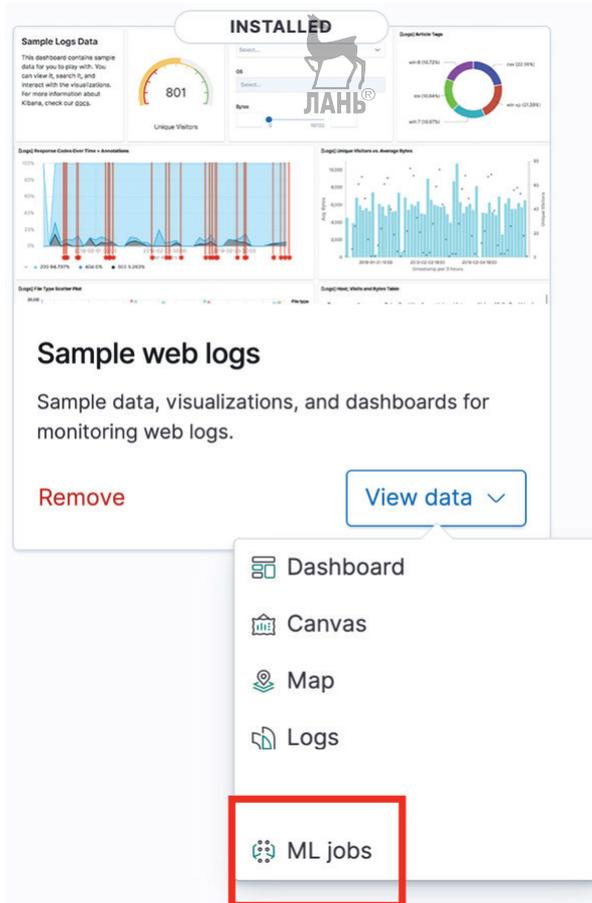


Рис. 6.5 ❖ Создание нескольких пробных заданий машинного обучения

4. Дайте трем примерам заданий префикс идентификатора задания (здесь был выбран `alert-demo-`) и убедитесь, что вы сняли флажок **Use full kibana_sample_data_logs data** (Использовать полные данные `kibana_sample_data_logs`), затем выберите время окончания не позднее 15 минут относительно текущего системного времени в вашем часовом поясе (рис. 6.6).



New job from index pattern kibana_sample_data_logs

Рис. 6.6 ❖ Присвоение префиксов имен примерам заданий и выбор текущего системного времени в качестве времени окончания

- Обратите внимание на рис. 6.6, что **8 апреля 2021 г. @ 11:00:00.00** было выбрано в качестве времени окончания, а дата на 11 дней раньше (**28 марта 2021 @ 00:00:00.00**) была выбрана в качестве времени начала (данные примера охватывают 11 дней с момента его установки). Текущее местное время на момент создания этого снимка экрана было 11:10 утра 8 апреля 2021 г. Это важно в том смысле, что мы пытаемся заставить эти образцы данных выглядеть так, будто они собраны в реальном времени. Нажмите кнопку **Create Jobs** (Создать задание), чтобы начать создание задания. Как только задание будет создано, вы увидите следующий экран (рис. 6.7).

	Job	Datafeed	Running
alert-demo-low_request_rate Find unusually low request rates kibana_sample_data kibana_sample_web_logs	✓	✓	✓
alert-demo-response_code_rates Find unusual event rates by HTTP response code (high and low) kibana_sample_data kibana_sample_web_logs	✓	✓	✓
alert-demo-uri_scanning Find client IPs accessing an unusually high distinct count of URLs kibana_sample_data kibana_sample_web_logs	✓	✓	✓

Рис. 6.7 ❖ Завершение инициализации примеров заданий

- Нам пока не нужно просматривать результаты. Вместо этого нам нужно убедиться, что эти три задания выполняются в реальном времени. На-

жмите ссылку **Anomaly Detection** (Обнаружение аномалий) в верхней строке меню, чтобы вернуться на страницу **Job Management** (Управление заданиями). Здесь мы видим, что наши три задания проанализировали некоторые данные, но теперь находятся в закрытом состоянии, а потоки данных в настоящее время остановлены (рис. 6.8).

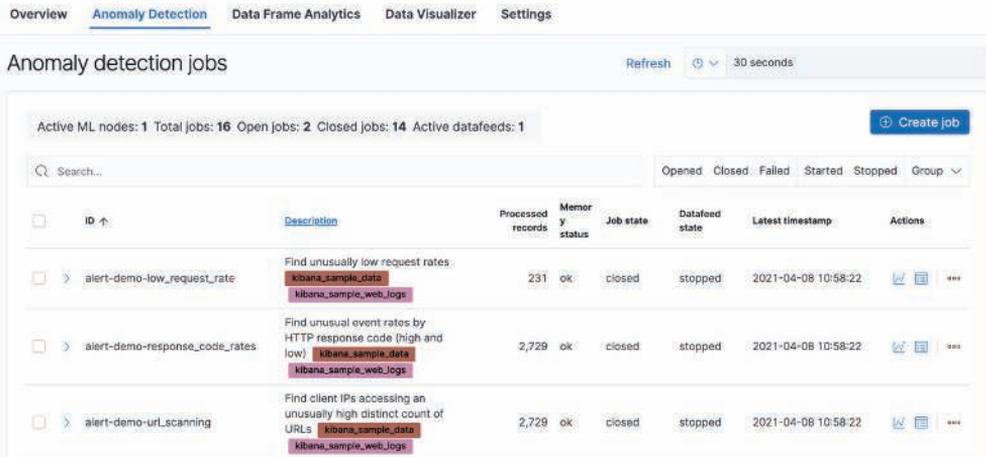


Рис. 6.8 ❖ Примеры заданий на экране управления заданиями

- Теперь нам нужно запустить эти три задания в реальном времени. Установите флажки рядом с каждым заданием, а затем щелкните значок шестеренки, чтобы открыть меню и выбрать **Start data feeds** (Запуск каналов данных) для всех трех заданий (рис. 6.9).

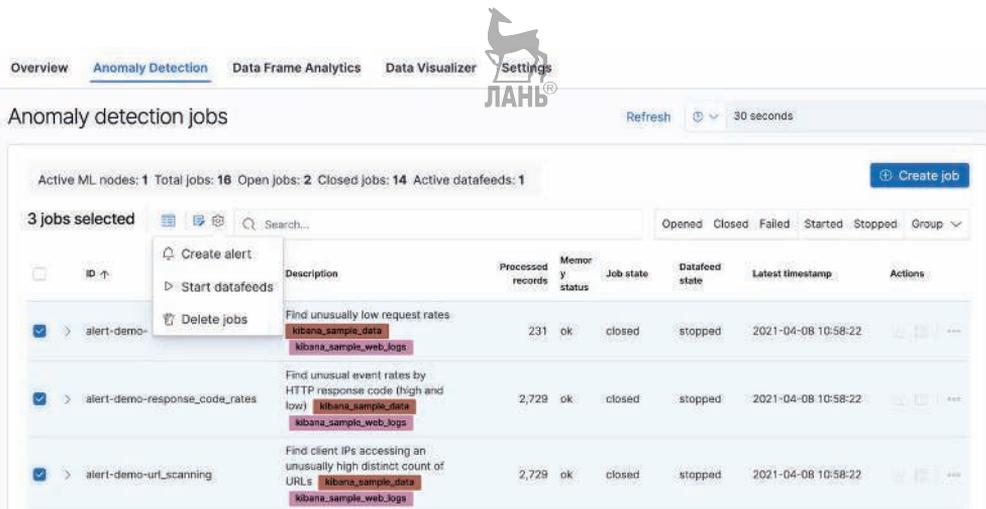


Рис. 6.9 ❖ Запуск потока данных для всех трех пробных заданий

8. Во всплывающем окне выберите верхнюю опцию списка для полей **Search start time** (Время начала поиска) и **Search end time** (Время окончания поиска), чтобы задание продолжало выполняться в реальном времени. Сейчас мы не станем выставлять флажок **Create alert after datafeed has started** (Создать оповещение после начала подачи данных), поскольку вскоре создадим свои собственные оповещения (рис. 6.10).

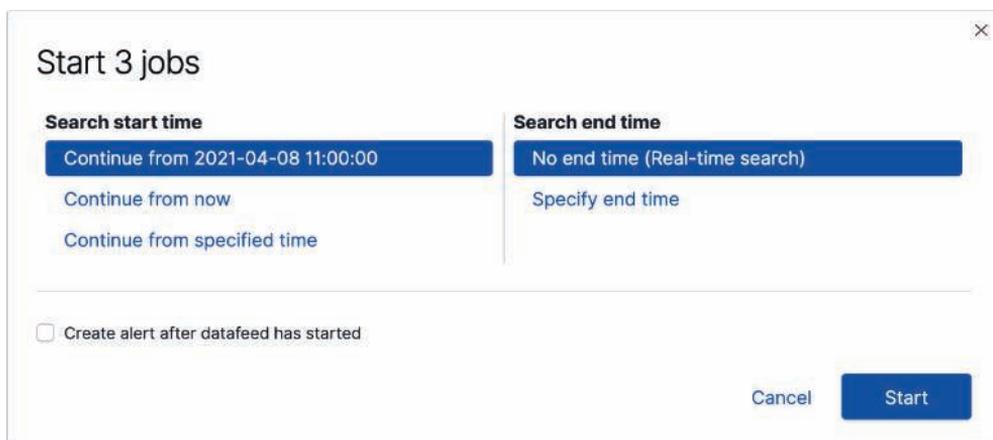


Рис. 6.10 ❖ Запуск потоков данных трех пробных заданий для запуска в реальном времени

9. После нажатия кнопки **Start** (Пуск) мы увидим, что три наших задания теперь находятся в открытом (работающем) состоянии (рис. 6.11).

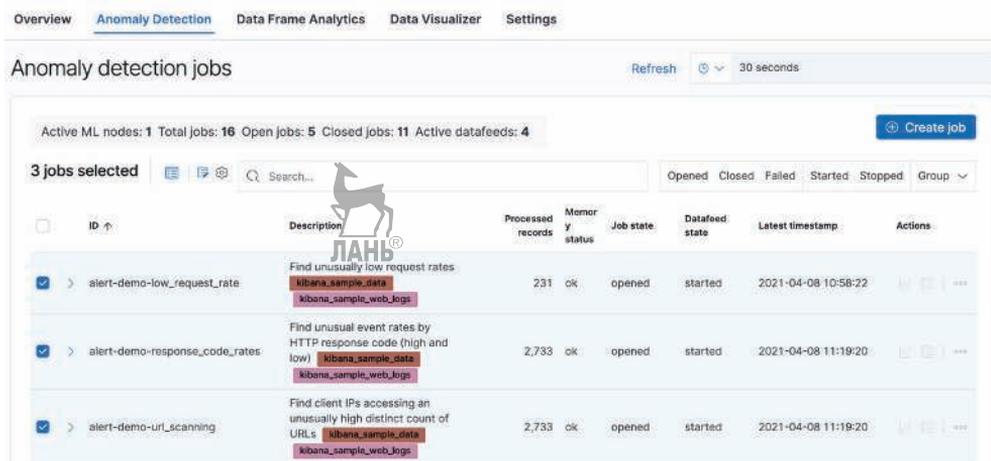


Рис. 6.11 ❖ Примеры заданий, которые теперь выполняются в режиме реального времени

Создав пробные задания и убедившись, что они работают, определим для них несколько оповещений.

Создание оповещений для пробных заданий

Теперь, когда наши задания выполняются в режиме реального времени, мы можем определить для них оповещения.

1. Для задания `alert-demo-response_code_rates` щелкните значок трюеточия «...», расположенный справа, и выберите опцию **Create alert** (Создать оповещение) (рис. 6.12).

The screenshot shows the 'Anomaly detection jobs' page in a web interface. At the top, there are navigation tabs: Overview, Anomaly Detection (selected), Data Frame Analytics, Data Visualizer, and Settings. Below the tabs, there's a 'Refresh' button and a dropdown set to '30 seconds'. A summary bar indicates: 'Active ML nodes: 1 Total jobs: 16 Open jobs: 5 Closed jobs: 11 Active datafeeds: 4'. A search bar is present. Below is a table with columns: ID, Description, Processed records, Memory status, Job state, Datafeed state, Latest timestamp, and Actions. Three jobs are listed:

ID	Description	Processed records	Memory status	Job state	Datafeed state	Latest timestamp	Actions
alert-demo-low_request_rate	Find unusually low request rates. kibana_sample_data kibana_sample_web_logs	231	ok	opened	started	2021-04-08	Stop datafeed
alert-demo-response_code_rates	Find unusual event rates by HTTP response code (high and low). kibana_sample_data kibana_sample_web_logs	2,733	ok	opened	started	2021-04-08	Create alert, Clone job, Edit job, Delete job
alert-demo-url_scanning	Find client IPs accessing an unusually high distinct count of URLs. kibana_sample_data kibana_sample_web_logs	2,733	ok	opened	started	2021-04-08	Delete job

Рис. 6.12 ❖ Создание оповещения для пробного задания

2. Далее появится всплывающее окно **Create alert**, и вы можете приступить к настройке желаемой конфигурации оповещения (рис. 6.13).
3. В соответствии с полями на рис. 6.13 нужно задать имя оповещению, а также указать, чтобы это оповещение проверяло наличие аномалий каждые 10 минут. Для этого задания интервал `bucket_span` установлен на 1 час, но параметр `frequency` установлен на 10 минут, поэтому промежуточные результаты будут доступны гораздо раньше, чем закончится сегмент. По этой же причине стоит включить промежуточные результаты в вашу конфигурацию оповещений, чтобы получить уведомление как можно скорее. В секции **Result type** (Тип результата) выберите **Bucket**, чтобы получить обобщенное описание аномальности, как обсуждалось ранее. Наконец, установите порог значимости **51**, чтобы оповещения генерировались только для аномалий с оценкой, превышающей это значение.

×

Create alert

Name

Tags (optional)

Check every ?

Notify ?

Anomaly detection alert

Alert when anomaly detection jobs results match the condition. [Documentation](#) ?


BETA

Select jobs or groups

 ✕

Result type

<p>Bucket</p> <p>How unusual was the job within the bucket of time?</p> <div style="background-color: #e0f2f1; padding: 5px; display: inline-block;">✓ Selected</div>	<p>Record</p> <p>What individual anomalies are present in a time range?</p> <div style="background-color: #e0e0e0; padding: 5px; display: inline-block;">Select</div>	<p>Influencer</p> <p>What are the most unusual entities in a time range?</p> <div style="background-color: #e0e0e0; padding: 5px; display: inline-block;">Select</div>
--	--	---

Select severity threshold

0

51

warning
 minor
 major
 critical

100

Include interim results

Cancel



✓ Save

Рис. 6.13 ❖ Создание конфигурации оповещения

4. Прежде чем продолжить, проверьте конфигурацию оповещений на прошлых данных. Введя значение 30d в проверочное поле, вы увидите, что за последние 30 дней было только одно оповещение, которое соответствовало заданному условию (рис. 6.14).

×

Create alert

✓ Selected

Select

Select

Select severity threshold

0

51

warning
minor
major
critical

100

Include interim results

Check the alert condition with an interval

Test

Triggers 1 time in the last 30d [Hide results](#)

Job IDs:
alert-demo-response_code_rates

Time:
2021-04-03T10:00:00.000Z

Anomaly score:
62

Top influencers:
response.keyword = 200 [79]

Top records:
count() 200 [82]

Actions

✓ slack alert
⊖

Cancel

✓ Save

Рис. 6.14 ❖ Проверка конфигурации оповещений на прошлых данных

- Наконец, вы можете настроить действие, которое будет выполняться при срабатывании оповещения. В данном случае наша система была предварительно настроена на использование Slack в качестве действия оповещения (рис. 6.15), поэтому мы на этом и остановимся, но есть много других вариантов, доступных пользователю (перейдите по адресу <https://www.elastic.co/guide/en/kibana/current/action-types.html>, чтобы изучить все доступные варианты и способы отправки сообщений с оповещениями).

×

Create alert

Top influencers:
response.keyword = 200 [79]

Top records:
count() 200 [82]

Actions

▼ slack alert ⊖

Run when
Anomaly score matched the condition
▼

Slack connector
Add connector

slack alert
▼

Message

Elastic Stack Machine Learning Alert:

- Job IDs: {{context.jobids}}
- Time: {{context.timestampIso8601}}
- Anomaly score: {{context.score}}

Alerts are raised based on real-time scores. Remember that scores may

Add action

Cancel✓ Save

Рис. 6.15 ❖ Настройка действия оповещения

6. Нажатие на кнопку **Save** (Сохранить), очевидно, сохранит оповещение, которое затем можно будет просмотреть и изменить через **Stack Management | Alerts and Actions** (Управление стеком | Оповещения и действия) в Kibana (рис. 6.16).
7. Теперь создайте еще одно оповещение для задания `alert-demo-url_scanning`. Пусть это будет оповещение типа **Record** (Запись), но с другими параметрами конфигурации, аналогичными предыдущему примеру (рис. 6.17).

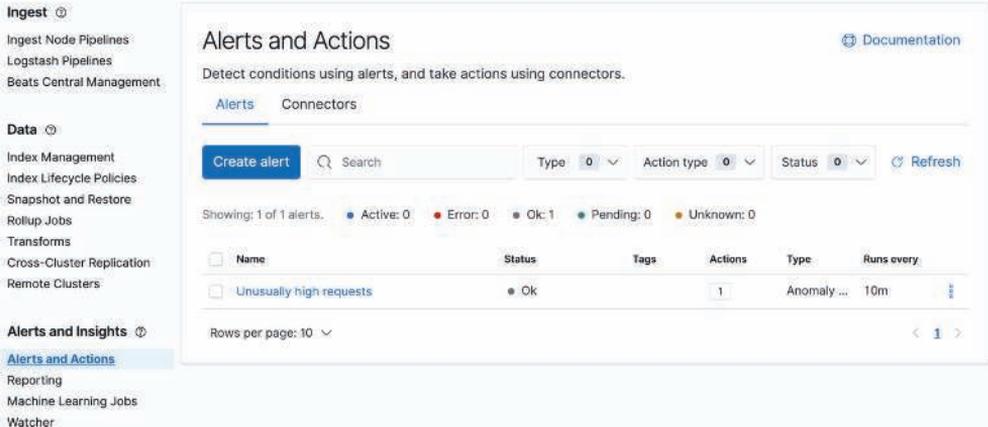


Рис. 6.16 ❖ Управление оповещениями

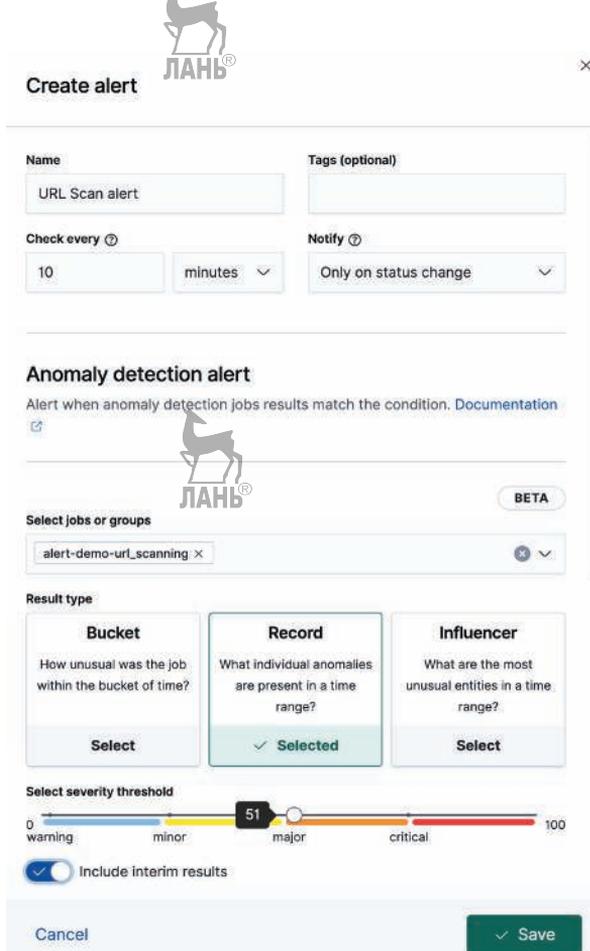


Рис. 6.17 ❖ Настройка оповещения для задания сканирования URL

Теперь, когда у вас настроены два оповещения, давайте перейдем к моделированию действительно аномальной ситуации в реальном времени, чтобы вызвать срабатывание оповещений.

Моделирование аномального поведения в реальном времени

Запуск моделирования аномального поведения в контексте этих примеров веб-журналов выглядит немного сложнее, чем настройка конфигурации заданий. Этот процесс потребует использования API Elasticsearch и выполнения нескольких команд через консоль Dev Tools в Kibana. Консоль – это место, где вы можете выполнять вызовы API в Elasticsearch и просматривать выходные данные (ответ) этих вызовов API.

❗ Если вы незнакомы с консолью, перейдите по адресу <https://www.elastic.co/guide/en/kibana/current/console-kibana.html>.

Мы проведем моделирование по двум направлениям: добавим несколько фальшивых документов в хранилище, за которым следит задание по обнаружению аномалий, а затем дождемся срабатывания оповещения. В этих документах будет показан всплеск запросов с фиктивного IP-адреса 0.0.0.0, ведущих к ответу с кодом 404, а также запросов случайных URL. Для запуска моделирования выполните следующие шаги.

1. Определите свое текущее время по UTC. Вы должны знать время в формате UTC (в отличие от времени вашего местного часового пояса), потому что документы, расположенные в хранилище Elasticsearch, имеют метку времени в формате UTC. Для этого вы можете просто воспользоваться любым онлайн-сервисом точного мирового времени (например, поиском в Google по запросу «текущее время UTC»). На момент написания этого абзаца текущим всемирным координированным временем было 16:41 8 апреля 2021 года. После преобразования в формат, который Elasticsearch ожидает для индекса `kibana_sample_data_logs`, оно будет иметь следующий вид:

```
"timestamp": "2021-04-08T16:41:00.000Z"
```

2. Давайте теперь вставим несколько новых фиктивных документов в хранилище `kibana_sample_data_logs` с меткой текущего времени (возможно, с небольшим буфером – округляя до следующей половины часа, в данном случае до 17:00). Замените значение поля `timestamp` на значение *своего текущего времени по UTC* и вызовите следующую команду *не менее 20 раз* в консоли Dev Tools для вставки документов:

```
POST kibana_sample_data_logs/_doc
{
  "timestamp": "2021-04-08T17:00:00.000Z",
  "event.dataset": "sample_web_logs",
```

```

"clientip": "0.0.0.0",
"response": "404",
"url": ""
}

```

3. Далее вы можете динамически изменять только те документы, которые вы только что вставили (в частности, поле `url`), чтобы имитировать уникальность всех URL, используя небольшой скрипт для рандомизации значения поля в вызове `API_update_by_query`:

```

POST kibana_sample_data_logs/_update_by_query
{
  "query": {
    "term": {
      "clientip": {
        "value": "0.0.0.0"
      }
    }
  },
  "script": {
    "lang": "painless",
    "source": "ctx._source.url = '/path/to/' + UUID.
randomUUID().toString();"
  }
}

```

4. Вы можете убедиться, что правильно создали набор уникальных случайных запросов с нашего фиктивного IP-адреса, посмотрев данные за нужное время в Kibana Discover (рис. 6.18).
5. Обратите внимание на рис. 6.18, что нам пришлось немного заглянуть в будущее, чтобы увидеть документы, которые были вставлены вручную (поскольку красная вертикальная линия на временной шкале около 12:45 является фактическим текущим системным временем в местном часовом поясе). Также обратите внимание, что вставленные нами документы имеют вполне корректно выглядящие случайные значения в поле `url`. Теперь, когда вы «заложили ловушку» для обнаружения аномалий и у вас есть готовые оповещения, вы можете просто сидеть, спокойно пить кофе и ждать, пока они сработают.

Получение и просмотр оповещений

Пока добавленное вами аномальное поведение ожидает обнаружения, вы можете подумать о том, в какой момент времени должны увидеть оповещение. Учитывая, что период времени заданий составляет 1 час, частота равна 10 мин, а задержки запросов составляют порядка 1–2 мин (при условии что ваши оповещения действительно будут искать промежуточные результаты и что они действительно выполняются с периодичностью 10 мин асинхронно по отношению к заданию обнаружения аномалий), вы должны ожидать появления оповещений между 13:12 и 13:20 по местному времени.

Так и есть – оповещения для двух заданий поступают в Slack в 13:16 и 13:18 по местному времени (рис. 6.19).

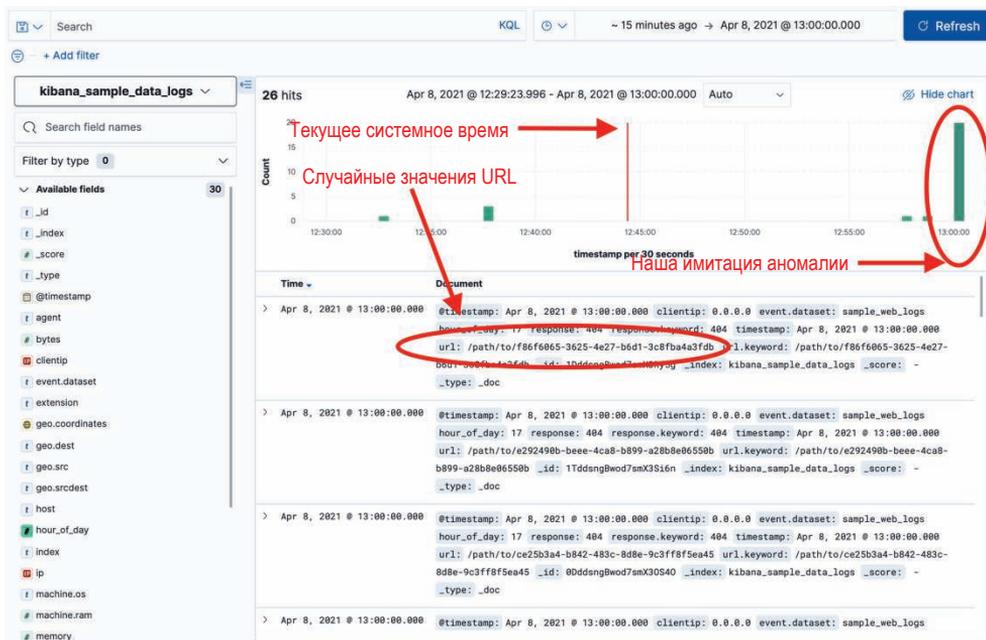


Рис. 6.18 ❖ Фиктивный всплеск аномальных событий, показанный в Discover

Верхнее оповещение на рис. 6.19, очевидно, предназначено для задания по обнаружению аномалий, которое подсчитывало количество событий для каждого `response.keyword` (поэтому было замечено, что количество документов с ответом 404 превышает ожидания), а нижнее оповещение – для другого задания, которое замечает аномально большое количество обращений к уникальным URL-адресам. Обратите внимание, что оба задания правильно определяют `clientip = 0.0.0.0` как фактор, влияющий на аномалии. В тексте оповещения предусмотрена возможность перейти по ссылке для прямого просмотра информации в Anomaly Explorer. Перейдя по ссылке во втором оповещении, вы попадаете в знакомое место для дальнейшего исследования аномалии (рис. 6.20).

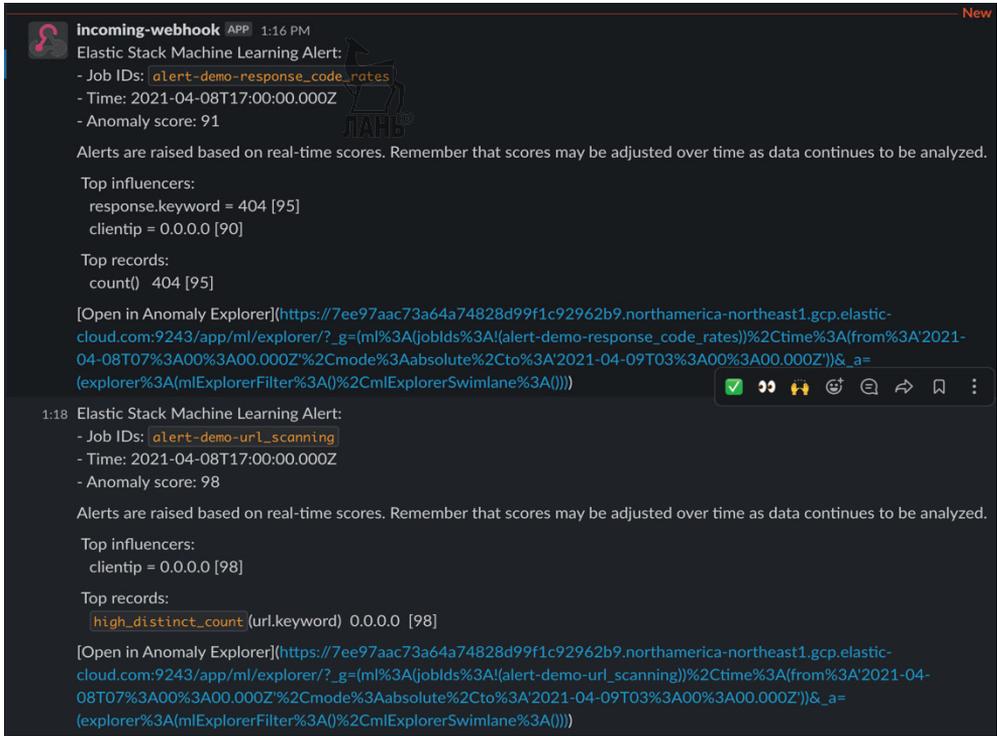


Рис. 6.19 ❖ Оповещения, полученные в клиенте Slack

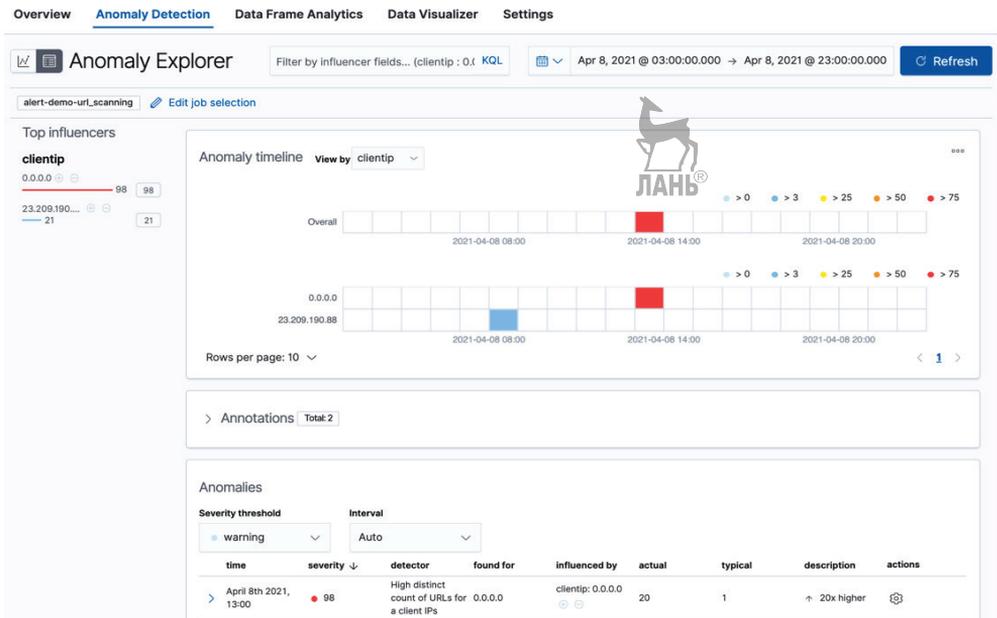


Рис. 6.20 ❖ Окно Anomaly Explorer после перехода по ссылке из оповещения

С помощью этого примера вы не только увидели, как использовать оповещения с применением фреймворка оповещения Kibana для заданий по обнаружению аномалий, но и смогли оценить тонкости работы как задания, так и оповещения в реальном времени. Настройки в потоке данных задания и интервале выборки оповещений действительно влияют на то, насколько близки к реальному времени могут быть оповещения. Мы могли бы, например, уменьшить значение параметра `frequency` и отключить опцию **Check every** в настройках оповещения, чтобы сэкономить несколько минут.

В следующем разделе мы не будем пытаться воспроизвести создание оповещений в реальном времени с помощью Watcher, но постараемся разобраться в соответствующих настройках, чтобы связать Watcher с заданием обнаружения аномалий, а также продемонстрировать несколько интересных примеров применения.

СОЗДАНИЕ ОПОВЕЩЕНИЙ С ПОМОЩЬЮ WATCHER

До версии 7.12 Watcher использовался как механизм оповещения об аномалиях, обнаруженных Elastic ML. Watcher – это очень гибкий собственный плагин для Elasticsearch, который может обрабатывать ряд задач автоматизации, а оповещения, безусловно, являются одной из них. В версиях 7.11 и более ранних пользователи могли либо создать свои собственные задания `watch` (экземпляр задачи автоматизации в Watcher) с нуля, чтобы оповещать о результатах задания по обнаружению аномалий, либо выбрать шаблон `watch` по умолчанию, созданный с помощью пользовательского интерфейса Elastic ML. Сначала мы рассмотрим вариант, предлагаемый по умолчанию, а затем обсудим некоторые идеи, касающиеся пользовательских экземпляров `watch`.

Использование устаревшего варианта `watch`

Теперь, когда оповещение о заданиях по обнаружению аномалий обрабатывается новой платформой оповещения Kibana, устаревший шаблон `watch` по умолчанию (плюс несколько других примеров) размещен в репозитории GitHub по адресу https://github.com/elastic/examples/tree/master/Alerting/Sample%20Watches/ml_examples.

Анализируя содержимое файла шаблона по умолчанию `default_ml_watch.json` и сопутствующую версию, в которой есть оповещение по электронной почте (`default_ml_watch_email.json`), мы видим, что они состоят из четырех основных разделов:

- `trigger` – определяет расписание `watch`;
- `input` – определяет входные данные для оценки;
- `condition` – оценивает, надо ли выполнять раздел `actions`;
- `actions` – список действий, которые необходимо предпринять, если условие `condition` выполнено.

! Полное описание всех возможностей Watcher вы можете найти в документации по Elastic по адресу <https://www.elastic.co/guide/en/elasticsearch/reference/current/how-watcher-works.html>.

Далее мы подробно обсудим каждый раздел.

trigger

В шаблонах Watcher для ML по умолчанию раздел `trigger` определяется следующим образом:

```
"trigger": {
  "schedule": {
    "interval": "82s"
  }
},
```



Здесь мы видим, что интервал срабатывания данного экземпляра `watch` в реальном времени составляет 82 секунды. Обычно это должно быть случайное значение от 60 до 120 секунд, чтобы при перезапуске узла все экземпляры `watch` не были синхронизированы и у каждого из них было свое время выполнения, распределенное более-менее равномерно, чтобы снизить потенциальную пиковую нагрузку на кластер. Также важно, чтобы это значение интервала было меньше или равно промежутку времени выполнения задания. Как было сказано ранее в этой главе, если он больше, чем длительность сегмента, Watcher может пропустить недавно сделанные записи об аномалиях. Поскольку интервал меньше (или даже намного меньше), чем период задания, вы также можете воспользоваться преимуществами расширенного оповещения, которое доступно при наличии промежуточных результатов, т. е. аномалий, которые можно определить, даже если вы не просмотрели все данные в пределах сегмента.

input

Раздел `input` начинается с секции `search`, в которой определяется следующий запрос по шаблону индекса `.ml-anomalies-*`:

```
"query": {
  "bool": {
    "filter": [
      {
        "term": {
          "job_id": "<job_id>"
        }
      },
      {
        "range": {
          "timestamp": {
            "gte": "now-30m"
          }
        }
      }
    ]
  }
}
```


Затем субагрегация запрашивает первый сегмент из списка, отсортированного по значению `anomaly_score`:

```
"aggs": {
  "top_bucket_hits": {
    "top_hits": {
      "sort": [
        {
          "anomaly_score": {
            "order": "desc"
          }
        }
      ],
      "_source": {
        "includes": [
          "job_id",
          "result_type",
          "timestamp",
          "anomaly_score",
          "is_interim"
        ]
      },
      "size": 1,
```

Затем в субагрегации `top_bucket_hits` следует ряд определений `script_fields`:

```
    "script_fields": {
      "start": {
        "script": {
          "lang": "painless",
          "source": "LocalDateTime.ofEpochSecond((doc[\"timestamp\"]').value.getMillis()-
            ((doc[\"bucket_span\"]').value * 1000)\n * params.padding)) /
            1000, 0, ZoneOffset.UTC).toString()+\":00.000Z\"",
          "params": {
            "padding": 10
          }
        }
      },
      "end": {
        "script": {
          "lang": "painless",
          "source": "LocalDateTime.ofEpochSecond((doc[\"timestamp\"]').value.
            getMillis()+((doc[\"bucket_span\"]').value * 1000)\n * params.
            padding)) / 1000, 0, ZoneOffset.UTC).toString()+\":00.000Z\"",
          "params": {
            "padding": 10
          }
        }
      }
    }
  }
}
```

```

    },
    "timestamp_epoch": {
      "script": {
        "lang": "painless",
        "source": "\"\"doc[\"timestamp\"].
value.getMillis()/1000\"\"\"
      }
    },
    "timestamp_iso8601": {
      "script": {
        "lang": "painless",
        "source": "\"\"doc[\"timestamp\"].
value\"\"\"
      }
    },
    "score": {
      "script": {
        "lang": "painless",
        "source": "\"\"Math.
round(doc[\"anomaly_score\"].value)\"\"\"
      }
    }
  }
}

```



Эти вновь определенные переменные будут использоваться механизмом Watcher для обеспечения большей функциональности в контексте задания. Некоторые из переменных представляют собой значения в другом формате (*score* – это просто округленная версия *anomaly_score*), в то время как *start* и *end* сыграют свою роль позже, определяя время начала и окончания, которые вычисляются как ± 10 длительностей сегмента с момента аномального сегмента. Позже эти значения понадобятся пользовательскому интерфейсу, чтобы показать соответствующий контекстный временной диапазон до и после аномального сегмента, чтобы пользователь увидел более ясную картину.

Агрегации *influencer_results* и *record_results* запрашивают три верхние оценки на уровнях фактора влияния и записи, но только результат агрегирования *record_results* используется в последующих частях экземпляра *watch* (и только в разделе *actions* файла *default_ml_watch_email.json*, который содержит некоторый текст электронной почты по умолчанию).

condition

Раздел *condition* – это место, где *input* проверяется на соответствие условиям, при которых нужно выполнить действие *action*. В этом случае раздел условий выглядит следующим образом:

```

"condition": {
  "compare": {
    "ctx.payload.aggregations.bucket_results.doc_count": {

```



```

    "gt": 0
  }
}
},

```

В данном случае выполняется проверка, вернула ли агрегация `bucket_results` какие-либо документы (где `doc_count` больше 0). Другими словами, если агрегация `bucket_results` вернула ненулевые результаты, это означает, что существуют документы, в которых значение `anomaly_score` превышает 75. Если это так, то будет вызвана секция `action`.

action

Раздел `action` по умолчанию в нашем случае состоит из двух частей: действие `log` для записи информации журнала в файл и действие `send_email` для отправки электронного письма. Для краткости мы не будем приводить здесь текст экземпляра `watch` (там действительно много текста). Действие `log` распечатает сообщение в выходной файл, который по умолчанию является файлом журнала Elasticsearch. Обратите внимание, что синтаксис сообщения использует язык шаблонов под названием Mustache (названный так из-за частого использования фигурных скобок – `mustache`, *англ. язык*). Проще говоря, переменные, помещенные в двойные фигурные скобки Mustache, будут заменены их фактическими значениями. В результате для одного из примеров заданий, которые мы создали ранее в этой главе, текст журнала, записанный в файл, может выглядеть следующим образом:

```
Alert for job [alert-demo-response_code_rates] at
[2021-04-08T17:00:00.000Z] score [91]
```

Это оповещение выглядит весьма похожим на сообщение Slack ранее в этой главе – ведь оно основано на той же информации. Оповещение по электронной почте может выглядеть следующим образом:

```
Elastic Stack Machine Learning Alert
Job: alert-demo-response_code_rates
Time: 2021-04-08T17:00:00.000Z
Anomaly score: 91
Click here to open in Anomaly Explorer.
Top records:
count() [91]
```

Очевидно, что данное сообщение предназначено не для краткого переказа информации пользователю, а для того, чтобы побудить его продолжить исследование, щелкнув ссылку в электронном письме.

Кроме того, следует отметить, что в тексте сообщения по электронной почте указаны первые три записи. В нашем примере есть только одна запись (детектор `count` с оценкой 91). Эта информация получена из агрегации `record_results`, которую мы описали ранее в разделе `input`.

Данный шаблон `watch`, используемый по умолчанию, является хорошим и полезным оповещением, которое предоставляет обобщенную информа-

цию о необычности набора данных с течением времени, но также полезно понимать особенности использования этого шаблона:

- основное условие для срабатывания оповещения – оценка аномалии сегмента выше определенного значения. Следовательно, оно не будет предупреждать об отдельных аномальных записях в сегменте в случае, если их оценка не поднимает общую оценку сегмента выше установленного порога;
- по умолчанию в выходных данных указывается не более трех наивысших оценок в сегменте, и только в версии для электронной почты;
- выбор доступных действий в этих примерах ограничен – только запись в журнал и отправка сообщения по электронной почте. Добавление других действий (сообщение Slack, веб-сокет и т. д.) потребует ручного редактирования шаблона.

В какой-то момент вам может понадобиться более полнофункциональный и сложный шаблон для полной настройки поведения экземпляра watch. Далее мы обсудим еще несколько примеров создания конфигураций watch с нуля.

Пользовательские шаблоны watch с уникальной функциональностью

Энтузиастам, которые хотят глубже изучить возможности Watcher, мы предлагаем рассмотреть несколько других примеров, представленных в репозитории GitHub. К ним относятся примеры одновременного запроса результатов нескольких заданий, программного объединения оценок аномалий и динамического сбора дополнительных указаний на первопричины других аномалий, коррелированных во времени.

Связанный ввод и сценарий условий

Хорошим примером интересного шаблона является `multiple_jobs_watch.json`, который демонстрирует возможность выполнения связанного ввода (выполнения нескольких запросов к результатам нескольких заданий), а также обработки более сложного динамического условия с помощью сценария:

```
"condition" : {
  "script" : {
// возвращает true, только если совокупная взвешенная оценка превышает 75
    "source" : "return ((ctx.payload.job1.aggregations.max_anomaly_score.value * 0.5) + (ctx.payload.job2.aggregations.max_anomaly_score.value * 0.2) + (ctx.payload.job3.aggregations.max_anomaly_score.value * 0.1)) > 75"
  }
},
```

Фактически этот сценарий означает, что оповещение срабатывает только в том случае, если совокупная взвешенная оценка аномалий трех разных за-

даний превышает значение 75. Другими словами, не все задания считаются одинаково важными, и учитывается вес каждого задания.

Передача информации между связанными входами

Еще одним уникальным аспектом связанных входных данных является то, что информация, полученная из одного входного потока, может передаваться в другой поток. Как показано в `chained_watch.json`, второй и третий входные потоки используют значение метки времени, полученной из первого запроса, как часть фильтра `range`:

```
{ "range": { "timestamp": { "gte": "{{ctx.payload.job1.hits.hits.0._source.timestamp}}|-{{ctx.metadata.lookback_window}}",
"lte": "{{ctx.payload.job1.hits.hits.0._source.timestamp}}"}},
```

Фактически это означает, что `Watcher` собирает вторичные аномалии в качестве свидетельств, извлеченных из временного окна, отмеренного до предположительно важной аномалии первого задания. Данный вид оповещений хорошо согласуется с ситуацией, которую мы скоро обсудим в главе 7. Это анализ, в котором реальная проблема приложения решается путем поиска коррелированных аномалий в окне вокруг аномалии KPI. Пример вывода такого шаблона `Watcher` может выглядеть следующим образом:

```
[CRITICAL] Anomaly Alert for job it_ops_kpi: score=85.4309 at
2021-02-08 15:15:00 UTC
Possibly influenced by these other anomalous metrics (within
the prior 10 minutes):
job:it_ops_network: (anomalies with at least a record score of
10):
field=In_Octets: score=11.217614808972602,
value=13610.62255859375 (typical=855553.8944717721) at 2021-02-
08 15:15:00 UTC
field=Out_Octets: score=17.00518, value=1.9079535783333334E8
(typical=1116062.402864764) at 2021-02-08 15:15:00 UTC
field=Out_Discards: score=72.99199, value=137.04444376627606
(typical=0.012289061361553099) at 2021-02-08 15:15:00 UTC
job:it_ops_sql: (anomalies with at least a record score of 5):
hostname=dbserver.acme.com field=SQLServer_Buffer_Manager_
Page_life_expectancy: score=6.023424, value=846.0000000000005
(typical=12.609336298838242) at 2021-02-08 15:10:00 UTC
hostname=dbserver.acme.com field=SQLServer_Buffer_Manager_
Buffer_cache_hit_ratio: score=8.337633, value=96.93249340057375
(typical=98.93088463835487) at 2021-02-08 15:10:00 UTC
hostname=dbserver.acme.com field=SQLServer_General_Statistics_
User_Connections: score=27.97728, value=168.15000000000006
(typical=196.1486370757187) at 2021-02-08 15:10:00 UTC
```

Здесь форматированием вывода, который сопоставляет результаты каждой из трех полезных нагрузок, управляет объемный сценарий `transform`, реализованный на Java-подобном языке сценариев `Painless`.

! Дополнительную информацию о языке сценариев Painless вы можете найти в документации Elastic по адресу <https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scriptingpainless.html>.

Если вас не пугает насыщенный кодом формат Watcher, вы можете использовать его как очень мощный инструмент для реализации некоторых интересных и полезных схем оповещений.

ЗАКЛЮЧЕНИЕ

Задания по обнаружению аномалий, безусловно, полезны сами по себе, но в сочетании с оповещениями в режиме, близком к реальному времени, пользователи могут использовать возможности автоматического анализа, при этом будучи уверенными в получении только значимых оповещений.

Изучив приемы эффективной фиксации результатов заданий по обнаружению аномалий с помощью оповещений в реальном времени, мы рассмотрели подробный пример использования новой инфраструктуры оповещений Kibana для простого определения некоторых интуитивно понятных оповещений и протестировали их с помощью реалистичного сценария. Затем рассмотрели примеры того, как опытный пользователь может использовать всю мощь Watcher для создания расширенных оповещений, если возможности Kibana не удовлетворяют его потребности.

В следующей главе вы увидите, как задания по обнаружению аномалий способствуют не только оповещению о важных ключевых показателях производительности, но и автоматическому анализу широкого набора данных при помощи Elastic ML, который в контексте конкретного приложения становится средством достижения чего-то наподобие «искусственного интеллекта», занятого отслеживанием проблемы приложения и определением ее первопричины.

Глава 7

.....

Выявление истинных причин аномалий

До этого момента мы подробно объясняли ценность обнаружения аномалий отдельно по метрикам и журналам. Это, конечно, очень важно. Однако в некоторых случаях знание того, что конкретная метрика или файл журнала содержит аномалии, не раскрывает нам полную картину происходящего. Например, аномалия может указывать на симптом, а не на причину проблемы. Чтобы лучше понять весь объем возникающей проблемы, часто бывает полезно изучить разные аспекты системы или взглянуть на ситуацию целиком. Этот подход включает в себя одновременный «интеллектуальный» анализ нескольких видов связанных наборов данных.

В этой главе рассмотрим следующие темы:

- истинное значение термина AIOps;
- понимание важности и ограничений KPI;
- выход за рамки KPI;
- организация данных для анализа;
- использование контекстной информации;
- переход к RCA.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Информация и примеры, представленные в данной главе, актуальны для версии 7.11 Elastic Stack и используют образцы наборов данных из репозитория GitHub, которые можно найти по адресу <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition>.

НАСТОЯЩЕЕ ЗНАЧЕНИЕ ТЕРМИНА AIOps

В главе 1 вы узнали, что многие компании захлебываются в постоянно растущем потоке ИТ-данных, при этом от них требуют «делать больше с меньшими затратами» (меньше людей, меньше расходов и т. д.). Часть этих данных со-

бирают и/или хранят с применением специализированных инструментов и платформ, но некоторые данные могут быть собраны на платформах данных общего назначения, таких как Elastic Stack. Остается открытым вопрос: какому проценту этих данных уделяется достаточное внимание? Мы имеем в виду долю собранных данных, которые активно проверяются людьми или отслеживаются с помощью автоматизированных средств (оповещения на основе правил, пороговых значений и т. д.). Даже оптимистичные оценки этой доли данных ограничиваются однозначными числами. Так почему же 90 % или более собираемых данных остаются без внимания? Правильный ответ заключается в том, что мы на самом деле не знаем всех причин.

Прежде чем упрекать IT-организации в том, что они собирают кучу данных, но не анализируют их, нам необходимо понять масштаб проблем, связанных с этими операциями. Типичное пользовательское приложение может делать следующее:

- охватывать сотни физических серверов;
- иметь десятки (если не сотни) микросервисов, каждый из которых может иметь десятки или сотни операционных показателей или записей журнала, описывающих его работу.

Комбинация этих показателей легко дает нам шести- или семизначное количество уникальных точек измерения. Больше того, под управлением IT-организации могут быть десятки или даже сотни таких приложений. Неудивительно, что количество данных, собираемых этими системами за день, можно легко измерить в терабайтах.

Поэтому вполне естественно, что подходящее решение может заключаться в сочетании автоматизации и искусственного интеллекта, чтобы уменьшить нагрузку на аналитиков. Какой-то умный маркетолог придумал термин **AIOps**¹, чтобы обозначить предполагаемое решение проблемы – делать то, что люди не могут (или не имеют времени) сделать вручную, при помощи интеллектуальной автоматизации. То, что *на самом деле* делает решение AIOps для достижения этой цели, часто остается на усмотрение взыскательного пользователя.

Итак, давайте разберемся в том, что на самом деле скрывается за сокращением AIOps, а красивые термины оставим маркетологам. Мы сформулируем перечень функций, которые должна выполнять интеллектуальная технология, чтобы помочь нам в трудной ситуации с обработкой данных:

- автономно проверять данные и оценивать их актуальность, важность и значимость на основе автоматически изученного набора ограничений, правил и поведения;
- отфильтровывать шум не относящегося к делу поведения, чтобы не отвлекать аналитиков от вещей, которые действительно имеют значение;
- вырабатывать нужное количество опережающих (прогнозных) оповещений о проблемах, которые назревают, но пока не привели к сбою или отказу;

¹ Artificial Intellect for IT Operations – деятельность по обработке и использованию данных в IT с применением искусственного интеллекта. – *Прим. перев.*

- автоматически собирать связанные/коррелированные свидетельства о наличии проблемы, чтобы помочь с *анализом первопричин* (root cause analysis, RCA);
- выявлять проблемы эксплуатационной эффективности для максимального повышения производительности инфраструктуры;
- предлагать действие или последовательность действий для исправления, основываясь на прошлых исправлениях и их эффективности.

Хотя этот список никоим образом не является исчерпывающим, мы можем сделать главный вывод – внедрение интеллектуальной автоматизации и анализа способно принести большую отдачу, позволяя IT-подразделениям значительно повысить эффективность и, таким образом, максимизировать бизнес-результаты.

Платформа Elastic ML (Elastic Machine Learning) может играть важную роль в реализации почти всех вышеупомянутых функций, за исключением разве что последнего пункта в списке. Вы уже видели, как Elastic ML может автоматически обнаруживать аномальное поведение, прогнозировать тенденции, вырабатывать прогнозные оповещения и т. д. Но мы также должны признать, что Elastic ML является платформой машинного обучения общего назначения – это не специализированный инструмент для IT-операций/наблюдения или аналитики безопасности. В связи с этим важно иметь исчерпывающее представление о том, как Elastic ML используется в контексте деятельности IT-подразделений.

Также важно отметить, что до сих пор существует большое количество операционных IT-подразделений, которые в настоящее время не используют интеллектуальную автоматизацию и анализ. Они часто заявляют, что хотели бы использовать подход на основе ИИ для улучшения своей работы, но не совсем готовы сделать решительный шаг. Итак, давайте попробуем опровергнуть расхожее представление о том, что единственный способ извлечь выгоду из ИИ – это сделать и внедрить все, что возможно, за один день. Давайте вместо этого создадим несколько практических приложений Elastic ML в контексте IT-операций и посмотрим, как их можно использовать для достижения большинства целей, сформулированных в предыдущем списке.

Мы начнем с понятия *ключевого показателя эффективности* (key performance indicator, KPI) и пояснения, почему это наилучший и логичный выбор для начала работы с Elastic ML.

ЗНАЧИМОСТЬ И ОГРАНИЧЕНИЯ KPI

Если вы стремитесь решить проблему масштабирования и добиться заметного прогресса в практическом применении собранных данных, то в первую очередь нужно обратить внимание на показатели, которые являются лучшими индикаторами производительности и работоспособности. Ключевые показатели эффективности, которые IT-организация выбирает для

измерения, отслеживания и оповещений, могут быть очень разнообразными, например:

- **качество обслуживания клиентов** – эта группа показателей отражает качество обслуживания клиентов в числовой форме, такой как время отклика приложений или количество ошибок;
- **доступность** – часто бывает важно отслеживать такие показатели, как время безотказной работы или среднее время простоя (mean time to repair, MTTR);
- **бизнес** – в эту группу могут входить показатели, которые напрямую измеряют эффективность бизнеса, например количество заказов в минуту или количество активных пользователей.

Как правило, эти типы показателей обычно отображаются на первом плане и в центре на большинстве операционных панелей мониторинга высокого уровня или в отчетах для всех сотрудников, от технических специалистов до руководителей. Поиск изображений в Google по фразе *KPI dashboard* (информационная панель KPI) вернет бесчисленное количество примеров диаграмм, датчиков, циферблатов, карт и других приятных для глаз визуализаций.

Несмотря на то что визуальное представление информации, которое можно охватить одним взглядом, имеет большую ценность, ручная проверка показателей по-прежнему сопряжена с фундаментальными проблемами:

- **интерпретация** – могут возникнуть трудности с пониманием разницы между нормальной работой и ненормальной, если только человек не знает этого заранее;
- **проблемы масштабирования** – несмотря на то что KPI уже представляет собой концентрированную выборку наиболее важных показателей, нередко возникает ситуация, когда показателей KPI больше, чем можно отобразить на одной панели мониторинга. В итоге вы можете получить перегруженную визуализацию или же длинные информационные панели, требующие прокрутки либо разбиения по страницам;
- **отсутствие проактивности** – многие информационные панели не имеют возможности привязывать показатели к оповещениям (особенно к проактивным, т. е. прогнозным, работающим на опережение), поэтому требуется постоянное наблюдение со стороны человека, если заранее известно, что важен нестабильный ключевой показатель эффективности.

Иными словами, правильный выбор и измерение KPI – чрезвычайно важный шаг в процессе выявления и отслеживания значимых индикаторов работоспособности и поведения IT-системы. Однако должно быть очевидно, что простое определение и отслеживание набора KPI с использованием только прямой визуализации не лишено существенных недостатков.

Мы утверждаем, что KPI – отличный кандидат на роль показателей, которые можно отслеживать с помощью механизма обнаружения аномалий Elastic ML. Например, предположим, что у нас есть данные, которые выглядят следующим образом (фрагмент набора данных `it_ops_kpi` в репозитории GitHub):

```
{
  '_index' : 'it_ops_kpi',
  '_type' : '_doc',
  '_id' : 'UqUsMngBF0h8A28xK-E3',
  '_score' : 1.0,
  '_source' : {
    '@timestamp' : '2021-01-29T05:36:09.000Z',
    'events_per_min' : 28,
    'kpi_indicator' : 'online_purchases'
  }
},
```

В данном случае KPI (поле с именем `events_per_min`) представляет собой суммарное общее количество покупок в минуту для некоторой онлайн-системы обработки транзакций. Мы можем без труда отслеживать изменения этого KPI с течением времени с помощью задания по обнаружению аномалий с функцией суммирования по полю `events_per_min` и промежутком времени в 15 мин. Неожиданный спад онлайн-продаж (до 921) обнаружен и помечен как аномальный (рис. 7.1).



Рис. 7.1 ❖ Анализ KPI с помощью типичного задания по обнаружению аномалий

Здесь KPI – это всего лишь единый обобщенный показатель. Если в данных есть другое категориальное поле, которое позволяет сегментировать показатель (например, продажи по идентификатору продукта, по категории

продукта, по географическому региону и т. д.), то ML может легко разделить анализ по этому полю на параллельные ветви анализа (как было показано в главе 3). Но давайте не будем отвлекаться от того, на чем мы сейчас сосредоточены: упреждающий анализ ключевого показателя, который кого-то, вероятно, волнует. Количество онлайн-продаж в единицу времени напрямую связано с поступающей выручкой и, следовательно, является очевидным KPI.

Однако, несмотря на важность понимания того, что с нашим KPI происходит что-то необычное, до сих пор нет понимания того, *почему* это происходит. Может быть, возникли проблемы в работе одной из серверных систем, обслуживающих это клиентское приложение? Или в новой версии пользовательского интерфейса возникла ошибка, из-за которой пользователям было сложнее завершить транзакцию? Или возникла проблема со сторонним поставщиком услуг обработки платежей? Ни на один из этих вопросов нельзя ответить, просто изучив KPI.

Чтобы получить представление о реальной картине происходящего, нам нужно будет расширить наш анализ, включив в него другие наборы релевантной и связанной информации.

Выходя за рамки KPI

В целом процесс выбора KPI редко вызывает затруднения, поскольку обычно очевидно, на какие показатели нужно обращать внимание в первую очередь (если онлайн-продажи падают, приложение, скорее всего, не работает). Но если мы хотим получить более полное представление о том, что послужило причиной возникновения операционной проблемы, то придется расширить наш анализ за пределы KPI и рассмотреть показатели, отражающие работу базовых систем и технологий, поддерживающих приложение.

К счастью, существует множество способов сбора всех видов данных для последующего централизованного анализа в Elastic Stack. Например, **Elastic Agent** – это единый унифицированный агент, который можно развернуть на узлах или контейнерах для сбора данных и отправки их в Elastic Stack. За кулисами основного процесса Elastic Agent запускает службы доставки Beats или Elastic Endpoint, необходимые для вашей конфигурации. Начиная с версии 7.11 Elastic Agent доступен в Kibana через пользовательский интерфейс Fleet и может использоваться для добавления и управления интеграциями популярных сервисов и платформ (рис. 7.2).

Используя различные интеграции, пользователь может легко собирать данные и централизовать их в Elastic Stack. Хотя эта глава не содержит руководство по использованию Fleet и Elastic Agent, важным моментом является то, что независимо от того, какие инструменты вы используете для сбора базовых данных приложения и системы, несомненно одно: данных будет много, очень много. Помните, что наша конечная цель – охватить вниманием и подвергнуть упреждающему анализу как можно больший процент от общего набора данных. Но, чтобы подвергнуть эти данные наиболее эффективно анализу с помощью Elastic ML, их необходимо правильно организовать.

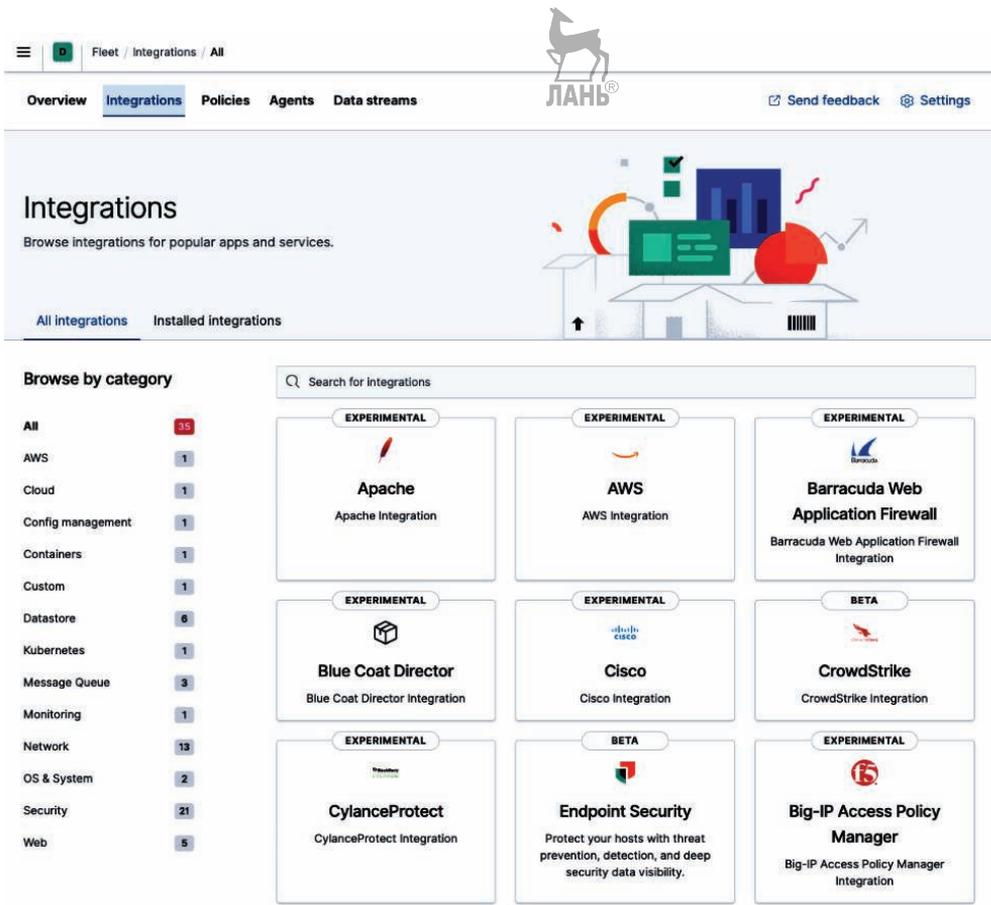


Рис. 7.2 ❖ Раздел интеграции в пользовательском интерфейсе Fleet

ОРГАНИЗАЦИЯ ДАННЫХ ДЛЯ АНАЛИЗА

Одна из самых приятных особенностей сбора данных через Elastic Agent заключается в том, что по умолчанию собранные данные нормализуются с использованием обобщенной схемы Elastic (Elastic Common Schema, ECS). ECS – это спецификация с открытым исходным кодом, которая определяет общую таксономию и соглашения об именах для данных, хранящихся в Elastic Stack. Благодаря этому данные становятся проще в управлении, анализе, визуализации и выявлении корреляции между разными типами данных, в том числе между показателями производительности и файлами журналов.

Даже если вы не используете Elastic Agent или более старые инструменты Elastic (такие как Beats и Logstash) и вместо этого полагаетесь на сторонние процессы сбора или извлечения данных, все равно рекомендуется согласовывать свои данные со спецификацией ECS, потому что ваши усилия окупятся

в будущем, если пользователи будут использовать эти данные для запросов, информационных панелей и, конечно же, для задач машинного обучения.

❗ Более подробную информацию о ECS можно найти в справочном разделе веб-сайта по адресу <https://www.elastic.co/guide/en/ecs/current/ecs-reference.html>.

Среди многих важных полей в ECS есть поле `host.name`, которое определяет, с какого хоста были собраны данные. По умолчанию большинство стратегий сбора данных в Elastic Stack включают размещение данных в хранилищах, ориентированных на тип данных и, следовательно, потенциально содержащих подборку документов с множества разных хостов. Возможно, некоторые из хостов в нашей среде поддерживают одно приложение (например, онлайн-покупки), но другие хосты поддерживают другое приложение (например, обработку счетов). Поскольку все хосты отправляют свои данные в одно хранилище, если мы хотим составить отчет и провести анализ данных для какого-то одного или группы приложений, то не сможем ориентироваться только на хранилище – нам понадобится возможность различать приложения.

Для этого у нас есть несколько вариантов:

- изменить запрос задания по обнаружению аномалий таким образом, чтобы он фильтровал данные только для хостов, связанных с интересующим приложением;
- при сборе данных добавлять в каждый документ дополнительную контекстную информацию, которая позже будет применяться для фильтрации запроса, сделанного заданием обнаружения аномалий.

Оба способа требуют настройки запроса канала данных, который задание обнаружения аномалий делает к необработанным данным в исходных хранилищах. Первый вариант может привести к относительно сложному запросу, а второй вариант требует промежуточного этапа дополнения данных с использованием настраиваемых конвейеров. Кратко обсудим каждый вариант.

Настраиваемые запросы для каналов данных

Когда вы создаете новое задание в пользовательском интерфейсе обнаружения аномалий, первым делом нужно выбрать либо шаблон хранилища, либо сохраненный поиск Kibana. Если выбрано первое, то вызывается запрос Elasticsearch `{'match_all':{}}` (возвращать каждую запись в хранилище). Если задание создается с помощью API или мастера расширенных заданий, пользователь может указать практически любой допустимый Elasticsearch DSL для фильтрации данных. Создание произвольной формы Elasticsearch DSL неопытными пользователями может привести к ошибкам. Более простым и понятным способом является использование сохраненных поисковых запросов Kibana.

Например, предположим, что у нас есть хранилище файлов журналов, а соответствующие хосты, связанные с приложением, которое мы хотим отслеживать и анализировать, состоят из двух серверов, `esxserver1.acme.com`

и `esxserver2.acme.com`. На странице **Discover** (Обнаружение) Kibana мы можем создать фильтрующий запрос на языке KQL, используя поле поиска в верхней части пользовательского интерфейса (рис. 7.3).

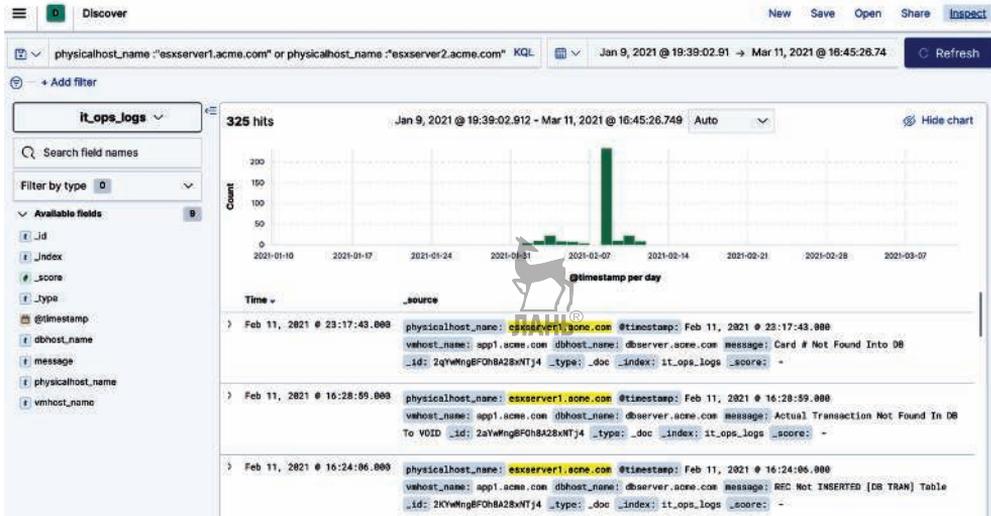


Рис. 7.3 ❖ Построение фильтрующего запроса с использованием KQL

Текст этого KQL-запроса будет следующим:

```
physicalhost_name:'esxserver1.acme.com' or physicalhost_
name:'esxserver2.acme.com'
```

Если вам интересно узнать о реальном Elasticsearch DSL, который вызывается Kibana для выполнения этого запроса, вы можете нажать кнопку **Inspect** (Осмотр) в правом верхнем углу и выбрать вкладку **Request** (Запрос), чтобы увидеть Elasticsearch DSL (рис. 7.4).

Вероятно, стоит отметить, что хотя в этом конкретном примере KQL-запрос транслируется в Elasticsearch DSL (например, с использованием `match_phrase`), это не единственный способ достичь желаемых результатов. Есть еще один способ – фильтрация запроса с использованием `terms`, но оценка преимуществ того или иного способа выходит за рамки этой книги.

Независимо от Elasticsearch DSL, который работает на заднем плане, ключевой момент заключается в том, что у нас есть запрос, который фильтрует необработанные данные, выделяя только серверы, имеющие отношение к приложению, которое мы хотели бы проанализировать с помощью Elastic ML. Чтобы сохранить этот отфильтрованный поиск, необходимо нажать кнопку **Save** в правом верхнем углу и присвоить поиску имя (рис. 7.5).

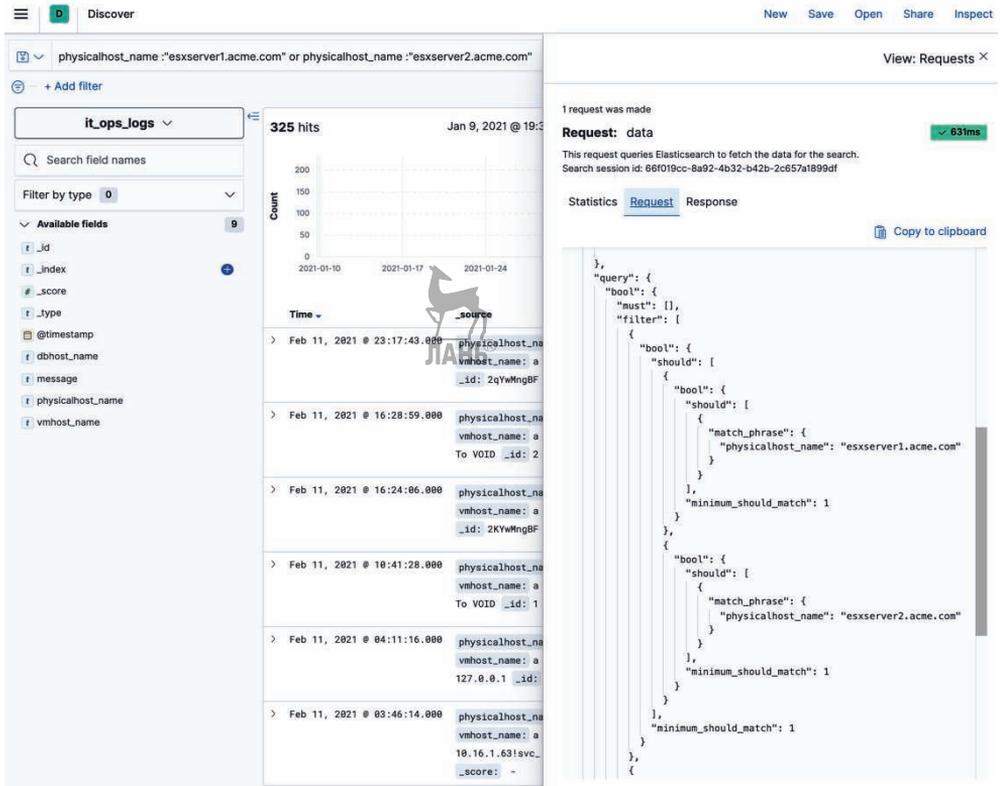


Рис. 7.4 ❖ Просмотр содержания Elasticsearch DSL, выполняющего фильтр KQL

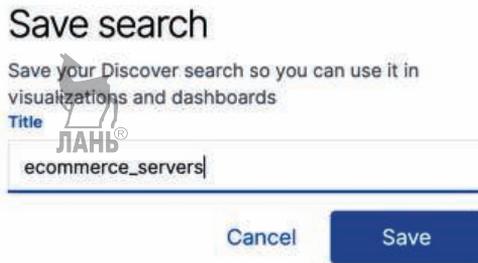


Рис. 7.5 ❖ Сохранение результатов поиска для последующего использования в Elastic ML

Позже вы можете выбрать этот сохраненный поиск при настройке нового задания по обнаружению аномалий (рис. 7.6).

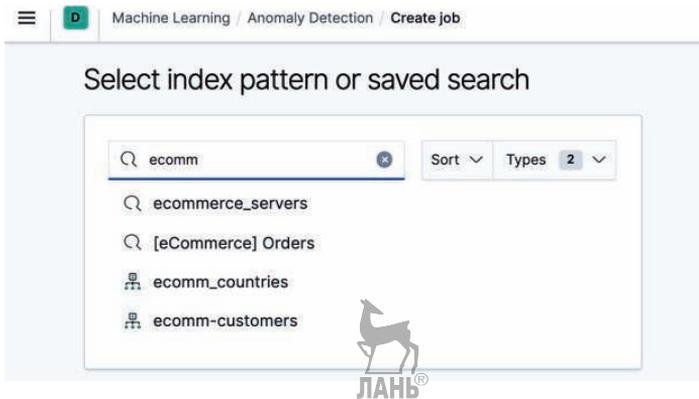


Рис. 7.6 ❖ Использование сохраненного поиска в задании по обнаружению аномалий

Таким образом, наша задача машинного обучения теперь будет выполняться только для хостов, представляющих интерес для этого конкретного приложения. Мы смогли эффективно ограничить и сегментировать анализ данных по хостам, которые, как мы определили, связаны с работой нужного приложения.

Дополнение получаемых данных

Другой вариант – перенести принятие решения о том, какие хосты каким приложениям принадлежат, на момент получения данных. Если частью процесса сбора данных является Logstash, вы можете использовать плагин фильтра для добавления дополнительных полей к данным при просмотре списка активов (файл, база данных и т. д.). Обратитесь к документации Logstash по адресу <https://www.elastic.co/guide/en/logstash/current/lookup-enrichment.html>, где показано, как динамически дополнять помещаемые в хранилище документы уточняющими полями для формирования контекста. Если вы не использовали Logstash (а просто задействовали Beats/Elastic Agent и приемный узел), возможно, более простым способом будет применение расширенной обработки. За дополнительной информацией обратитесь к справочной документации по адресу <https://www.elastic.co/guide/en/elasticsearch/reference/current/ingest-enriching-data.html>.

Например, вы можете добавить поле `application_name` и динамически заполнять значение этого поля соответствующим именем приложения (здесь показан фрагмент JSON):

```
'host': 'wasinv2.acme.com',
'application_name': 'invoice_processing',
```

Или так:

```
'host': 'www3.acme.com',
'application_name': 'online_purchases',
```

После того как значение этого поля установлено и вставлено в сохраненные документы, вы должны использовать поле `application_name` вместе с фильтрующим запросом для задания обнаружения аномалии (как описано ранее), чтобы ограничить анализ данных нужным значением поля. Может показаться, что дополнение данных требует лишних усилий, но они должны окупить себя в долгосрочной перспективе, поскольку этот метод легче обновлять по мере изменения или расширения перечня имен активов, в то время как первый метод требует жесткого кодирования имен активов при отборе заданий ML.

Теперь, когда вы организовали свои данные и, возможно, даже дополнили их, давайте посмотрим, как использовать контекстную информацию, чтобы сделать задания по обнаружению аномалий более эффективными.

ИСПОЛЬЗОВАНИЕ КОНТЕКСТНОЙ ИНФОРМАЦИИ

Когда ваши данные организованы и дополнены, вы можете использовать контекстную информацию двумя основными способами: с помощью аналитических *разделений* (*split*) и статистических *факторов влияния* (*influencer*).

Аналитическое разделение

Вы уже видели, что задание по обнаружению аномалий можно разделить по любому категориальному полю. Таким образом, вы можете построить индивидуальную модель поведения отдельно для каждого экземпляра этого поля (категории). Это может быть чрезвычайно ценно, особенно в случае, когда каждому экземпляру нужна собственная модель.

Возьмем, к примеру, случай, когда у нас есть данные по разным регионам мира (рис. 7.7).

Какими бы ни были данные (ключевые показатели эффективности продаж, показатели использования и т. д.), очевидно, что у них есть выраженные особенности, уникальные для каждого региона. В этом случае имеет смысл разделить по регионам любой анализ на обнаружение аномалий, чтобы извлечь пользу из этой уникальности, и тогда вы сможете обнаружить аномалии в поведении, характерные для каждого региона.

Давайте также представим, что в каждом регионе парк серверов поддерживает обработку приложений и транзакций, но они сбалансированы по нагрузке и в равной степени влияют на производительность приложения в целом. Следовательно, на вкладе каждого конкретного сервера нет ничего уникального. Скорее всего, нет смысла разделять анализ по серверам.

Естественно, мы пришли к выводу, что разделение по регионам более эффективно, чем по серверам. Но что, если на конкретном сервере в регионе возникают проблемы, напрямую влияющие на обнаруженные аномалии? Разве вы не хотели бы, чтобы эта информация была доступна немедленно, вместо того чтобы вручную проводить дальнейшую диагностику? Ответ можно получить, исследуя факторы влияния.

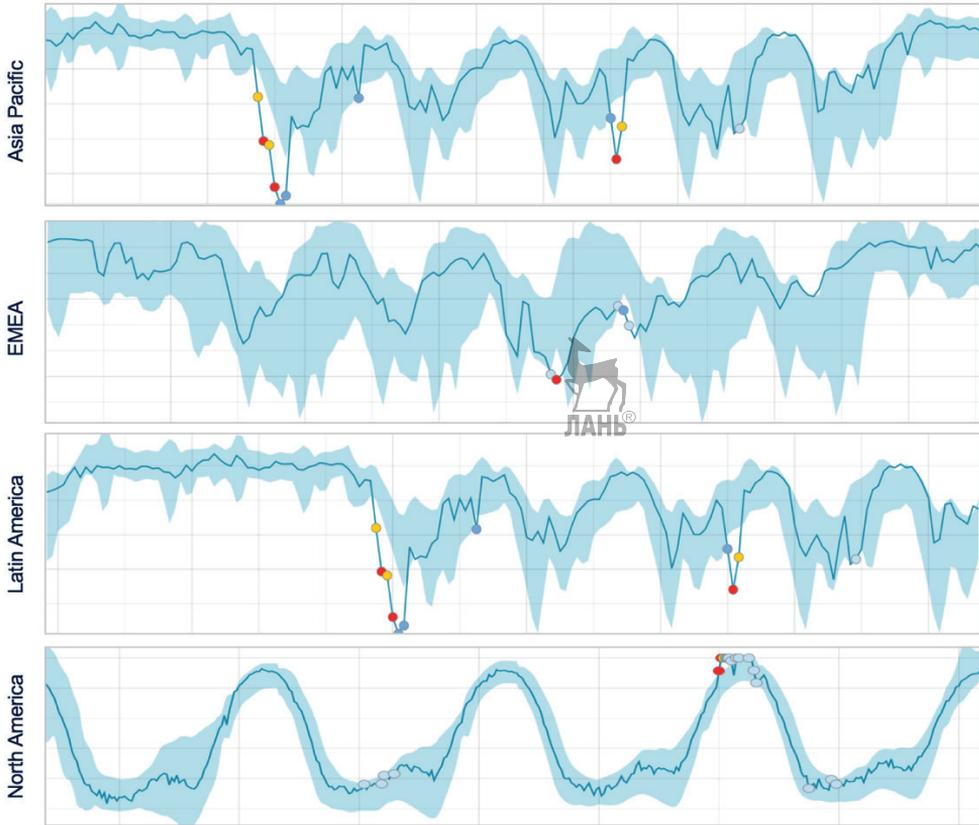


Рис. 7.7 ❖ Различное поведение данных в зависимости от региона

Статистические факторы влияния

Вы познакомились с факторами влияния в главе 5. Напомним, что *фактор влияния* – это поле, описывающее сущность, о которой вы хотели бы знать, влияет ли она на существование аномалии или, по крайней мере, вносит ли в нее заметный вклад. Любое поле, выбранное в качестве кандидата на роль фактора влияния, не обязательно должно быть частью логики детектора, хотя имеет смысл в качестве факторов влияния выбирать поля, которые используются для разбиения. Также важно, чтобы факторы влияния были выбраны при создании заданий по обнаружению аномалий, поскольку их невозможно добавить в конфигурацию позже.

Нужно понимать, что процесс поиска потенциальных факторов влияния происходит после того, как задание по обнаружению аномалий обнаружит аномалию. Другими словами, это не влияет на вычисления вероятностей, выполняемые в процессе обнаружения аномалии. После определения аномалии ML будет последовательно просматривать все экземпляры каждого поля кандидата на фактор влияния и удалять вклад этого экземпляра в данные

в определенном временном сегменте. Если после удаления оставшиеся данные больше не являются аномальными, то в соответствии с принципом исключения вклад этого экземпляра следует считать влиятельным и оценивать соответствующим образом (со значением `influencer_score` в результатах).

Однако в следующем разделе вы увидите, как факторы влияния могут быть задействованы при просмотре результатов не только одного задания по обнаружению аномалии, но даже нескольких связанных заданий. Далее мы рассмотрим процесс группирования и совместного просмотра заданий, перетекающий в процесс анализа первопричин (RCA).

АНАЛИЗ ПЕРВОПРИЧИН АНОМАЛИИ

Сейчас настало время обсудить, как объединить полученные ранее знания и навыки для построения комплексного процесса выявления причин аномалии. Если вы стремитесь повысить эффективность IT-операций и более комплексно отслеживать работоспособность приложений, вам необходимо применить на практике то, что вы узнали в предыдущих разделах, и соответствующим образом настроить задания по обнаружению аномалий. С этой целью давайте рассмотрим реальный сценарий, в котором Elastic ML помог найти первопричину эксплуатационного сбоя.

История проблемы

Этот сценарий в общих чертах основан на реальном сбое приложения, хотя данные были несколько упрощены и очищены, чтобы скрыть настоящего клиента. Проблема заключалась в приложении розничной торговли, которое обрабатывало транзакции по подарочным картам. Иногда приложение внезапно переставало работать, и транзакции не обрабатывались. Причем это становилось известно лишь тогда, когда отдельные магазины начинали звонить в штаб-квартиру компании, чтобы пожаловаться на неполадки. Основная причина проблемы оставалась неизвестной, и клиент не мог ее установить. Поскольку он так и не нашел первопричину и устранял сбой простой перезагрузкой серверов приложений, проблема продолжала возникать в произвольные моменты времени и мучила клиента в течение нескольких месяцев.

Следующие данные были собраны и включены в анализ, чтобы понять происхождение проблемы (они доступны в репозитории GitHub):

- обобщенный (1-минутный) подсчет объема транзакции (основной KPI);
- журналы приложений (частично структурированные текстовые сообщения) от механизма обработки транзакций;
- метрики производительности SQL сервера из базы данных, которая обслуживает механизм обработки транзакций;
- метрики производительности использования сети, в которой работает механизм обработки транзакций.

Таким образом, для анализа данных были настроены четыре задания машинного обучения:

- **it_ops_kpi** – использует функцию `sum` для подсчета количества транзакций, обрабатываемых за минуту;
- **it_ops_logs** – использует функцию `count` детектора `mlcategory` для подсчета количества сообщений журнала по типам, но применяет динамическую категоризацию на основе машинного обучения для определения различных типов сообщений;
- **it_ops_sql** – простой анализ среднего значения (`mean`) каждой метрики SQL-сервера в хранилище;
- **it_ops_network** – простой анализ среднего значения (`mean`) каждой метрики производительности сети в хранилище.

Эти четыре задания были настроены и выполняли анализ данных, когда в приложении возникла проблема. Были обнаружены аномалии, особенно в KPI, отслеживающем количество обрабатываемых транзакций. Фактически это тот же самый KPI, который вы видели в начале этой главы, где неожиданное падение количества обработанных заказов было основным индикатором возникновения проблемы (рис. 7.8).



Рис. 7.8 ❖ KPI количества обработанных транзакций

Однако основная причина не была понятна до тех пор, пока аномалия этого ключевого показателя эффективности не была сопоставлена с аномалиями в трех других заданиях машинного обучения, которые просматривали

данные в базовом приложении и инфраструктуре. Давайте посмотрим, как приемы визуальной корреляции и общих факторов влияния позволили обнаружить основную причину.

Корреляция и общие факторы влияния

Помимо аномалии в KPI обработанных транзакций (в которой происходит неожиданный провал), три других задания по обнаружению аномалий (производительность сети, журналы приложений и производительность SQL-сервера) были наложены на тот же временной интервал в инструменте Anomaly Explorer. На следующем снимке экрана показаны результаты наложения (рис. 7.9).



Рис. 7.9 ❖ Anomaly Explorer отображает результаты нескольких заданий

В частности, обратите внимание, что в тот день, когда возникла проблема с KPI (8 февраля 2021 г., как показано на рис. 7.8), три других задания на рис. 7.9 демонстрируют коррелированные аномалии (обведены красной линией). При более детальном рассмотрении (нажав на красную плитку для задания `it_ops_sql`) вы можете увидеть, что были проблемы с несколькими метриками SQL-сервера, которые одновременно принимали аномальные значения (рис. 7.10).



Рис. 7.10 ❖ Anomaly Explorer отображает аномалии SQL-сервера

❗ Заштрихованная область диаграмм обозначает окно времени, связанное с шириной выбранной плитки на строке просмотра. Это окно может быть больше, чем период анализа (как в данном случае), и поэтому заштрихованная область может содержать несколько отдельных аномалий в течение этого периода времени.

Если мы посмотрим на аномалии в задании по обнаружению аномалий журнала приложения, то обнаружим много ошибок, и все они ссылаются на базу данных, что дополнительно подтверждает нестабильность SQL-сервера (рис. 7.11).

Однако интересные вещи происходили и в сети (рис. 7.12).

В частности, наблюдался большой всплеск сетевого трафика (представленный метрикой `Out_Octets`) и высокий всплеск количества пакетов, отбрасываемых на уровне сетевого интерфейса (представленный метрикой `Out_Discards`).

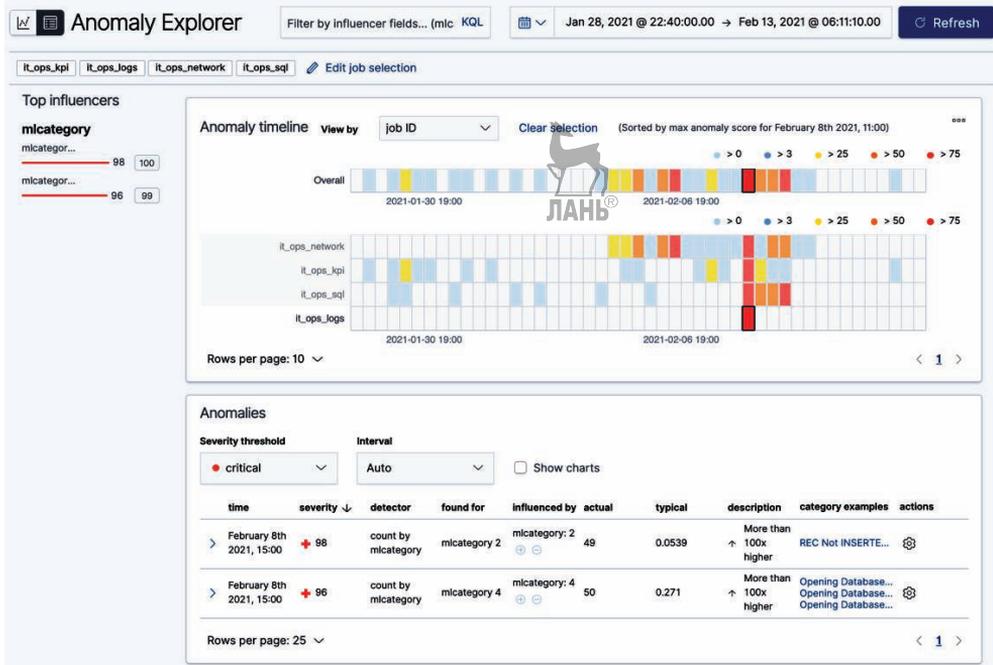


Рис. 7.11 ❖ Anomaly Explorer отображает аномалии в журнале приложения

На тот момент возникло явное подозрение, что этот скачок в сети может иметь какое-то отношение к проблеме с базой данных. И хотя корреляция не всегда является причинно-следственной связью, этого было достаточно, чтобы побудить аналитиков вернуться к некоторым историческим данным по предыдущим отключениям. При каждом предыдущем отключении также наблюдался большой всплеск сетевого трафика и количества отвергнутых пакетов.

Конечной причиной скачка трафика в сети оказалось решение по перемещению виртуальных машин VMware на новые серверы ESX. Кто-то неправильно сконфигурировал сетевой коммутатор, и VMware отправляла этот массивный всплеск трафика через VLAN приложения вместо VLAN управления. Когда это происходило (конечно, в случайные моменты), приложение для обработки транзакций временно теряло соединение с базой данных и пыталось восстановить соединение. Однако в соответствующем коде переподключения был критический недостаток, заключающийся в том, что код не пытался переподключиться к базе данных по удаленному IP-адресу, принадлежащему SQL-серверу. Вместо этого он пытался переподключиться к localhost (IP-адрес 127.0.0.1), где, конечно же, такой базы данных не было. Ключ к разгадке этой ошибки был замечен в одной из строк журнала, которые Elastic ML отображал в разделе **Examples** (Примеры), обведенном на рис. 7.13.



Рис. 7.12 ❖ Anomaly Explorer отображает аномалии сетевых данных

Таким образом, после возникновения проблемы подключение к SQL-серверу было возможно только в том случае, если сервер приложений полностью перезагружен, файлы конфигурации запуска повторно прочитаны и IP-адрес SQL-сервера правильно определен. Вот почему полная перезагрузка всегда решала проблему.

Следует отметить, что факторы влияния в пользовательском интерфейсе также помогают сузить круг потенциальных виновников аномалий (рис. 7.14).



Anomalies

Severity threshold: warning Interval: Auto Show charts

time	severity ↓	detector	found for	influenced by	actual	typical	description	category examples	actions
> February 8th 2021, 15:00	+ 98	count by micategory	micategory 2	micategory: 2	49	0.0539	More than 100x higher	REC Not INSERTE...	
February 8th 2021, 15:00	+ 96	count by micategory	micategory 4	micategory: 4	50	0.271	More than 100x higher	Opening Database... Opening Database... Opening Database...	

Details Category examples

Terms ⓘ
Opening Database DRIVER SQL Server SERVER network dbmsocn address DATABASE svc_prod AnsiNPW No

Regex ⓘ
.*?Opening.*?Database.*?DRIVER.*?SQL.*?Server.*?SERVER.*?network.*?dbmsocn.*?address.*?DATABASE.*?svc_prod.*?AnsiNPW.*?No.*

Examples

Opening Database = DRIVER={SQL Server};SERVER=10.16.1.63;network=dbmsocn;address=10.16.1.63
1433;DATABASE=,uid=dbadmin1;pwd=####;AnsiNPW=No

Opening Database = DRIVER={SQL Server};SERVER=127.0.0.1;network=dbmsocn;address=127.0.0.1
DATABASE=svc_prod;Trusted_Connection=Yes;AnsiNPW=No

Opening Database = DRIVER={SQL Server};SERVER=sssvcdbj1.acme.com;network=dbmsocn;address=sssvcdbj1.acme.com
1433;DATABASE=svc_prod;uid=dbadmin1;pwd=####;AnsiNPW=No

Рис. 7.13 ❖ Anomaly Explorer показывает основную причину проблемы повторного подключения

Факторы влияния, набравшие наибольшее количество баллов за период времени, выбранный на панели инструментов, перечислены в разделе **Top influencers** слева. Для каждого фактора отображается максимальная оценка (в любом сегменте) вместе с общей оценкой фактора за временной диапазон панели мониторинга (суммируется по всем сегментам). И если несколько заданий отображаются вместе, то те факторы влияния, которые являются общими для разных заданий, имеют более высокие суммы, что поднимает их рейтинг выше.

Это довольно важный момент, потому что теперь очень легко увидеть общие черты в аномалиях разных заданий. Например, если `esxserver1.acme.com` – единственный физический хост, который выступает в качестве фактора влияния при просмотре нескольких заданий, нам сразу понятно, на какой машине сосредоточиться; мы знаем, что это не очень распространенная проблема.

В конце концов, заказчик смог устранить проблему, исправив неверную конфигурацию сети и ошибку в коде переподключения к базе данных. Ему удалось довольно быстро исключить рабочие версии и найти основную причину, потому что Elastic ML позволил сузить фокус расследования, тем самым сэкономив время и предотвратив возникновение проблем в будущем.



Рис. 7.14 ❖ Anomaly Explorer показывает наиболее значимые факторы влияния

ЗАКЛЮЧЕНИЕ

Elastic ML, безусловно, может увеличить охват данных, на которые IT-подразделения и организации обращают внимание, и, таким образом, существенно увеличить ценность всего объема собираемых данных. Возможность организовывать данные, находить в них корреляции и комплексно анализировать связанные аномалии по типам данных имеет решающее значение для изоляции проблем и выявления первопричин. Это сокращает время простоя приложения и уменьшает вероятность повторения проблемы.

В следующей главе вы увидите, как другие приложения, входящие в пакет в Elastic Stack (APM, Security и Logs), используют Elastic ML, чтобы обеспечить быстрое развертывание, адаптированное для конкретных случаев использования.



Глава 8

.....

Другие приложения Elastic Stack для обнаружения аномалий



Когда два года назад было опубликовано первое издание этой книги, в Elastic Stack не существовало решений, использующих Elastic ML для поиска аномалий в конкретной предметной области. Однако с тех пор Elastic ML стал *поставщиком* средств обнаружения аномалий для доменных решений, предоставляя индивидуальные конфигурации заданий, которые пользователи могут активировать одним щелчком мыши.

В этой главе мы рассмотрим, что Elastic ML привносит в различные приложения Elastic Stack:

- обнаружение аномалий в Elastic APM;
- обнаружение аномалий в приложении Logs;
- обнаружение аномалий в приложении Metrics;
- обнаружение аномалий в приложении Uptime;
- обнаружение аномалий в приложении Elastic Security.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Информация в этой главе актуальна для версии 7.12 Elastic Stack.



ОБНАРУЖЕНИЕ АНОМАЛИЙ® В ELASTIC APM

Elastic APM выводит мониторинг приложений и управление производительностью на совершенно новый уровень, позволяя пользователям оснащать инструментальными средствами код своих приложений, чтобы получить

детальное представление о производительности отдельных микросервисов и транзакций. В сложных средах это может привести к большому количеству измерений и создать потенциально парадоксальную ситуацию – когда более высокая наблюдаемость достигается за счет детализации измерений, но при этом аналитик, которому приходится тщательно изучать полученные результаты, начинает тонуть в потоке данных.

К счастью, Elastic APM и Elastic ML – это союз, заключенный на небесах. Обнаружение аномалий не только автоматически адаптируется к уникальным характеристикам каждого типа транзакции с помощью машинного обучения без учителя, но также легко масштабируется для обработки больших объемов данных, которые может генерировать APM.

Хотя пользователь всегда может создавать задания по обнаружению аномалий для любых данных временных рядов в любом хранилище, независимо от типа, есть веский аргумент в пользу того, чтобы просто использовать готовые конфигурации заданий для данных Elastic APM, поскольку формат данных уже известен.

Включение обнаружения аномалий для APM

Чтобы воспользоваться преимуществом обнаружения аномалий в данных APM, вам, очевидно, необходимо иметь данные APM, собранные для определенных объявленных служб, и иметь собранные данные, размещенные в хранилищах, доступных через шаблон хранилища `арм-*`.

1. Если вы еще не настроили обнаружение аномалий в своих данных APM, вы увидите индикатор в верхней части экрана, сообщающий, что его необходимо настроить (рис. 8.1).

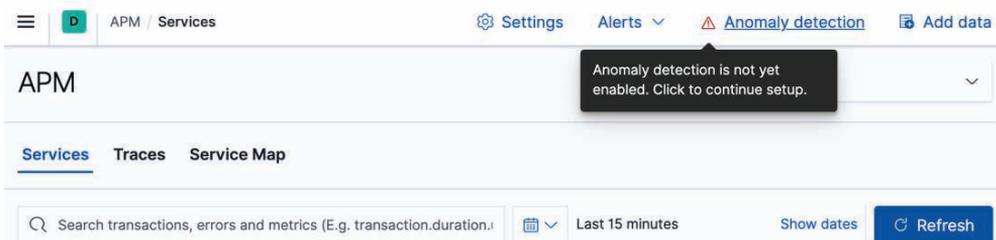


Рис. 8.1 ❖ Индикатор, показывающий, что обнаружение аномалий еще не включено в APM

2. Чтобы продемонстрировать, как будет выглядеть конфигурация обнаружения аномалий, было создано простое приложение в стиле Hello World для Node.js, обслуживаемое Elastic APM. Это приложение (под названием `муарр`) также было помечено тегом среды `dev`, чтобы обозначить, что оно является приложением для разработки (все это делается в агенте APM для конфигурации Node.js) (рис. 8.2).

```

// Add this to the VERY top of the first file loaded in your app
var apm = require('elastic-apm-node').start({

  // Override the service name from package.json
  // Allowed characters: a-z, A-Z, 0-9, -, _, and space
  serviceName: '',

  // Use if APM Server requires a secret token
  secretToken: 'mAh8tevR2KIsq4KQiA',

  // Set the custom APM Server URL (default: http://localhost:8200)
  serverUrl: 'https://103497ec58cc41c49128f066aba2fcd8.apm.westus2.azure.elastic-cloud.com:443',

  // Set the service environment
  environment: 'dev'
});

var express = require('express');
var app = express();
app.get('/', function (req, res) {
  res.send('Hello World!');
});
app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});

```

Рис. 8.2 ❖ Пример приложения Node.js, оснащенного Elastic APM

3. При просмотре внутри Elastic APM служба будет выглядеть, как показано на рис. 8.3.

The screenshot shows the Elastic APM 'Services' page. At the top, there are navigation links for 'Settings', 'Alerts', 'Anomaly detection', and 'Add data'. The main heading is 'APM' with a dropdown menu for 'Environment' set to 'All'. Below this, there are tabs for 'Services', 'Traces', and 'Service Map'. A search bar is present with the text 'Search transactions, errors and metrics (E.g. tra...'. To the right of the search bar, there is a date selector set to 'Last 15 minutes', a 'Show dates' button, and a 'Refresh' button. On the left side, there is a 'Filters' section with 'HOST' and 'AGENT NAME' filters. The main content area displays a table with the following data:

Name	Environment	Latency (avg.)	Throughput ↓	Error rate %
myapp	dev	3.2 ms	0.3 tpm	0%

At the bottom right of the table, there is a pagination control showing '< 1 >'.

Рис. 8.3 ❖ Пример приложения Node.js в пользовательском интерфейсе Elastic APM

4. Чтобы включить обнаружение аномалий для этой службы, просто перейдите в **Settings** (Настройки), нажмите **Anomaly detection** (Обнаружение аномалий), а затем нажмите кнопку **Create ML Job** (Создать задание машинного обучения) (рис. 8.4).

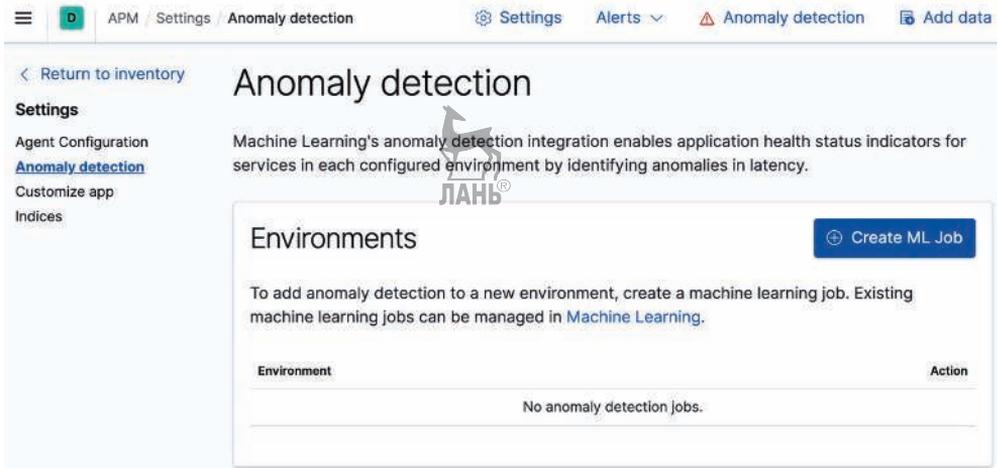


Рис. 8.4 ❖ Создание заданий машинного обучения для данных APM

5. Затем укажите имя среды, для которой вы хотите создать задание (рис. 8.5).

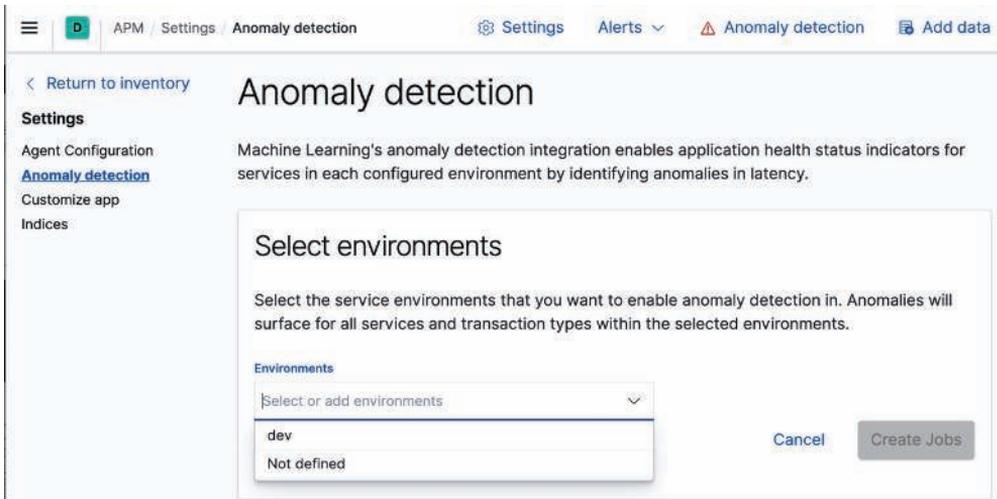


Рис. 8.5 ❖ Указание среды, для которой создается задание машинного обучения

6. Здесь мы выберем `dev`, так как это единственный вариант, доступный для нашего приложения. После выбора среды и нажатия кнопки `Create Jobs` (Создать задание) мы получаем подтверждение, что задание было создано и оно указано в таблице (рис. 8.6).

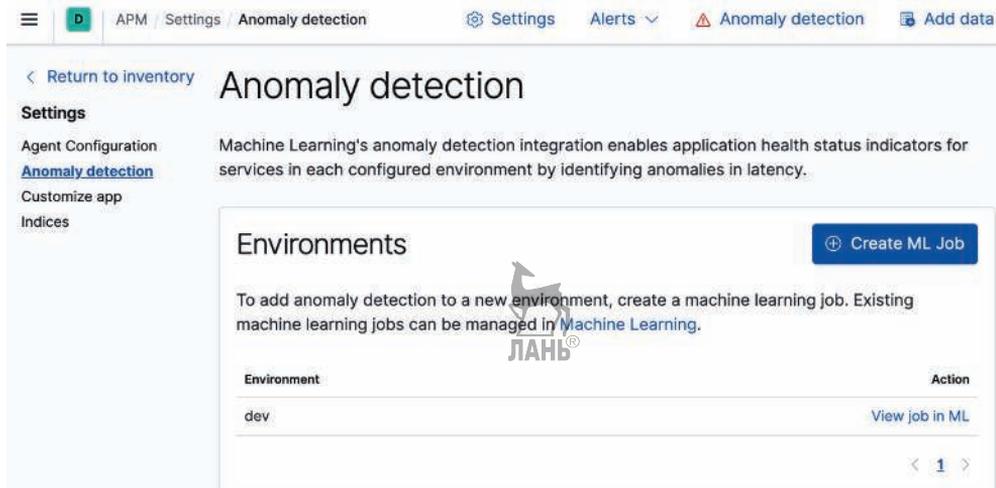


Рис. 8.6 ❖ Список созданных заданий обнаружения аномалий в APM

- Если мы перейдем в приложение Elastic ML, чтобы просмотреть сведения о задании, которое было создано для нас, то увидим экран как на рис. 8.7.

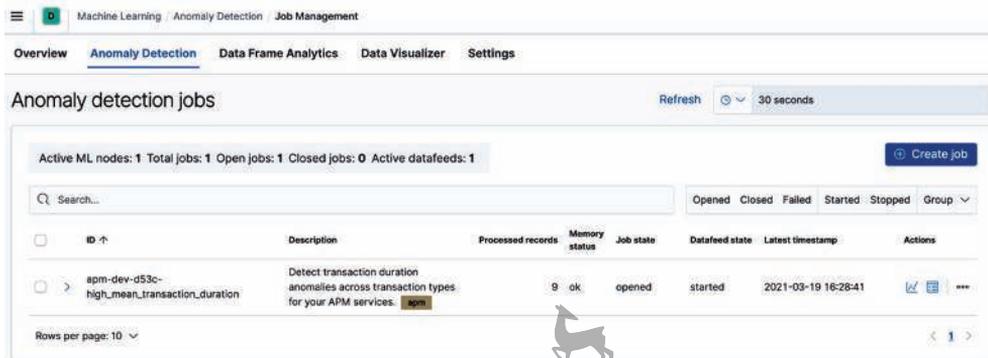


Рис. 8.7 ❖ Список созданных заданий по обнаружению аномалий в ML

- Продолжая изучать задание, мы можем увидеть фактическую конфигурацию детектора (рис. 8.8).

The screenshot shows the configuration page for a detector in the APM interface. The main configuration area is titled 'Job config' and includes the following details:

- Detector:** `high_mean("transaction.duration.us") by "transaction.type" partition_field_name="service.name"`
- Data description:** `high duration by transaction type for an APM service`
- Influencers:** `transaction.type` and `service.name`
- Analysis config:** `bucket_span` is set to `15m`

Рис. 8.8 ❖ Конфигурация детектора для задания APM

Обратите внимание, что при этом используется функция `high_mean` поля `transaction.duration.us`, разделенного как по `transaction.type`, так и по `service.name`. Также заметим, что значение длительности сегмента `bucket_span` для данного задания составляет 15 минут, что может быть, а может и не быть идеальным параметром для вашей среды.

❗ Следует отметить, что эта конфигурация является единственно возможной при использовании подхода одним щелчком из пользовательского интерфейса APM, поскольку конфигурация жестко запрограммирована. Если вы хотите настроить или создать свои собственные конфигурации, вы можете либо создать их с нуля, либо клонировать это задание и установить свой собственный диапазон сегмента и/или логику детектора.

9. Мы можем щелкнуть вкладку **Datafeed**, чтобы увидеть следующие настройки потока запрашиваемых данных (рис. 8.9).

The screenshot shows the configuration page for a datafeed in the APM interface. The main configuration area is titled 'Datafeed' and includes the following details:

- Datafeed ID:** `datafeed-apm-dev-d53c-high_mean_transaction_duration`
- Job ID:** `apm-dev-d53c-high_mean_transaction_duration`
- Query:** `{ "bool": { "filter": [{ "term": { "processor.event": "transaction" } }, { "exists": { "field": "transaction.duration.us" } }, { "term": { "service.environment": "dev" } }] } }`
- Indices:** `apm-*`
- Scroll size:** `1000`
- Delayed data check config:** `{ "enabled": true }`
- State:** `started`

The 'Timing stats' section shows the following values:

- `job_id`: `apm-dev-d53c-high_mean_transaction_duration`
- `search_count`: `0`
- `bucket_count`: `3`
- `average_search_time_per_bucket_ms`: `29.333`
- `exponential_average_search_time_per_hour_ms`: `88`

Рис. 8.9 ❖ Конфигурация потока данных для задания APM

Обратите внимание, что мы запрашиваем только те документы, которые соответствуют "service.environment" : "dev", как мы указали при настройке задания в пользовательском интерфейсе APM.

- Мы можем нажать на вкладку **Datafeed preview** (Предварительный просмотр потока данных), чтобы просмотреть образец наблюдений, которые будут переданы в Elastic ML для этого конкретного задания, как показано на рис. 8.10.

ID ↑	Description	Processed records	Memory status	Job state	Datafeed state	Latest timestamp	Actions
apm-dev-d53c-high_mean_transaction_duration	Detect transaction duration anomalies across transaction types for your APM services. apm	31	ok	opened	started	2021-03-19 16:50:00	View Refresh More

Job settings Job config Datafeed Counts JSON Job messages **Datafeed preview** Forecasts Annotations Model snapshots

```

1 - [
2   {
3     "@timestamp": 1616184554393,
4     "service.name": "myapp",
5     "transaction.duration.us": 10095,
6     "transaction.type": "request"
7   },
8   {
9     "@timestamp": 1616184554597,
10    "service.name": "myapp",
11    "transaction.duration.us": 1383,
12    "transaction.type": "request"
13  },
14  {
15    "@timestamp": 1616184563812,
16    "service.name": "myapp",
17    "transaction.duration.us": 1876,
18    "transaction.type": "request"
19  },
20 ]

```

Рис. 8.10 ❖ Предварительный просмотр потока данных для задания APM

Теперь, когда задание APM настроено и запущено, давайте посмотрим, где в пользовательском интерфейсе APM будут отражаться результаты задания по обнаружению аномалий.

Просмотр результатов задания по обнаружению аномалий

Помимо просмотра результатов задания в пользовательском интерфейсе Elastic ML, есть три ключевых места, в которых результаты заданий по обнаружению аномалий отображаются в пользовательском интерфейсе APM:

- **обзор служб** – это представление в пользовательском интерфейсе APM является индикатором работоспособности высокого уровня, который отражает максимальную оценку аномалий в результатах поиска аномалий для этой конкретной службы (рис. 8.11);

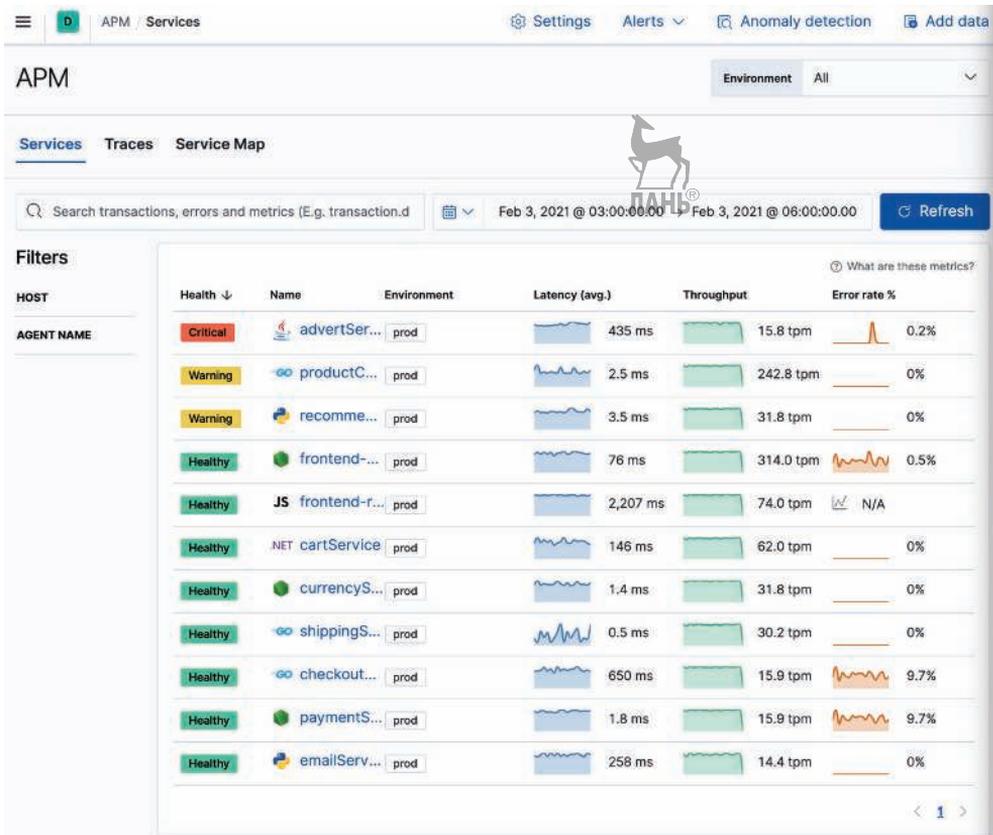


Рис. 8.11 ❖ Обзор служб в пользовательском интерфейсе APM

- **карта служб** – это динамическое представление показывает зависимости транзакций приложения с цветовым индикатором аномалии, основанным на максимальной оценке аномалий в результатах обнаружения аномалий для этой конкретной службы (рис. 8.12);
- **график продолжительности транзакции** – эта диаграмма под основным представлением транзакций в APM показывает ожидаемые границы и цветную аннотацию, когда оценка аномалии превышает 75 для определенного типа транзакции.

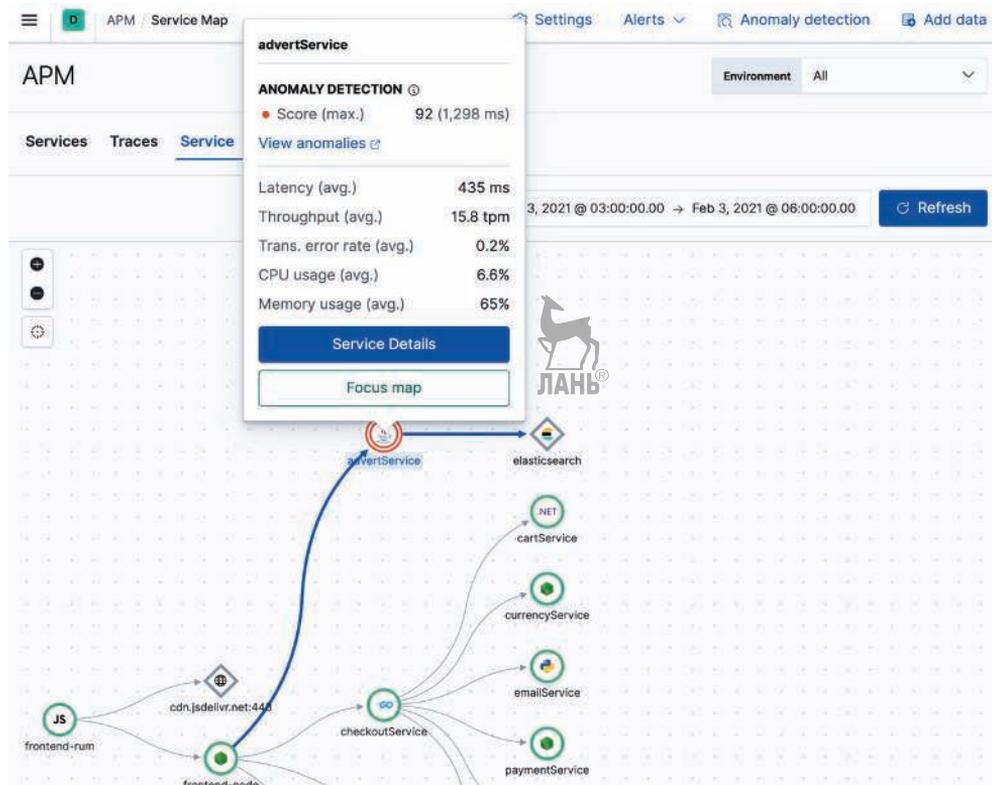


Рис. 8.12 ❖ Карта служб в пользовательском интерфейсе APM

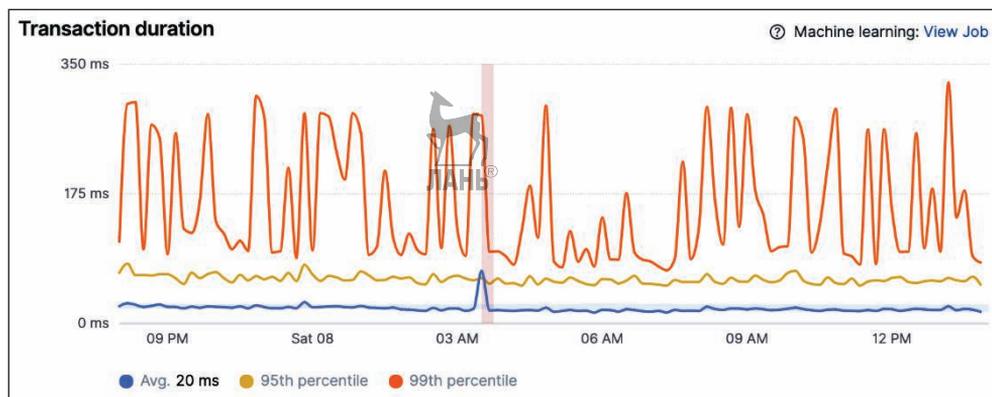


Рис. 8.13 ❖ График продолжительности транзакции в пользовательском интерфейсе APM

Эти полезные индикаторы аномальности помогают пользователю в дальнейшем исследовании и устранении проблем с использованием широких возможностей пользовательского интерфейса APM и ML. Далее мы рассмотрим еще один способ интеграции Elastic ML с данными из APM, используя *распознаватель данных*.

Создание заданий машинного обучения с помощью распознавателя данных

Хотя «распознаватель данных» (the data recognizer) не является официальным маркетинговым названием функции в Elastic ML, на самом деле он может быть очень полезен, потому что помогает пользователю создавать предварительно настроенные задания для распознаваемых данных.



Предварительно настроенные задания для распознавателя данных определены и хранятся в следующем репозитории GitHub: https://github.com/elastic/kibana/tree/master/x-pack/plugins/ml/server/models/data_recognizer/modules.

Рабочий процесс создания нового задания с помощью распознавателя выглядит следующим образом.

Если в процессе создания задания на обнаружение аномалий входные данные задания (шаблон хранилища или сохраненный поиск) соответствуют шаблону поиска, известным одному из предопределенных модулей распознавания, то вы можете предложить пользователю создать одно из таких предопределенных заданий. В качестве иллюстрации можно взять типичный пример Node.js, представленный ранее в этой главе. Вместо того чтобы создавать задание обнаружения аномалий из пользовательского интерфейса APM, можно пройти через пользовательский интерфейс Elastic ML и выбрать шаблон хранилища `арм-*`:

- выбрав шаблон хранилища, вы увидите два предварительно настроенных задания, предлагаемых в дополнение к обычным мастерам заданий (рис. 8.14);
- первое задание с названием **APM** – это задание того же типа, которое мы уже создали ранее в этой главе из пользовательского интерфейса APM. Второй вариант (**APM: Node.js**) на самом деле представляет собой набор из трех заданий (рис. 8.15);
- третье задание – это снова тот же тип задания, который мы уже создали ранее в этой главе из пользовательского интерфейса APM, но два других уникальны. Идея предлагать пользователям варианты готовых заданий, если исходные данные «распознаны», не уникальна для данных APM, и вы можете увидеть эти задания в других ситуациях или сценариях использования (например, при выборе хранилищ для образцов данных Kibana, данных из nginx и т. д.).

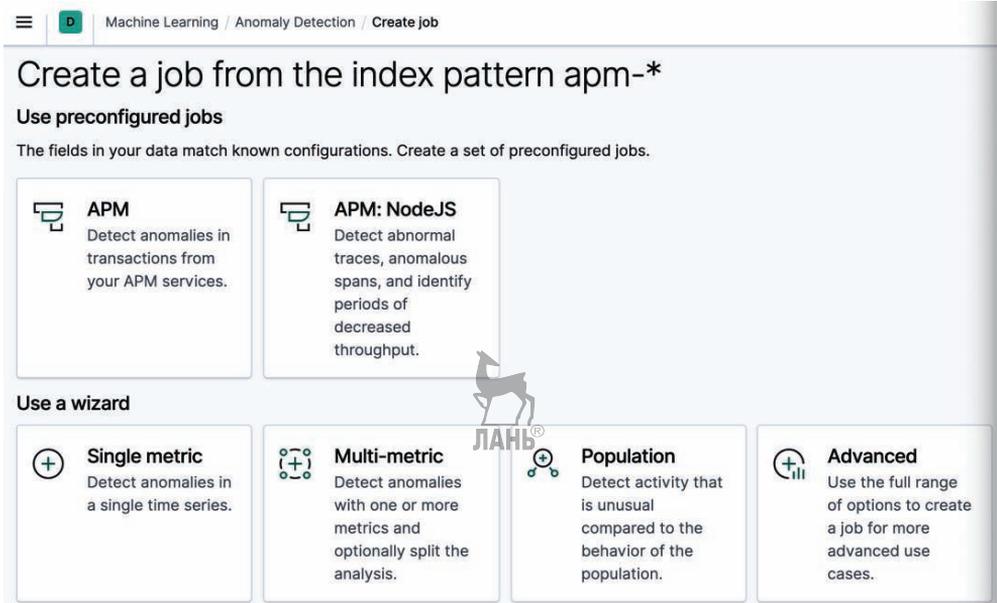


Рис. 8.14 ❖ Задания, предлагаемые распознавателем данных

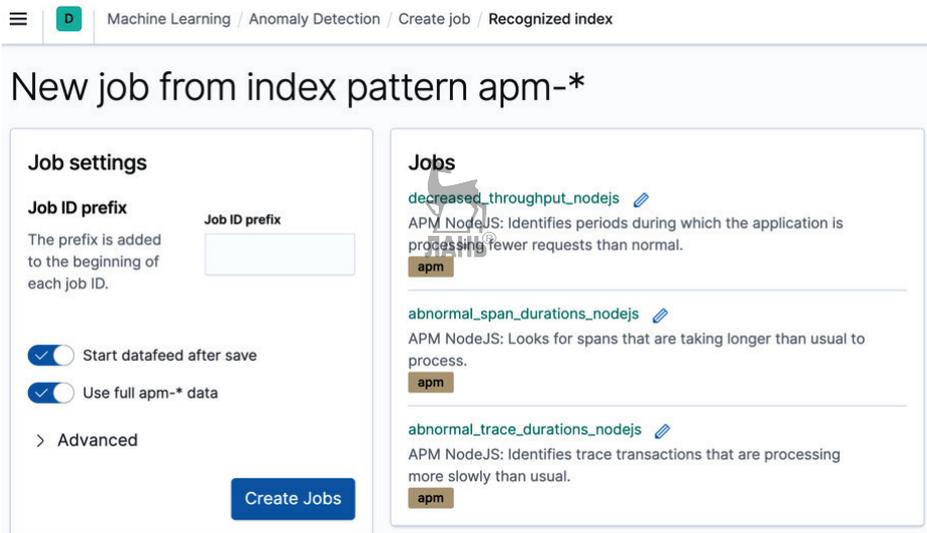


Рис. 8.15 ❖ Задания Node.js, предлагаемые распознавателем данных

Теперь, когда вы увидели, как Elastic ML встроен в приложение APM, давайте выясним, как обнаружение аномалий используется приложением Logs.

ОБНАРУЖЕНИЕ АНОМАЛИЙ В ПРИЛОЖЕНИИ LOGS

Приложение Logs в разделе **Observability** интерфейса Kibana предлагает такое же представление ваших данных, как и приложение Discover. Тем не менее пользователям, которые больше ценят возможность просмотра своих журналов в реальном времени независимо от хранилищ, в которых они расположены, понравится приложение Logs (рис. 8.16).

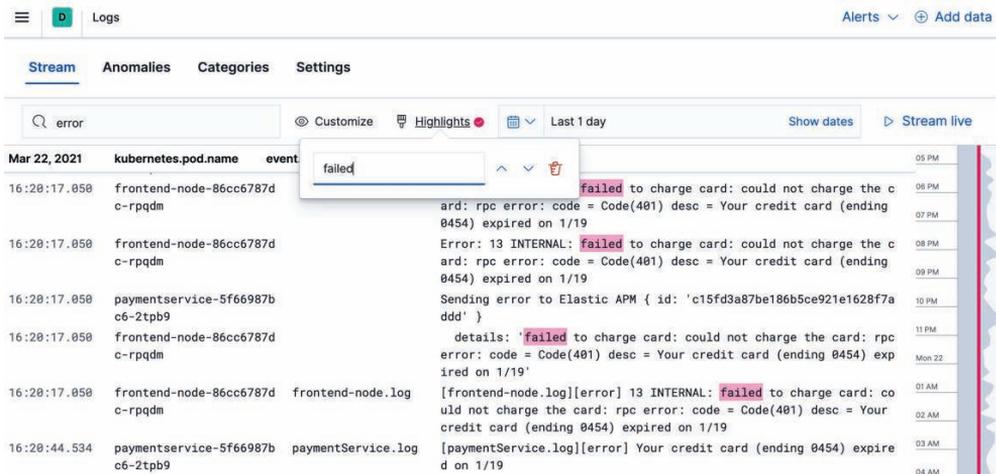


Рис. 8.16 ❖ Приложение Logs – часть раздела **Observability** в Kibana

Обратите внимание, что на экране есть вкладки **Anomalies** (Аномалии) и **Categories** (Категории). Давайте начнем с раздела **Categories**.

Категории журналов

Категоризация Elastic ML, впервые показанная еще в главе 3, применяется в качестве общего подхода к любому хранилищу неструктурированных данных журнала. Однако в приложении Logs категоризация используется с некоторыми более строгими ограничениями на данные. Ожидается, что данные будут в представлении в формате обобщенной схемы данных Elastic (Elastic Common Schema, ECS) с набором определенных полей (особенно полем `event.dataset`).

! Набор данных журналов из главы 7 дублируется для этой главы в репозитории GitHub для использования с приложением Logs, но с добавлением поля `event.dataset`. Если вы импортируете набор через средство загрузки файлов в ML, обязательно замените имя поля на `event.dataset` вместо предлагаемого по умолчанию `event_dataset`.

Причину этого ограничения можно понять, учитывая, что приложение Logs пытается создать для вас задание категоризации шаблонным способом. Следовательно, необходимо быть уверенным в соблюдении соглашения об именах полей. Очевидно, что на это нельзя рассчитывать, если задание создано в приложении ML, где ответственность за объявление имен поля категоризации и поля сообщения лежит на пользователе.

Если вы все же настроите приложение Logs для выполнения категоризации, то результат будет выглядеть примерно так, как на рис. 8.17, на котором показаны все отдельные категории журналов, отсортированные по максимальной оценке аномалии.

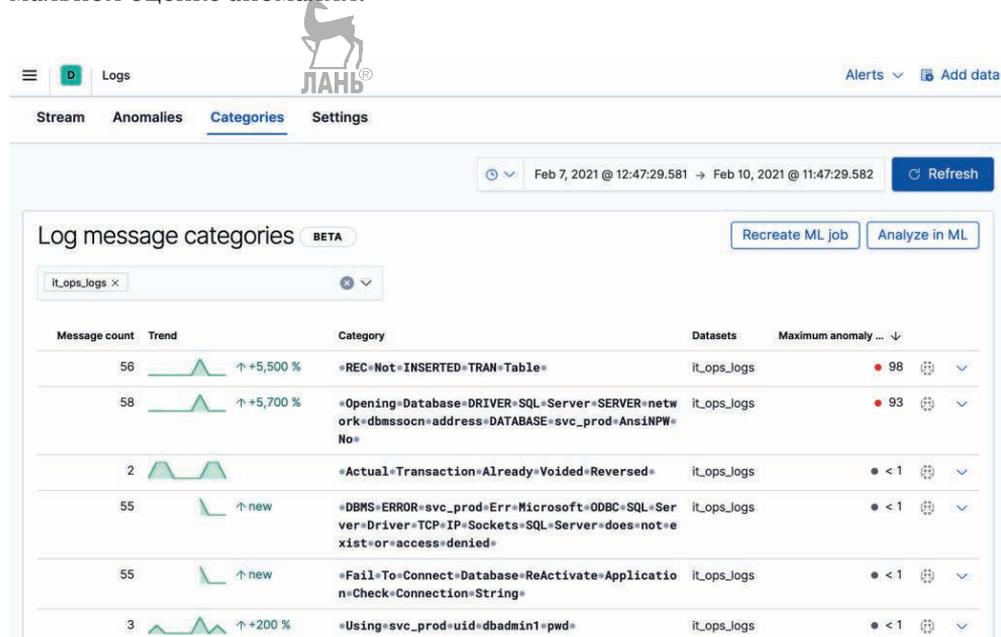


Рис. 8.17 ❖ Приложение Logs, отображающее результаты категоризации из Elastic ML

Затем пользователи могут нажать кнопку **Analyze in ML** (Анализировать в машинном обучении), чтобы перейти к обозревателю аномалий Anomaly Explorer в пользовательском интерфейсе машинного обучения для дальнейшей проверки.

Журнал аномалий

Раздел **Anomalies** приложения Logs обеспечивает представление, аналогичное обзору Anomaly Explorer (рис. 8.18).

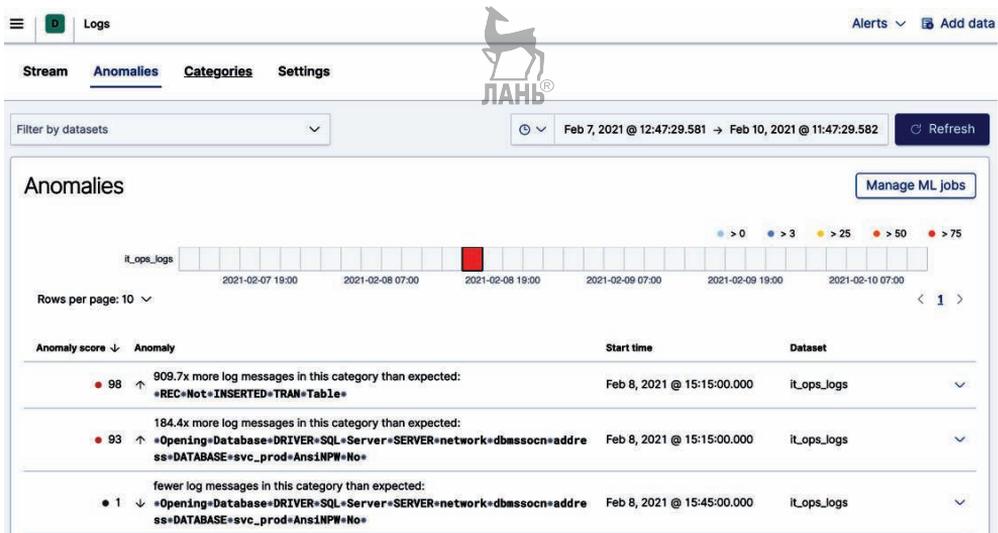


Рис. 8.18 ❖ Приложение Logs отображает представление, подобное обозревателю аномалий в ML

Этот раздел также позволяет пользователю управлять заданиями по обнаружению аномалий, если нажать кнопку **Manage ML jobs** (Управление заданиями машинного обучения) (рис. 8.19).

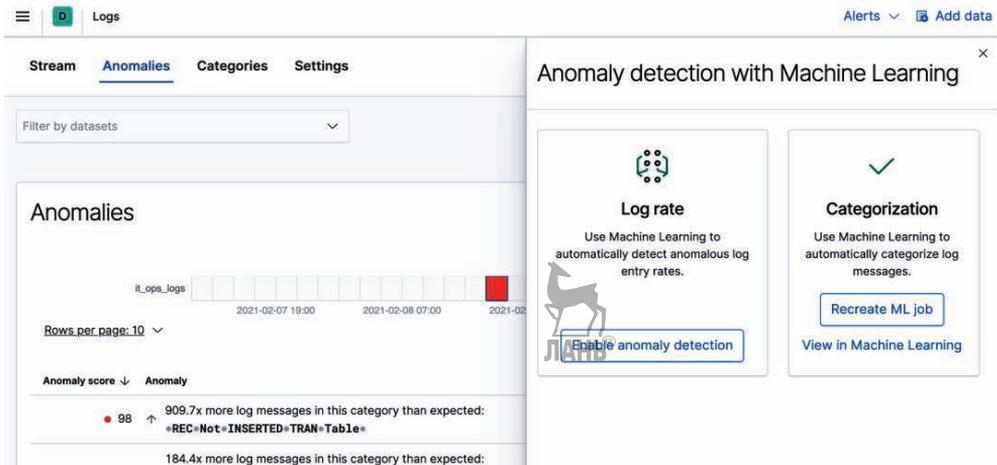


Рис. 8.19 ❖ Приложение Logs позволяет управлять заданиями ML

Задание **Log rate** (Скорость наполнения журнала), отображаемое в левой части рис. 8.18, представляет собой простой детектор count на основе поля event.dataset.

Пользователи должны помнить, что эти задания машинного обучения здесь можно только сбросить, но не удалить окончательно – чтобы удалить

эти задания, вы должны перейти на страницу управления заданиями по обнаружению аномалий в приложении машинного обучения.

Очевидно, что опытный пользователь может заниматься созданием и управлением заданиями по обнаружению аномалий для своих журналов в приложении ML. Но приятно, что приложение Logs раскрывает возможности Elastic ML таким образом, что эти функции становятся очевидными и легко реализуемыми. Давайте продолжим эту тенденцию и посмотрим, как Elastic ML используется в приложении Metrics.

Обнаружение аномалий в приложении Metrics

Приложение Metrics, также являющееся частью раздела **Observability** интерфейса Kibana, позволяет пользователям просматривать свои данные с точки зрения ресурсов и метрик:

- в представлении **Inventory** (Доступные ресурсы) пользователи видят общую карту отслеживаемых ресурсов. Такие объекты, как хосты, модули или контейнеры, могут быть организованы и отфильтрованы для настройки представления, включая шкалу состояния с цветовой кодировкой, как показано на рис. 8.20.

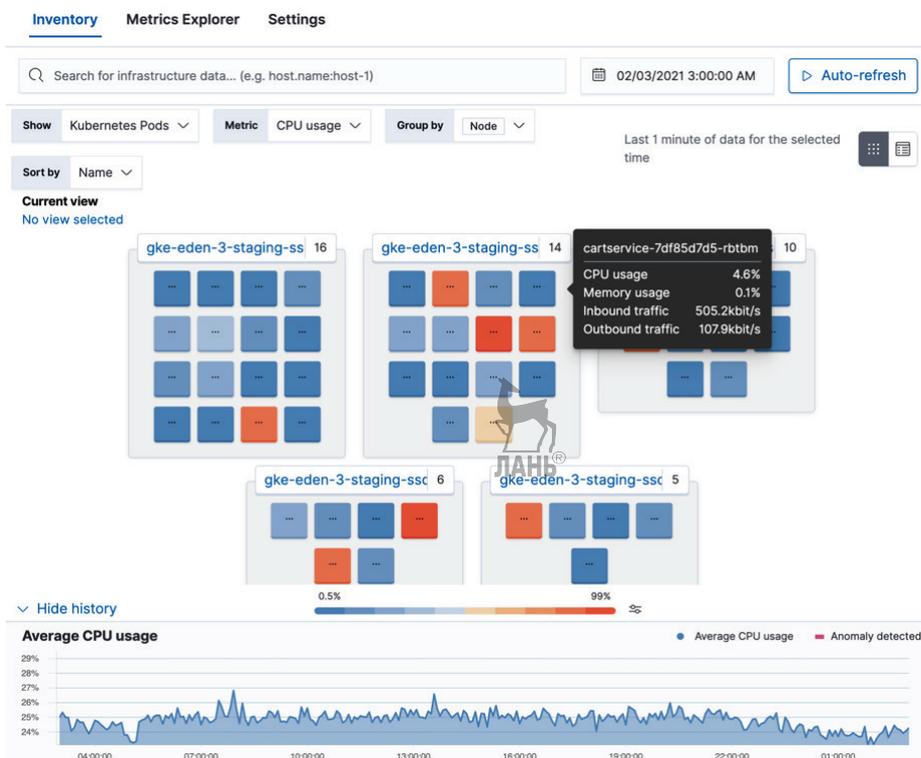


Рис. 8.20 ❖ Представление Inventory в приложении Metrics

Обратите внимание, что на нижней панели будут отображаться обнаруженные аномалии. В настоящее время это единственное место для просмотра аномалий в приложении Metrics;

- чтобы включить встроенные задания по обнаружению аномалий через приложение Metrics, нажмите кнопку **Anomaly detection** (Обнаружение аномалий) вверху, чтобы активировать всплывающее меню конфигурации (рис. 8.21);

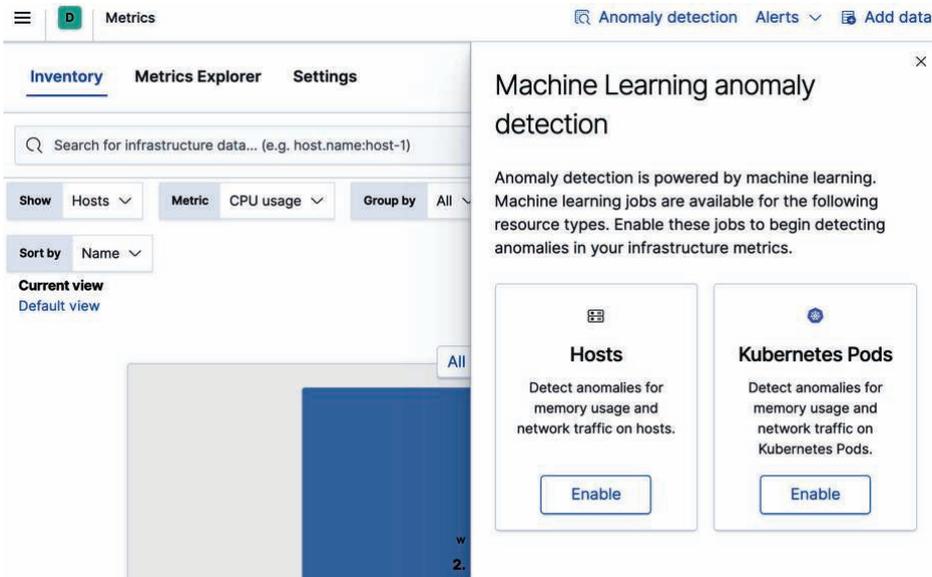


Рис. 8.21 ❖ Управление заданиями по обнаружению аномалий в приложении Metrics

- если, например, мы решим включить обнаружение аномалий для хостов, нажатие соответствующей кнопки **Enable** (Включить) покажет конфигурацию как на рис. 8.22;
- обратите внимание, что в случае необходимости поле раздела предлагается как часть конфигурации. Когда вы нажимаете кнопку **Enable jobs** (Включить задания), от вашего имени для вас создаются три разных задания по обнаружению аномалий, которые можно просмотреть в разделе **Job Management** пользовательского интерфейса Elastic ML (рис. 8.23);
- чтобы установить минимальный балл аномалии, необходимый для отображения аномалий в приложении Metrics, перейдите на вкладку **Settings** (Настройки) и настройте параметр **Anomaly Severity Threshold** (Порог серьезности аномалии) (рис. 8.24).

Enable machine learning for hosts

Settings can not be changed once the jobs are created. You can recreate jobs anytime, however, the previously detected anomalies are removed.

When does your model begin?

By default, machine learning jobs analyze the last 4 weeks of data and continue to run indefinitely.

How do you want to partition your data?

Partitions enable you to build independent models for groups of data that share similar behavior. For example, you can partition by machine type or cloud availability zone.

Filter

By default, machine learning jobs analyze all of your metric data.

Start date: 02/23/2021 03:27 PM

Partition field: host.name

Filter (optional): Search for infrastructure data...

Cancel **Enable jobs**

Show history 0%

Рис. 8.22 ❖ Конфигурация обнаружения аномалий на хостах в приложении Metrics

Machine Learning Anomaly Detection Job Management

Overview **Anomaly Detection** Data Frame Analytics Data Visualizer Settings

Anomaly detection jobs Refresh 30 seconds

Active ML nodes: 1 Total jobs: 5 Open jobs: 5 Closed jobs: 0 Active datafeeds: 5 **Create job**

groups:(metrics)

ID	Description	Processed records	Memory status	Job state	Datafeed state	Latest timestamp	Actions
kibana-metrics-ui-default-default-hosts_memory_usage	Metrics: Hosts - Identify unusual spikes in memory usage across hosts.	79	ok	opened	started	2021-03-23 15:37:20	🔍 📄 ⋮
kibana-metrics-ui-default-default-hosts_network_in	Metrics: Hosts - Identify unusual spikes in inbound traffic across hosts.	4	ok	opened	started	2021-03-23 15:39:50	🔍 📄 ⋮
kibana-metrics-ui-default-default-hosts_network_out	Metrics: Hosts - Identify unusual spikes in outbound traffic across hosts.	4	ok	opened	started	2021-03-23 15:39:50	🔍 📄 ⋮

Rows per page: 10

Рис. 8.23 ❖ Задания по обнаружению аномалий для хостов, созданные приложением Metrics

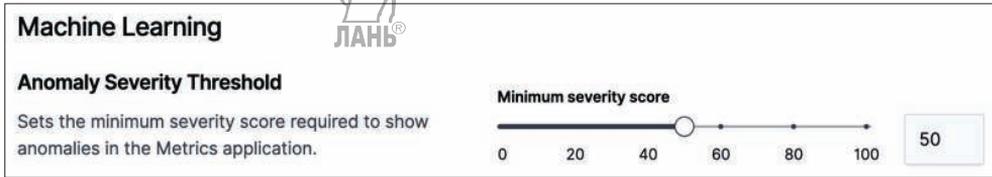


Рис. 8.24 ❖ Настройка порога серьезности аномалии в приложении Metrics

Интеграция Elastic ML с приложением Metrics помогает пользователю быстро начать использовать обнаружение аномалий в своих данных метрик. Давайте теперь кратко рассмотрим интеграцию с приложением Uptime.

ОБНАРУЖЕНИЕ АНОМАЛИЙ В ПРИЛОЖЕНИИ UPTIME

Приложение Uptime позволяет отслеживать доступность и время отклика сервисов с помощью различных сетевых протоколов, включая HTTP/S, TCP и ICMP.

1. Приложение Uptime, которое часто относят к инструментам *синтетического мониторинга*, использует Heartbeat для активного зондирования конечных точек сети из одного или нескольких мест (рис. 8.25).

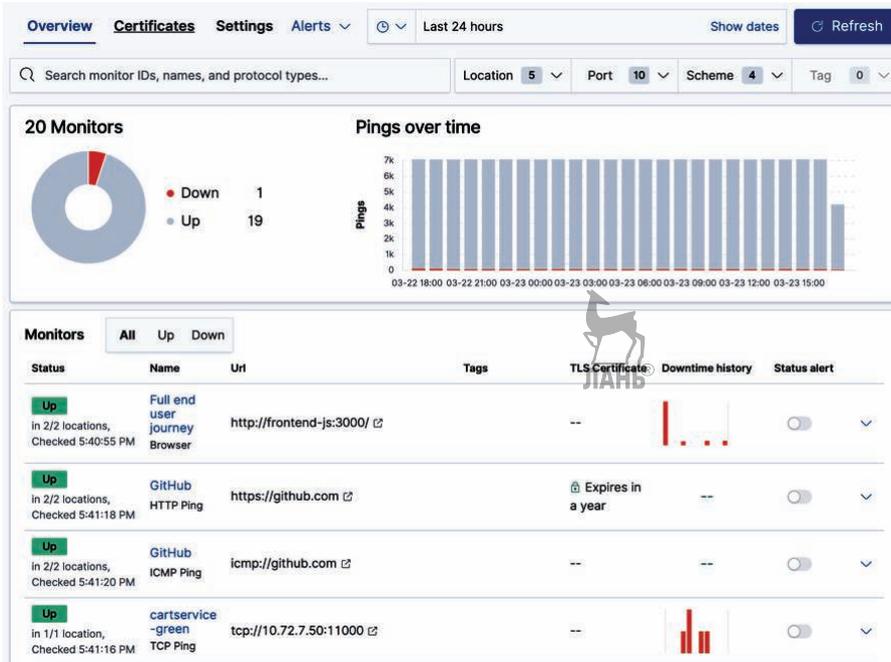


Рис. 8.25 ❖ Приложение Uptime в Kibana

- Если вы хотите включить обнаружение аномалий для конкретного монитора, просто нажмите на имя монитора, чтобы просмотреть подробные сведения о нем. Вы увидите кнопку **Enable anomaly detection** (Включить обнаружение аномалий) на панели **Monitor duration**, отображающей результаты в ходе мониторинга (рис. 8.26).

currencyservice-green Enable status alert Alerts ▾

🕒 Last 24 hours Show dates Refresh

Up in 1 location

Overall availability **99.97 %**

Url <tcp://10.72.7.41:6000>

Monitor ID **auto-tcp-0X54A370C321A56B9F**

Type **tcp**

Tags [Set tags](#)

Monitoring from

Location	Availability	Last check
nyc-gcp-us-east1	99.97 %	5:57:56 PM

Monitor duration [Enable anomaly detection](#)

Pings over time

History All Up Down Location 5 ▾

Status	Location	IP	Duration	Error
Up	nyc-gcp-us-east1	10.72.7.41	0 ms	--

Checked 5:57:56 PM

Рис. 8.26 ❖ Включение обнаружения аномалий для монитора времени работоспособности

- Нажатие кнопки **Enable anomaly detection** создает задание в фоновом режиме и предлагает пользователю возможность создать предупреждение об аномалиях, обнаруженных в задании (рис. 8.27).
- Как только задание на обнаружение аномалий станет доступным, любые обнаруженные аномалии также будут отображаться на странице сведений о мониторе на панели **Monitor duration** (рис. 8.28).

currencyservice-green Enable status alert

Last 24 hours Show dates

Up in 1 location

Overall availability **99.97 %**

Uri <tcp://10.72.7.41:6000>

Monitor ID **auto-tcp-0X54A370C321A56B9F**

Type **tcp**

Tags [Set tags](#)

Monitor duration (Anomalies: 0) [Anomaly detection](#)

History All Up Down Location

Status	Location	Checked
Up	nyc-gcp-us-east1	6:00:16 PM
Up	nyc-gcp-us-east1	6:00:06 PM

Create alert

Name

Tags (optional)

Check every

Notify

Uptime Duration Anomaly

Alert when the Uptime monitor duration is anomalous. [Documentation](#)

WHEN MONITOR **auto-tcp-0X54A370C321A56B9F**
HAS ANOMALY WITH SEVERITY ● Critical

Actions

Select an action type

Email

IBM Resilient

Index

Jira

Microsoft Teams

PagerDuty

Server log

ServiceNow

Cancel Save

Рис. 8.27 ❖ Создание оповещения для задания обнаружения аномалий в приложении Uptime

Full end user journey Enable status alert

Last 2 days Show dates Refresh

Up in 2 locations

Overall availability **97.47 %**

Uri <http://frontend-js:3000/>

Monitor ID **Full end user journey**

Type **browser**

Tags [Set tags](#)

Monitor duration (Anomalies: 6) [Anomaly detection](#)

Monitoring from

Location	Availability	Last check
New York	97.60 %	6:04:57 PM
New York Mobile/3G	97.33 %	6:04:54 PM

Pings over time

Рис. 8.28 ❖ Аномалии, отображаемые в приложении Uptime

И снова интеграция Elastic ML с другим приложением из группы Observability в Elastic Stack позволяет пользователям невероятно легко воспользоваться преимуществами сложного обнаружения аномалий. Но мы также знаем, что Elastic ML может делать кое-какие интересные вещи в отношении анализа популяций и редких событий. В следующем разделе вас ждет интеграция ML с Elastic SIEM.

ОБНАРУЖЕНИЕ АНОМАЛИЙ В ПРИЛОЖЕНИИ ELASTIC SECURITY



Elastic Security – это настоящая ^{квинтэссенция} целевого приложения в Elastic Stack. Созданное с нуля для реализации рабочего процесса аналитика по безопасности, всеобъемлющее приложение Elastic Security само по себе достойно отдельной книги. Однако в основе приложения Elastic Security лежит функция обнаружения, в которой правила, созданные пользователем и Elastic, применяются для создания оповещений при выполнении условий, заданных в правилах. Как вы увидите чуть позже, Elastic ML играет важную роль в функции обнаружения.

Готовые задания по обнаружению аномалий

Большинство правил обнаружения в Elastic Security статичны, но многие из них поддерживаются предварительно созданными заданиями обнаружения аномалий, которые работают с данными, собранными из Elastic Agent или Beats, или эквивалентными данными, соответствующими полям ECS, применимым для каждого типа задания. Чтобы увидеть полный список заданий по обнаружению аномалий, предоставляемых Elastic, просмотрите определение конфигурации потока данных и задания в папках `security_*` и `siem_*` в следующем репозитории GitHub: https://github.com/elastic/kibana/tree/7.12/x-pack/plugins/ml/server/models/data_recognizer/modules.

(Вы можете подставить номер последней версии выпуска на место, которое сейчас занимает 7.12.)

Проницательный читатель заметит, что многие заранее созданные задания используют либо популяционный анализ, либо обнаружение редких событий. Каждый из этих способов обнаружения аномалий хорошо согласуется с целями аналитика по безопасности – где обнаружение нового поведения или поведения, которое выделяет пользователей или объекты из толпы, часто является признаком компрометации системы.

Предварительно созданные задания по обнаружению аномалий доступны для просмотра на вкладке **Detections** (Обнаружения) в Elastic Security и имеют тег ML (рис. 8.29).

Если нажать на **ML job settings** (Настройки задания ML) в правом верхнем углу экрана, откроется список настроек, в котором пользователь сможет увидеть все задания в библиотеке – даже те, которые недоступны (отмечены значком предупреждения) (рис. 8.30).

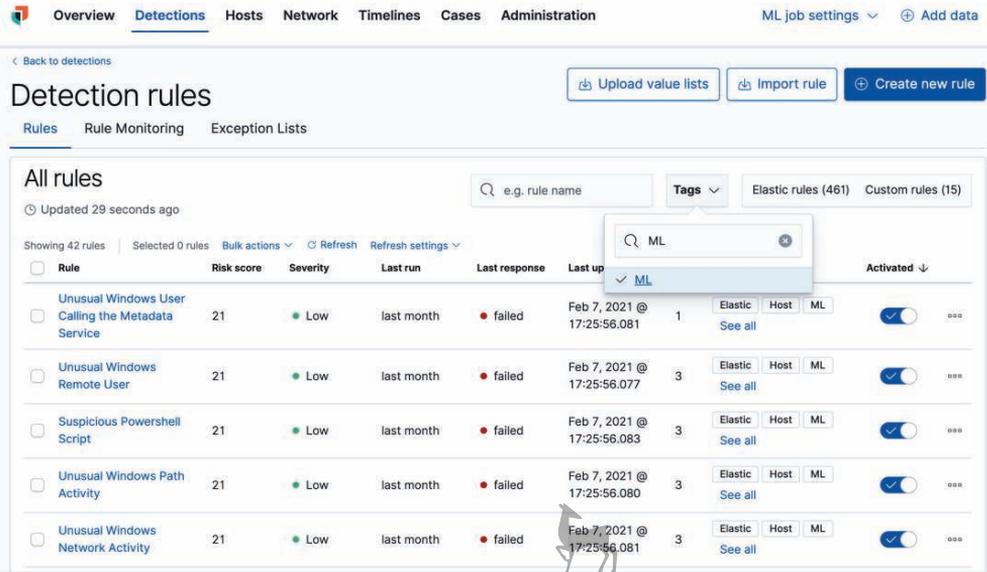


Рис. 8.29 ❖ Задания машинного обучения в разделе правил обнаружения приложения Security

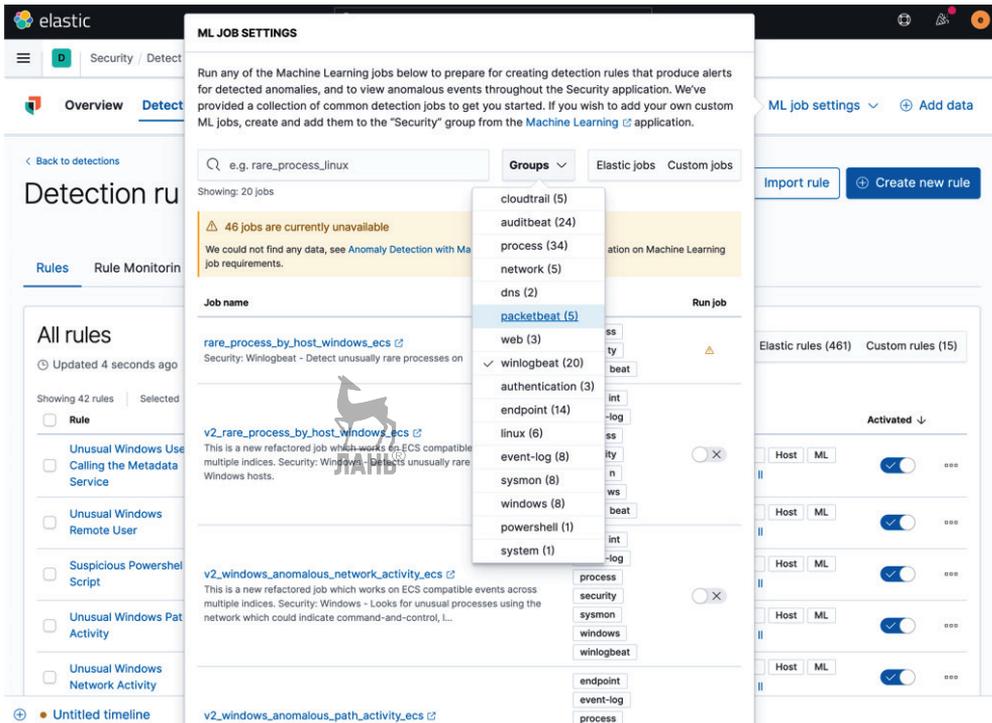


Рис. 8.30 ❖ Список заданий в разделе ML job settings приложения Security

Задания помечаются как недоступные, если необходимые данные в настоящее время не размещены в хранилищах Elasticsearch. Если задание доступно, вы можете активировать его, нажав на переключатель, и задание по обнаружению аномалий будет выполнено в фоновом режиме. Конечно, вы всегда можете просмотреть задания, созданные в пользовательском интерфейсе управления заданиями Elastic ML (рис. 8.31).

The screenshot displays the 'Anomaly detection jobs' page in the Elastic ML interface. At the top, there are navigation tabs: Overview, Anomaly Detection (selected), Data Frame Analytics, Data Visualizer, and Settings. Below the tabs, there's a 'Refresh' button and a refresh interval set to '30 seconds'. A summary bar indicates 'Active ML nodes: 1 Total jobs: 8 Open jobs: 1 Closed jobs: 7 Active datafeeds: 1' and a 'Create job' button. A search bar is present above the table. The table has columns: ID, Description, Processed records, Memory status, Job state, Datafeed state, Latest timestamp, and Actions. Two jobs are listed:

ID	Description	Processed records	Memory status	Job state	Datafeed state	Latest timestamp	Actions
v2_rare_process_by_host_windows_ecs	This is a new refactored job which works on ECS compatible events across multiple indices. Security: Windows - Detects unusually rare processes on Windows hosts. endpoint	1,293	ok	opened	started	2021-02-07 15:54:39	🔗 📄 ⋮
v2_windows_anomalous_network_activity_ecs	This is a new refactored job which works on ECS compatible events across multiple indices. Security: Windows - Looks for unusual processes using the network which could indicate command-and-control, lateral movement, persistence, or data exfiltration activity. endpoint	11,086	ok	closed	stopped	2021-02-07 15:59:59	🔗 📄 ⋮

Рис. 8.31 ❖ Задания Elastic ML, созданные Security и представленные в пользовательском интерфейсе управления заданиями

Теперь, когда вы знаете, как включить задания по обнаружению аномалий, пора посмотреть, как они создают оповещения об обнаружении для приложения Security.

Оповещения на основе заданий обнаружения аномалий

Если вы вернетесь к представлению **Detections**, показанному на рис. 8.28, и нажмете кнопку **Create new rule** (Создать новое правило), то увидите, что можете выбрать **Machine Learning** в качестве типа правила (рис. 8.32).

Если вы захотите выбрать конкретное задание машинного обучения с помощью раскрывающегося списка, вам также будет предложено выбрать пороговое значение оценки аномалии, которое необходимо превысить, чтобы вызвать оповещение об обнаружении аномалии (рис. 8.33).

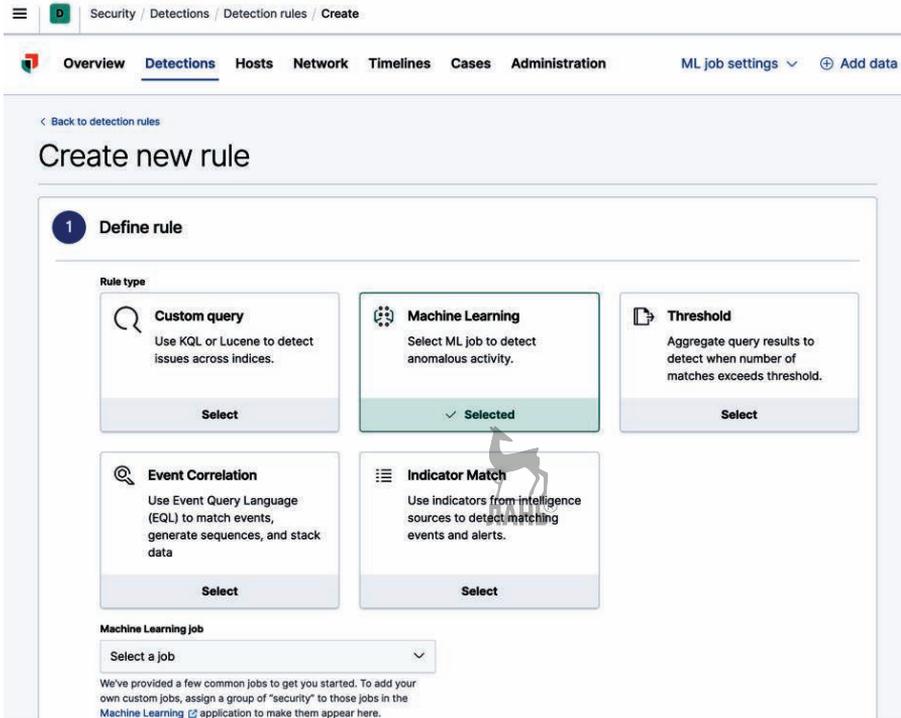


Рис. 8.32 ❖ Создание правила обнаружения на основе машинного обучения

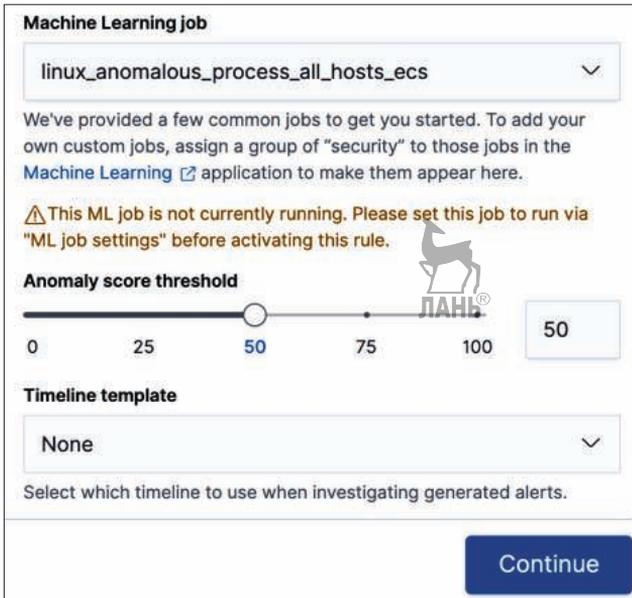


Рис. 8.33 ❖ Выбор задания ML и порогового значения для правила обнаружения

Разумеется, если задание в данный момент не выполняется, появится предупреждение о том, что задание необходимо запустить. Также примечательно, что если вы создали свое собственное задание (то есть не использовали одно из предварительно созданных заданий), но назначили его **Job Group** (группе заданий) с названием Security в приложении Elastic ML, это настраиваемое задание также будет кандидатом на подачу оповещений, и его можно будет выбрать в раскрывающемся списке. Оставшуюся часть настройки правила обнаружения можно оставить на усмотрение читателя, поскольку она не относится к машинному обучению.

Приложение Elastic Security в значительной степени опирается на обнаружение аномалий, чтобы дополнить традиционные статические правила динамическим поведенческим анализом пользователя или объекта, который может выявить заметные события, часто заслуживающие пристального исследования. Будет интересно понаблюдать за тем, насколько глубоко и широко машинное обучение будет применяться в этом новом варианте использования спустя некоторое время!

ЗАКЛЮЧЕНИЕ

Elastic ML глубоко интегрирован со многими приложениями в Elastic Stack, предоставляя пользователям простые в использовании функциональные возможности. Это показывает, насколько важную роль играет Elastic ML в составе Elastic Stack и насколько его функции схожи с другими ключевыми функциями стека, такими как агрегирование.

Поздравляем, вы дошли до конца первой половины этой книги! Мы надеемся, что теперь вы знаете все, что нужно об обнаружении аномалий при помощи Elastic ML.

Дальше мы займемся «обратной стороной» Elastic ML – аналитикой фрейм-ов данных, и вы узнаете, как использовать другие методы машинного обучения (включая создание моделей, обучаемых с учителем, и логический вывод), чтобы задействовать аналитические решения в принципиально новых вариантах использования.

.....

АНАЛИЗ ФРЕЙМОВ ДАННЫХ

В этом разделе рассмотрены основы анализа фреймов данных, его полезность и способы его реализации. В нем описаны различные типы анализа и их отличия друг от друга. В конце раздела представлены полезные советы и примеры, которые можно использовать для получения максимальной отдачи от Elastic ML.

Эта часть книги состоит из следующих глав:

- главы 9 «Введение в анализ фреймов данных»;
- главы 10 «Обнаружение выбросов»;
- главы 11 «Классификационный анализ»;
- главы 12 «Регрессия»;
- главы 13 «Заключение»;
- приложение «Полезные советы».

Глава 9

.....

Введение в анализ фреймов данных

В первой части этой книги мы подробно рассказали об обнаружении аномалий – основной функции машинного обучения, которая напрямую интегрирована в Elastic Stack. В этой и следующей главах вы познакомитесь с новыми функциями машинного обучения, интегрированными в стек. К ним относятся *обнаружение выбросов*, новый метод обучения без учителя для обнаружения необычных точек данных в хранилищах, не относящихся к временным рядам, а также две функции обучения с учителем: классификация и регрессия.

Алгоритмы обучения с учителем используют размеченные наборы данных – например, набор данных, описывающий различные аспекты образцов биологической ткани, а также информацию о том, является ли ткань злокачественной – для обучения модели. Затем эту модель можно использовать для прогнозирования ранее не встречавшихся точек данных (или образцов тканей, если продолжить наш пример). Когда целью прогноза является дискретная переменная или категория, например является ли ткань злокачественной или нет, метод обучения с учителем называется *классификацией*. Когда целью является непрерывная числовая переменная, например цена продажи квартиры или почасовая цена на электроэнергию, такой метод обучения с учителем известен как *регрессия*. В совокупности эти три новые разновидности машинного обучения (обнаружение выбросов, классификация и регрессия) известны как *анализ фреймов (окон) данных*. Мы обсудим каждую из этих разновидностей более подробно в следующих главах.

Хотя они решают разные задачи и имеют разные цели, все они работают под капотом общей *технологии преобразования данных*, которая позволяет нам преобразовывать и агрегировать данные, представленные в виде последовательности транзакций или потока в формат на основе объектов. Этот ориентированный на объект формат нужен для многих алгоритмов, которые применяются в анализе фреймов данных, и поэтому, прежде чем углубиться в изучение новых функций машинного обучения, мы собираемся детально рассказать о том, как преобразовать ваши данные в формат, который больше подходит для последующих технологий машинного обучения. Мы также собираемся представить краткий обзор Painless – встроенного в Elasticsearch языка сценариев, который является хорошим инструментом для любых спе-

циалистов по обработке данных или инженеров, работающих с машинным обучением в Elastic Stack.

Богатая экосистема библиотек как для обработки данных, так и для машинного обучения существует и за пределами Elastic Stack. Основным стимулом для развития этих приложений является популярность языка Python. Из-за его повсеместного распространения в сообществах специалистов по науке о данных и инженерии данных во второй части этой главы мы сосредоточимся на совместном использовании Python и Elastic Stack, уделяя особое внимание новому клиенту Elasticsearch для обработки данных – Eland. В этой главе мы рассмотрим следующие темы:

- основы преобразования данных;
- использование Painless для сложных конфигураций преобразования;
- совместное использование Python и Elasticsearch.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Материал в этой главе требует использования Elasticsearch версии 7.9 или выше и Python версии 3.7 или выше. Примеры кода и фрагменты, необходимые для данной главы, размещены в репозитории GitHub книги по адресу <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter09>. Когда для некоторых примеров требуется конкретная новая версия Elasticsearch, об этом говорится до начала описания примера.

ОСНОВЫ ПРЕОБРАЗОВАНИЯ ДАННЫХ

В этом разделе вы погрузитесь в тему преобразования потоковых или событийных данных, таких как журналы, в данные, представляющие некую сущность (объект).

Чем полезны преобразования?

Каковы наиболее распространенные типы данных, загружаемых в Elasticsearch? Зачастую это документы, которые содержат записи, зависящие от времени или последовательности записей, например журналы веб-сервера, покупки клиентов в интернет-магазине, комментарии, опубликованные на платформе социальных сетей, и т. д.

Хотя этот вид данных полезен для понимания поведения наших систем на отрезке времени и идеально подходит для использования с такими технологиями, как обнаружение аномалий, наборы данных на основе потоков или событий сложно заставить работать с функциями анализа фреймов без предварительной агрегации или преобразования. Например, рассмотрим

интернет-магазин, в котором регистрируются покупки, сделанные клиентами. За год каждый покупатель может совершить десятки или сотни транзакций. Если магазин затем захочет найти способ использовать обнаружение выбросов для выявления необычных клиентов, ему придется преобразовать все точки данных транзакций для каждого клиента и суммировать определенные ключевые показатели, такие как средняя сумма денег, потраченная на покупку, или количество покупок за календарный месяц. На рис. 9.1 представлена упрощенная иллюстрация, которая изображает процесс извлечения записей транзакций из покупок электронной коммерции, сделанных двумя покупателями, и преобразования их в набор, ориентированный на объект и описывающий общее количество товаров, приобретенных этими покупателями, а также среднюю стоимость их заказа.

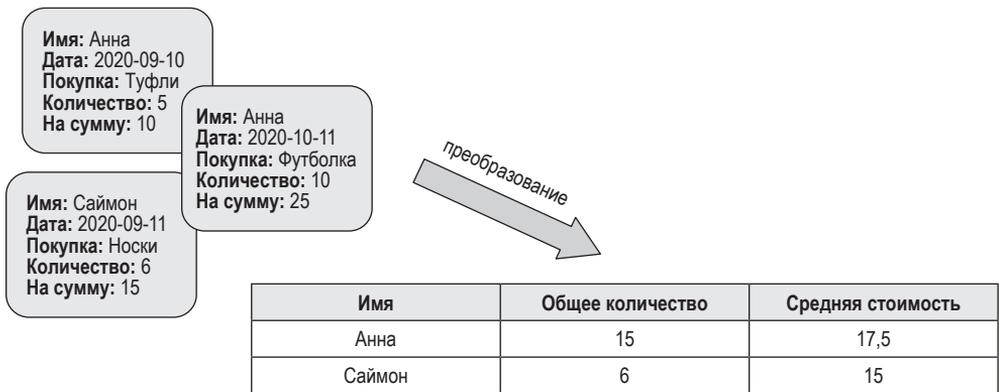


Рис. 9.1 ❖ Схема преобразования транзакций интернет-магазина в набор данных, ориентированный на объекты

Чтобы выполнить преобразование, показанное на рис. 9.1, мы должны сгруппировать каждый из документов в хранилище транзакций по имени клиента, а затем выполнить два вычисления: просуммировать количество элементов в каждом документе транзакции, чтобы получить общую сумму, а также вычислить среднюю цену покупок для каждого покупателя. Выполнение этих операций вручную для всех транзакций для каждого из тысяч потенциальных клиентов было бы чрезвычайно трудоемким занятием, и именно здесь вступают в силу преобразования.

Анатомия преобразований

Хотя мы собираемся начать наше путешествие по преобразованиям с простых примеров, многие реальные варианты использования намного сложнее. При применении преобразований к вашим собственным проектам данных полезно помнить о двух вещах, которые помогут вам справиться с задачей: *сводная таблица (pivot)* и *агрегирование (aggregation)*.

Давайте посмотрим, как эти два инструмента дополняют друг друга, помогая преобразовать потоковый документ в набор, представляющий объекты. В нашем сценарии использования клиентской аналитики у нас есть много различных признаков, описывающих каждого клиента: имя клиента, цена, которую он заплатил за каждый из своих продуктов при оформлении заказа, список приобретенных товаров, дата покупки, местоположение клиента и т. д.

Первое, что нам нужно выбрать, – это объект представления, для которого мы собираемся построить наш объектно-ориентированный набор. Давайте начнем с очень простого примера и скажем, что наша цель – выяснить, сколько денег в среднем каждый покупатель потратил на покупку в течение выбранного периода времени и сколько они потратили в целом. Таким образом, объект представления, для которого мы хотим построить новый набор данных – нашу сводную таблицу, – это имя клиента.

С большинством клиентов в нашем исходном наборе данных связано более одной транзакции. Следовательно, если мы попытаемся сгруппировать наш набор по имени клиента, для каждого клиента у нас будет несколько документов. Чтобы успешно составить сводную таблицу, нам нужно решить, какие агрегированные количества (например, среднюю цену за заказ клиента) мы хотим включить в наш набор, представляющий объект. От этого, в свою очередь, зависит, какие агрегации мы определим в нашей конфигурации преобразования. Давайте посмотрим, как это работает, на практическом примере.

Использование преобразований для анализа заказов интернет-магазина

В этом разделе для иллюстрации некоторых основных принципов преобразования, изложенных раньше, мы будем использовать образец набора данных Kibana E-Commerce.

1. Импортируйте образец набора данных заказов **Sample eCommerce orders** из панели данных **Sample data** Kibana, показанной на рис. 9.2, нажав кнопку **Add data** (Добавить данные). Будет создано новое хранилище с именем `kibana_sample_data_ecommerce`, заполненное выбранным набором данных.
2. Перейдите к мастеру преобразований **Transforms**, вызвав раскрывающееся меню панели Kibana с помощью кнопки в верхнем левом углу, затем перейдите в раздел **Stack Management** (Управление стеком) и нажмите **Transforms** в меню **Data**.
3. В окне **Transforms** нажмите на **Create your first transform** (Создать первое преобразование), чтобы вызвать мастер преобразований. Вам будет предложено выбрать исходное хранилище, которое преобразование будет использовать для создания вашей сводной таблицы и агрегатов. В данном случае нас интересует хранилище `kibana_sample_data_ecommerce`, которое вы должны выбрать на панели, показанной на рис. 9.3. Исходные хранилища, отображаемые в вашем интерфейсе

Kibana, могут немного отличаться в зависимости от хранилищ, доступных в настоящее время в вашем кластере Elasticsearch.



Рис. 9.2 ❖ Импорт набора данных Sample eCommerce orders из панели данных Kibana Sample

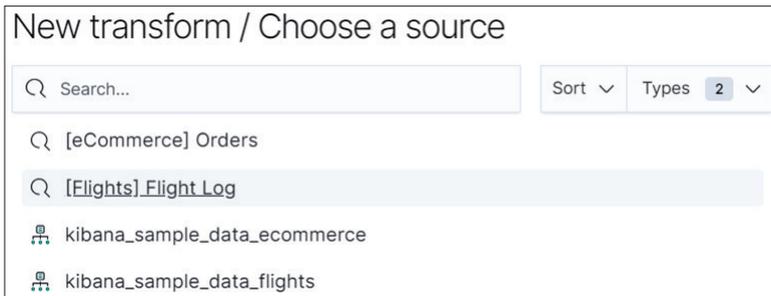


Рис. 9.3 ❖ Для нашего примера выберите kibana_sample_data_ecommerce

4. После выбора исходного хранилища мастер преобразования откроет диалоговое окно, в котором представлен предварительный просмотр исходных данных (рис. 9.4), а также позволит нам выбрать сводный объект с помощью раскрывающегося селектора в разделе **Group by** (Группировать по). В данном случае нужно выбрать поле с именем `customer_full_name`.
5. Теперь, когда определен объект для формирования сводных данных, мы перейдем к следующей части построения преобразования: агрегациям. В данном случае нас интересует вычисление средней суммы денег, которую покупатель потратил в интернет-магазине на один заказ. Во время каждой транзакции, которая записывается в документе в исходном хранилище, общая сумма, уплаченная клиентом, сохраняется в поле `taxful_total_price`. Следовательно, агрегирование, которое мы определяем, будет работать с этим полем. В меню **Aggregations** (Агрегаты) выберите `taxful_total_price.avg`. После того как вы щелкнете по этому пункту, под ним

появится поле предварительного просмотра сводного набора данных, как показано на рис. 9.5.

1 Configuration

Index pattern
kibana_sample_data_ecommerce

e.g. method: "GET" or status: "404" KQL Edit JSON query

Histogram charts 36 columns hidden Sort fields

category	currency	customer_birth_date	customer_first_name	customer_full_name	customer_gender	customer_id	customer_la
Men's Clothing	EUR		Eddie	Eddie Underwood	MALE	38	Underwood
Woman's Clothing	EUR		Mary	Mary Bailey	FEMALE	20	Bailey
["Women's Shoes"; "Wom...	EUR		Gwen	Gwen Butler	FEMALE	26	Butler
["Women's Shoes"; "Wom...	EUR		Diane	Diane Chandler	FEMALE	22	Chandler
["Men's Clothing"; "Men's ...	EUR		Eddie	Eddie Weber	MALE	38	Weber

Rows per page: 5

Group by
customer_full_name.keyword Edit JSON config

Рис. 9.4 ❖ Выберите в списке **Group by** объект, для которого вы хотите представить сводную информацию

Aggregations

taxful_total_price.avg Edit Close

Add an aggregation ...

Columns Sort fields

customer_full_name....	taxful_total_price.avg
Abd Adams	49.4921875
Abd Allison	170
Abd Bailey	127.484375
Abd Baker	67
Abd Ball	91.484375

Rows per page: 5

< 1 2 3 4 5 ... 20 >

> Next

Рис. 9.5 ❖ Предварительный просмотр преобразованных данных для быстрой проверки, все ли настроено должным образом

- Наконец, мы настроим последние два элемента: идентификатор задания преобразования и имя целевого хранилища, в которое будут помещены документы, описывающие наши сводные данные. Рекомендуется оставить флажок **Create index pattern** (Создать шаблон хранилища) установленным, как показано на рис. 9.6, чтобы можно было легко перейти к целевому хранилищу на вкладке **Discover** и просмотреть результаты.

Идентификатор преобразования будет использоваться для определения задания преобразования и целевого хранилища.

7. Чтобы запустить задание преобразования, не забудьте нажать **Next** в мастере преобразования после выполнения инструкций, описанных в шаге 6, а затем **Create and start** (Создать и запустить), после чего будет запущено задание преобразования и создан сводный набор данных, представляющий объект.
8. После завершения преобразования (индикатор выполнения достигнет 100 %, если все идет хорошо) вы можете нажать кнопку **Discover** в нижней части мастера преобразования и просмотреть преобразованные документы.

2 Transform details

Transform ID

Transform description

Destination index

Create index pattern
 Continuous mode

> Advanced settings

Рис. 9.6 ❖ Каждому преобразованию нужно присвоить идентификатор

Как было сказано в начале этого раздела, мы видим из образца документа на рис. 9.7, что задание преобразования взяло набор данных, представляющий последовательность транзакций, описывающих каждую покупку, сделанную клиентом в нашем интернет-магазине, и преобразовывало его в набор, представляющий объект, который содержит результаты конкретного аналитического преобразования (расчет средней цены, уплачиваемой клиентом), и сгруппированный по полному имени клиента.

Expanded document

Table JSON

f _id	QW0H6FA5Jvsfn4Jj0_LF32QAAAAAAAAA
f _index	ecommerce_avg_total_price_dest
# _score	0
f _type	_doc
f customer_full_name.keyword	Abd Adams
# taxful_total_price.avg	49.492

Рис. 9.7 ❖ Результатом выполнения задания преобразования является целевой набор данных, где каждый документ содержит агрегирование для каждого объекта. В данном случае это средняя цена `taxful_total_price`, уплачиваемая каждым покупателем за один заказ

Поздравляем – вы создали и запустили свое первое задание преобразования! Хотя оно довольно простое по своей природе, эта базовая конфигурация задания является хорошим строительным блоком для создания более сложных преобразований, которые мы рассмотрим в следующих разделах.

Более сложные конфигурации сводной таблицы и агрегирования

В предыдущем разделе мы рассмотрели две части преобразования: сводную таблицу и агрегирование. В последующем примере наша цель состояла в том, чтобы, используя преобразования в демонстрационном наборе данных интернет-магазина, узнать среднюю сумму денег, которую клиенты тратят на один заказ. Решая эту задачу, мы выяснили, что каждый документ, содержащий транзакцию, имеет поле с именем `customer.full_name`, и мы использовали это поле для составления сводной таблицы из исходных данных. Наше агрегирование представляло собой среднее значение поля, в котором записана общая сумма денег, потраченных клиентом на заказ.

Однако не все вопросы, которые мы могли бы задать о наших данных интернет-магазина, поддаются простой сводке или группировке в конфигурациях, подобных рассмотренной ранее. Давайте разберем некоторые более сложные конфигурации преобразований, способствующие выполнению других исследований, которые мы, возможно, захотим провести с набором данных интернет-магазина. Если вы хотите узнать обо всех доступных конфигурациях сводной таблицы, ознакомьтесь с документацией API по следующему адресу: <https://www.elastic.co/guide/en/elasticsearch/reference/master/put-transform.html>.

Предположим, что мы хотим узнать среднюю сумму денег, потраченную на один заказ за неделю, и сколько уникальных клиентов совершили покупки за этот период. Чтобы ответить на эти вопросы, нам нужно будет создать новую конфигурацию преобразования.

1. Вместо того чтобы опираться на имя клиента, мы должны построить *гистограмму дат* из поля `order_date`, которое, как следует из названия, хранит информацию о том, когда был размещен заказ. Мастер преобразования упрощает эту работу, поскольку `date_histogram(order_date)` является одним из предварительно настроенных параметров, отображаемых в раскрываемом списке **Group by**.
2. После того как вы выбрали `date_histogram(order_date)` в раскрываемом списке **Group by**, обратите внимание на правую часть панели на рис. 9.8. Она должна содержать условные обозначения для длины интервала группировки, используемого в гистограмме даты (например, `1m` для интервала в 1 минуту). В нашем случае мы заинтересованы в группировке по неделям, поэтому нам нужно выбрать в списке `1w` (`week`, `неделя`).

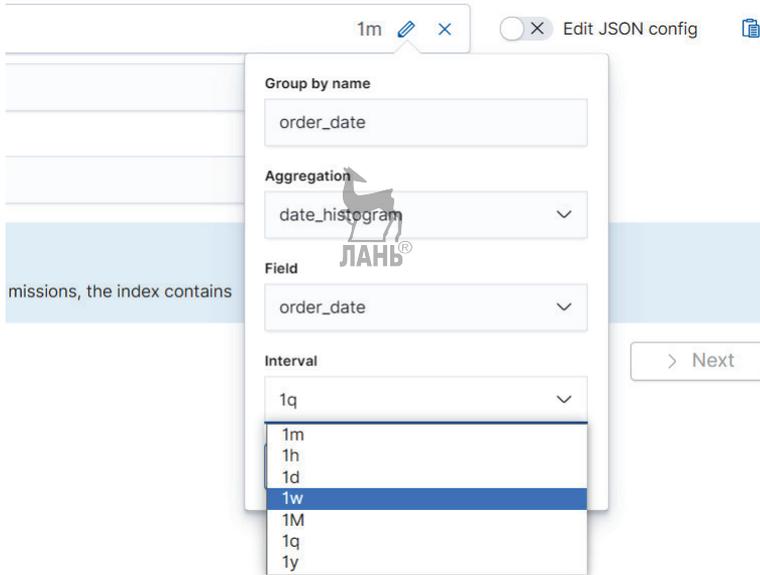
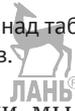


Рис. 9.8 ❖ Выберите частоту группировки по дате из раскрывающегося списка

3. Далее для нашего агрегирования выберем уже знакомую среднюю цену (`total_taxful_price`). После того как мы сделаем свой выбор, мастер преобразования отобразит предварительный просмотр, в котором будет показана средняя цена, уплачиваемая клиентом за заказ, в группах заказов по разным неделям, для нескольких выборочных записей. Предварительный просмотр предназначен для проверки получившихся результатов. Поскольку задания преобразования могут быть ресурсоемкими, на этом этапе рекомендуется временно остановиться и внимательно изучить предварительные результаты, чтобы убедиться, что данные преобразованы в желаемый формат.
4. Иногда нам может потребоваться опросить наши данные способами, которые не получается реализовать простыми одноуровневыми группировками, подобными той, которую мы исследовали на предыдущих шагах. Как вы сейчас увидите, можно создавать вложенные группировки. Предположим, что в нашем гипотетическом примере интернет-магазина владельцам также будет интересно узнать среднюю сумму денег, потраченную за неделю по географическим регионам. Чтобы решить эту проблему, вернемся к мастеру преобразования и добавим второе поле для группировки. В этом случае мы будем группировать данные по `geoip.region_name`. Как и раньше, мастер отображает предварительные результаты для просмотра, когда мы выбираем поле группировки. Как и в предыдущем случае, лучше потратить время на проверку строк, отображаемых в области предварительного просмотра, чтобы убедиться, что данные были преобразованы правильно.

- ✓ Щелкните переключатель **Columns** (Столбцы) над таблицей предварительного просмотра, чтобы изменить порядок столбцов.



Помимо создания вложенной группировки, мы также можем добавить к нашему преобразованию несколько агрегаций. Предположим, что помимо средней суммы, потраченной на одного покупателя в неделю по регионам, нам также будет интересно узнать количество уникальных покупателей, разместивших заказы в нашем магазине за эту неделю. Давайте посмотрим, как мы можем добавить эту агрегацию к нашему преобразованию.

- В раскрывающемся меню **Aggregations** в мастере прокрутите список вниз, пока не найдете опцию **entity cardinality** (мощность объекта) для поля `customer.full_name.keyword`, и выберите ее. Соответствующая агрегация будет добавлена в вашу конфигурацию преобразования, и в предварительном просмотре должен появиться один дополнительный столбец.

Теперь вы можете выполнить шаги, описанные в руководстве в предыдущем разделе, чтобы назначить идентификатор и целевое хранилище для преобразованных данных, а также создать и запустить задание. Мы оставляем вам эти шаги в качестве упражнений.

В предыдущих двух разделах мы исследовали два ключевых компонента преобразований: сводную таблицу и агрегирование, а также рассмотрели два разных пошаговых примера, чтобы показать, как простые и усложненные комбинации сводной таблицы и агрегирования могут использоваться для исследования наших данных и получения различных сведений.

Внимательные читатели могли заметить, что на рис. 9.6 мы оставили флажок **Continuous mode** (Непрерывный режим) снятым. В следующем разделе мы более подробно рассмотрим, что означает запуск преобразования в непрерывном режиме.



Различие между пакетными и непрерывными преобразованиями

Первое преобразование, которое мы создали в предыдущем разделе, было простым и выполнялось только один раз. Задание преобразования прочитало исходный набор данных `kibana_sample_data_ecommerce`, который мы указали в мастере преобразования, выполнило вычисления средней цены заказа каждого клиента, а затем записало полученные документы в целевое хранилище. Поскольку наше преобразование выполняется только один раз, любые изменения в нашем исходном наборе `kibana_sample_data_ecommerce`, которые происходят после запуска задания преобразования, больше не будут отражаться в данных, оказавшихся в целевом хранилище. Такое преобразование, которое выполняется только один раз, называется *пакетным преобразованием*.

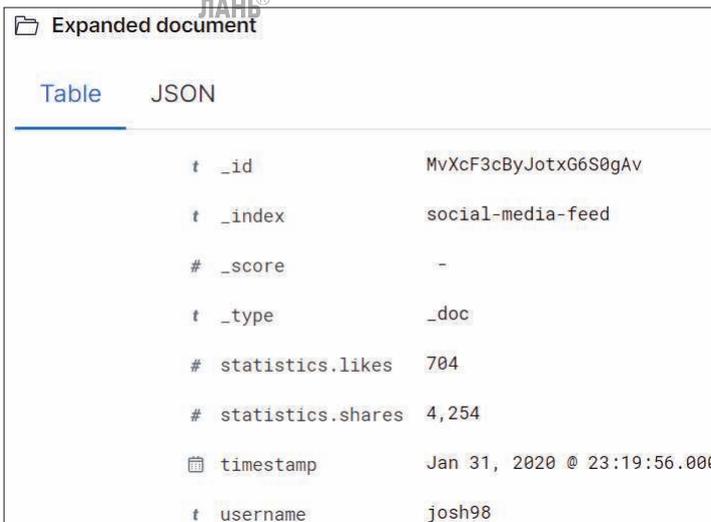
Во многих реальных сценариях, которые создают записи транзакций (например, в нашем примере вымышленного интернет-магазина), в исходное хранилище постоянно добавляются новые документы. Это означает, что наш свод-

ный набор данных, представляющий объекты, почти сразу устареет. Одним из решений, позволяющих поддерживать синхронизацию целевого хранилища с исходным, является постоянное удаление целевого хранилища и повторный запуск задания пакетного преобразования через регулярные промежутки времени. Однако это непрактично и заставляет выполнять вручную большой объем работы. Здесь вступают в игру *непрерывные преобразования*.

Если у нас есть исходное хранилище, которое постоянно обновляется, и мы хотим использовать его для создания сводного хранилища, представляющего объект, то мы должны использовать непрерывное преобразование. Давайте рассмотрим непрерывные преобразования более подробно, чтобы понять, чем они отличаются от пакетных преобразований и какие важные параметры конфигурации следует учитывать при запуске непрерывного преобразования.

Прежде всего подготовим почву для проблемы, которую мы пытаемся решить. Предположим, у нас есть вымышленная платформа для микроблогов в социальных сетях, где пользователи публикуют короткие посты, назначают им категории и устанавливают взаимосвязи с другими пользователями, а также с заранее определенными темами. Можно поделиться постом и поставить лайк. Также записывается статистика для каждого поста. Мы написали код Python, чтобы помочь сгенерировать этот набор данных. Этот код и сопутствующие инструкции по его запуску доступны в репозитории GitHub по адресу <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter09>. После запуска генератора у вас появится хранилище под названием `social-media-feed`, которое будет содержать ряд документов.

Каждый документ в наборе данных содержит пост, опубликованный пользователем в социальной сети. Для краткости мы исключили текст поста из документа. На рис. 9.9 показан образец документа в хранилище `social-media-feed`.



Expanded document	
Table	JSON
<code>t</code> <code>_id</code>	<code>MvXcF3cByJotxG6S0gAv</code>
<code>t</code> <code>_index</code>	<code>social-media-feed</code>
<code>#</code> <code>_score</code>	<code>-</code>
<code>t</code> <code>_type</code>	<code>_doc</code>
<code>#</code> <code>statistics.likes</code>	<code>704</code>
<code>#</code> <code>statistics.shares</code>	<code>4,254</code>
<code>📅</code> <code>timestamp</code>	<code>Jan 31, 2020 @ 23:19:56.000</code>
<code>t</code> <code>username</code>	<code>josh98</code>

Рис. 9.9 ❖ Образец документа в хранилище ленты социальной сети содержит имя пользователя, время отправки сообщения на платформу, а также некоторую базовую статистику о взаимодействии поста с пользователями

В следующем разделе вы увидите, как использовать этот набор данных вымышленной платформы социальных сетей, чтобы реализовать непрерывные преобразования.



Анализ данных социальных сетей с помощью непрерывных преобразований

В этом разделе мы будем использовать представленный ранее набор данных для изучения концепции непрерывных преобразований. Как было сказано в предыдущем разделе, пакетное преобразование уместно для однократного анализа, когда мы либо готовы проанализировать моментальный снимок набора данных в определенный момент времени, либо у нас нет набора данных, который изменяется. В большинстве реальных приложений дело обстоит иначе. Файлы журналов непрерывно дополняются новыми записями, многие платформы социальных сетей работают круглосуточно, а платформы электронной коммерции обслуживают клиентов во всех часовых поясах и, таким образом, генерируют поток данных о транзакциях. Здесь вступают в игру непрерывные преобразования.

Давайте посмотрим, как мы можем проанализировать средний уровень взаимодействия (количество лайков и репостов, полученных пользователем социальной сети) с помощью непрерывных преобразований.

1. Перейдите к мастеру преобразований. На странице **Stack Management** (Управление стеком) слева под разделом **Data** выберите **Transforms**.
2. Действуя, как в предыдущих разделах, начнем с создания преобразования. В качестве исходного хранилища выберите шаблон хранилища `social-media-feed`. Вы должны увидеть представление, подобное изображенному на рис. 9.10.

1 Configuration

Index pattern
social-media-feed

e.g. method : "GET" or status : "404" KQL Edit JSON query

Histogram charts Columns Sort fields

statistics.likes	statistics.shares	timestamp	username
429	5,339	Jan 3, 2020 @ 07:30:53.0...	josh98
212	9,311	Jan 3, 2020 @ 05:31:29.0...	johnny
927	4,924	Jan 2, 2020 @ 11:09:47.0...	Jenny
691	2,721	Jan 27, 2020 @ 22:50:51.1...	James
14	4,391	Jan 6, 2020 @ 09:08:08.0...	Amelia

Rows per page: 5

< 1 2 3 4 5 ... 2000 >

Рис. 9.10 ❖ Мастер преобразований показывает образец записей в хранилище ленты социальных сетей

3. В данном случае нас интересует вычисление агрегированных показателей взаимодействия каждого поста для каждого имени пользователя. Таким образом, наша конфигурация **Group by** будет включать имя пользователя, в то время как наши агрегации будут вычислять общее количество лайков и репостов для каждого пользователя, среднее количество лайков и репостов для каждого пользователя, а также общее количество постов, опубликованных каждым пользователем. Окончательная конфигурация **Group by** и **Aggregations** должна выглядеть примерно так, как показано на рис. 9.11.

The screenshot shows a configuration panel with two main sections: 'Group by' and 'Aggregations'.
 - Under 'Group by', there is a text input field containing 'username.keyword' with edit and delete icons. Below it is a dropdown menu labeled 'Add a group by field ...'.
 - Under 'Aggregations', there are four text input fields: 'statistics.likes.avg', 'statistics.likes.sum', 'statistics.shares.avg', and 'statistics.shares.sum', each with edit and delete icons. Below them is a dropdown menu labeled 'Add an aggregation ...'.
 - On the right side of the 'Group by' section, there is a toggle switch labeled 'Edit JSON config'.

Рис. 9.11 ❖ Конфигурация **Group by** и **Aggregations** для нашего примера непрерывного преобразования

4. Переведите переключатель **Continuous mode** во включенное состояние и убедитесь, что в поле **Date field** выбрано значение `timestamp`, как показано на рис. 9.12.

The screenshot shows a configuration panel for 'Continuous mode'.
 - At the top, there is a toggle switch labeled 'Continuous mode' which is turned on (checked).
 - Below it is a section titled 'Date field' with a dropdown menu currently showing 'timestamp'. Below the dropdown is the text: 'Select the date field that can be used to identify new documents.'
 - Below that is a section titled 'Delay' with a text input field containing '60s'. Below the input field is the text: 'Time delay between current time and latest input data time.'
 - At the bottom, there is a link labeled '> Advanced settings'.

Рис. 9.12 ❖ Включите переключатель **Continuous mode**, чтобы процесс преобразования периодически проверял исходное хранилище и добавлял новые документы в целевое хранилище

- После нажатия кнопки **Create and start** вы сможете вернуться на страницу **Transforms** и увидите, что выполняется задание непрерывного преобразования для хранилища `social-media-feed`. Обратите внимание на тег `continuous` (непрерывный) в описании задания.



Рис. 9.13 ❖ Непрерывные преобразования, отображаемые на странице **Transforms**. Обратите внимание, что задание помечено тегом `continuous`

- Давайте вставим несколько новых сообщений в хранилище `social-media-feed` и посмотрим, как изменится статистика для пользователя по имени `Carl` после добавления нового документа в исходное хранилище. Чтобы вставить новый пост, откройте консоль Kibana **Dev Console** и выполните следующую команду REST API:

```
POST social-media-feed/_doc
{
  "username": "Carl",
  "statistics": {
    "likes": 320,
    "shares": 8000
  },
  "timestamp": "2021-01-18T23:19:06"
}
```

- Теперь, когда мы добавили новый документ в хранилище `social-media-feed`, мы вправе ожидать, что этот документ будет выбран заданием непрерывного преобразования и добавлен в целевое хранилище `social-media-feed-engagement`. На рис. 9.14 показана преобразованная запись для пользователя по имени `Carl`.

В предыдущем примере очень упрощенно показано, как работают непрерывные преобразования и как вы можете создать свое собственное непрерывное преобразование с помощью мастера преобразований, доступного в Kibana. В главе 13 мы вернемся к теме непрерывных преобразований, когда продемонстрируем, как комбинировать обученные модели ML, логический вывод и преобразования.

Table	JSON
t _id	Q_ssi2m7mTL0e294yatDGW4AAAAAAAAA
t _index	social-media-feed-engagement
# _score	0
t _type	_doc
# statistics.likes.avg	320
# statistics.likes.sum	640
# statistics.shares.avg	8,000
# statistics.shares.sum	16,000
t username.keyword	Carl

Рис. 9.14 ❖ Целевое хранилище непрерывного преобразования содержит запись для нового имени пользователя Carl, которое мы добавили вручную через Kibana Dev Console

А пока сделаем краткий экскурс в мир языка сценариев Painless. Хотя мастера преобразований и множества предустановленных конфигураций группировки и агрегирования, которые он предлагает, достаточно для многих наиболее распространенных сценариев использования анализа данных, более продвинутые пользователи захотят определять свои собственные агрегации. Очевидный способ сделать это – использовать встроенный скриптовый язык Elasticsearch Painless.

В следующем разделе мы совершим небольшой тур по миру Painless, который подготовит вас к созданию собственных расширенных конфигураций преобразования.

ИСПОЛЬЗОВАНИЕ PAINLESS ДЛЯ РАСШИРЕННЫХ КОНФИГУРАЦИЙ ПРЕОБРАЗОВАНИЯ

Как вы видели раньше, стандартные настройки сводной таблицы и агрегирования позволяют нам анализировать и опрашивать наши данные различными способами. Однако для более нестандартных или расширенных вариантов использования встроенные функции могут быть недостаточно гибкими. В этих случаях нам нужно будет написать собственные конфигурации сводных данных и агрегирования. Для этого и предназначен гибкий язык сценариев Painless, встроенный в Elasticsearch.

Далее мы познакомим вас с Painless, продемонстрируем некоторые инструменты, которые полезны при работе с Painless, а затем покажем, как можно применить этот язык для создания пользовательских конфигураций преобразования.

Знакомство с Painless

Painless – это язык сценариев, встроенный в Elasticsearch. Мы рассмотрим Painless с точки зрения переменных, конструкций управления выполнением, операций и функций. Это основные строительные блоки, которые помогут вам разработать собственные сценарии для применения в преобразованиях данных.

Вполне вероятно, что многие читатели этой книги имеют некоторый опыт программирования. Возможно, вы создавали скрипты очистки данных с помощью Python, программировали Linux с помощью сценариев bash

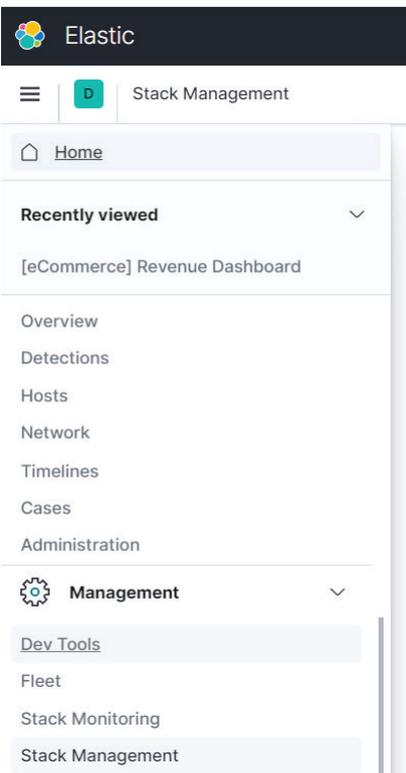


Рис. 9.15 ❖ Ссылка на страницу Dev Tools находится в нижней части бокового меню Kibana.

Нажмите на нее, чтобы получить доступ к интерактивной среде отладки Painless Lab

или разрабатывали корпоративное программное обеспечение на Java. Хотя эти языки имеют много различий и предназначены для разных целей, все они имеют общие строительные блоки, которые помогают людям, читающим язык, понимать их. Хотя существует почти бесконечное количество подходов к обучению языку программирования, подход, который мы выбрали, основан на усвоении следующих основных тем о Painless: переменные, операции (такие как сложение, вычитание и различные проверки условий), управление выполнением (конструкции if-else и циклы for) и функции. Эти концепции должны быть понятны пользователям, знакомым с любым другим языком программирования. В дополнение к этому мы рассмотрим некоторые особенности, присущие только Painless, такие как различные контексты выполнения.

При изучении нового языка программирования важно иметь рабочую площадку, на которой можно экспериментировать с синтаксисом. К счастью, начиная с версии Elasticsearch 7.10 приложение Dev Tools содержит новую отладочную среду Painless Lab, где вы можете опробовать образцы кода, представленные в этой главе, а также любой код, который вы разработаете самостоятельно. Чтобы попасть в Painless Lab,

перейдите в **Dev Tools**, как показано на рис. 9.15, а затем в верхнем меню страницы **Dev Tools** выберите **Painless Lab**.

После перехода по ссылке откроется встроенный редактор кода Painless, как показано на рис. 9.16.



Рис. 9.16 ❖ В Painless Lab есть встроенный редактор кода.

Окно **Output** показывает результат выполнения кода, введенного в окне редактора

Редактор кода в Painless Lab по умолчанию предлагает некоторые примеры функций и объявления переменных в качестве иллюстрации того, как можно нарисовать фигуру в окне вывода на рис. 9.16 при помощи Painless. Сейчас вы можете удалить этот код, чтобы освободить место для ваших собственных экспериментов, которые вы будете проводить при чтении оставшейся части этой главы.



Полную спецификацию языка Painless можно найти по адресу <https://www.elastic.co/guide/en/elasticsearch/painless/master/painless-lang-spec.html>. Вы можете использовать ее как справочник и ресурс для получения дополнительной информации по темам, рассмотренным далее в этой главе.



Переменные, операторы и управление выполнением

Первое, что мы обычно делаем на новом языке программирования, – учимся манипулировать значениями. Для этого мы присваиваем этим значениям имена или переменные. Painless имеет типы, и прежде чем переменная может быть назначена, она должна быть объявлена вместе с ее типом. Синтаксис объявления переменной выглядит так: идентификатор_типа имя_переменной;.

Применение этого синтаксиса на практике показано в следующем блоке кода, где мы объявляем переменные `a` и `b` для хранения целочисленных значений, переменную `my_string` для хранения строкового значения и переменную `my_float_array` для хранения массива значений с плавающей запятой:

```
int a;
int b;
String my_string;
float[] my_float_array;
```

На данный момент переменные еще не содержат ненулевых значений. Они только что инициализированы и готовы к использованию оператора присваивания, который присваивает каждой переменной значение соответствующего типа. Если вы попытаетесь скопировать предыдущий блок кода в редактор Painless Lab, то увидите в окне вывода результат `null`, как показано на рис. 9.17.

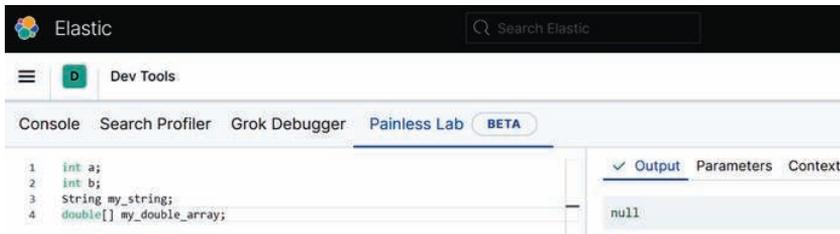


Рис. 9.17 ❖ Слева инициализируются переменные Painless различных типов. Справа на панели вывода отображается значение `null`, поскольку этим переменным еще не присвоено значение

! Редактор кода Painless Lab отображает только результат выполнения последнего оператора.

Теперь присвоим этим переменным какие-нибудь значения, чтобы продолжить эксперименты. Присвоение значений показано в следующем блоке кода. В первых двух строках мы присваиваем целочисленные значения нашим целочисленным переменным `a` и `b`. В третьей строке мы назначаем строковое значение "hello world" строковой переменной `my_string`, а в последней строке инициализируем новый массив значениями с плавающей запятой:

```
a = 1;
b = 5;
my_string = "hello world";
my_double_array = new double[] {1.0, 2.0, 2.5};
```

Чтобы продемонстрировать работу некоторых операторов Painless, выполним с этими переменными различные действия. Полный список доступных операторов представлен в спецификации языка Painless по адресу <https://www.elastic.co/guide/en/elasticsearch/painless/current/painless-operators.html>. Следующие блоки кода иллюстрируют основные математические операции: сложение, вычитание, деление и умножение, а также операцию модуля или взятие остатка:

```
int a;
int b;
```

```

a = 1;
b = 5;

// Сложение
int addition;
addition = a+b;

// Вычитание
int subtraction;
subtraction = a-b;

// Умножение
int multiplication;
multiplication = a*b;

// Целочисленное деление
int int_division;
int_division = a/b;

// Остаток от деления
int remainder;
remainder = a%b;

```



Попробуйте ввести эти примеры кода в редакторе Painless Lab, и вы сможете увидеть результаты выполнения последнего оператора, как показано в примере сложения на рис. 9.18.

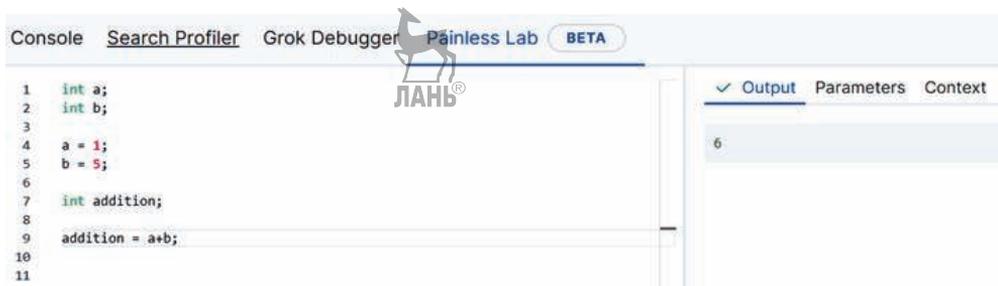


Рис. 9.18 ❖ Использование редактора кода Painless Lab и консоли для выполнения операции сложения. Результат, сохраненный в переменной под названием `addition` в редакторе кода слева, отображается на вкладке **Output** справа

Помимо математических операций, мы также рассмотрим логические операторы. Они жизненно важны для многих сценариев и конфигураций Painless, а также для операторов управления выполнением кода, которые мы рассмотрим позже.

Следующие фрагменты кода показывают, как объявить переменную для хранения логического (истина/ложь) значения и как использовать операторы сравнения. Полный список логических операторов Painless представлен в спецификации, доступной по адресу <https://www.elastic.co/guide/en/elastic-search/painless/current/painless-operators-boolean.html>:

```
boolean less_than = 4<5;
boolean greater_than = 4>5;
boolean less_than_or_equal = 4 <=5;
boolean greater_than_or_equal = 4 >= 5;
```

В качестве упражнения скопируйте предыдущий блок кода в редактор Painless Lab. При желании вы можете просмотреть содержимое каждой из этих переменных, введя ее имя и точку с запятой в последней строке редактора кода, и значение, хранящееся в переменной, отобразится в окне вывода справа, как показано на рис. 9.19.

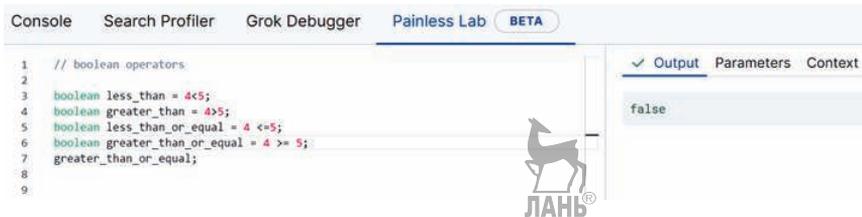


Рис. 9.19 ❖ Введя имя переменной и точку с запятой в редакторе кода Painless Lab, вы увидите значение переменной на вкладке **Output**

Хотя представленный здесь оператор `boolean` полезен во многих численных вычислениях, мы, вероятно, не смогли бы написать эффективные операторы управления выполнением кода без операторов равенства `==` и неравенства `!=`, которые проверяют, равны ли две переменные. В следующем блоке кода показано, как использовать эти операторы, на нескольких практических примерах:

```
// логический оператор проверки на равенство

boolean two_equal_strings = "hello" == "hello";
two_equal_strings;

// логический оператор проверки на неравенство

boolean not_equal = 5!=6;
not_equal;
```

И последнее, но не менее важное: в нашем обзоре логических операторов Painless – это блок кода, иллюстрирующий применение оператора `instanceof`, который проверяет, действительно ли данная переменная является экземпляром типа и возвращает `true` или `false`. Это полезно при создании кода, который вы хотите использовать только с переменными указанного типа:

```
// логический оператор instanceof проверяет, действительно ли
// переменная является экземпляром указанного типа

// проверка переменной is_integer вернет результат true (истина)

int int_number = 5;
```

```
boolean is_integer = int_number instanceof int;
is_integer;
```

В заключительной части этого раздела мы рассмотрим один из самых важных строительных блоков кода на языке Painless: операторы `if-else` и циклы `for`. Синтаксис операторов `if-else` показан в следующей блоке кода:

```
int a = 5;
int sum;

if (a < 6){
    sum = a+5;
}
else {
    sum = a-5;
}

sum;
```



В этом блоке кода мы объявляем целочисленную переменную `a` и присваиваем ей целочисленное значение 5. Затем объявляем другую целочисленную переменную `sum`. Эта переменная будет изменяться в соответствии с ветвью выполнения, определяемой оператором `if-else`. Наконец, оператор `if-else` сначала проверяет, меньше ли целочисленная переменная `a`, чем 6, и если да, то сохраняет результат сложения `a` и целого числа 5 в переменной `sum`. В противном случае значение, присвоенное переменной `sum`, является результатом вычитания 5 из `a`.

Если вы наберете этот код в редакторе кода Painless Lab, консоль вывода отобразит значение суммы, равное 10 (как показано на рис. 9.20), чего мы и ожидали, основываясь на предыдущих рассуждениях.



Рис. 9.20 ❖ Оператор `if-else` приводит к присвоению переменной `sum` значения 10

Далее рассмотрим цикл `for`, который полезен для различных задач анализа и обработки данных с помощью Painless. В цикле `for` мы будем перебирать строковую переменную и вычислять, сколько вхождений буквы `a` встречается в строке. Это очень простой пример, но он поможет вам понять синтаксис, чтобы вы могли применять его в своем коде:

```
// инициализация строки и переменной counter
String sample_string = "a beautiful day";
int counter = 0;

for (int i=0;i<sample_string.length();i++){
    // извлечение буквы из строки при помощи метода substring
    String letter = sample_string.substring(i, i+1);

    // использование оператора if для проверки, является ли текущий символ строки буквой а
    if (letter=="a") {
        // если да, то увеличиваем значение counter на 1
        counter++
    }
}

counter;
```

Вставьте этот образец кода в окно редактора Painless Lab, и вы убедитесь, что переменная count отобразит на панели **Output** значение 3, как мы и ожидали, поскольку в строке «a beautiful day» три раза встречается буква «a».

Функции

После знакомства с переменными и математическими и условными операторами настало время рассмотреть *функции*. Иногда нам приходится использовать одни и те же строки кода снова и снова в нескольких разных сценариях и конфигурациях. На этом этапе было бы разумно упаковать строки, к которым мы обращаемся снова и снова, в многократно используемый фрагмент кода, на который мы можем ссылаться по имени из наших сценариев Painless.

Вернемся к примеру, где мы использовали цикл for с оператором if для подсчета количества экземпляров буквы «a» в заданной строке. Предположим, мы хотим повторно использовать эту функциональность и сделать ее более универсальной. Это прекрасная возможность упаковать данный фрагмент кода в функцию Painless.

Создание функции в Painless состоит из трех этапов. Сначала мы пишем заголовок функции, в котором указывается тип значения, возвращаемого функцией, а также имя функции. Заголовок понадобится нам для обозначения функции, когда мы будем использовать ее в наших скриптах или потоках данных, которые рассмотрим более подробно при обсуждении логического вывода в главе 13. Каркас нашей функции, которую мы назовем letterCounter, выглядит так:

```
int letterCounter(){
}
```

Ключевое слово int перед именем функции определяет тип значения, возвращаемого этой функцией. Поскольку нас интересует количество вхождений

определенной буквы в конкретной строке, мы будем возвращать целое число. Скобки после имени `letterCounter` будут содержать параметры, которые принимает функция. На данный момент мы не указали никаких параметров, поэтому в скобках ничего нет. Наконец, две фигурные скобки обозначают место для *тела функции* – здесь будет находиться вся логика функции.

Теперь, когда вы знаете, из чего состоит заголовок функции, давайте заполним тело функции (пространство между фигурными скобками) кодом, который мы написали в предыдущем разделе при изучении циклов `for`. Теперь наша функция должна выглядеть примерно так:

```
int letterCounter(){
    // инициализация строки и переменной counter

    String sample_string = "a beautiful day";
    int counter = 0;

    for (int i=0;i<sample_string.length();i++){

        // извлечение буквы из строки при помощи метода substring
        String letter = sample_string.substring(i, i+1);

        // использование оператора if для проверки, является ли текущий символ строки буквой а
        if (letter=="a") {
            // если да, то увеличиваем значение counter на 1
            counter++
        }
    }
    return counter;
}

letterCounter();
```

Если вы посмотрите, чем заканчивается функция, то заметите, что единственное различие между нашим циклом `for`, находящимся внутри тела функции, состоит в том, что теперь мы добавили оператор `return`. Он позволяет возвращать интересующее нас значение, хранящееся в переменной `count`, в код, вызывающий функцию, что подводит нас к следующему вопросу. Теперь, когда у нас есть первая функция на языке Painless и она делает что-то интересное, как нам вызвать эту функцию?

В среде Painless Lab мы просто набираем `letterCounter()`; , как показано на рис. 9.21. Результат, возвращаемый этой функцией, как мы и ожидали, исходя из нашего предыдущего анализа этого примера кода, равен 3.

Теперь, когда у нас есть работающая функция, давайте посмотрим, как сделать эту функцию более обобщенной, что вам часто придется делать при работе либо с преобразованиями, которые мы обсуждали в этой главе, либо с различными конвейерами загрузки и обработчиками сценариев, которые мы обсудим в главе 13. В настоящее время наша функция `letterCounter` очень специфична. Она только вычисляет количество появлений конкретной буквы «а» в конкретной строке «a beautiful day».

The screenshot shows the Painless Lab IDE interface. The main editor displays the following Java code:

```

1
2 int letterCounter(){
3     // initialize the string and the counter variable
4
5     String sample_string = "a beautiful day";
6     int counter = 0;
7
8     for (int i=0;i<sample_string.length();i++){
9
10
11     // get a letter from the string using the substring method
12     String letter = sample_string.substring(i, i+1);
13
14     //use an if-statement to check if the current letter being processed is an a
15     if (letter=="a") {
16         // is yes, increment the counter
17         counter++
18     }
19     return counter;
20 }
21
22 letterCounter();

```

On the right side, the 'Output' tab is active, showing the result of the function call: '3'. A watermark logo for 'ЛАНЬ' is visible in the center of the code editor.

Рис. 9.21 ❖ Пример функции letterCounter, показанный вместе с примером вызова функции в среде Painless Lab

Для того чтобы сделать этот фрагмент кода действительно полезным, необходимо изменить как фразу, так и букву, вхождения которой мы подсчитываем. С помощью функции мы можем сделать это с минимальным дублированием кода, объявив *параметры функции*. Поскольку мы хотим иметь возможность менять и букву, и фразу, давайте будем передавать их в функцию в виде параметров. После доработки определение нашей функции будет выглядеть следующим образом:

```

int letterCounter(String sample_string, String count_letter){
    // инициализация строки и переменной counter
    int counter = 0;

    for (int i=0;i<sample_string.length();i++){

        // извлечение буквы из строки при помощи метода substring
        String letter = sample_string.substring(i, i+1);

        // использование оператора if для проверки, является ли текущий символ строки буквой a
        if (letter==count_letter) {
            // если да, то увеличиваем значение counter на 1
            counter++
        }
    }
    return counter;
}

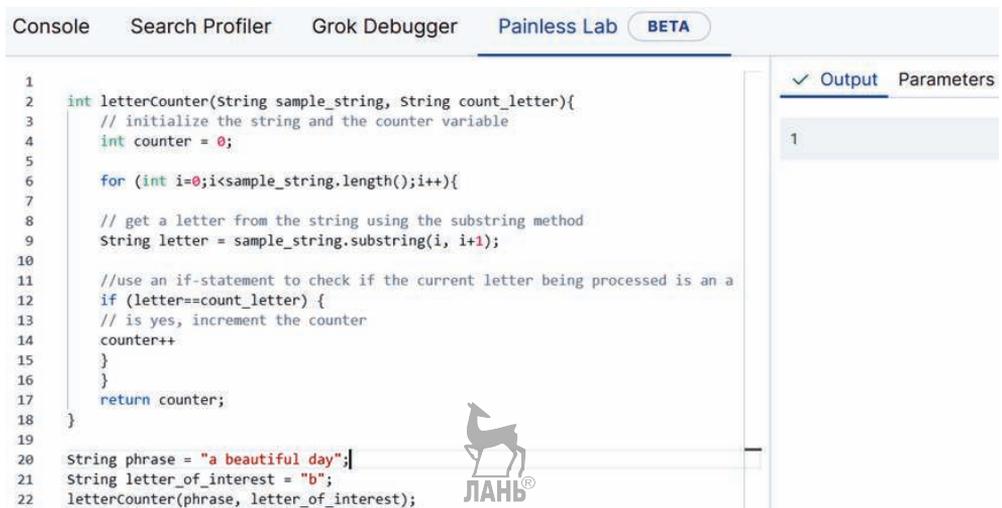
```

Обратите внимание, что теперь между скобками в заголовке функции мы определили два параметра: первый параметр `sample_string` представляет фразу, в которой мы хотим подсчитать вхождения второго параметра – буквы, которую представляет `count_letter`.

Чтобы вызвать эту функцию, мы сначала определим новые переменные, которые будут содержать нашу фразу (снова «a beautiful day») и интересующую нас букву – на этот раз букву «b» вместо «a». После этого мы передадим обе эти переменные в вызов функции, как показано ниже:

```
String phrase = "a beautiful day";
String letter_of_interest = "b";
letterCounter(phrase, letter_of_interest);
```

Поскольку буква «b» в интересующей нас фразе встречается только один раз, мы ожидаем, что результатом выполнения этой функции будет 1, как показано на рис. 9.22.



The screenshot shows the Painless Lab interface with a code editor on the left and an output panel on the right. The code in the editor is as follows:

```
1
2 int letterCounter(String sample_string, String count_letter){
3     // initialize the string and the counter variable
4     int counter = 0;
5
6     for (int i=0;i<sample_string.length();i++){
7
8         // get a letter from the string using the substring method
9         String letter = sample_string.substring(i, i+1);
10
11        //use an if-statement to check if the current letter being processed is an a
12        if (letter==count_letter) {
13            // is yes, increment the counter
14            counter++;
15        }
16    }
17    return counter;
18 }
19
20 String phrase = "a beautiful day";
21 String letter_of_interest = "b";
22 letterCounter(phrase, letter_of_interest);
```

The output panel on the right shows the result of the function call, which is the number 1.

Рис. 9.22 ❖ Результат вызова функции показан на панели **Output** справа

Теперь вы знаете все, что нужно для написания собственного кода на языке Painless! Это пригодится в главе 13, где мы будем использовать расширенные функции Painless для извлечения признаков и написания обработчиков сценариев.

СОВМЕСТНОЕ ИСПОЛЬЗОВАНИЕ PYTHON И ELASTICSEARCH

В последние годы Python стал доминирующим языком во многих проектах с интенсивным использованием данных. Благодаря простым в применении библиотекам машинного обучения и анализа данных многие ученые и инженеры, занятые обработкой данных в своей повседневной деятельности, теперь в значительной степени полагаются на Python. Поэтому ваши знания

об использовании машинного обучения в Elastic Stack не будут полными, если мы не расскажем о работе с Elastic Stack в Python.

В этом разделе мы рассмотрим три официальных клиента Python Elasticsearch, исследуем различия между ними и обсудим, в каких случаях лучше использовать каждый из них. Мы продемонстрируем, как можно автоматизировать использование Elastic Stack ML с помощью клиентов Elasticsearch. Кроме того, мы более подробно рассмотрим Eland, новый собственный клиент для анализа данных, который обеспечивает эффективный анализ данных в памяти при поддержке Elasticsearch. После изучения того, как работает Eland, мы покажем, как можно применять комбинацию Eland с Jupyter Notebook – интерактивной средой с открытым исходным кодом – для анализа данных, хранящихся в Elasticsearch.

Краткий обзор клиентов Python Elasticsearch

Любой, кто использовал Kibana Dev Tools Console для связи с Elasticsearch, знает, что большинство задач выполняются через REST API. Вы можете вставлять, обновлять и удалять документы, вызвав нужную конечную точку с правильными параметрами. Неудивительно, что существует несколько уровней абстракции, на которых может быть написана клиентская программа, вызывающая эти конечные точки REST API. Низкоуровневый клиент `elasticsearch-py` (<https://elasticsearch-py.readthedocs.io/en/v7.15.1/>) предоставляет тонкую оболочку Python для вызовов REST API, которые обычно выполняются через консоль Kibana Dev Console или через какое-то приложение, способное отправлять HTTP-запросы. Следующий уровень абстракции реализован клиентом Elasticsearch DSL (<https://elasticsearch-dsl.readthedocs.io/en/latest/>). Наконец, наиболее абстрактным клиентом является Eland (<https://eland.readthedocs.io/en/v7.14.1b1/>), сильной стороной которого является табличное представление фреймов данных. В следующих примерах мы увидим, что это значит для специалистов по данным, желающих использовать Eland с Elasticsearch.

Помимо доступных клиентов Elasticsearch, мы рассмотрим различные среды выполнения, которые доступны любым инженерам данных или специалистам по данным, желающим работать с Python и Elasticsearch. Это, в свою очередь, приведет нас к знакомству с блокнотами Jupyter и всей экосистемой Jupyter, о которых следует знать всем желающим работать с машинным обучением и Elastic Stack.

Первый и, вероятно, наиболее привычный способ выполнить программу Python – это реализовать логику нашей программы в виде текстового файла (скрипта), сохранить его с расширением `.py`, чтобы указать, что он содержит код Python, а затем вызвать скрипт с помощью командной строки, как показано на рис. 9.23.

Второй способ – использовать интерактивную среду Python REPL, как показано на рис. 9.24. Вызов `python` (или `python3`) в нашей командной строке запустит интерактивную среду Python, в которой мы можем писать функции, определять переменные и выполнять все виды кода Python. Хотя эта среда

полезна для небольших или быстрых экспериментов, на практике в среде REPL будет сложно работать над долгосрочным или более крупным проектом совместного анализа данных. Поэтому для большинства проектов, связанных с Python, анализом данных и Elastic Stack, предпочтительной средой является интегрированная среда разработки, которая предоставляет редактор кода вместе с различными инструментами для поддержки как программирования, так и выполнения.

```
camilla@LAPTOP-MMIGNS8K:~$ python sample_python.py
hello world! I am a Python script
camilla@LAPTOP-MMIGNS8K:~$ _
```

Рис. 9.23 ❖ Один из способов работы с Python – сохранить программу в текстовом файле или скрипте, а затем выполнить его из командной строки

```
camilla@LAPTOP-MMIGNS8K:~$ python3
Python 3.5.2 (default, Oct  8 2019, 13:06:37)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> my_string = "hello world"
>>> for letter in my_string:
...     print(letter)
...
h
```

Рис. 9.24 ❖ Пример работы интерактивной оболочки REPL, которая идеально подходит для быстрых экспериментов с языком программирования Python

Среда разработки, специально созданная для анализа данных, – это Jupyter Notebook, показанный на рис. 9.25.

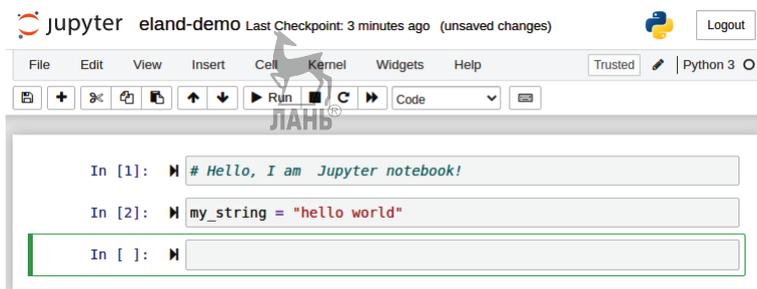


Рис. 9.25 ❖ Пример рабочего окна Jupyter Notebook

Jupyter Notebook может быть добавлен в Python через центральную службу управления пакетами, такую как `pip`, и запускается путем ввода команды `jupyter notebook` в командной строке. Jupyter запускается в браузере, на-

пример Chrome или Firefox, и предоставляет среду, в которой фрагменты кода, текстовые абзацы, графики и визуализации (как интерактивные, так и статические) могут существовать бок о бок. Мы полагаем, что среда Jupyter Notebook и экосистема библиотек, которая сформировалась вокруг нее, заслуживают намного большего внимания, чем мы можем уделить в этой главе, и поэтому предлагаем тем читателям, которые хотят эффективно использовать в анализе данных комбинацию инструментов Elasticsearch и Python, ознакомиться с перечнем дополнительной литературы в конце этой главы.

Зачем нам нужен Eland?



У читателей может возникнуть вопрос: зачем понадобился еще один клиент Python для Elasticsearch, когда у сообщества уже есть два клиента на выбор? Более того, зачем строить всю программную библиотеку вокруг идеи объекта Data Frame? Из ответов на оба этих вопроса можно составить еще одну книгу, поэтому представленные здесь краткие ответы упускают некоторые тонкости. Тем не менее мы надеемся, что у заинтересованного читателя после прочтения этого раздела возникнет понимание причин появления Eland и почему он был разработан на основе идеи фрейма данных.

Хотя сегодня Python доминирует во многих областях анализа данных и машинного обучения, так было не всегда. В частности, в начале 2010-х годов в экосистеме преобладал язык статистической обработки R, который имел очень полезную конструкцию – объект фрейма данных, позволяющий анализировать строки данных в табличной структуре (концепция, которая, несомненно, знакома пользователям Excel). Примерно в то же время Уэс МакКинни, тогда работавший в нью-йоркской финансовой компании AQR Capital, начал работать над библиотекой, призванной облегчить жизнь аналитикам данных Python. Эта работа завершилась выпуском pandas, библиотеки анализа данных с открытым исходным кодом, которую используют тысячи ученых и инженеров по данным.

Одной из ключевых функций, сделавших библиотеку pandas полезной и простой в использовании, был объект DataFrame. Подобно объекту R, этот объект упростил манипулирование данными и их анализ в табличной форме. Хотя библиотека pandas очень мощная и содержит множество встроенных функций и методов, она начинает сталкиваться с ограничениями, когда набор данных, который вы хотите проанализировать, слишком велик, чтобы поместиться в памяти.

В этих случаях аналитики часто прибегают к выборке данных из различных баз данных, например Elasticsearch, где они хранятся, экспортируют их в плоские файлы, а затем считывают в свой процесс Python, чтобы проанализировать с помощью pandas или другой библиотеки. Хотя этот подход, безусловно, работает, рабочий процесс стал бы проще и эффективнее, если бы pandas напрямую взаимодействовала с базой данных. Что, если бы мы могли прозрачно связать объект DataFrame с Elasticsearch, чтобы аналитик данных мог сосредоточиться на анализе данных, вместо того чтобы беспокоиться об управлении подключением и экспорте данных из Elasticsearch?

Это основная идея Eland. Надеюсь, в следующих разделах вам станет ясно, как эта философия реализована в библиотеке.

Знакомство с Eland

Поскольку Eland является сторонней библиотекой, вам сначала необходимо установить ее, чтобы использовать в своем коде Python. Инструкции по установке для различных операционных систем приведены в репозитории GitHub по адресу <https://github.com/elastic/eland>. Мы предполагаем, что читатели, желающие ознакомиться с материалом в этом разделе книги, следовали инструкциям, приведенным в ссылке, для завершения установки библиотеки. Примеры и снимки экрана в этой главе будут проиллюстрированы с использованием среды Jupyter Notebook, но также можно запустить образцы кода, представленные в этой главе, в автономной среде (например, из Python REPL или из скрипта Python). Необходимость использовать именно Jupyter Notebook будет четко оговорена в соответствующих примерах.

1. Первый шаг, который вы должны сделать, чтобы использовать Eland, – это импортировать его в свою среду. В Python это делается с помощью оператора `import`, как показано на рис. 9.26. Обратите внимание, что при использовании библиотеки, такой как Eland, для нее обычно назначается псевдоним. Во фрагменте кода, показанном на рис. 9.26, мы присвоили `eland` псевдоним `ed`, используя ключевое слово `as`. Это избавит нас от необходимости печатать длинное название в будущем, поскольку мы будем использовать имя библиотеки много раз при доступе к ее объектам и методам.

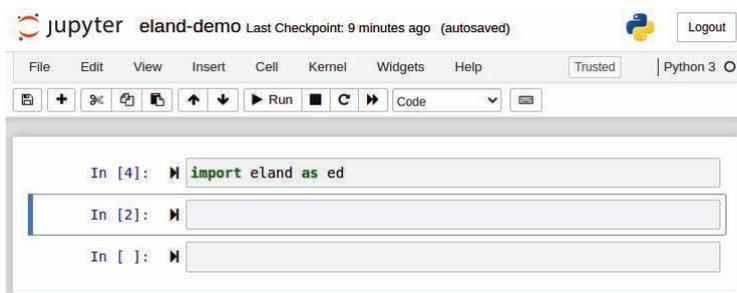


Рис. 9.26 ❖ Импорт Eland в блокнот Jupyter

2. После импорта библиотеки в Jupyter мы можем приступить к изучению. Начнем с самых простых вещей, которые можно делать с Eland: создания объекта `DataFrame`. Чтобы создать `DataFrame`, нам нужно указать две вещи: URL нашего кластера Elasticsearch (например, `localhost`, если мы запускаем Elasticsearch локально на порту 9200 по умолчанию) и имя хранилища Elasticsearch. Эти два параметра передаются в конструктор `DataFrame`, как показано на рис. 9.27.

```
In [4]: In [2]: In [ ]:
```

```
import eland as ed

INDEX_NAME = "kibana_sample_data_ecommerce"
ES_URL = "localhost"

ed_frame = ed.DataFrame(ES_URL, INDEX_NAME)
```

Рис. 9.27 ❖ Для создания DataFrame в Eland необходимо указать URL кластера Elasticsearch и имя хранилища, которое содержит данные, предназначенные для анализа

- Одна из первых задач, которые интересно выполнить, начиная изучать новый набор данных, – это просмотр того, как выглядят данные (обычно достаточно увидеть несколько строк, чтобы понять общую суть данных) и некоторые общие статистические свойства набора данных. Для просмотра образца данных нужно вызвать метод `head` нашего объекта `DataFrame`, как показано на рис. 9.28.

```
In [11]: ## basic data analysis operations with the data frame
ed_frame.head()
```

```
Out[11]:
```

	category	currency	customer_birth_date	customer_first_name	customer_full_name	customer_gender	customer_id	customer_la
OvVYAXcB9qtQdZZ-nCe1	[Men's Shoes, Men's Clothing]	EUR	NaT	Abdulraheem AI	Abdulraheem AI Carr	MALE	33	
O_VVYAXcB9qtQdZZ-nCe1	[Women's Clothing]	EUR	NaT	Selena	Selena Allison	FEMALE	42	
PPVYAXcB9qtQdZZ-nCe1	[Men's Clothing, Men's Accessories]	EUR	NaT	Phil	Phil Foster	MALE	50	
PRVYAXcB9qtQdZZ-nCe1	[Men's Clothing]	EUR	NaT	Fitzgerald	Fitzgerald Mcdonald	MALE	11	
PvVYAXcB9qtQdZZ-nCe1	[Men's Clothing, Men's Shoes]	EUR	NaT	Sultan AI	Sultan AI Fleming	MALE	19	

5 rows x 46 columns

Рис. 9.28 ❖ Вызов метода `head` объекта `DataFrame` покажет нам первые 5 строк набора данных

- Просмотр статистики достигается путем вызова метода `describe`, который знаком пользователям `pandas`, как показано на рис. 9.29.

```
In [12]: ed_frame.describe()
```

```
Out[12]:
```

	day_of_week_l	products.base_price	products.base_unit_price	products.discount_amount	products.discount_percentage	products.min_price	prod
count	4675.000000	10087.000000	10087.000000	10087.000000	10087.000000	10087.000000	100
mean	3.114866	34.886523	34.774993	0.100922	0.160603	17.337877	
std	1.930189	27.805031	25.691300	1.125216	1.785010	12.841733	
min	0.000000	5.988281	5.988281	0.000000	0.000000	2.759766	
25%	1.000000	16.984375	16.984375	0.000000	0.000000	8.699489	
50%	3.000000	25.844879	25.834470	0.000000	0.000000	13.464828	
75%	5.000000	43.906848	43.958546	0.000000	0.000000	22.500000	
max	6.000000	1080.000000	540.000000	15.000000	20.000000	259.250000	10

Рис. 9.29 ❖ Метод `describe` отображает основные статистические свойства числовых столбцов в наборе данных

5. Помимо получения общего обзора набора данных, мы можем легко получить доступ к отдельным значениям конкретного поля в хранилище, используя метод `get` в сочетании с именем поля, как показано на рис. 9.30.

```
In [13]: ed_frame.get('customer_full_name')

Out[13]: OvVYAXcB9qtQd2Z-nCe1    Abdulraheem Al Carr
         O_VYAXcB9qtQd2Z-nCe1    Selena Allison
         PPVYAXcB9qtQd2Z-nCe1    Phil Foster
         PfvYAXcB9qtQd2Z-nCe1    Fitzgerald Mcdonald
         PvvYAXcB9qtQd2Z-nCe1    Sultan Al Fleming
         ...
         B_RYAXcByJotxG6S3ern    Yahya Rivera
         BfRYAXcByJotxG6S3ern    Mary Lambert
         BvRYAXcByJotxG6S3ern    Jim Gilbert
         CPRYAXcByJotxG6S3ern    Mary Hampton
         CfRYAXcByJotxG6S3ern    Jackson Hopkins
         Name: customer_full_name, Length: 4675, dtype: object
```

Рис. 9.30 ❖ Можно работать со значениями отдельных полей в хранилище, используя метод `get`

6. Наконец, мы можем вычислить агрегаты для наших числовых столбцов, используя метод `aggregate`. В примере, показанном на рис. 9.31, мы выбираем два числовых столбца `total_unique_products` и `taxful_total_price` и вычисляем сумму, минимум и максимум значений в этих полях для всех документов в хранилище.

```
In [22]: ## perform aggregations on numerical columns
         ed_frame[['total_unique_products', 'taxful_total_price']].aggregate(['sum', 'min', 'max'])

Out[22]:
```

	total_unique_products	taxful_total_price
sum	10087	350884.128906
min	1	6.988281
max	4	2250.000000

Рис. 9.31 ❖ В Eland возможно вычисление агрегатов по выбранным столбцам с использованием метода `aggregate`

Хотя приведенные здесь примеры относительно просты, мы надеемся, что они продемонстрировали, насколько легко можно объединить Elasticsearch, работу на Python и среду анализа данных, такую как Jupyter Notebook, в один цельный рабочий процесс анализа данных. Полученные навыки работы с Eland пригодятся в главе 13, когда мы будем рассматривать более сложные варианты использования.

ЗАКЛЮЧЕНИЕ



В этой главе мы погрузились в мир анализа фреймов данных – совершенно новую ветвь инструментов машинного обучения и преобразования данных,

которые предлагают эффективные способы использования данных, сохраненных в Elasticsearch. Помимо обзора новых методов машинного обучения с учителем и без учителя, которые мы подробнее рассмотрим в следующих главах, вы изучили три важные темы: преобразования, использование языка сценариев Painless и интеграция Python и Elasticsearch. Эти темы лягут в основу следующих глав.

Вы изучили два компонента преобразований данных – сводные таблицы и агрегаты, а также два возможных режима выполнения преобразования: пакетный и непрерывный. Пакетное преобразование выполняется только один раз и генерирует преобразованные данные по срезу исходного хранилища в определенный момент времени. Это отлично работает для наборов данных, которые мало меняются, или когда преобразование данных должно выполняться только в определенный момент времени. Для многих реальных случаев использования, таких как ведение журнала или наш знакомый пример интернет-магазина, отслеживаемая система и анализируемые данные постоянно меняются. Приложение постоянно регистрирует действия своих пользователей, интернет-магазин постоянно регистрирует новые транзакции. Для анализа таких потоковых наборов данных предпочтительными инструментами являются непрерывные преобразования.

Хотя предварительно настроенные параметры, доступные в мастере преобразования, который мы продемонстрировали в наших примерах, подходят для большинства ситуаций, более опытные пользователи могут нуждаться в настройке своих собственных агрегатов. Для этого (и в целом чтобы иметь возможность настроить многие более сложные конфигурации, которые мы обсудим в следующих главах) пользователи должны быть знакомы с Painless, языком сценариев, встроенным в Elasticsearch. Вы узнали, как объявлять переменные в Painless, как выполнять над ними математические и логические действия, как создавать более сложные программы, используя условные операторы и оператор цикла, и, наконец, как оформить полезные фрагменты кода в виде функций. Все это пригодится в следующих главах!

И последнее, но не менее важное: вы узнали об использовании Python при анализе данных, хранящихся в Elasticsearch. Вы познакомились с существующими клиентами Python для Elasticsearch, `elasticsearch-py` и `elasticsearch-dsl`, и получили первые навыки использования третьего и новейшего клиента Eland.

В следующей главе мы займемся изучением первого из трех новых методов машинного обучения, добавленных в Elastic Stack, – обнаружения выбросов.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Для получения дополнительной информации об экосистеме Jupyter и, в частности, о Jupyter Notebook ознакомьтесь с исчерпывающей документацией Project Jupyter по адресу <https://jupyter.org/documentation>.

Если вы новичок в Python и хотели бы получить обзор экосистемы языка и различных доступных инструментов, ознакомьтесь с увлекательным руководством по адресу <https://docs.python-guide.org/>.

Чтобы узнать больше о проекте pandas, прочтите документацию по адресу <https://pandas.pydata.org/>.

Для получения дополнительной информации о встроенном языке сценариев Painless ознакомьтесь с официальной спецификацией языка по адресу <https://www.elastic.co/guide/en/elasticsearch/painless/current/painless-lang-spec.html>.



Глава 10

.....

Обнаружение выбросов

В первом разделе этой книги мы подробно обсудили обнаружение аномалий, функцию, которая позволяет нам обнаруживать необычное поведение в данных временных рядов при помощи машинного обучения без учителя. Этот механизм хорошо работает, когда мы хотим определить, например, испытывает ли одно из наших приложений необычную задержку в определенный момент времени или хост в нашей корпоративной сети передает необычное количество байтов.

В этой главе вы узнаем о второй функции машинного обучения без учителя в Elastic Stack: обнаружении выбросов, которое позволяет нам обнаруживать необычные объекты в хранилищах, не основанных на временных рядах. К интересным примерам применения этого механизма можно отнести обнаружение необычных клеток в образце ткани, исследование необычных домов или участков на местном рынке недвижимости и обнаружение необычных двоичных файлов, установленных на вашем компьютере.

Обнаружение выбросов в Elastic Stack основано на совокупности или группе из четырех различных методов обнаружения выбросов. Два из этих методов основаны на плотности, то есть они пытаются определить, какие точки данных в вашем хранилище находятся далеко от основной массы данных, и два основаны на расстоянии, то есть они пытаются определить, какие точки находятся на наибольшем расстоянии от всех других точек. Хотя по отдельности каждый из четырех алгоритмов имеет свои сильные и слабые стороны, взятые вместе, они обеспечивают надежное обнаружение выбросов. Позже в этой главе мы обсудим, что делает каждый из этих алгоритмов на концептуальном уровне.

Помимо изучения технологии, которая обеспечивает обнаружение выбросов, мы рассмотрим, чем обнаружение выбросов отличается от обнаружения аномалий, как настроить задание обнаружения выбросов в Elasticsearch, как интерпретировать результаты обнаружения выбросов, а также как понять, какие признаки были ответственны за то, что данная точка была объявлена выбросом. В этой главе мы рассмотрим следующие темы:

- принцип работы механизма обнаружения выбросов;
- применение обнаружения выбросов на практике;

- оценка качества обнаружения выбросов с помощью API Evaluate;
- настройка гиперпараметров для обнаружения выбросов.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Материал в этой главе основан на использовании Elasticsearch версии 7.9 или выше. Числовые значения в этой главе были получены с помощью Elasticsearch 7.10. Примеры кода, используемые здесь, находятся в репозитории GitHub по адресу <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter10>.

ПРИНЦИП РАБОТЫ МЕХАНИЗМА ОБНАРУЖЕНИЯ ВЫБРОСОВ

Обнаружение выбросов может предложить новый взгляд на набор данных за счет обнаружения необычных точек, но как работает обнаружение выбросов в Elastic Stack? Чтобы понять, как устроен механизм обнаружения выбросов, давайте начнем с концептуального размышления о том, как бы вы спроектировали такой алгоритм, а затем посмотрим, как ваши концептуальные идеи могут быть формализованы в четырех отдельных алгоритмах, составляющих ансамбль обнаружения выбросов в Elasticsearch.

Предположим, что у нас есть двумерный набор измерений веса и обхвата тыкв и мы хотим выяснить, какие тыквы являются выбросами в этой популяции (возможно, мы хотим использовать эту информацию, чтобы выяснить, почему они являются выбросами). Хорошим первым шагом было бы построить график данных, чтобы увидеть, есть ли какие-либо очевидные точки данных, которые кажутся далекими от других (рис. 10.1).

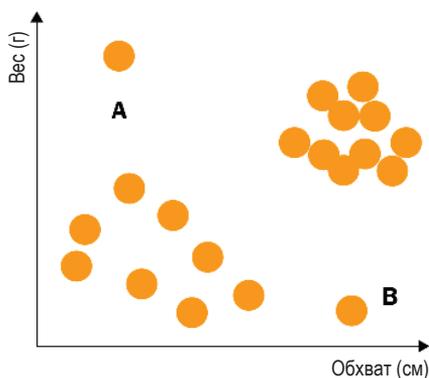


Рис. 10.1 ❖ Точки A и B выглядят выбросами в этом наборе данных, потому что они расположены далеко от общей массы данных

Человеческие глаза исключительно хорошо улавливают визуальные закономерности, и беглый взгляд на график на рис. 10.1 скажет вам, что точки А и В являются выбросами. Какие интуитивные рассуждения привели нас к такому выводу? Наша зрительная система сообщает нам, что обе точки А и В являются в некотором смысле далекими от двух других отдельных групп точек в двумерном пространстве. Такое наблюдение и его формализация – это ключевой момент, на котором основаны методы обнаружения выбросов, используемые в Elastic Stack.

Обзор четырех методов обнаружения выбросов

Как было сказано ранее в этой главе, алгоритм обнаружения выбросов в Elastic Stack представляет собой совокупность или группу из четырех различных методов обнаружения выбросов. Эти методы можно подразделить на две категории: на основе расстояния и на основе плотности. Мы рассмотрим каждую из них по очереди в следующих разделах.

Методы, основанные на расстоянии

Как вы сами только что убедились, зрительная система человека обладает невероятной способностью улавливать выбросы на двумерных изображениях, и суть этого навыка заключается в обнаружении точек, которые кажутся далекими от общей массы точек данных. Это наблюдение реализуют два метода на основе расстояния: *расстояние до k -ближайших соседей* и *среднее расстояние до k -ближайших соседей*.

Предположим, у нас есть двумерный набор данных, который распределен в пространстве, как показано на рис. 10.2, и предположим, что мы выбрали значение k равным 3. На этом этапе мы просто выбираем произвольно низкое значение k в целях иллюстрации. Это означает, что для точки А на рис. 10.2 мы находим третью ближайшую точку и вычисляем расстояние от точки А до нее (отмечено более толстой стрелкой на рис. 10.2). Этот подход

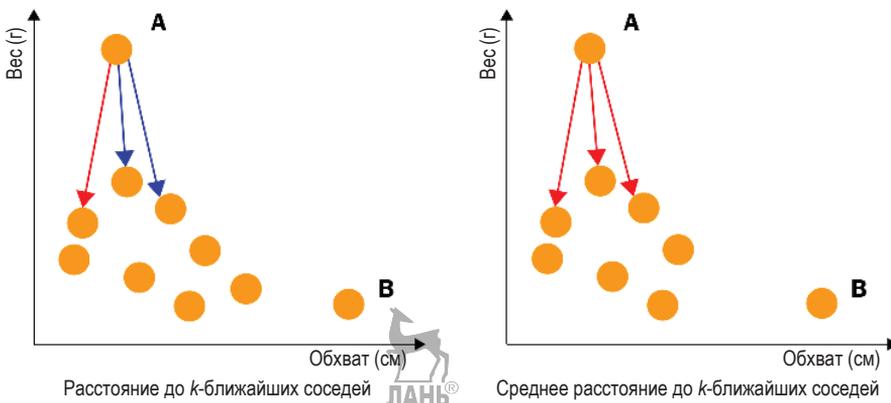


Рис. 10.2 ❖ Расстояние до k -ближайшего соседа от точки А и среднее расстояние до k -ближайшего соседа для точки А при $k = 3$

отличается простотой, но он также подвержен шуму. Чтобы сделать его надежнее, мы можем вычислить среднее расстояние до k -ближайших соседей (показано на рис. 10.2 справа).

Хотя методы, основанные на расстоянии, великолепны по своей простоте и интерпретируемости, они неспособны уловить некоторые тонкости пространственного распределения данных, в частности насколько разреженным или плотным является окружение каждой точки данных. Чтобы уловить эти свойства, мы должны использовать методы, основанные на плотности.

Методы, основанные на плотности

Одним из факторов, который методы, основанные на расстоянии, не могут полностью уловить, является разница между плотностью точек в окрестности интересующей нас точки и плотностью точек вокруг ее соседей. Вычисление локальной факторной меры выброса (local outlier factor) точки отмечает именно это: насколько окрестности данной точки отличаются от окрестностей других точек.

Рисунок 10.3 иллюстрирует основную идею этого метода для $k = 3$. В этом случае мы сравниваем окрестность точки А с окрестностями трех ее ближайших соседей (показано пунктирными кружками на диаграмме на рис. 10.3). Значение 1 для локальной факторной меры означает, что окрестность точки А сравнима с окрестностями ее соседей – она не более разреженная и не более плотная. Значение больше 1 означает, что окрестность А более разреженная, чем окрестность его соседей, и потенциально может быть выбросом. И наоборот, значение меньше 1 означает, что точка плотно окружена своими соседями и, следовательно, вряд ли будет выбросом.

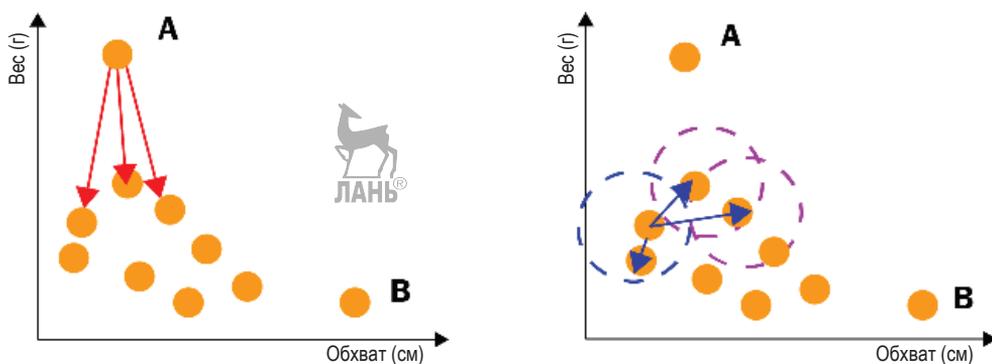


Рис. 10.3 ❖ Локальная факторная мера выброса сравнивает окрестности точки А с окрестностями k -ближайших соседей

Последний метод в нашей подборке из четырех методов – это локальный фактор выбросов, основанный на расстоянии (local distance-based outlier factor, LDOF). Подобно локальной факторной мере выброса (local outlier factor, LOF), цель LDOF – сравнить окрестность заданной точки А с окрестностями ее соседей.

В этом случае мы вычисляем среднее расстояние до k -ближайших соседей A для некоторого фиксированного значения k , $avg(A)$. Затем для каждого k -ближайшего соседа A вычисляем попарные расстояния и берем их среднее значение, $avgkk(A)$.

Наконец, мы проверяем отношение $avg(A)/avgkk(A)$, чтобы увидеть, насколько оно близко к значению 1. Если отношение приближается к 1, это означает, что точка A окружена локальной плотной группой других точек, и, таким образом, маловероятно, что она является выбросом.

Окончательная общая оценка выброса, присваиваемая каждой точке данных, представляет собой комбинацию значений, полученных с помощью четырех вышеуказанных методов. Чем ближе значение к 1, тем больше вероятность того, что точка будет выбросом.

Хотя иногда мы просто хотим выяснить, какие точки в наших наборах данных являются выбросами, в других случаях мы также хотим понять, почему алгоритм обнаружения выбросов предполагает, что конкретная точка является выбросом. Есть ли какая-то особенность, значение поля или, возможно, группа значений, которая делает эту точку зрения необычной? Это тема, которую мы рассмотрим в следующем разделе.

Оценка влияния характеристики

Давайте ненадолго вернемся к нашему вымышленному набору данных о тыквах из начала этой главы. Предположим, мы анализируем этот набор данных с помощью алгоритма обнаружения выбросов. После завершения анализа для каждой тыквы мы получаем оценку от 0 до 1, которая определяет необычность тыквы. Помимо оценки, нам также может быть интересно понять, какая именно характеристика – вес тыквы или ее обхват – способствовала ее необычности.

Это как раз та проблема, которую стремится решить оценка *влияния характеристики* (feature influence). Вкратце: алгоритм присваивает каждой характеристике (или полю, если использовать термины, которые мы обычно используем для описания документов Elasticsearch) оценку от 0 до 1, которая описывает, насколько важной была характеристика для принятия решения о том, что точка данных является выбросом. Общая сумма баллов влияния по всем характеристикам в сумме составляет 1.

Давайте подробнее рассмотрим влияние характеристик с помощью набора данных тыкв на рис. 10.4. Предположим, что наш алгоритм обнаружения выбросов идентифицировал точки A и B как выбросы в этом наборе данных. Теперь давайте посмотрим, как оценки влияния характеристик – веса и обхвата тыквы – будут соотноситься друг с другом в точках A и B .

Тыква A имеет вес, который выходит далеко за пределы нормального диапазона веса тыкв, но ее обхват находится где-то посередине значений обхвата тыкв левого кластера. Таким образом, мы ожидаем, что оценка влияния веса тыквы будет высокой для точки A , но оценка влияния обхвата будет низкой.

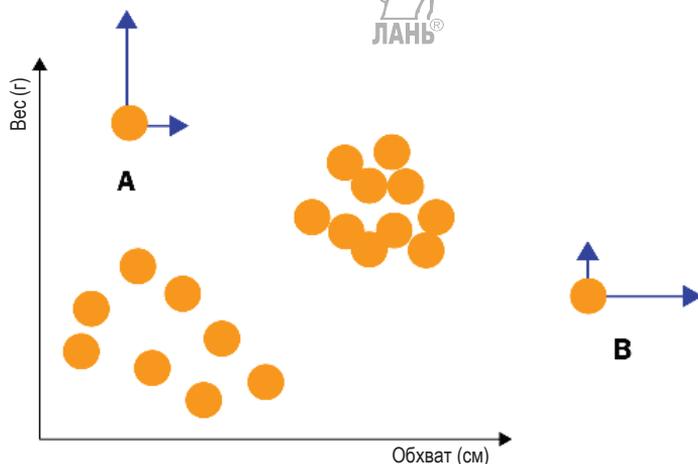


Рис. 10.4 ❖ Показатели влияния характеристики зависят от того, насколько важна данная характеристика при определении необычности точки данных

Теперь давайте посмотрим на тыкву-выброс В, где ситуация обратная. В то время как вес тыквы В попадает в средний диапазон набора данных, обхват тыквы В намного больше, чем любая другая точка данных. Таким образом, для тыквы В оценка влияния обхвата будет выше, чем оценка влияния веса.

Как рассчитывается оценка влияния характеристик для каждой точки?

При интерпретации оценок влияния характеристик часто бывает полезно точно знать, что входит в расчет. Давайте ненадолго вернемся к нашему двумерному набору тыквенных данных, чтобы проиллюстрировать шаги, которые входят в расчет оценки влияния характеристик. Мы хотим установить, какое влияние конкретная характеристика X , скажем вес тыквы, оказывает на окончательный результат выброса. Естественный способ попытаться количественно оценить этот эффект – представить, что мы вообще не включаем эту функцию в наши вычисления выбросов. На рис. 10.5 показано, как это будет выглядеть на практике для нашего примера с тыквой. Для каждой точки данных тыквы мы проецируем значение характеристики «Вес» на нулевой уровень и смотрим, насколько изменилась необычность каждой точки данных.

По рис. 10.5 видно, что для точки А удаление характеристики или проецирование значения веса на 0 в конечном итоге приведет к тому, что точка А перестанет быть выбросом. Следовательно, мы можем сделать вывод, что характеристика веса имеет большое влияние на необычность точки А. С другой стороны, если мы посмотрим на точку В на правом графике на рис. 10.5, то увидим, что в результате обнуления характеристики веса необычность точки не изменилась. Следовательно, мы можем сделать вывод, что характеристика «Вес» не имеет большого отношения к необычности точки В, и, следовательно, значение ее влияния будет низким.

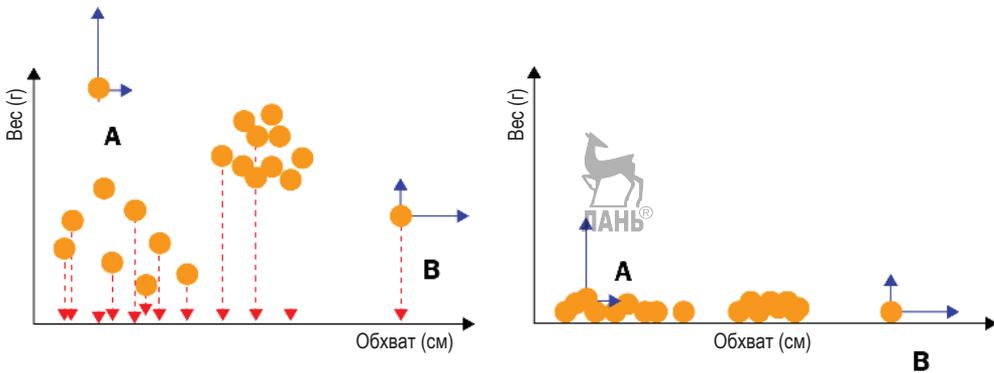


Рис. 10.5 ❖ Влияние характеристик рассчитывается путем выяснения того, насколько необычность точки данных изменится, если данная характеристика будет обнулена

Чем обнаружение выбросов отличается от обнаружения аномалий?

Читая эту главу, вы могли заметить, что как обнаружение выбросов, так и обнаружение аномалий – это методы обучения без учителя, которые пытаются достичь схожей цели: найти необычные или обособленные точки данных. Тогда возникает естественный вопрос: *чем обнаружение выбросов отличается от обнаружения аномалий?* В этом разделе мы собираемся объяснить основные различия между ними. Перечень различий представлен в табл. 10.1, которую вы скоро увидите.

Сравнение вероятностных моделей и экземпляров

Чтобы сделать различие между обнаружением аномалий и обнаружением выбросов более понятным, давайте сначала кратко рассмотрим доступные методы обнаружения аномалий. Механизм обнаружения аномалий позволяет нам обнаруживать необычные особенности в данных на основе временных рядов. Для этого временной ряд разбивается на дискретные интервалы времени, называемые сегментами, а затем к отдельным значениям в сегменте применяется функция детектора, такая как среднее значение или сумма. Каждое значение сегмента затем используется как отдельная точка данных в распределении вероятностей, которое постоянно обновляется, по мере того как детектор аномалий видит все больше и больше данных. Сегменты, вероятность появления которых в соответствии с распределением вероятностей низкая, помечаются как аномалии.

Вместо построения вероятностной модели, которая отслеживает эволюцию наших данных во времени, для обнаружения выбросов используется ансамбль (или группа) из четырех методов – два метода на основе расстояния и два метода на основе плотности, которые были рассмотрены ранее в этой

главе. Чем дальше точка данных находится от общей массы данных в наборе, тем больше вероятность, что она будет выбросом. Для набора данных не строится вероятностная модель.

Подсчет оценок

Это существенное различие в двух механизмах приводит нас, в свою очередь, к разнице в оценках. Оценка аномальности сегмента определяется тем, насколько маловероятно его появление в рамках модели, которую детектор аномалий построил, исходя из данных. Чем ниже вероятность, тем аномальнее сегмент.

Напротив, при обнаружении выбросов мы вычисляем оценку выбросов, а не вероятности. Оценка выбросов – это непрерывная мера в диапазоне от 0 до 1, которая отражает сводную меру того, насколько далеко данная точка данных находится от общей массы данных во всем наборе. Как вы видели чуть ранее, этот показатель вычисляется с использованием четырех различных методов. Чем выше показатель выброса, тем более аномальной или необычной является точка данных в наборе.

Характеристики данных

Помимо оценки, еще одно важное различие между этими двумя методами – это тип данных, для которых они предназначены. Обнаружение аномалий подходит только для данных временных рядов, в то время как обнаружение выбросов можно использовать для одно- или многомерных наборов данных, которые могут содержать или не содержать компонент, основанный на времени.

Потоковая и пакетная обработка

Наконец, последнее существенное различие между двумя методами обучения без учителя заключается в том, насколько они поддаются обновлению, если в индекс вводятся новые данные. Пользователи, знакомые с обнаружением аномалий, знают, что этот метод отлично подходит для потоковой передачи данных. Сразу после поступления и обработки нового пакета вероятностная модель может быть обновлена для отражения новых данных.

И наоборот, обнаружение выбросов не работает в режиме, близком к реальному времени, как обнаружение аномалий. Если группа новых точек данных добавляется в исходное хранилище, мы должны повторно запустить задание обнаружения выбросов. Причина этого в том, что обнаружение выбросов – это метод на основе экземпляров, который использует пространственное и плотностное распределение точек данных, чтобы определить, какие из них являются нормальными, а какие – выбросами. Любые новые точки могут изменить пространственное распределение данных до такой степени, что точки, ранее классифицированные как выбросы, перестанут быть выбросами, и, следовательно, повторная оценка новых данных требует, чтобы оценка выбросов была пересчитана для всего набора данных в пакете.

Таблица 10.1. Основные различия между обнаружением аномалий и обнаружением выбросов

Обнаружение аномалий	Обнаружение выбросов
На основе модели: обучение вероятностной модели для обнаружения сегментов, имеющих низкую вероятность появления	На основе экземпляра: мы не строим вероятностную модель, а используем пространственный и плотностный анализы целого пакета данных
Оценка: на основе вероятностей; чем ниже вероятность появления точки данных, тем выше ее оценка аномальности	Оценка: чем сильнее точка отдалена от основной массы данных, тем выше ее оценка как потенциального выброса
Данные: набор данных обязательно имеет компонент времени	Данные: набор данных является одно- или многомерным и не обязан (хотя может) иметь компонент времени
Поток: вероятностная модель может меняться по мере того, как получает больше новых данных, чтобы обновлять свои прогнозы	Поток: если в хранилище добавлены новые точки данных, оценки выбросов должны быть пересчитаны заново для полного набора, потому что добавление новых точек может полностью изменить картину распределения

ПРИМЕНЕНИЕ ОБНАРУЖЕНИЯ ВЫБРОСОВ НА ПРАКТИКЕ

В этом разделе мы рассмотрим практический пример обнаружения выбросов с использованием общедоступного набора данных, описывающих физико-химические свойства вина. Этот набор данных доступен для загрузки из репозитория Калифорнийского университета в Ирвине (UCI) по адресу <https://archive.ics.uci.edu/ml/datasets/wine+quality>.

Набор данных о винах состоит из двух файлов CSV: один описывает физико-химические свойства белого вина, другой – красного. В этом пошаговом руководстве мы будем использовать набор данных о белом вине, но вы также можете использовать данные для красного вина, поскольку большинство шагов, описанных здесь, применимы к обоим наборам.

Сначала импортируем набор данных в наш кластер Elasticsearch с помощью инструмента Data Visualizer, который вы можете найти в приложении Machine Learning в Kibana. Мы создадим хранилище для набора данных белого вина и назовем его winequality-white (рис. 10.6).

Беглый взгляд на данные на вкладке **Discover** говорит нам, что каждый документ представляет собой одно вино и содержит информацию о содержании алкоголя, кислотности¹, pH, содержании сахара и прочие химические характеристики, а также качественную оценку, присвоенную дегустаторами. Цель нашего исследования – использовать обнаружение выбросов, чтобы определить, какие вина являются выбросами с точки зрения их химического

¹ Кислотность (субацидность) вина как органолептический показатель и pH (чисто химический показатель) – это разные понятия. Правильное сочетание органических кислот определяет освежающую бодрость вкуса, а неправильное способно «убить» вино, сделав его тусклым и кислым на вкус. – *Прим. перев.*

состава, а затем посмотреть, коррелирует ли это с оценкой качества, полученной от дегустаторов. Наша гипотеза заключается в том, что вина, необычные по своим характеристикам химического состава, также будут выделяться по качеству. Для проверки этой гипотезы выполните следующие действия.

1. Создайте задание обнаружения выбросов с помощью мастера Data Frame Analytics, как показано на рис. 10.7.

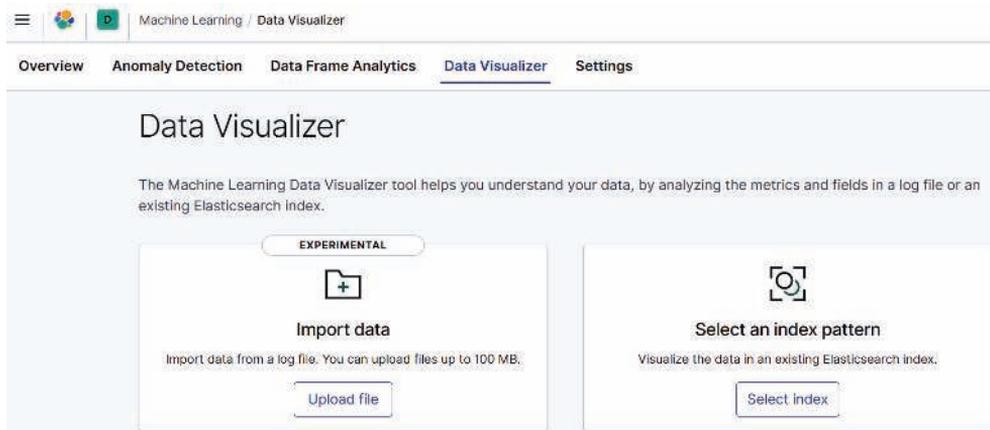


Рис. 10.6 ❖ Инструмент визуализации данных находится в приложении Machine Learning в Kibana и удобен для импорта небольших файлов данных, которые можно использовать для экспериментов



Рис. 10.7 ❖ Используйте мастер Data Frame Analytics для создания задания обнаружения выбросов

2. Поскольку мы заинтересованы в сопоставлении вин, которые выделяются по химическому составу, с винами, которые выделяются оценкой качества, мы исключим оценку качества из задания по обнаружению выбросов, как показано на рис. 10.8.



Рис. 10.8 ❖ Исключите оценку качества из задания по обнаружению выбросов, сняв флажок рядом с именем поля

- Мы будем использовать настройки по умолчанию для остальных параметров конфигурации. Как только задание будет завершено, мы можем проверить результаты с помощью средства просмотра результатов Data Frame Analytics (рис. 10.9).

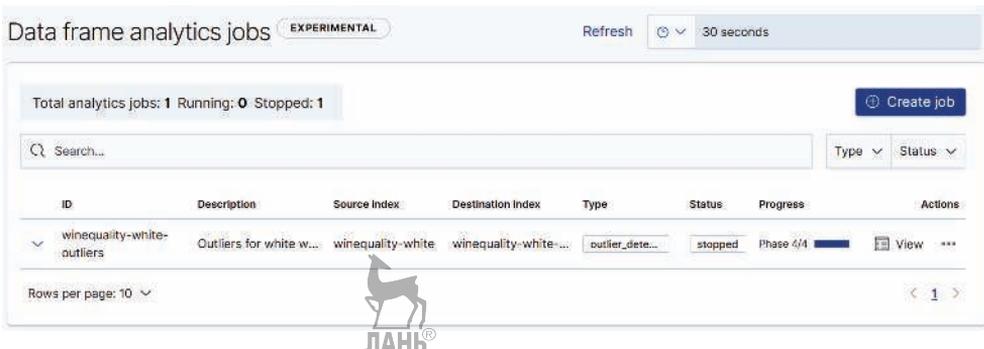


Рис. 10.9 ❖ Используйте пользовательский интерфейс Data Frame Analytics, чтобы узнать о завершении задания

- Вид пользовательского интерфейса результатов анализа фрейма данных показан на рис. 10.10. В этом интерфейсе есть несколько ключевых элементов, которые помогут нам определить, какие точки данных (какие из наших образцов белого вина) были выбросами и какие ключевые особенности определили их необычность. Давайте рассмотрим каждый элемент по очереди.

Слева на рис. 10.10 пользовательский интерфейс отображает количество `ml.outlier_score`. Оценка выброса – это дробное значение в интервале от 0 до 1, которое показывает, насколько эта точка данных отличается от набора. Чем ближе данная оценка к 1, тем сильнее точка удалена от основной массы данных, и наоборот.

Остальные столбцы в таблице показывают нам значения, извлеченные из других полей набора данных. Каждая ячейка закрашена в соответствии со значением градиента от 0 до 1, что отражает влияние характе-

ристики, другими словами, насколько значимой была характеристика в определении необычности точки данных. Чем темнее оттенок синего в данной ячейке, тем важнее эта характеристика для необычности точки.

Например, посмотрев на значения в столбце `ml.outlier_score` на рис. 10.10, мы можем увидеть, что каждое из четырех самых необычных вин в наборе данных набрало 0,998 балла оценки выбросов.

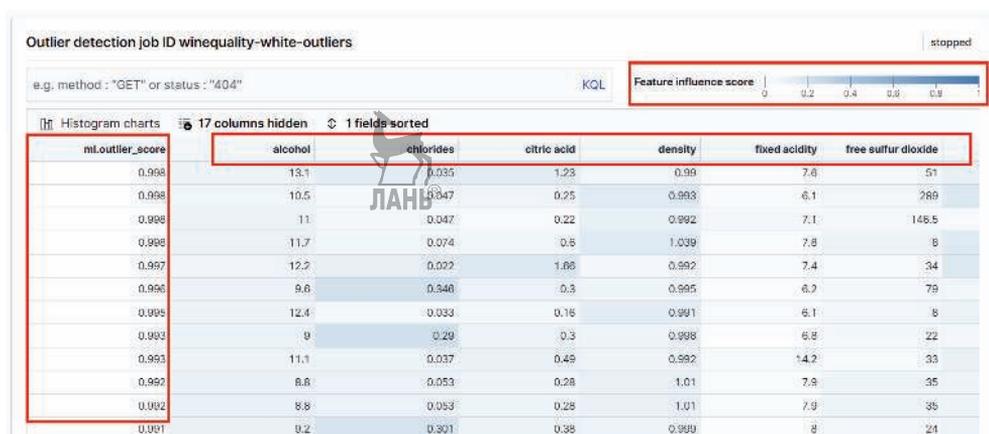


Рис. 10.10 ❖ Пользовательский интерфейс результатов задания обнаружения выбросов отображает сводную таблицу, в которой представлены оценки выбросов для каждой точки данных, а также набор полей (в алфавитном порядке) с цветовой кодировкой и оценкой значимости характеристики

⚠ Здесь возникает интересный вопрос: *каков порог, при котором мы объявляем точку выбросом?* Можно ли сказать, что все точки с оценкой выше 0,5 являются выбросами? Или мы установим более консервативный порог и скажем, что только точки с оценкой выше 0,9 являются выбросами? Процесс установки порогового значения для непрерывной оценки, чтобы разделить точки на две группы – нормальные и выбросы, известен как *бинаризация* и часто определяется путем объединения знаний о предметной области и целей, которые есть у пользователя (т.е. на основе экспертного мнения). Однако при наличии размеченного набора данных (например, набора, в котором каждая точка данных уже снабжена меткой нормы или выброса) можно провести несколько более систематический процесс выбора порога. Мы вернемся к этой теме в следующем разделе, когда рассмотрим API Evaluate.

- Теперь вернемся к пользовательскому интерфейсу результатов и посмотрим на окраску ячеек, чтобы решить, сможем ли мы почерпнуть интересную информацию о том, какие факторы делают то или иное вино необычным. Мы можем включить отображение скрытых столбцов в пользовательском интерфейсе и добавить все оставшиеся характеристики, чтобы увидеть полную картину влияния характеристик для самых верхних удаленных точек данных, как показано на рис. 10.11.

Outlier detection job ID winequality-white-outliers

e.g. method: "GET" or status: "404"

Histogram charts 13 columns hidden 1 fields sorted

ml.outlier_score	alcohol	chlorides	citric acid	density	fixed acidity	free sulfur dioxide	pH	residual sugar	sulphates	total sulfur dioxide	volatile acidity
0.998	13.1	0.035	1.23	0.99	7.0	51	3.03	4.6	0.43	294	0.25
0.998	10.5	0.047	0.23	0.993	6.1	289	3.44	2.9	0.64	440	0.26
0.998	11	0.047	0.23	0.992	7.1	148.5	3.24	2	0.37	307.5	0.69
0.998	11.7	0.074	0.4	1.029	7.8	8	3.39	65.8	0.69	180	0.965
0.997	12.2	0.022	1.86	0.992	7.4	34	3.25	2.1	0.55	113	0.2
0.996	9.6	0.346	0.3	0.995	6.2	79	3.29	6.6	0.58	200	0.37
0.995	12.4	0.033	0.16	0.991	6.1	8	3.35	4.4	0.47	109	1.1
0.993	9	0.29	0.3	0.998	6.8	22	3.08	13	0.67	193	0.67
0.993	11.1	0.037	0.49	0.992	14.2	33	3.15	1.1	0.54	156	0.27
0.992	8.8	0.053	0.28	1.01	7.9	35	3.15	31.6	0.38	176	0.33
0.992	8.8	0.053	0.28	1.01	7.9	35	3.15	31.6	0.38	176	0.33
0.991	9.2	0.301	0.38	0.999	8	24	2.94	12.1	0.48	220	0.61
0.989	12	0.041	1	0.998	7.5	33	3.24	19.5	0.38	148	0.4
0.988	8.9	0.058	0.35	1.002	7.5	129	3.44	17.8	0.43	212	0.23
0.987	10.2	0.052	0.45	0.999	9.8	34	3.12	8.6	0.59	187	0.93
0.986	13.8	0.036	0	0.99	4.7	23	3.53	3.4	0.92	134	0.785
0.985	11	0.037	0.29	0.994	9.4	124	2.9	8.5	0.38	208	0.24
0.985	12.6	0.039	0.21	0.992	9.6	21	3	2	1	120	0.855
0.98	9.6	0.039	0.24	0.995	6.2	138.5	3.53	1.7	0.53	272	0.255
0.98	12	0.029	0.36	0.99	4.2	93	3.65	1.8	0.89	161	0.17
0.978	9.2	0.239	0.49	0.999	7.6	42	2.96	13	0.51	220	0.47
0.978	10.2	0.046	0.46	0.997	9.9	34	3.02	1.4	0.49	185	1.005
0.973	9.8	0.271	0.2	0.994	7.1	24	3.11	1.6	0.63	140	0.36
0.968	12	0.032	1	0.997	7.7	42	3.29	10.95	0.5	164	0.43
0.965	9.3	0.211	0.42	0.999	8	37	2.99	12.6	0.58	213	0.55

Рис. 10.11 ❖ Влияние характеристик, отображаемое для всех полей в наборе данных

ml.outlier_score	quality
0.998	6
0.998	3
0.998	3
0.998	6
0.997	6
0.996	5
0.995	4
0.993	4
0.993	6
0.992	6

Рис. 10.12 ❖ Самые необычные белые вина, отсортированные по убыванию оценки выбросов, а также оценка качества, присвоенная дегустаторами

Как видно из отмеченных областей, выделяющиеся вина необычны по разным причинам. Для первой точки данных полями с наивысшими оценками влияния являются содержание хлоридов и лимонной кислоты, в то время как для следующих трех наиболее важными характеристиками при определении необычности, по-видимому, являются плотность и pH вина.

6. Наконец, мы можем вернуться к вопросу, который задали в начале этого раздела. *Коррелирует ли необычность вина с количественной оценкой качества, присвоенной ему дегустаторами?* Чтобы проверить, можем ли мы с первого взгляда выявить какую-либо корреляцию, давайте добавим оценку качества к набору данных вместе с оценкой выбросов (рис. 10.12).

Как мы видим, ни одно из 10 самых выдающихся белых вин не попадает в категорию лучших (оценка качества 9). Вместо этого большинство из них набрало 3–6 баллов. Хотя это неубедительное доказательство, у нас есть намек на то, что вина с необычным химическим составом обычно не получают высокую оценку вкуса!

ОЦЕНКА КАЧЕСТВА ОБНАРУЖЕНИЯ ВЫБРОСОВ С ПОМОЩЬЮ API EVALUATE

В предыдущем разделе мы упомянули тот факт, что пользователю может быть сложно установить пороговое значение оценки выброса для разделения точек данных на две группы. В этом разделе мы продемонстрируем подход к решению этой проблемы, если у вас есть размеченный набор данных, который для каждой точки содержит эталонные значения, указывающие, является ли точка выбросом. Прежде чем мы перейдем к демонстрации примера, давайте разберемся с некоторыми ключевыми показателями достоверности, которые используются при оценке качества алгоритма обнаружения выбросов.

Один из простейших способов измерить достоверность алгоритма – вычислить количество точек данных, которые он правильно предсказал как выбросы; другими словами, найти количество *истинных положительных результатов* (true positives, TP). Кроме того, нужно знать количество *истинно отрицательных результатов* (true negative, TN): сколько нормальных точек данных было правильно предсказано как нормальное. Кроме того, нам нужно отметить, сколько раз алгоритм обнаружения выбросов совершал одну из двух возможных ошибок: либо нормальные точки были ошибочно помечены как выбросы, т. е. *ложноположительные результаты* (false positive, FP), либо наоборот – *ложноотрицательные* (false negative, FN).

Эти четыре показателя могут быть удобно представлены в табличной форме, известной как *матрица неточностей* (confusion matrix). Пример матрицы несоответствий показан на рис. 10.13.

		Предсказание	
		Выброс	Норма
Реальность	Выброс	TP	FN
	Норма	FP	TN

Рис. 10.13 ❖ Матрица неточностей, отображающая истинно положительные, истинно отрицательные, ложноположительные и ложноотрицательные значения

Следующие два показателя, *точность* (precision) и *отклик* (recall), могут быть определены на основе четырех описанных выше показателей.

Точность – это доля истинно положительных результатов по отношению к общему количеству всех предсказанных выбросов. Отклик, с другой стороны, – это доля истинно положительных результатов по отношению к количеству точек, которые на самом деле являются выбросами. Эти определения можно записать в виде следующих уравнений:

$$\textit{precision} = \textit{tp} / (\textit{tp} + \textit{fp});$$
$$\textit{recall} = \textit{tp} / (\textit{tp} + \textit{fn}).$$

Основываясь на определениях, данных в предыдущих параграфах, можно подумать, что для вычисления количества TP, TN и т. д. необходимо, чтобы каждой из точек в нашем целевом индексе была присвоена метка класса.

Однако результатом задания обнаружения выбросов, как мы уже говорили, является не метка двоичного класса, которая обозначает каждую точку как выброс или норму, а числовая оценка выброса в диапазоне от 0 до 1.

Это представляет для нас проблему, когда дело доходит до вычисления желаемых показателей, потому что мы должны принять решение и указать граничную точку. Все точки, у которых оценка больше, чем у граничной точки, относятся к классу выбросов, а остальные точки, у которых оценка ниже, чем у граничной точки, относятся к классу нормы. Мы назовем это *порогом бинаризации*.

Определить точное значение этого порога не так-то просто, что возвращает нас к первоначальной цели данной главы – использованию API Evaluate для сравнения различных упомянутых ранее показателей точности при разных пороговых значениях, чтобы мы могли сделать осознанный выбор.

Далее мы рассмотрим практические примеры, иллюстрирующие применение этих знаний на практике.

1. Возьмем общедоступный набор данных, размещенный в репозитории UCI по адресу <https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29>. Чтобы лучше соответствовать цели этого упражнения, мы немного изменили набор данных, создав дополнительное поле с именем `Outlier`, в котором указано, является ли данная точка данных выбросом или нет. Измененный набор данных находится в файле с именем `breast-cancer-wisconsinoutlier.csv`, который можно скачать из репозитория книги по адресу <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter10>.

После загрузки набора данных вы можете использовать функцию импорта в визуализаторе данных Data Visualizer для импорта набора данных. Напомним, что об использовании Data Visualizer и импорте данных рассказано в разделе «Применение обнаружения выбросов на практике» этой главы.

Набор данных описывает признаки, характерные для тканей злокачественной и доброкачественной опухолей молочной железы, и включает поле `Class`, которое может принимать значение 2 (доброкачественная) или 4 (злокачественная). Для примера мы будем считать выбросами точки данных, помеченные как злокачественная опухоль (рис. 10.14). Здесь стоит упомянуть, что API Evaluate (<https://www.elastic.co/guide/en/elasticsearch/reference/current/evaluate-dfanalytics.html>), который мы

будем использовать, чтобы понять, насколько хорошо работает алгоритм обнаружения выбросов, выполняемый для эталонных меток истинности, требует, чтобы метка истинности была логическим значением 0 (для нормальной точки данных) и 1 для точки данных, которая является выбросом. Поэтому мы немного скорректировали исходный набор данных, добавив дополнительное поле под названием `Outlier`, в которое перенесли значение поля `Class` в формате, подходящем для использования в API Evaluate. Образец документа из набора данных показан на рис. 10.14.

```
# Bare_Nuclei      1
# Bland_Chromatin  3
# Class            2
# Clump_Thickness  5
# Marginal_Adhesion 1
# Mitoses          1
# Normal_Nucleoli  1
# Outlier          0
# Sample_code_number 1,000,025
# Single_Epithelial_Cell_Size 2
# Uniformity_of_Cell_Shape 1
# Uniformity_of_Cell_Size 1
t _id              o2GL1nUBtUA9eMQqL7vN
t _index           breast-cancer-wisconsin
# _score           0
t _type            doc
```

Рис. 10.14 ❖ Каждая точка в наборе данных помечена меткой класса. Мы преобразовали метку класса в логическую метку и сохранили ее в новом поле под названием `Outlier`

2. Воспользуемся мастером Data Frame Analytics, чтобы создать задание обнаружения выбросов для этого набора данных. Мы исключим поля, содержащие метку класса, метку эталона, а также номер выборки, как показано на рис. 10.15.
3. После завершения задания мы можем использовать целевое хранилище, в котором находятся результаты задания по обнаружению выбросов, чтобы вычислить, насколько хорошо сработал наш алгоритм обнаружения выбросов по сравнению с эталонными метками.

Included fields
9 fields included in the analysis



Search... Is included Is not included

<input type="checkbox"/>	Field name	Mapping	Is included	Is required	Reason
<input checked="" type="checkbox"/>	Bare_Nuclei	long	Yes	No	
<input checked="" type="checkbox"/>	Bland_Chromatin	long	Yes	No	
<input type="checkbox"/>	Class	long	No	No	field not in includes list
<input checked="" type="checkbox"/>	Clump_Thickness	long	Yes	No	
<input checked="" type="checkbox"/>	Marginal_Adhesion	long	Yes	No	
<input checked="" type="checkbox"/>	Mitoses	long	Yes	No	
<input checked="" type="checkbox"/>	Normal_Nucleoli	long	Yes	No	
<input type="checkbox"/>	Outlier	long	No	No	field not in includes list
<input type="checkbox"/>	Sample_code_number	long	No	No	field not in includes list
<input checked="" type="checkbox"/>	Single_Epithelial_Cell_Size	long	Yes	No	

Rows per page: 10 < 1 2 >

Рис. 10.15 ❖ Исключение полей Class, Outlier и Sample_code_number из задания обнаружения выбросов

4. Мы будем взаимодействовать с API Evaluate через консоль Dev Tools в Kibana. Давайте рассмотрим пример вызова API Evaluate:

```
POST _ml/data_frame/_evaluate
{
  "index": "breast-cancer-wisconsin-outlier",
  "evaluation": {
    "outlier_detection": {
      "actual_field": "Outlier",
      "predicted_probability_field": "ml.outlier_score",
      "metrics": {
        "confusion_matrix": {"at": [0.25, 0.5, 0.75]},
        "precision": {"at": [0.25, 0.5, 0.75]},
        "recall": {"at": [0.25, 0.5, 0.75]}
      }
    }
  }
}
```

Первая важная часть вызова – это целевое хранилище задания по обнаружению выбросов. Это хранилище содержит прогнозы нашего алгоритма, поэтому необходимо указать его имя. Вторая важная часть – actual_field. Это поле, которое содержит метку истинности наших данных. В нашем случае это поле под названием outlier. Наконец, мы

переходим к определению показателей, которые API должен возвращать для нас, а также пороговых значений, при которых они должны быть рассчитаны.

Параметры в вызове REST API позволяют нам указать широкий спектр возможных пороговых значений или пороговых значений, для которых необходимо вычислить метрики точности алгоритма. В предыдущем примере мы попросили API Evaluate вернуть значения показателей точности алгоритма при трех разных порогах бинаризации: 0,25, 0,5 и 0,75, – но в равной степени мы могли бы выбрать другой набор значений.

5. Рассмотрим результаты, возвращаемые API Evaluate:

```
{
  "outlier_detection" : {
    "confusion_matrix" : {
      "0.25" : {
        "tp" : 0,
        "fp" : 15,
        "tn" : 429,
        "fn" : 239
      },
      "0.5" : {
        "tp" : 0,
        "fp" : 5,
        "tn" : 439,
        "fn" : 239
      },
      "0.75" : {
        "tp" : 0,
        "fp" : 1,
        "tn" : 443,
        "fn" : 239
      }
    },
    "precision" : {
      "0.25" : 0.0,
      "0.5" : 0.0,
      "0.75" : 0.0
    },
    "recall" : {
      "0.25" : 0.0,
      "0.5" : 0.0,
      "0.75" : 0.0
    }
  }
}
```



Как видите, API Evaluate вернул ответ, в котором каждая метрика вычисляется три раза – один раз для каждого из пороговых значений, которые были указаны в вызове REST API.

Различные значения матрицы неточностей указывают на то, что алгоритм обнаружения выбросов работает довольно плохо, когда дело доходит до этого конкретного набора данных. Ни одно пороговое значение не дает истинно положительных результатов, а это означает, что мы не смогли обнаружить никаких выбросов с настройками по умолчанию. В следующем разделе вы увидите, как настройка гиперпараметров помогает достичь лучших результатов при обнаружении выбросов.

НАСТРОЙКА ГИПЕРПАРАМЕТРОВ ДЛЯ ОБНАРУЖЕНИЯ ВЫБРОСОВ

Для более продвинутых пользователей мастер Data Frame Analytics предлагает возможность конфигурировать и настраивать гиперпараметры – нечто наподобие ручек и циферблатов, которые точно настраивают работу алгоритма обнаружения выбросов. Доступные гиперпараметры показаны на рис. 10.16. Например, мы можем указать в задании обнаружения выбросов только определенный тип метода обнаружения выбросов вместо комбинации методов, использовать определенное значение для числа ближайших соседей, которые применяются в вычислениях, и предположить, что к выбросам относится определенная часть данных.

Обратите внимание, что хотя поэкспериментировать с этими настройками и почувствовать, как они влияют на конечные результаты, полезно, но если вы хотите настроить какие-либо из них для производственного сценария использования, вам следует внимательно изучить характеристики ваших данных и хорошо понимать, как эти характеристики будут взаимодействовать с выбранными вами настройками гиперпараметров. Дополнительная информация о каждом из этих гиперпараметров доступна в документации по адресу <https://www.elastic.co/guide/en/elasticsearch/reference/current/put-dfanalytics.html>.

В нашем случае мы знаем, что набор данных тканей содержит около 30 % образцов, относящихся к злокачественным опухолям. Следовательно, количество ожидаемых нами выбросов также близко к этому значению. Мы можем настроить это как параметр **Outlier fraction** (Доля выбросов) и перезапустить задание, как показано на рис. 10.16.

Давайте повторно создадим задание по обнаружению выбросов с этим новым гиперпараметром и сравним результат с показанным на рис. 10.16.

1. Выполните шаги, описанные во время создания нашего первого задания по обнаружению выбросов в разделе «Оценка выбросов с помощью API Evaluate» этой главы, но настройте параметр **Outlier fraction** в диалоговом окне **Hyperparameters** (Гиперпараметры), как показано на рис. 10.16. Создайте и запустите задание обнаружения выбросов.

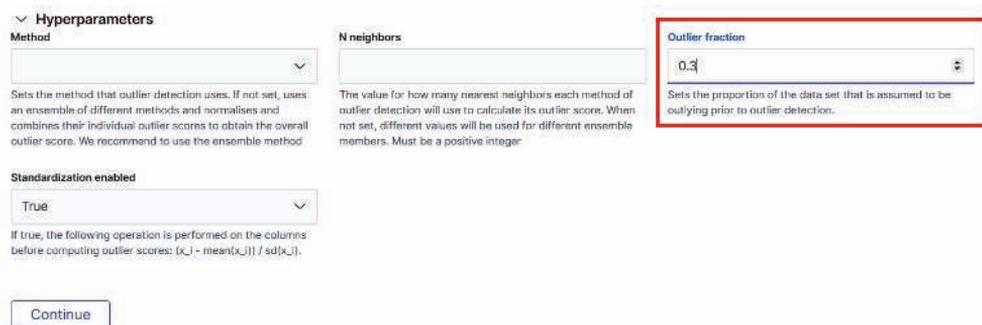


Рис. 10.16 ❖ Можно точно настроить поведение задания по обнаружению выбросов, настроив гиперпараметры с помощью мастера Data Frame Analytics

- После завершения задания мы можем повторно запустить команды API Evaluate для этого нового хранилища результатов. Мы использовали название `breast-cancer-wisconsin-outlier-fraction` в качестве имени целевого хранилища, которое содержит результаты задания с настроенными гиперпараметрами. Следовательно, наш новый вызов API Evaluate выглядит следующим образом:

```
POST _ml/data_frame/_evaluate
{
  "index": "breast-cancer-wisconsin-outlier-fraction",
  "evaluation": {
    "outlier_detection": {
      "actual_field": "Outlier",
      "predicted_probability_field": "ml.outlier_score",
      "metrics": {
        "confusion_matrix": {"at": [0.25, 0.5, 0.75]},
        "precision": {"at": [0.25, 0.5, 0.75]},
        "recall": {"at": [0.25, 0.5, 0.75]}
      }
    }
  }
}
```

- Давайте посмотрим, насколько изменилась наша матрица неточностей для трех разных пороговых значений. Ответ, который мы получаем от API Evaluate, выглядит следующим образом:

```
{
  "outlier_detection": {
    "confusion_matrix": {
      "0.25": {
        "tp": 239,
        "fp": 210,
        "tn": 234,
        "fn": 0
      },
```



```

    "0.5" : {
      "tp" : 86,
      "fp" : 48,
      "tn" : 396,
      "fn" : 153
    },
    "0.75" : {
      "tp" : 0,
      "fp" : 13,
      "tn" : 431,
      "fn" : 239
    }
  },
  "precision" : {
    "0.25" : 0.532293986636971,
    "0.5" : 0.6417910447761194,
    "0.75" : 0.0
  },
  "recall" : {
    "0.25" : 1.0,
    "0.5" : 0.3598326359832636,
    "0.75" : 0.0
  }
}
}
}

```



Как видно из значений матрицы неточностей, у нас немного лучше с точки зрения обнаружения истинных положительных результатов, истинных выбросов, но немного хуже с точки зрения обнаружения ложноотрицательных результатов.

Сравнение показателей оценки из текущего задания по обнаружению выбросов и задания по обнаружению выбросов, которое мы создали в разделе «Оценка качества обнаружения выбросов с помощью API Evaluate», свидетельствует о том, что выбор гиперпараметров может иметь существенное влияние на результат задания по обнаружению выбросов. Как же тогда правильно выбрать разумные значения гиперпараметров?

Есть много тонкостей и сложных тем, в которые можно погрузиться при выборе гиперпараметров, но лучший совет – помнить об итеративном характере процесса. Лучше начать с размеченного набора данных и оценить качество результатов, используя настройки по умолчанию (другими словами, не изменяя ничего в диалоговом окне **Hyperparameters** мастера Data Frame Analytics).

Значения по умолчанию обычно представляют собой разумный компромисс и проверяются на различных наборах данных. Если качество результатов неудовлетворительное, можно приступить к корректировке и точной настройке различных гиперпараметров. Хороший первый шаг – исправить известные проблемы и изучить, как это влияет на качество результатов. Например, нам было известно, что набор данных по раку груди содержит около 30 % выбросов, или 0,3 от общего объема данных, что позволило нам скорректировать этот параметр и добиться немного лучшего качества работы алгоритма.

ЗАКЛУЧЕНИЕ

Завершая главу, давайте вспомним второе назначение машинного обучения без учителя в Elastic Stack: обнаружение выбросов. Обнаружение выбросов можно использовать для обнаружения необычных точек данных в одномерных или многомерных наборах.

Алгоритм основан на комбинации из четырех отдельных мер: двух мер, основанных на расстоянии до k -ближайших соседей, и двух мер, основанных на плотности. Комбинация этих мер позволяет определить, насколько далеко данная точка данных находится от своих соседей, а также от общей массы данных в наборе данных. Необычность точки выражается в виде числовой оценки выброса, которая варьируется в интервале от 0 до 1. Чем ближе оценка данной точки к 1, тем более необычной она является в наборе данных.

В дополнение к оценке выброса для каждого объекта или точки мы вычисляем величину, известную как влияние характеристики. Чем выше влияние характеристики из данного поля, тем больше это поле отвечает за то, что данная точка является необычной. Эти оценки влияния характеристик можно использовать, чтобы понять, почему конкретная точка получила определенную оценку выбросов.

В отличие от другой функции обучения без учителя в Elastic Stack – обнаружения аномалий, обнаружение выбросов не требует, чтобы данные имели временную составляющую или были временными рядами любого типа. Кроме того, алгоритм обнаружения выбросов, в отличие от обнаружения аномалий, не формирует вероятностную модель, чтобы понять, какие точки данных имеют низкую вероятность возникновения. Вместо этого он использует измерения на основе расстояния и плотности для вычисления необычности. Из-за этой разницы в методологиях обнаружение выбросов не может работать в режиме, близком к режиму реального времени, обновляя свои оценки выбросов по мере добавления новых данных в исходное хранилище. Вместо этого, если мы хотим оценить необычность новых точек по мере их добавления в хранилище, мы должны повторно запустить расчет обнаружения выбросов для всего исходного хранилища в пакетном режиме.

В следующей главе мы оставим методы обучения без учителя позади и погрузимся в захватывающий мир обучения с учителем, начиная с задачи классификации.



Классификационный анализ

Алгоритмы машинного обучения принято разделять на три основных класса: обучение с учителем, обучение без учителя и обучение с подкреплением. Третья разновидность машинного обучения выходит как за рамки данной книги, так и возможностей Elastic Stack. Обучение без учителя является темой глав, посвященных обнаружению аномалий, а также главы 10 об обнаружении выбросов. В этой главе мы наконец переходим к обучению с учителем. Elastic Stack предлагает две разновидности обучения с учителем: классификацию и регрессию. Эта глава будет посвящена классификации, а следующая – регрессии.

Цель обучения с учителем – взять размеченный набор данных и извлечь из него закономерности, закодировать знания, полученные из набора данных, в структуре, которую мы называем *моделью*, а затем использовать эту обученную модель для прогнозирования, исходя из ранее не встречавшихся выборок данных. Это характерно как для классификации, так и для регрессии. Классификация используется для прогнозирования дискретных меток или классов, а регрессия – для непрерывных значений.

Задачи классификации встречаются нам повсюду. Патолог-гистолог, изучающий образцы тканей пациентов, должен классифицировать каждый из них как злокачественный или доброкачественный; рабочий конвейера, изучающий детали машин, должен классифицировать каждую как бракованную или качественную; аналитик, изучающий данные об оттоке клиентов, пытается предсказать, будет ли клиент продлевать или отменять подписку на услугу, и т. д.

Знакомство с классификацией и принципом ее работы в Elastic Stack также приведет нас к тесному соприкосновению с рядом других тем, понимание которых актуально для любого практикующего специалиста, желающего в полной мере использовать классификацию для решения своих задач. Эти темы включают конструирование признаков, разделение набора данных на обучающий и тестовый наборы, понимание того, как измерить точность классификатора и почему один и тот же показатель, применяемый к обучающему и тестовому набору, означает разные вещи, как использовать важность признака, чтобы понять, в какой мере каждый признак способствовал присвоению метки класса, а также многие другие вопросы, обсуждение которых составит основную часть этой главы.

В данной главе мы рассмотрим следующие темы:

- классификация: от данных к обученной модели;
- учимся выполнять классификацию;
- деревья решений с градиентным усилением;
- гиперпараметры;
- интерпретация результатов.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Материал в этой главе требует наличия Elasticsearch версии 7.9 или новее. Примеры были протестированы с использованием Elasticsearch версии 7.10.1, но должны работать с любой версией Elasticsearch более поздней, чем 7.9. Обратите внимание, что для запуска примеров в этой главе требуется лицензия Platinum. Если для конкретного примера или раздела нужна более поздняя версия Elasticsearch, это будет упомянуто в тексте.

КЛАССИФИКАЦИЯ: ОТ ДАННЫХ К ОБУЧЕННОЙ МОДЕЛИ

Обучение классифицирующей модели на исходном наборе данных – это многоэтапный процесс. Мы начнем главу с обзорного знакомства с процессом обучения (рис. 11.1), который начинается с размеченного набора обучающих данных (рис. 11.1, А).

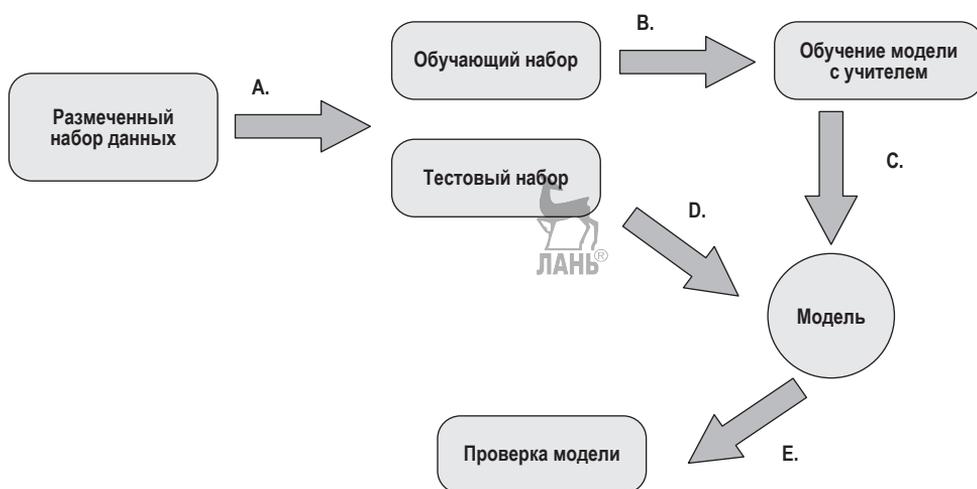


Рис. 11.1 ❖ Обобщенная схема процесса обучения с учителем, который берет помеченный набор данных и выводит обученную модель

Из имеющегося набора данных обычно выделяют обучающую часть, которая будет загружена в алгоритм обучения (рис. 11.1, В). Результатом работы алгоритма обучения является обученная модель (рис. 11.1, С). Затем обученную модель применяют к классификации тестовых данных (рис. 11.1, D), которые также заранее выделили из всего набора данных. Точность работы модели с тестовыми данными представляют в виде оценочных показателей, которые можно использовать, чтобы определить, достаточно ли хорошо модель обобщается на ранее не встречавшиеся выборки данных.

Каждый из этих этапов будет дополнительно разъяснен в практических пошаговых инструкциях, представленных в этой главе. Чтобы связать практические примеры с более абстрактными теоретическими аспектами машинного обучения, мы также представим концептуальное понимание того, что такое обучение с учителем. Это приведет нас к обсуждению этапа извлечения признаков – что такое признаки, и как они влияют на точность классификатора? Наконец, мы рассмотрим, как оценить точность классификатора и как разделение данных на набор для обучения и тестирования помогает нам измерить два разных типа точности.

Классифицирующие модели учатся на данных

Один из ключевых принципов машинного обучения заключается в том, что классификатор машинного обучения – это программный продукт, который учится на данных. Иными словами, по мере обучения модель все лучше и лучше выполняет классификацию (в случае классифицирующих моделей), поскольку она видит или обрабатывает все больше и больше данных. Но в конце концов, модель машинного обучения – это не живое существо, а программа, поэтому возникает вопрос: что именно мы имеем в виду, когда говорим, что модель учится на данных?

Ответ на этот вопрос приведет нас к важному понятию *решающей границы* (decision boundary). Давайте рассмотрим эту концепцию с помощью уже знакомого вымышленного двумерного набора данных, в котором записаны вес и обхват тыкв. Предположим, что набор данных содержит измерения различных классов тыкв, и наша цель – научить модель относить произвольные тыквы к тому или иному классу на основе их веса и обхвата.

Если мы построим наш набор данных тыквы в двух измерениях, он может выглядеть примерно так, как показано на рис. 11.2.

Измерения веса представлены на горизонтальной оси, а измерения обхвата – на вертикальной. Если точка данных представлена кругом, она соответствует тыкве первого класса. Если квадратом, то это тыква второго класса. Внимательно рассмотрите распределение квадратов и кругов на рис. 11.2 и представьте, что вам нужно создать простое правило, позволяющее различать (классифицировать) тыквы первого и второго классов, и записать его таким образом, чтобы другой человек тоже смог им воспользоваться и сразу определить, к какому из двух классов принадлежит новая тыква, которую он только что измерил.

Очень простой способ решить эту задачу – изучить двумерное изображение и начертить линию, которая приблизительно отделяет один класс от другого, например как показано на рис. 11.3.

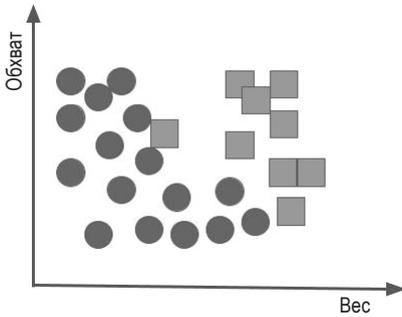


Рис. 11.2 ❖ Вымышленный двумерный набор данных, изображающий измерения обхвата и веса двух разных классов тыкв, представленных кругами и квадратами

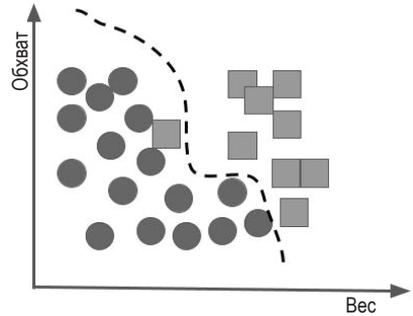


Рис. 11.3 ❖ Решающая граница отделяет друг от друга два класса тыкв, представленных в виде точек данных в двумерном пространстве

Несколько точек данных окажутся не по ту сторону линии, но это просто неизбежно при разработке классификаторов – очень немногие классификации в конечном итоге оказываются идеальными! Вы только что изобразили решающую границу – линию или гиперпространственную плоскость (если мы работаем с набором данных с несколькими переменными), которая отделяет членов одного класса от другого. Теперь любой, кто хочет классифицировать только что измеренные тыквы, может взять вашу схему и нанести на нее измерения своей собственной тыквы, чтобы увидеть, на какой стороне решающей границы окажется их точка данных, как показано на рис. 11.4.

Это очень обобщенное представление процесса, выполняемого алгоритмами классификации. Они берут набор обучающих данных (наши исходные размеры тыквы и классы, к которым принадлежит каждая измеренная тыква), а затем применяют различные тесты и преобразования, чтобы найти решающую границу. Этот процесс настройки модели на соответствие специально подготовленным данным называется обучением, и он применим не только к алгоритмам классификации, но и к алгоритмам регрессии. Решающая граница кодируется в обученной модели и затем может использоваться для прогнозирования класса будущих, ранее неизвестных точек данных.

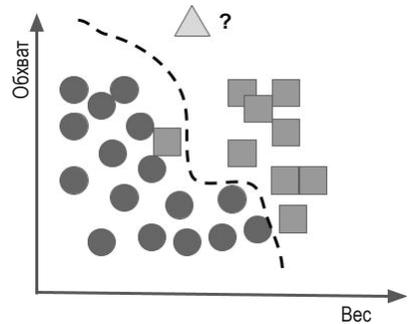


Рис. 11.4 ❖ Новая точка данных, представленная треугольником, находится на правой стороне решающей границы и, скорее всего, относится к «квадратному» классу

Глядя на приведенный выше вымышленный пример, следует отметить, что нам сильно повезло. Двумерный график распределения наших тыкв по весу и объёму дал картину, на которой мы смогли легко провести решающую границу, которая, хотя и несовершенна, тем не менее даёт хорошие результаты для большинства точек данных. Однако в реальном мире очень немногие наборы данных поддаются столь простой и точной классификации. Как вы увидите позже, атрибуты, которые мы выбираем для представления наших точек данных, могут иметь большое влияние на то, насколько разделим набор данных, и, следовательно, на то, как хорошо с этими данными будет работать классифицирующая модель.

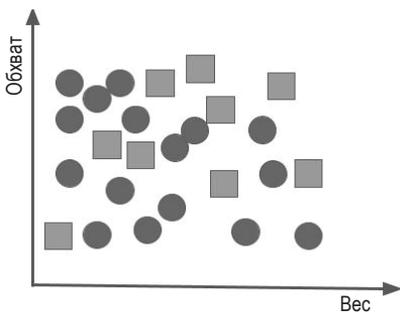


Рис. 11.5 ❖ Эти признаки не обеспечивают значимого разделения двух классов

Предположим, что вместо характеристик веса и объёма для наших тыкв мы выбрали процентное содержание химического соединения X, которое было обнаружено в тыкве, и количество воды, которое использовалось для ее полива во время роста. В этом случае двумерный график распределения точек данных (рис. 11.5) с использованием этих признаков вообще не позволяет провести какую-либо решающую границу!

Как мы увидим в следующем разделе, *конструирование признаков* – это большая тема сама по себе и важный этап предварительной обработки, который требует внимательного отношения при запуске любого проекта машинного обучения, будь то в Elastic Stack или за его пределами.

Конструирование признаков

В предыдущем разделе на примере вымышленного набора данных измерений тыкв мы проиллюстрировали концепцию решающих границ, которые являются продуктом обучения классификатора на специально подготовленных обучающих данных. Давайте рассмотрим следующую тему, процесс выбора и управления признаками таким образом, чтобы они подходили для классификации, на основе более реалистичного примера – классификации вредоносных программ. В связи с тем, что миллионы новых вариантов вредоносных программ выпускаются в мир почти каждый день, становится все труднее с высокой достоверностью отличать безопасные двоичные файлы от вредоносных, используя традиционные подходы, основанные на правилах. Поскольку мы имеем дело с большим объемом и разнообразием входных данных, которые невозможно зафиксировать с помощью негибких правил, идеальным решением является машинное обучение. Как именно нам нужно предварительно обработать обучающие данные – в данном случае вредоносные и безопасные двоичные файлы, – чтобы алгоритм машинного обучения смог их понять? Этот вопрос и ответ на него подводят нас к целой области исследований в области машинного обучения, известной как конструирование признаков.

Процесс конструирования признаков включает в себя использование знаний о проблеме, имеющихся у экспертов в предметной области, и их применение к обучающим данным. Например, аналитик вредоносных программ может сказать нам, что строки в обычных двоичных файлах часто длиннее, чем строки во вредоносных двоичных файлах, и это может быть полезным признаком для классификации. Следовательно, в рамках процесса конструирования признаков мы должны разработать способ вычисления средней длины строки в каждом из двоичных файлов в наших обучающих данных и использовать этот показатель в качестве признака.

Было бы полезно выделить время на выяснение того, что есть у экспертов в проблемной области знаний и какие признаки были использованы для достижения современных результатов. Это важно, потому что, как вы убедились на вымышленном примере тыквы, от выбора признаков зависит, сможет ли наша модель выделить и формализовать различия между классами. Например, предположим, что мы обучили нашу модель классификации вредоносных программ использовать размер двоичного файла в байтах и наличие буквы «а» в имени двоичного файла в качестве признаков. Если окажется, что и вредоносные, и безопасные двоичные файлы имеют одинаковый размер в байтах и оба содержат букву «а» в своих именах, наш классификатор будет бесполезен, потому что он будет рассматривать признаки, которые не имеют значения, когда дело доходит до отличия вредоносных двоичных файлов от безопасных.

Конструирование признаков зачастую является итеративным процессом. Следует начать с предположения о том, какие признаки могут дать хорошие результаты, обучить модель и оценить ее на тестовом наборе, а затем постепенно добавлять, убирать или изменять признаки до тех пор, пока не будет достигнуто желаемое качество результатов. Позже в этой главе мы рассмотрим, как именно измеряют качество модели и как это реализовать с помощью машинного обучения в Elastic Stack.

Оценка модели

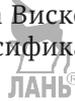
Последняя тема, которая завершает этот вводный раздел о классификации, – это оценка модели. Как узнать, насколько хорошо работает наша модель? Существуют различные методы количественной оценки качества модели, и мы подробно рассмотрим эти методы и их значение в следующем разделе этой главы. Важное понятие, связанное с измерением качества модели, – это то, на каких данных оно измеряется.

Чтобы измерить, насколько хорошо работает модель, нам нужно подать ей на вход размеченный набор данных, а затем сравнить метки классов, которые модель предсказала, с эталонными метками и подсчитать, сколько ошибок модель сделала. Один из доступных наборов данных – это обучающие данные, но если мы воспользуемся этими данными для оценки качества модели, то фактически мы будем показывать модели те же данные, на которых она обучалась. Хотя это даст нам хорошую оценку *ошибки обучения*, это ничего не скажет нам о том, насколько хорошо модель будет *обобщать*, т. е. делать прогнозы на основе незнакомых данных.

Чтобы оценить качество модели на данных, которые ей незнакомы, мы должны заранее отделить часть обучающих данных. Эту часть данных, также известную как *набор тестовых данных*, ни в коем случае нельзя использовать для обучения модели. Вместо этого после завершения процесса обучения мы воспользуемся моделью, чтобы сделать прогнозы для набора тестовых данных и посмотреть, сколько точек тестовых данных классифицировано неправильно. Эта мера даст нам представление о том, насколько хорошо модель будет обобщать свои прогнозы на незнакомые данные. Количество ошибок, совершаемых моделью при прогнозировании тестового набора данных, называется *ошибкой обобщения*.

Для измерения этих ошибок будут использоваться несколько показателей, таких как достоверность, точность и отклик (некоторые из них вы встречали в главе 10). Чтобы освежить новые знания, вы можете перечитать раздел «Оценка качества обнаружения выбросов с помощью API Evaluate» в главе 10.

Мы обсудим каждую из рассмотренных выше идей более подробно в следующих главах, а пока давайте посмотрим, как эти концепции реализуются на практике. Будем использовать общедоступный набор данных по раку молочной железы у пациентов из штата Висконсин, чтобы создать задание анализа фрейма данных для целей классификации образцов ткани.



ПРОСТОЙ ПРИМЕР КЛАССИФИКАЦИИ

Далее мы рассмотрим пошаговый процесс настройки задания по классификации, использующего общедоступный набор данных по раку молочной железы у пациентов из штата Висконсин. Исходный набор данных доступен по адресу [https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+\(original\)](https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)). В этом упражнении мы будем использовать слегка очищенную версию набора данных, которая не нуждается в предварительной очистке данных (важный шаг в жизненном цикле проекта машинного обучения, но его обсуждение выходит за рамки данной книги) и позволит сосредоточиться на основных шагах настройки задания классификации.

1. Загрузите очищенный файл с набором данных `breast-cancer-wisconsin-outlier.csv` из репозитория <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter11> и сохраните его локально на вашем компьютере. В своем экземпляре Kibana перейдите к приложению Machine Learning из меню слева и щелкните вкладку **Data Visualizer** (Визуализатор данных). Вы перейдете к загрузчику файлов **File Uploader**. Нажмите **Upload file** (Загрузить файл) и выберите загруженный CSV-файл.

После успешной загрузки (проверьте свой шаблон хранилища в разделе **Discover** и кратко просмотрите несколько документов, чтобы убедиться, что с набором данных все в порядке) вернитесь в приложение Machine Learning и вместо вкладки **Data Visualizer** выберите **Data Frame Analytics** (Аналитика фрейма данных). Должно появиться представление, изображенное на рис. 11.6.



Рис. 11.6 ❖ Обзор заданий аналитики фрейма данных в настоящее время пуст

- Нажмите синюю кнопку **Create job**. Вы перейдете к мастеру аналитики фрейма данных **Data frame analytics**, который немного похож на мастер преобразований **Transforms**, знакомый вам по главе 9, и позволит вам легко создать задание классификации, регрессии или обнаружения выбросов. В данном случае выберите задание классификации в раскрывающемся списке на рис. 11.7.

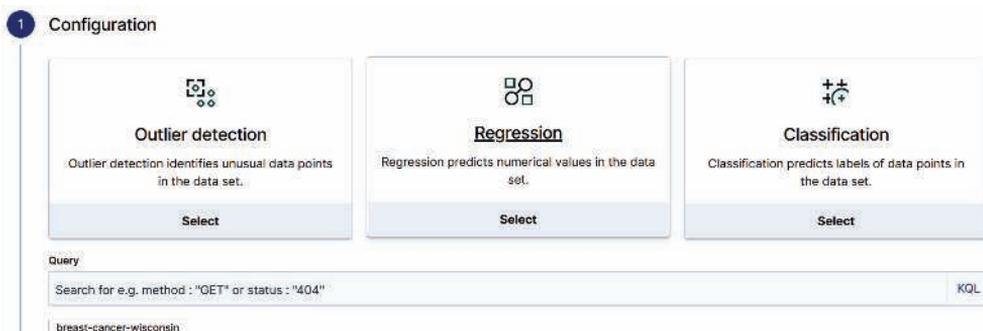


Рис. 11.7 ❖ Мастер **Data frame analytics** помогает создать три различных типа заданий. Для нашего текущего случая выберите **Classification**

- Далее перейдем к выбору зависимой переменной. Как вы уже знаете, цель классификации – научиться предсказывать, к какому классу принадлежит определенная ранее не встречавшаяся точка данных. Переменная, обозначающая этот класс, называется *зависимой переменной* – ее значение зависит от обобщений, которые наш классификатор сделает из других точек данных. Для этого классификатор должен сначала изучить обучающие данные, и поэтому мы должны сообщить мастеру, какая переменная содержит метку класса. В этом наборе данных метка класса хранится в переменной `Class`, поэтому мы выбираем ее из раскрывающегося списка. После того как мы сделаем выбор, мастер отобразит список включенных и исключенных полей. На этом шаге будьте внимательны. Если

в вашем наборе данных есть поле, которое может действовать как псевдоним для зависимой переменной, то вы, скорее всего, получите классификатор, который по умолчанию просто проверяет значение поля-псевдонима вместо предсказания на основе набора данных.

Поясним, что это означает на практике. Предположим, что в нашем случае все обучающие образцы рака были организованы таким образом, что сначала идут злокачественные ткани, а потом – доброкачественные. Это означает, что номер образца фактически будет псевдонимом класса. Если номер образца меньше, скажем, 30, то классификатор научится предсказывать злокачественное новообразование. Если больше, то доброкачественное. Поэтому следует проявлять осторожность и заранее изучать свои данные, а затем удалять все переменные, которые, как ожидается, не будут нести значимую информацию для предсказания зависимой переменной.

Даже если переменная, обозначающая номер выборки, не имеет характер псевдонима, мы все равно не ожидаем, что она несет много значимой информации, которая могла бы помочь в выводе зависимой переменной. Фактически она только увеличивает объем памяти, занимаемой заданием, поэтому лучше исключить ее с самого начала.

Как показано на рис. 11.8, мы исключили переменную `Sample_code_number` из задания, сняв флажок.

<input type="checkbox"/>	Field name	Mapping	Is included	Is required	Reason
<input checked="" type="checkbox"/>	Clump_Thickness	long	Yes	No	
<input checked="" type="checkbox"/>	Uniformity_of_Cell_Size	long	Yes	No	
<input checked="" type="checkbox"/>	Bare_Nuclei	long	Yes	No	
<input checked="" type="checkbox"/>	Bland_Chromatin	long	Yes	No	
<input checked="" type="checkbox"/>	Class	long	Yes	Yes	
<input checked="" type="checkbox"/>	Marginal_Adhesion	long	Yes	No	
<input checked="" type="checkbox"/>	Mitoses	long	Yes	No	
<input checked="" type="checkbox"/>	Normal_Nucleoli	long	Yes	No	
<input checked="" type="checkbox"/>	Outlier	long	Yes	No	
<input type="checkbox"/>	Sample_code_number	long	Yes	No	

Rows per page: 10 ▾

< 1 2 >

Рис. 11.8 ❖ Исключение `Sample_code_number` из задания классификации, чтобы избежать эффекта переменной-псевдонима

- Теперь мы готовы перейти к следующему шагу настройки, который включает в себя выбор параметра **Training percent** (Доля обучающих данных). Напомним, что в начале этого раздела мы говорили о том, как производительность классификатора может быть оценена на двух различных наборах данных, полученных из исходного набора: на обучающем и тестовом наборах данных. Выполнять такое разделение вручную в Elasticsearch было бы довольно трудно, особенно если вы имеете дело с большим набором данных. Чтобы упростить эту задачу, мастер задания Data Frame Analytics включает параметр конфигурации, который позволяет нам установить, какую часть набора данных мы хотим выделить для обучения модели, а какую часть оставить для тестирования. Это соотношение устанавливается в мастере с помощью ползунка, показанного на рис. 11.9.

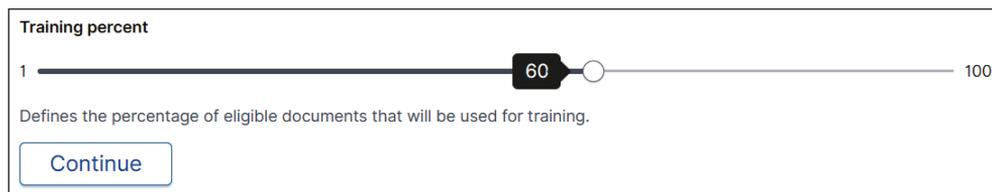


Рис. 11.9 ❖ Ползунок выбора доли обучающих данных

Как выбрать нужную долю обучающих данных? Как и на многих других этапах создания программы машинного обучения, ответ на этот вопрос во многом будет зависеть от размера вашего набора данных и от того, о чем вы заботитесь больше – о получении точной оценки качества модели на наборе данных для обучения или на наборе данных для тестирования. Позже мы обсудим разницу между этими двумя показателями более подробно, но пока достаточно сказать, что оценка качества модели на тестовых данных говорит нам о том, насколько хорошо модель будет работать, когда вы начнете применять ее к неизвестным выборкам, поэтому в большинстве случаев определенно стоит выделить часть данных, чтобы получить оценку того, насколько хорошо ваша модель будет работать после ее развертывания в производственной среде.

Другой аспект – это размер набора данных. Как правило, если ваш набор данных содержит более 100 000 документов, начните с меньшей доли обучающих данных – около 10 % или 15 %, а затем, при необходимости, постепенно увеличивайте долю и повторяйте обучение.

Поскольку в нашем случае весь набор данных содержит чуть менее 700 документов, мы установим долю обучающих данных равной 60 %.

- После выбора доли обучающих данных мастер предлагает перейти к дополнительным параметрам **Additional options**, как показано на рис. 11.10. Мы пока оставим эти настройки по умолчанию, но вернемся к их обсуждению в следующих, более сложных примерах.

2 Additional options

Advanced configuration

Feature importance values

0

Specify the maximum number of feature importance values per document to return.

Model memory limit

100mb

The approximate maximum amount of memory resources that are permitted for analytical processing.

> Hyperparameters

Continue

Prediction field name

Defines the name of the prediction field in the results. Defaults to <dependent_variable>_prediction.

Maximum number of threads

1

The maximum number of threads to be used by the analysis. The default value is 1.

Top classes

2

The number of categories for which the predicted probabilities are reported.

Рис. 11.10 ❖ Дополнительные параметры для мастера заданий классификации Data Frame Analytics

6. Наконец, после нажатия кнопки **Continue** (Продолжить) мы установим идентификатор задания, а все остальное оставим по умолчанию. Установите флажок **Start immediately** (Начать немедленно), а затем создайте и запустите задание, нажав кнопку **Create**. Вернитесь на главную страницу аналитики фрейма данных. Вы должны увидеть только что созданное задание на панели управления заданиями, как показано на рис. 11.11.

Data frame analytics EXPERIMENTAL Refresh 30 seconds

Jobs Models

Total analytics jobs: 1 Running: 0 Stopped: 1 Create job

Search...

ID	Description	Memory status	Source index	Destination index	Type	Status	Progress	Actions
breast-cancer-wisconsin-classification		ok	breast-cancer-wis...	breast-cancer-wis...	classification	stopped	Phase 8/8	View

Rows per page: 10

Рис. 11.11 ❖ На панели обзора заданий аналитики фреймов данных отображается сводка текущих заданий аналитики фреймов данных

Когда статус задания отображается как **stopped** (остановлено), а индикатор выполнения показывает, что все этапы задания завершены, используйте меню **Actions** (Действия) для перехода к результатам. Для этого нажмите в меню ссылку **View** (Просмотр).

7. На странице результатов отображается большой объем информации, которая жизненно важна для понимания того, как работает классификатор, только что обученный на нашем наборе данных. Пример страницы результатов показан на рис. 11.12. Обратите внимание на кнопки **Testing** (Тестирование) и **Training** (Обучение) в правом верхнем углу. Эти переключатели определяют,

по каким частям набора данных рассчитываются оставшиеся метрики, визуализации и таблицы на странице. При первой загрузке представления результатов эти метрики рассчитываются для всего набора данных, который включает как набор данных для обучения, так и набор данных для тестирования.

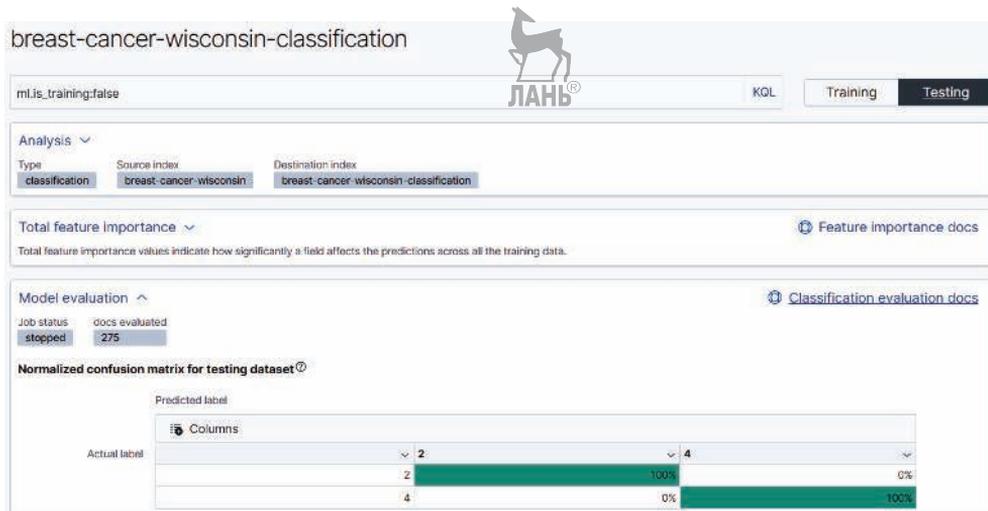


Рис. 11.12 ❖ На странице результатов задания классификации отображается матрица неточностей

Переключите режим отображения, чтобы увидеть результаты данных тестирования и посмотреть на матрицу неточностей. Помните, что целью этого упражнения было научиться предсказывать, является ли данный образец ткани злокачественным или доброкачественным. Класс 4 обозначает злокачественные ткани, а класс 2 – доброкачественные. Поскольку это набор данных, содержащий эталонные метки для каждой выборки, то в хранилище результата каждая выборка будет иметь две метки: фактическую метку класса и метку прогнозируемого класса.

Матрица неточностей суммирует количество экземпляров, в которых фактическая метка класса (отображается в левой части матрицы, изображенной на рис. 11.12) совпадает с предсказанной меткой класса (отображается горизонтально в матрице выше), и количество экземпляров, в которых произошло несоответствие меток. В главе 10 мы подробно исследовали матрицу неточностей и вывели словарь терминов для представления величин в матрице. Экземпляры, в которых фактическая метка класса совпадает с предсказанной, называются истинно положительными и истинно отрицательными, а случаи несовпадения – ложноположительными и ложноотрицательными.

Если вы посмотрите на матрицу неточностей на рис. 11.12, то увидите что-то странное. В ней нет ни ложноположительных, ни ложноотрицательных результатов. Судя по всему, наш классификатор отлично справился с тестовым набором данных!

Воздержитесь от ликования и взгляните на полученный результат с большим скептицизмом. Некоторые результаты слишком хороши, чтобы быть правдой, и, как вы вскоре убедитесь, в данном случае так и есть. Если мы достигаем идеальных результатов на тестовом наборе (помните, что это точки данных, которые модель не видела во время обучения, и поэтому у нее не было возможности научиться точно классифицировать их заранее), обычно виновником является признак в наборе данных, который действует как псевдоним для зависимой переменной, поэтому классификатор становится «ленивым» и просто по умолчанию проверяет значение признака-псевдонима при назначении класса. Давайте внимательно посмотрим на детали задания, чтобы разобраться, какие значения были включены в анализ, – возможно, это даст нам ключ к пониманию того, что происходит. Вернитесь на страницу управления заданием Data Frame Analytics и нажмите на значок направленной вниз стрелки рядом с названием задания слева. На появившейся панели перейдите на вкладку **JSON**. Она будет похожа на рис. 11.13.

```

12+ {
13   "dest": {
14     "index": "breast-cancer-wisconsin-classification",
15     "results_field": "ml"
16   },
17   "analysis": {
18     "classification": {
19       "dependent_variable": "Class",
20       "num_top_features_importance_values": 0,
21       "class_assignment_objective": "maximize_minimum_recall",
22       "num_top_classes": 2,
23       "prediction_field_name": "Class_prediction",
24       "training_percent": 98,
25       "analytics_seed": 885172881287188988
26     }
27   },
28   "analyzed_fields": {
29     "includes": [
30       "Raw_Nucleoli",
31       "Class",
32       "Class_Thickness",
33       "Marginal_Adhesion",
34       "Mitoses",
35       "Normal_Nucleoli",
36       "Outlier",
37       "Single_Epithelial_Cell_Size",
38       "Uniformity_of_Cell_Shape",
39       "Uniformity_of_Cell_Size"

```

Рис. 11.13 ❖ Конфигурация задания классификации в формате JSON

Если вы внимательно посмотрите на раздел `analysis_fields`, то увидите, что есть поле под названием `outlier`. Это дублированное поле, которое мы создали из поля `Class` в главе 10. Именно это поле-псевдоним заставило наши результаты выглядеть лучше, чем они есть на самом деле.

8. Давайте теперь создадим задание, но исключим как поле `outlier`, так и поле `Sample_code_number`. После того как эта работа будет завершена, новые результаты для тестового набора данных будут выглядеть, как показано на рис. 11.14.

Как видно из результатов на рис. 11.14, исключение переменной `outlier` вместе с переменной `Sample_code_number` привело к 98 % истинно положительных результатов, а также к некоторым ложноположительным и ложноотрицательным срабатываниям, что является более реалистичным результатом, чем идеальная классификация, которая у нас была раньше.



Рис. 11.14 ❖ Матрица неточностей отображает результаты задания классификации после исключения переменной outlier

Прежде чем мы углубимся в примеры классификации, было бы хорошо понять, что именно происходит в стеке Elastic ML при обучении нашей классифицирующей модели. Поэтому следующий раздел будет посвящен рассказу о том, как работают деревья решений.

ДЕРЕВЬЯ РЕШЕНИЙ С ГРАДИЕНТНЫМ УСИЛЕНИЕМ

Конечная цель задачи классификации – взять незнакомые точки данных и попытаться сделать вывод, к какому из нескольких возможных классов они принадлежат. Мы берем размеченный обучающий набор, который содержит репрезентативное количество точек данных, извлекаем из него соответствующие признаки, которые позволяют нам найти решающую границу, а потом кодируем знания об этой границе в классифицирующей модели. Затем эта модель принимает решения о том, к какому классу принадлежит определенная точка данных. Но как модель извлекает и запоминает знания? Сейчас мы постараемся ответить на этот вопрос.

В соответствии с традицией, которой мы придерживаемся на протяжении всей книги, давайте начнем с концептуального рассмотрения того, какие подходы люди используют для принятия сложных решений. Знакомый инструмент, который многие из нас использовали раньше для принятия решений, когда задействовано несколько потенциально сложных факторов, – это блок-схема. На рис. 11.15 показан пример блок-схемы, которую можно составить, чтобы решить, какую верхнюю одежду надеть с учетом сегодняшней погоды.

На каждом этапе блок-схема задает вопрос (например, насколько тепло или холодно, идет дождь на улице или нет) и на основе нашего ответа перенаправляет нас к другой части блок-схемы и новому набору вопросов. В конечном итоге, отвечая на вопросы в блок-схеме и следуя по ее ветвям, мы приходим к решению или метке класса.

Модель, созданная как часть процесса обучения в Elastic Stack, фактически делает нечто очень похожее. В области машинного обучения этот алгоритм известен как *дерево решений*. Хотя версия, используемая в стеке Elastic ML, намного сложнее, чем описанная здесь, основные концепции те же.

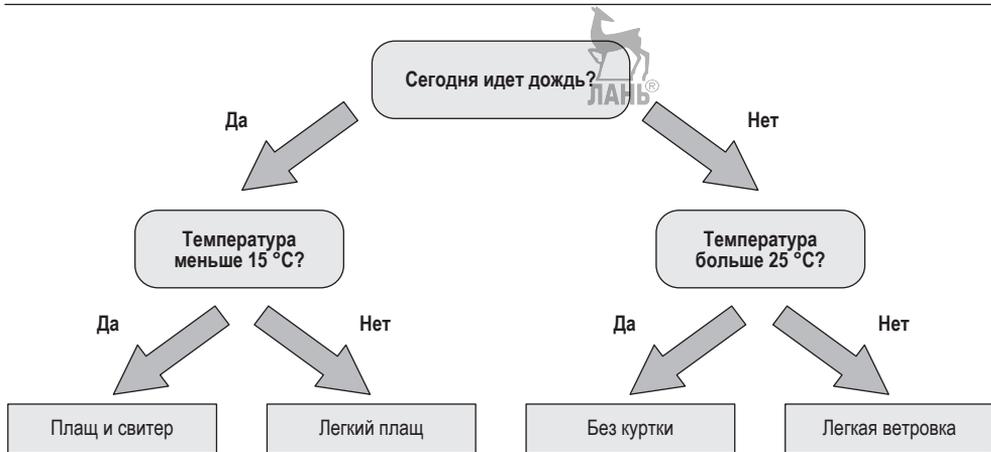


Рис. 11.15 ❖ Пример блок-схемы принятия решений

Давайте подробнее рассмотрим деревья решений.

Введение в деревья решений

Как строится наше дерево решений? Мы начинаем с разделения набора данных на две группы, используя значение определенного признака и определенный порог. Мы определяем эту функцию и этот порог, просматривая все доступные признаки (или поля, если использовать терминологию Elasticsearch), а затем находим одну пару пороговых значений признака, которая дает *наиболее чистое разделение* (purest split). Что мы понимаем под наиболее чистым разделением? Чтобы понять идею чистоты узлов и то, как она влияет на построение дерева решений и последующее использование дерева решений для классификации незнакомых точек данных, мы должны сделать шаг назад и изучить общую картину.

Как вы помните, наша цель состоит в том, чтобы, по сути, спроектировать блок-схему принятия решений, по которой мы можем пройти, когда нам нужно классифицировать новую точку данных. Мы определяем путь классификации на блок-схеме, глядя на последний узел, к которому ведет наш обход. В деревьях решений этот последний узел называется *листовым*, или *конечным*, *узлом* и состоит из всех точек обучающих данных, которые оказались там в результате последовательных разделений, выполненных с использованием определенного признака и определенного порога. Поскольку процедура классификации обычно несовершенна, листовые узлы не ограничатся точками, принадлежащими только одному классу, а вместо этого будут смешанными. Степень «засоренности» узла чужими точками данных может быть определена количественно по различным параметрам и называется *чистотой* узла или листа. Лист или узел, который содержит только точки данных одного класса, является *наиболее чистым*.

Наличие чистых узлов в дереве решений – это очень хорошо, потому что это означает, что как только мы достигли конечного узла в нашей блок-схеме,

мы можем быть достаточно уверены, что точка данных, которую мы сейчас пытаемся классифицировать, принадлежит к тому же классу, что и точки данных, уже помещенные в этот конечный узел. На практике стремление к абсолютной чистоте узлов может привести к слишком разветвленному дереву решений (в конце концов, мы могли бы достичь наивысшей чистоты, просто создав столько листовых узлов, сколько есть точек данных, – таким образом, каждый лист содержал бы ровно одну точку данных ровно одного класса и, следовательно, имел бы идеальную чистоту), поэтому листовые узлы всегда будут смешанными.

Вычисление доли точек данных в листовом узле, принадлежащем данному классу, дает нам оценку вероятности того, что точка данных, которую мы пытаемся классифицировать, принадлежит этому классу. Например, предположим, что у нас есть дерево решений, разделяющее точки данных на класс А и класс В. Один из конечных узлов имеет 80 выборок из класса А и 20 выборок из класса В. Если во время классификации новая точка данных попадает в этот узел, тогда вероятность того, что она принадлежит к классу А, составляет 0,8.

Это, конечно, очень упрощенное представление о том, как на самом деле работает алгоритм дерева решений, но, опираясь на него, мы можем двигаться дальше.



Градиентное усиление

Часто одно дерево решений само по себе не дает сильного классификатора. Специалисты по анализу данных и практики машинного обучения со временем обнаружили, что древовидные классификаторы могут быть сильнее в сочетании со специальными схемами обучения, которые итеративно их улучшают. Одна из таких схем известна как *градиентное усиление* (gradient boosting).

Мы не будем вдаваться в технические подробности и лишь отметим, что процесс усиления обучает последовательность деревьев решений, и каждое новое дерево решений лучше предыдущего. Процедура усиления достигает этого, беря точки данных, которые были неправильно классифицированы предыдущей итерацией дерева решений, и повторно обучает новое дерево решений, чтобы улучшить классификацию этих ранее неправильно классифицированных точек.

ГИПЕРПАРАМЕТРЫ



В предыдущем разделе мы кратко рассмотрели принцип работы дерева решений. В частности, мы установили, что одним из критериев определения того, на каком шаге следует разбить дерево решений (другими словами, когда добавить новую ветвь к блок-схеме), является чистота узлов. Мы также отметили, что если позволить обучающему алгоритму сосредоточиться ис-

ключительно на чистоте узлов в качестве критерия для построения дерева решений, это быстро приведет к деревьям, у которых количество конечных узлов сравняется с количеством точек обучающих данных (так называемое *избыточное обучение*, или *переобучение*, *overfit*). Эти деревья решений настолько настроены на обучающие данные, что они не только выделяют наиболее важные признаки для классификации точки данных, но даже моделируют шум в данных, как если бы это был реальный сигнал. Следовательно, хотя этот вид дерева решений, максимально оптимизированный для некоторых признаков, будет прекрасно работать с обучающими данными, он будет намного хуже работать с тестовым набором данных и плохо обобщать незнакомые данные, хотя это является конечной целью обучения модели.

Чтобы избежать этих ошибок, процедура обучения для создания дерева решений из набора обучающих данных имеет несколько гиперпараметров. *Гиперпараметр* – это своего рода ручка настройки расширенной конфигурации, которую можно крутить до тех пор, пока вы не найдете оптимальные настройки для обучения своей модели. Эти ручки управляют такими параметрами, как количество деревьев, которые обучаются в нашей последовательности усиления, насколько глубоко разветвляется каждое дерево, сколько признаков используется для обучения дерева и т. д.

API Data Frame Analytics предоставляет следующие гиперпараметры для классификации: `eta`, `feature_bag_fraction`, `gamma`, `lambda` и `max_trees`. Мы рассмотрим каждый из них по очереди, но прежде чем углубимся в изучение того, что они означают и как влияют на результирующие деревья решений, которые обучаются на наших наборах данных, давайте поговорим об оптимизации гиперпараметров.

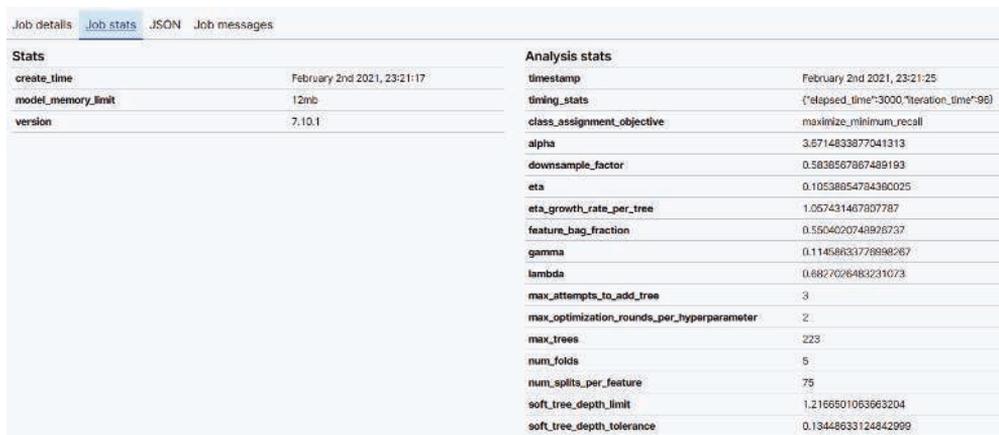
Если вы опытный пользователь и вам доводилось обучать деревья с градиентным усилением при помощи других фреймворков, вы, вероятно, имеете представление о том, какие значения гиперпараметров лучше всего подходят для вашего конкретного набора обучающих данных и задачи. Но как подойти к выбору этих значений, если вы начинаете с нуля? Систематический процесс поиска лучших значений гиперпараметров называется *оптимизацией гиперпараметров*. В этом процессе набор обучающих данных разбивается на блоки перекрестной проверки. Чтобы убедиться, что каждый блок перекрестной проверки является репрезентативным для всего набора обучающих данных, процедура выборки, которая генерирует эти блоки, гарантирует, что пропорция членов каждого класса в блоке перекрестной проверки примерно такая же, как и во всем наборе данных.

После того как набор данных был разделен на блоки, мы можем перейти к следующей части оптимизации гиперпараметров. У нас есть пять гиперпараметров, каждый из которых может принимать широкий диапазон значений. Как разработать систематический метод поиска наилучшего сочетания гиперпараметров? Один из вариантов – создать многомерную сетку, где каждая точка сетки соответствует определенной комбинации значений гиперпараметров. Например, у нас может быть `eta=0,5`, `feature_bag_fraction=0,8`, `gamma=0,7`, `lambda=0,6` и `max_trees=50`. Выбрав эти параметры, мы берем K блоков перекрестной проверки и обучаем модель на $K - 1$ блоках, отложив K -й блок для тестирования. Затем выполняем эту процедуру K раз для одного и того

же набора гиперпараметров и каждый раз пропускаем разный K -й блок для тестирования.

Потом повторяем эту процедуру для следующего набора возможных значений гиперпараметров до тех пор, пока не найдем комбинацию значений, которая лучше всего работает на отложенном K -м тестовом наборе. Как вы, наверное, догадались, повторение этой процедуры даже для небольшого количества комбинаций гиперпараметров может быть довольно дорогостоящим с точки зрения времени и вычислений. Поэтому на практике используют различные методы оптимизации, чтобы получить достаточно хороший набор гиперпараметров с учетом приемлемой стоимости вычислений.

Если вам интересно, какие гиперпараметры выбраны для вашей модели с помощью оптимизации гиперпараметров, вы можете перейти на страницу **Data Frame Analytics** в Kibana и нажать **Manage Jobs**. Вы попадете на страницу управления заданиями **Data Frame Analytics**. Найдите идентификатор задания классификации, которое вы хотите изучить, и щелкните значок с направленной вниз стрелкой слева рядом с идентификатором задания. Откроется раскрывающийся список с подробными сведениями о задании. Нажмите на панель **Job stats** (Статистика задания), как показано на рис. 11.16. Будет показана информация о задании вместе с его статистическими данными.



Stats		Analysis stats	
create_time	February 2nd 2021, 23:21:17	timestamp	February 2nd 2021, 23:21:25
model_memory_limit	12mb	timing_stats	{"elapsed_time":3000,"iteration_time":96}
version	7.10.1	class_assignment_objective	maximize_minimum_recall
		alpha	3.6714833877041313
		downsample_factor	0.5838567887489193
		eta	0.10538654784360025
		eta_growth_rate_per_tree	1.057431487807787
		feature_bag_fraction	0.5504020748926737
		gamma	0.11458833778998287
		lambda	0.6827026483231073
		max_attempts_to_add_tree	3
		max_optimization_rounds_per_hyperparameter	2
		max_trees	223
		num_folds	5
		num_splits_per_feature	75
		soft_tree_depth_limit	1.2166501063963204
		soft_tree_depth_tolerance	0.13448633124842999

Рис. 11.16 ❖ Панель статистики задания отображает основную информацию о задании классификации, а также информацию о гиперпараметрах, определенных процедурой оптимизации

Прежде чем мы закончим этот раздел главы, давайте кратко рассмотрим пять вышеупомянутых гиперпараметров: η , $\text{feature_bag_fraction}$, γ , λ и max_trees , – чтобы понять, что они означают и какой аспект процесса обучения дерева решений зависит от каждого из них. Но сначала стоит кратко рассказать о том, как строятся деревья решений с градиентным усилением. Это поможет оценить значение гиперпараметров в контексте задачи и прояснить, каким образом каждый из них влияет на окончательную форму

последовательности деревьев решений, создаваемых в результате градиентного усиления.

Как вы знаете из предыдущих разделов, в основе деревьев решений с градиентным усилением лежит простое дерево решений. Дерево решений строится путем рекурсивного деления набора данных на все меньшие и меньшие секции на основе пороговых значений для определенных признаков. Например, если мы классифицируем точки данных, представляющие цветки ириса с различными параметрами, то можем принять решение, что будем разделять набор данных по длине лепестков. Все точки данных с длиной лепестков менее 2 см попадают в левый узел, остальные – в правый. Мы выбираем этот признак – длину лепестка в данном случае – путем перебора всех возможных признаков в наборе данных и проверки чистоты узлов, что приводит к разделению с использованием именно этого признака. Как вы можете себе представить, в многомерных наборах данных может потребоваться очень много времени с вычислительной точки зрения, чтобы перебрать все признаки и проверить, насколько чистыми являются узлы, полученные в результате деления по каждому из этих признаков. Чтобы сократить необходимое время вычислений, мы можем выбрать для тестирования только часть признаков, и именно эта часть определяется параметром `feature_bag_fraction`.

После того как мы обучили наше первое дерево решений на основе обучающего набора данных, процесс усиления требует, чтобы мы взяли точки данных, которые были неправильно классифицированы первым деревом решений, и построили последующую итерацию, направленную на улучшение первого дерева решений. Таким образом, к концу этой процедуры у нас будет последовательность деревьев решений, которую в публикациях по машинному обучению иногда называют лесом. Скорость, с которой растет этот лес, другими словами – насколько длинной будет окончательная последовательность деревьев решений, зависит от параметра `eta`. Чем меньше значение `eta`, тем больше будет лес, а также тем лучше этот лес будет обобщаться на новые точки данных.

В нашем разделе, посвященном деревьям решений, мы говорили о том, что если применить простую оптимизацию по критерию соответствия обучающему набору данных, дерево может расти до тех пор, пока не станет идеально соответствовать обучающим данным. Хотя это выглядит хорошо на обучающих данных, на самом деле, если позволить дереву чрезмерно соответствовать обучающим данным, у нас получится модель, которая плохо обобщается. Чтобы ограничить рост отдельного дерева решений, применяются два гиперпараметра – `gamma` и `lambda`. Чем выше значение `gamma`, тем большее предпочтение отдается деревьям меньшего размера, что помогает уменьшить переобучение. Как и в случае с параметром `gamma`, чем выше значение `lambda`, тем меньше будут деревья решений.

Хотя на практике вы всегда можете положиться на процесс оптимизации гиперпараметров, хорошо знать хотя бы на концептуальном уровне, что означает каждый из этих параметров и как они влияют на окончательную обученную модель.

ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ

В предыдущем разделе мы рассмотрели теоретические основы деревьев решений и основные подходы к их построению. В этом разделе мы вернемся к примеру классификации образцов биологических тканей и более подробно рассмотрим формат результатов, а также обсудим их интерпретацию.

Ранее в этой главе мы создали обученную модель, чтобы предсказать, является ли определенный образец ткани молочной железы вредоносным или доброкачественным (напомним, что в этом наборе данных злокачественная опухоль относится к классу 2, а доброкачественная – к классу 4). Фрагмент результатов классификации для этой модели показан на рис. 11.17.

```
# ml.Class_prediction      4
ml.is_training            false
# ml.prediction_probability 0.814
# ml.prediction_score      0.814

ml.top_classes            {
  "class_name": 4,
  "class_probability": 0.8136328521395665,
  "class_score": 0.8136328521395665
},
{
  "class_name": 2,
  "class_probability": 0.1863671478604335,
  "class_score": 0.14323736037702758
}
```

Рис. 11.17 ❖ Результаты классификации для выборки из набора данных по раку молочной железы в Висконсине

Располагая обученной моделью, мы можем брать незнакомые точки данных и делать прогнозы. В какой форме представлены эти прогнозы? В простейшем случае точке данных назначается метка класса (поле `ml.Class_prediction` на рис. 11.17 является примером такого прогноза). В нашем примере эта метка может принимать одно из двух значений: 2 (злокачественная) и 4 (доброкачественная).

Однако такой способ классификации точек данных скрывает определенную проблему процесса прогнозирования: степень уверенности в своих прогнозах. Вы можете увидеть важность количественной оценки неопределенности, связанной с нашими прогнозами, если вспомните о таком банальном явлении, как ежедневные прогнозы погоды. Допустим, у вас есть прогноз, что в среду пойдет дождь с вероятностью 80 %, а в четверг – с вероятностью 16 %. Оба дня в прогнозе названы «дождливыми», но вероятность того, что вы забудете дома зонтик в среду, будет меньше, чем в четверг. Все это говорит о том, что при оценке наших классификаторов машинного обучения следует знать не только метку класса, присвоенную данной точке данных,

но и то, насколько уверена модель в этой метке. В Elastic Stack есть два показателя уверенности модели в присвоенных метках классов: *вероятность класса* и *оценка класса*. Далее мы рассмотрим их более подробно.

В дополнение к знанию того, насколько модель уверена в данном прогнозе, часто бывает очень полезно знать, какие признаки точки данных были наиболее важны для отнесения точки к одному классу по сравнению с другим. Вклад признака точки данных в классификацию зависит от показателя значимости этого признака.

Вероятность класса

Как было сказано выше, обычно недостаточно просто знать метку, которую классификатор машинного обучения назначил точке данных. Нам также необходимо знать, какова была вероятность такого назначения. На рис. 11.17 вероятности для классов 2 и 4 показаны во вложенной структуре под полем `ml.top_classes`. Метка класса, присвоенная точке данных, равна 4, а вероятность того, что точка данных принадлежит этому классу, равна 0,814 (округленно). Модель уверена, что эта точка данных действительно относится к классу 4.

Оценка класса

Хотя во многих случаях присвоение метки класса на основе класса, который получает более высокую вероятность, является достаточно хорошим правилом, это не лучший выбор для всех наборов данных. Например, для наборов данных, в которых классы сильно не сбалансированы, лучшим вариантом может быть использование оценки класса. Это значение рис. 11.17 обозначено как `ml.prediction_score`, а более точное значение `ml.top_classes.class_score` можно видеть во вложенной структуре меток классов и вероятностей классов.

Оценка класса вычисляется из вероятности класса, но таким образом, чтобы учитывать, хотите ли вы максимизировать точность или минимальный отклик. Другими словами, она зависит от того, насколько терпимой является неправильная классификация точек данных тех классов, которые меньше представлены в наборе обучающих данных.



Подробное объяснение того, как рассчитываются оценки класса, можно найти в документации Elastic по адресу <https://www.elastic.co/guide/en/machine-learning/current/dfa-classification.html#dfa-classification-class-score>, а для более подробного ознакомления – в блокноте Jupyter Notebook по адресу <https://github.com/elastic/examples/blob/master/Machine%20Learning/Class%20Assignment%20Objectives/classification-class-assignment-objective.ipynb>.

Важность признака

Изучая прогнозы модели машинного обучения, мы не только заботимся о прогнозируемой метке класса, вероятности этой метки класса, а также, потенциально, об оценке класса; обычно мы также хотим знать, какие признаки

способствовали принятию моделью определенного решения. Это зависит от такого показателя, как *важность признака* (feature importance). Каждому полю, используемому в процессе обучения (поля, которые были выбраны как **Included**, т. е. включенные во время настройки задания классификации), можно присвоить значение важности признака, но обычно нас интересуют только наиболее важные признаки, то есть поля с наивысшими значениями важности.

Поэтому, чтобы не загромождать хранилище результатов Elasticsearch в нашем кластере результатами каждого задания машинного обучения, мы можем выбрать в конфигурации задания классификации количество основных значений важности признака, которые будут записаны для каждой классифицированной точки данных. Для нашей конфигурации установлено значение 4 в примере, показанном на рис. 11.18.

2 Additional options

Advanced configuration

Feature importance values

4

Specify the maximum number of feature importance values per document to return.

Model memory limit

12mb

The approximate maximum amount of memory resources that are permitted for analytical processing.

Prediction field name

Class_prediction

Defines the name of the prediction field in the results. Defaults to <dependent_variable>_prediction.

Maximum number of threads

1

The maximum number of threads to be used by the analysis. The default value is 1.

Top classes

2

The number of categories for which the predicted probabilities are reported.

> **Hyperparameters**

Рис. 11.18 ❖ Эта конфигурация будет записывать 4 значения важности функций для каждого документа

После завершения задания по классификации каждый документ в хранилище результатов будет иметь, помимо предсказанного класса, вероятность и оценку класса, а также четыре верхних значения важности признаков для данной точки данных. Сокращенный фрагмент значений важности признаков для выборки данных показан на рис. 11.19.

Этой точке данных назначена метка класса 4. Среди признаков, которые больше всего способствовали присвоению именно этого класса, были значения полей `Bag_Nuclei` и `Marginal_Adhesion`.

Помимо изучения значений важности признаков для каждой отдельной точки данных, мы также можем изучить, какие признаки важны для классификации в наборе данных в целом. Эта диаграмма показана на рис. 11.20 и доступна на экране Data Frame Analytics (вы можете получить доступ к этому представлению, перейдя на страницу управления заданиями аналитики фрейма данных, выбрав задание, для которого вы настроили, сколько значений важности признаков должно быть записано, а затем нажав на кнопку **View**).



```

# ml.Class_prediction
ml.feature_importance
{
  "feature_name": "Bare_Nuclei",
  "classes": [
    {
      "importance": -1.8031243941016428,
      "class_name": 2
    },
    {
      "importance": 1.8031243941016428,
      "class_name": 4
    }
  ]
},
{
  "feature_name": "Marginal_Adhesion",
  "classes": [
    {
      "importance": -0.7605405795083313,
      "class_name": 2
    },
    {
      "importance": 0.7605405795083313,
      "class_name": 4
    }
  ]
}

```

Рис. 11.19 ❖ Два значения важности признаков для выборочной точки данных



Рис. 11.20 ❖ Совокупные значения важности признаков для всего набора данных

ЗАКЛЮЧЕНИЕ

В этой главе вы глубоко погрузились в обучение с учителем. Вы узнали, что это такое, какую роль играют обучающие данные при построении модели, что означает обучение модели, какие признаки следует использовать для достижения оптимального качества модели, а также как модель оценивается и что означают различные показатели качества.

Узнав об основах обучения с учителем в целом, вы более внимательно изучили классификацию и узнали, как создавать и запускать задания классификации в Elastic Stack, а также как оценивать обученные модели, создаваемые этими заданиями. В дополнение к рассмотрению основных поня-

тий, таких как матрица неточностей, вы также рассмотрели ситуации, когда следует скептически относиться к результатам, которые кажутся слишком хорошими, чтобы быть правдой, потенциальные причины, по которым результаты классификации иногда могут казаться идеальными, и почему это не обязательно означает, что обученная модель хороша.

Затем вы более подробно рассмотрели алгоритм, на котором основан механизм классификации в Elastic Stack: деревья решений с градиентным усилением. Чтобы детально разобраться в том, как работают деревья решений, вы исследовали, как строится отдельное дерево решений, узнали, что подразумевают под чистотой узлов в контексте деревьев решений, а также как чрезмерная оптимизация деревьев решений может привести к созданию моделей, которые подвержены переобучению и плохо обобщаются. Для тонкой настройки процесса выращивания дерева решений из набора данных применяются несколько дополнительных обобщенных параметров конфигурации – гиперпараметров. По умолчанию они устанавливаются процедурой, известной как оптимизация гиперпараметров, но опытные пользователи также могут настроить их вручную.

В последнем разделе этой главы мы вернулись к нашему исходному набору данных классификации рака молочной железы, чтобы дополнительно изучить формат результатов и значение вероятности класса, оценки класса и того, как важность признаков влияет на отнесение точки данных к тому или иному классу.

В следующей главе вы узнаете, как деревья решений можно использовать для решения задач, в которых зависимая переменная, которую нужно предсказать, является не дискретным значением, как в случае классификации, а непрерывной величиной.



ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Для получения дополнительной информации о том, как рассчитывается оценка класса, ознакомьтесь с примерами текста и кода в блокноте Jupyter по адресу <https://github.com/elastic/examples/blob/master/Machine%20Learning/Class%20Assignment%20Objectives/classification-class-assignment-objective.ipynb>.

Глава 12



Регрессия

В предыдущей главе вы изучали классификацию – один из двух методов обучения с учителем, доступных в Elastic Stack. Однако классификация подходит отнюдь не для всех задач. Что, если вы захотите спрогнозировать цены на квартиры в вашем районе? Или сколько денег средний покупатель потратит в нашем интернет-магазине? В этих случаях значение, которое нас здесь интересует, не относится к одному из дискретных классов, а представляет собой переменную величину, способную принимать множество непрерывных значений в определенном диапазоне.

Именно эту проблему решает *регрессионный анализ* данных. Вместо того чтобы предсказывать, к какому классу принадлежит точка данных, мы можем предсказывать непрерывное значение. Хотя конечная цель регрессионного анализа немного отличается от цели классификации, основной алгоритм, используемый для регрессионного анализа, аналогичен тому, с которым вы познакомились в главе 11. Поэтому вы уже много знаете о том, как работает регрессия.

Поскольку результатом регрессионного анализа являются непрерывные значения вместо дискретной метки класса, способы, которыми мы оцениваем качество регрессионной модели, немного отличаются от способов, которые мы исследовали для классификации в предыдущей главе. Вместо того чтобы использовать матрицы неточностей и различные метрики, вычисленные из количества правильно и неправильно помеченных выборок, мы вычисляем агрегированные метрики, которые указывают, насколько далеки непрерывные значения, предсказанные для нашего набора данных, от фактических значений, представленных в этом обучающем наборе. Позже в этой главе мы подробнее рассмотрим, как это работает на практике.

В данной главе мы рассмотрим следующие темы:

- использование регрессионного анализа для прогнозирования цен на жилье;
- как деревья решений применяются для создания регрессионных моделей.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Для материалов этой главы потребуется кластер Elasticsearch версии 7.10.1 или новее. Некоторые примеры могут включать снимки экрана или опера-

ции, которые доступны только в более поздних версиях Elasticsearch. В таких случаях в тексте будет явно указано, какая более поздняя версия требуется для выполнения примера.

ИСПОЛЬЗОВАНИЕ РЕГРЕССИОННОГО АНАЛИЗА ДЛЯ ПРОГНОЗИРОВАНИЯ ЦЕН НА ЖИЛЬЕ

В предыдущей главе мы рассмотрели первый из двух методов обучения с учителем в Elastic Stack – классификацию. Цель классификационного анализа – использовать размеченный набор данных для обучения модели, способной предсказать метку класса для незнакомой точки данных. Например, мы могли бы обучить модель на историческом наборе данных об образцах клеток в сочетании с информацией о том, является ли клетка злокачественной, и использовать эту модель для прогнозирования класса клеток, данные о которых отсутствуют в обучающем наборе. В классификации класс или зависимая переменная, которую мы хотим спрогнозировать, всегда является дискретной величиной. Если же нам нужно предсказать значение непрерывной величины в определенный момент времени, значит, речь идет о регрессии.

Прежде чем приступить к изучению теоретических основ регрессии, давайте рассмотрим практический пример обучения регрессионной модели в Elasticsearch. Набор данных, который мы будем использовать, доступен на Kaggle (<https://www.kaggle.com/harlfoxem/housesalesprediction>) и описывает цены на дома, проданные в районе штата Вашингтон в США в период с 2014 по 2015 год. Исходный набор данных был немного изменен, чтобы упростить загрузку в Elasticsearch; он доступен в репозитории GitHub по адресу <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter12>. Чтобы воспользоваться этим набором данных для обучения регрессионной модели, выполните следующие действия:

1. Загрузите набор данных в Elasticsearch, используя предпочитаемый вами метод. При желании вы можете применить функцию **Upload file**, доступную в визуализаторе данных **Data Visualizer** приложения **Machine Learning**, как показано на рис. 12.1.
2. После окончания загрузки данных их можно просмотреть в **Data Visualizer**. В этом представлении сразу видно, какие поля присутствуют в данных и каково распределение значений для каждого поля. Например, для нашего набора данных о ценах на жилье мы можем видеть на гистограмме значений цен на рис. 12.2, что большинство цен на дома в этом наборе данных находятся в диапазоне от 200 000 до 900 000 долларов США, при этом небольшая часть домов продается по цене более 900 000 долларов США (рис. 12.2).

Просмотр значений и распределения данных в визуализаторе может быстро предупредить нас о потенциальных проблемах, таких как недопустимые или отсутствующие значения в заданном поле набора данных.

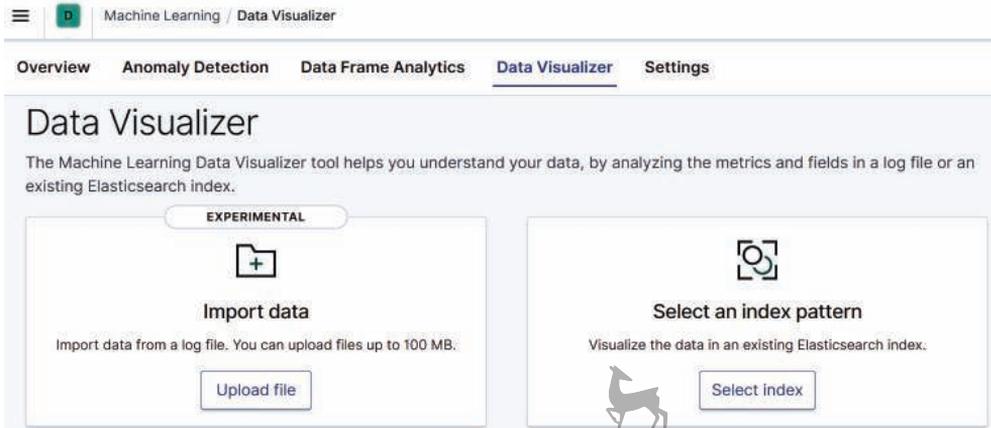


Рис. 12.1 ❖ Загрузка файла через Data Visualizer приложения Machine Learning

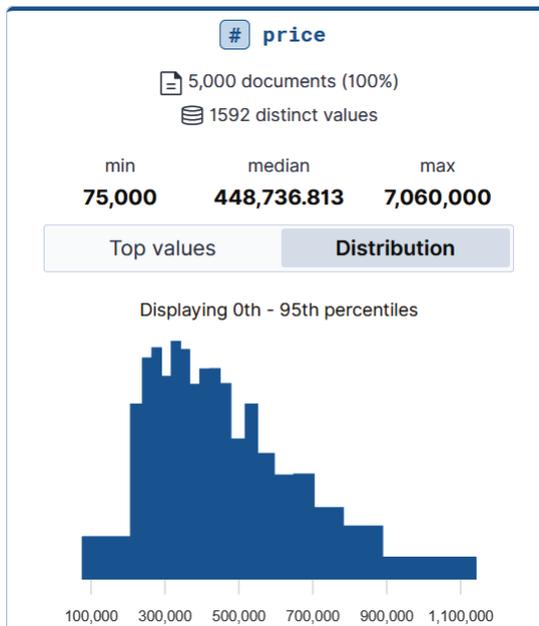


Рис. 12.2 ❖ Распределение значений поля цены, показанное в визуализаторе данных

Затем перейдем к мастеру **Data Frame Analytics**. В левом выпадающем меню Kibana нажмите на **Machine Learning**, чтобы перейти на главную страницу приложения. На этой странице в разделе **Data Frame Analytics**, если вы еще не создали задание анализа фрейма данных, нажмите кнопку **Create job**. Она приведет вас к мастеру аналитики фрейма данных, который мы уже видели в главах 10 и 11.

Если вы уже создали задание, сначала нажмите кнопку **Manage jobs**. Вы перейдете к странице **Data Frame Analytics**, где увидите список существующих заданий (например, если вы создали какое-либо задание, следуя пошаговым инструкциям в предыдущих главах) и кнопку **Create job**.

3. После того как вы выбрали шаблон исходного хранилища (он должен совпадать с именем шаблона хранилища, который вы выбрали при импорте или загрузке данных на шаге 1), выберите **Regression** в селекторе заданий мастера Data Frame Analytics, как показано на рис. 12.3.

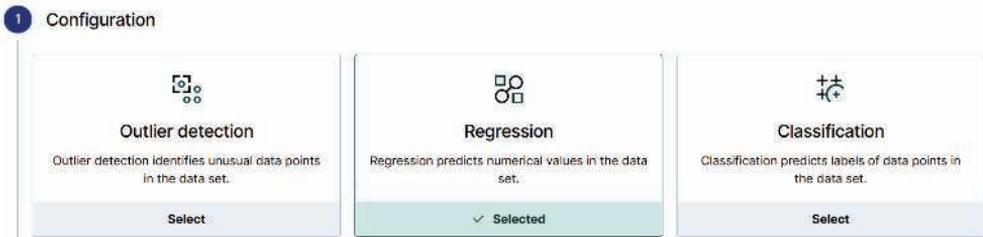


Рис. 12.3 ❖ Выберите **Regression** в качестве типа задания в селекторе заданий Data Frame Analytics

4. Теперь выберем зависимую переменную **Dependent Variable** и поля, которые будут включены в анализ. Как вы знаете из предыдущих глав, зависимая переменная обозначает поле, содержащее метки, которые мы хотим использовать для обучения нашей модели с учителем. Для задач классификации это поле содержало метки класса, к которому принадлежала конкретная точка данных (глава 11). Для задач регрессии это поле будет содержать непрерывные значения, прогнозирования которых мы хотим научить нашу модель. В данном случае мы хотим спрогнозировать цену дома и поэтому выбираем поле `price`. После выбора зависимой переменной перейдем к выбору полей, которые будут включены или исключены из нашего анализа. Хотя многие поля в наборе данных предоставляют полезную информацию для прогнозирования значения зависимой переменной (цены), есть несколько полей, которые, как мы знаем с самого начала, не будут коррелировать с ценой, поэтому их следует удалить. Первый из них – это идентификатор (`id`) точки данных. Это просто номер строки, обозначающий положение точки данных в исходном файле данных, и поэтому нельзя ожидать, что он будет содержать полезную информацию. На самом деле его наличие может принести больше вреда, чем пользы. Например, если исходный файл данных был организован таким образом, что все недорогие дома были расположены в начале файла с небольшими значениями идентификаторов, а все дорогие дома – в конце файла, модель может ошибочно принять идентификатор данных за важный фактор для определения цены дома, хотя мы знаем, что это всего лишь особенность конкретного набора данных. Это, в свою очередь, может

отрицательно сказаться на качестве модели в отношении будущих, пока еще незнакомых точек данных. Если мы предположим, что значение идентификатора точек данных растет, любая новая точка данных, добавленная в набор, будет автоматически иметь более высокий номер идентификатора, чем любая точка данных в обучающих данных, и, таким образом, модель будет предполагать, что любая новая точка данных соответствует дому с более высокой ценой.

Кроме того, мы также исключим информацию о широте и долготе местоположения дома, поскольку эти переменные не могут интерпретироваться нашим упрощенным алгоритмом машинного обучения как географические местоположения и будут просто интерпретированы как числа. Окончательная конфигурация задания должна выглядеть, как показано на рис. 12.4.

Dependent variable

price

Included fields

18 fields included in the analysis

is_included:false

Is included Is not included

<input type="checkbox"/>	Field name	Mapping	Is included	Is required	Reason
<input type="checkbox"/>	id	long	No	No	field not in includes list
<input type="checkbox"/>	lat	double	No	No	field not in includes list
<input type="checkbox"/>	location	keyword	No	No	field not in includes list
<input type="checkbox"/>	long	double	No	No	field not in includes list

Рис. 12.4 ❖ Поля, не включенные в анализ

- После настройки зависимой переменной, а также включенных и исключенных полей мы можем перейти к дополнительным элементам конфигурации. Хотя мы оставляем большинство значений в этом разделе по умолчанию, мы изменим количество значений важности признаков с 0 на 4 (рис. 12.5). Если установлено количество 0, значения важности признаков не записываются. Если установлено количество 4, для каждой точки данных записываются четыре наиболее важных значения признаков. Как кратко сказано в главе 11, значения важности признаков записываются отдельно для каждого документа и помогают определить, почему модель определенным образом классифицирует конкретный документ. Мы вернемся к этому вопросу чуть позже в данной главе.

Мы оставляем остальные настройки по умолчанию и запускаем задание, прокручивая страницу мастера до конца и нажимая кнопку **Create and start**.

2 Additional options

Advanced configuration

Feature importance values <input type="text" value="4"/> Specify the maximum number of feature importance values per document to return.	Prediction field name <input type="text"/> Defines the name of the prediction field in the results. Defaults to <dependent_variable>_prediction.	Model memory limit <input type="text" value="36mb"/> The approximate maximum amount of memory resources that are permitted for analytical processing.
Maximum number of threads <input type="text" value="1"/> The maximum number of threads to be used by the analysis. The default value is 1.		

Рис. 12.5 ❖ Настройки для значений важности признаков

6. После выполнения перечисленных выше шагов и запуска задания вернитесь на страницу **Data Frame Analytics**. На ней вы увидите обзор всех заданий, которые вы создали, включая новое задание регрессии. Когда работа будет завершена, щелкните меню справа и выберите **View**, как показано на рис. 12.6.

Рис. 12.6 ❖ Выберите **View**, чтобы увидеть результаты для интересующего вас задания

Вы окажетесь на странице **Exploration** (Исследование), которая позволяет нам исследовать различные метрики недавно обученной регрессионной модели. Первое, что требует внимания на этой странице, – это переключатель набора данных обучения/тестирования (**Training/Testing**), который показан на рис. 12.7.



Рис. 12.7 ❖ Переключатель обучения/тестирования в средстве просмотра результатов

Важно помнить об этом переключателе при просмотре метрик на странице **Exploration**, потому что оценки модели означают разные вещи

в зависимости от того, какая из метрик выбрана. В данном случае нас интересует, как модель будет работать, когда она попытается делать прогнозы для незнакомых точек данных. Набор данных, который наиболее хорошо соответствует этой задаче, – это набор данных тестирования – другими словами, набор данных, который не участвовал в процессе обучения.

7. Прокрутите страницу вниз и взгляните на метрики оценки модели для набора данных тестирования, которые показаны на рис. 12.8.



Рис. 12.8 ❖ Ошибка обобщения

Мы более подробно разберем, что означает каждый из этих показателей, позже в этой главе, а пока вы можете рассматривать их как совокупные меры того, насколько близки прогнозы цен на жилье, сделанные моделью, к фактическим ценам на жилье.

8. Наконец, многих читателей может заинтересовать то, какие из полей в нашем наборе данных были наиболее важными при вычислении окончательного прогноза модели. Мы можем узнать это, взглянув на раздел **Total feature importance** (Общая важность признака) на странице Exploration (рис. 12.9).

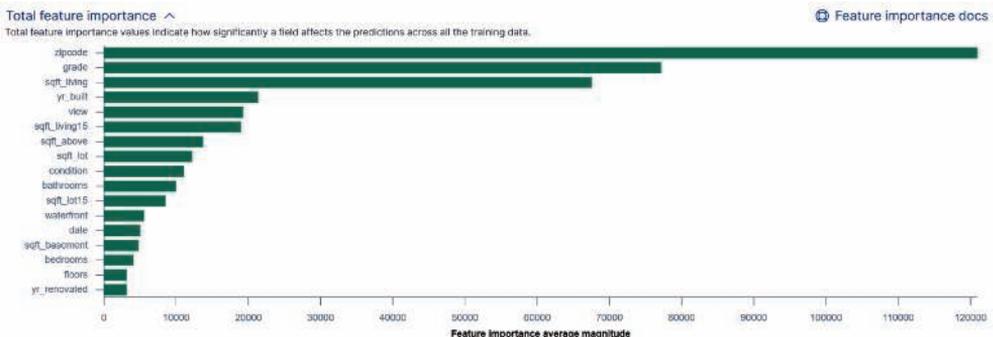


Рис. 12.9 ❖ Общая важность признака

Как следует из рисунка, наиболее важным фактором для определения продажной цены дома в King Country в штате Вашингтон является почтовый индекс (`zip_code`), т. е. местоположение дома. К наименее важным факторам относятся год, в котором дом был отремонтирован (`yr_renovated`), и количество этажей (`floors`) в нем.

Следует иметь в виду, что признаки, показанные на этой диаграмме, являются наиболее важными для набора данных в целом. Однако признаки, определяющие продажную цену конкретной точки в наборе данных, могут сильно отличаться от общей картины. Немного ранее в этом пошаговом руководстве, во время создания задания в мастере Data Frame Analytics, мы настроили количество значений важности признаков равным 4. Это означает, что после обучения модели четыре наиболее существенных значения важности признаков будут записаны в хранилище результатов. Давайте рассмотрим образец документа в нашем хранилище результатов `king-county-houses-regression`. Этот документ показан на рис. 12.10.

```
m1.feature_importance {
  {
    "feature_name": "grade",
    "importance": -49697.953304322975
  },
  {
    "feature_name": "sqft_living",
    "importance": -29824.77907346486
  },
  {
    "feature_name": "yr_built",
    "importance": -10351.602066855963
  },
  {
    "feature_name": "zipcode",
    "importance": -194892.77601876782
  }
}
```

Рис. 12.10 ❖ Значимость признаков для одного из документов в хранилище результатов

Как мы видим на рис. 12.10, для этого конкретного дома четыре наиболее важных признака – это `grade` (категория), `sqft_living` (жилая площадь дома, измеренная в квадратных футах), `yr_built` (год постройки дома) и `zipcode` (почтовый индекс), соответствующий району города, в котором расположен дом. Следует отметить, что все значения важности характеризуют здесь отрицательны, а это означает, что они способствуют снижению цены на дом.

Взгляните на четыре верхних значения важности признаков в другом документе (рис. 12.11) в хранилище результатов и сравните их с предыдущими значениями важности (рис. 12.10), чтобы увидеть, насколько сильно они могут различаться.

```

m1.feature_importance {
  {
    "feature_name": "sqft_living",
    "importance": 43226.39584919779
  },
  {
    "feature_name": "sqft_lot15",
    "importance": 37270.72453155121
  },
  {
    "feature_name": "yr_built",
    "importance": 36016.55250544813
  },
  {
    "feature_name": "zipcode",
    "importance": 179641.80964027002
  }
}

```

Рис. 12.11 ❖ Четыре наиболее важных признака, влияющих на цену этого дома

Как видите, дом, представленный в документе на рис. 12.11, имеет некоторые общие черты с домом на рис. 12.10, но также имеет некоторые уникальные особенности. Кроме того, характерные признаки этого дома положительно влияют на его цену.

Теперь, когда вы вкратце узнали, как обучать регрессионную модель и оценивать ее результаты, возникает естественный вопрос: что дальше? Если бы вы использовали эту модель в реальной жизни, например для прогнозирования потенциальных цен на еще не проданные дома в округе Кинг или другом месте, то могли бы применить логический вывод для обучения и развертывания этой модели. Более подробно мы рассмотрим данный вопрос в главе 13. В следующем разделе мы вернемся к нашему обсуждению деревьев решений, которое начали в главе 11, и посмотрим, как принципы работы этого алгоритма применимы к задачам регрессии.

ИСПОЛЬЗОВАНИЕ ДЕРЕВЬЕВ РЕШЕНИЙ В РЕГРЕССИОННОМ АНАЛИЗЕ

Как мы говорили в предыдущих главах, регрессионные модели основаны на обучении с учителем. Принцип обучения с учителем заключается в том, чтобы взять размеченный набор данных (например, набор данных, содержащий характеристики домов и их рыночную цену – зависимую переменную) и преобразовать знания из этих данных в обученную модель. Затем эту обученную модель можно использовать для прогнозирования рыночных цен домов, которые модель ранее не видела. Когда зависимая переменная, которую мы пытаемся предсказать, является непрерывной переменной, в отличие от дискретной переменной из задач классификации, мы имеем дело с регрессией.

Регрессия – задача извлечения информации, представленной в реальных наблюдениях или данных, – это область машинного обучения, которая включает в себя методы, гораздо более обширные, чем метод дерева решений, который используется в машинном обучении Elasticsearch. Однако мы ограничимся здесь концептуальным обсуждением того, как регрессия может быть выполнена с использованием деревьев решений (упрощенная версия процесса, который происходит внутри Elastic Stack), а читатели, заинтересованные в получении дополнительной информации о регрессии, могут обратиться к литературе. Например, книга *Mathematics for Machine Learning* (<https://mml-book.github.io/>) содержит хорошее введение в регрессию.

В главе 11 мы начали обсуждение деревьев решений применительно к задачам классификации с рассмотрения блок-схемы, которую вы могли бы построить, если бы хотели взять дом и попытаться вычислить его рыночную цену. Пример вымышленной блок-схемы, которую вы могли бы построить, представлен на рис. 12.12.

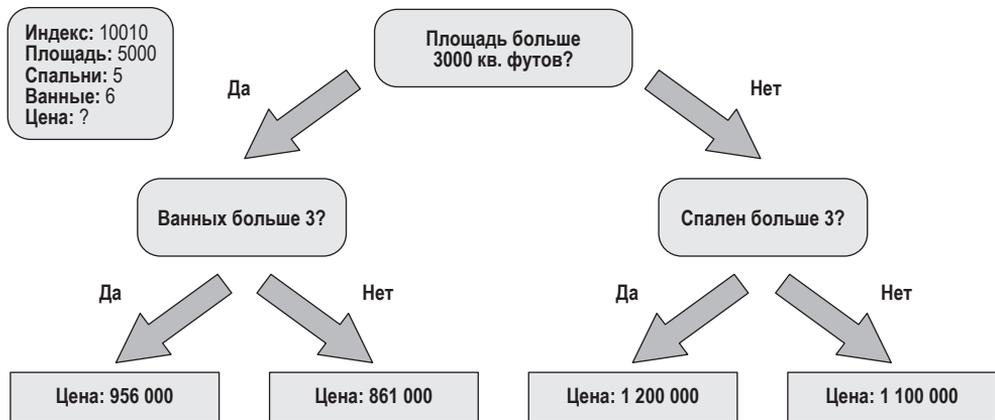


Рис. 12.12 ❖ Примерная блок-схема, иллюстрирующая решения, которые вы можете принять, чтобы попытаться спрогнозировать рыночную цену дома

На вымышленной блок-схеме, показанной на рис. 12.12, у нас есть образец дома, изображенный рамкой в верхнем левом углу. В этом доме 5000 квадратных футов жилой площади, 5 спален и 5 ванных комнат. Мы хотим спрогнозировать окончательную цену продажи дома на основе данных атрибутов и собираемся использовать для этого блок-схему. Начинаем с вершины блок-схемы (или *корня* дерева решений) и продвигаемся вниз к конечным узлам (узлам, которые не имеют нижестоящих узлов). Жилая площадь нашего дома составляет 5000 квадратных футов, поэтому мы даем ответ «Да» для первого узла и снова «Да» для второго дочернего узла. Это приводит нас к конечному или листовому узлу, который содержит прогнозируемую цену нашего дома: 956 000 долларов США.

Возникает естественный вопрос: как создать блок-схему, подобную показанной на рис. 12.12? Для создания этой блок-схемы (или дерева решений)

требуются два компонента: размеченный набор данных, который содержит характеристики каждого дома и его продажную цену, и алгоритм обучения, который берет этот набор данных и использует его для построения законченной блок-схемы или дерева решений, которое затем можно использовать для прогнозирования цен на другие дома (это делается с помощью вывода, который подробно рассматривается в главе 13).

Процесс обучения дерева решений на основе размеченного набора данных включает создание узлов, подобных тем, которые изображены на рис. 12.12. Эти узлы разбивают набор данных на все меньшие и меньшие подмножества, пока мы не достигнем ситуации, когда каждое подмножество удовлетворяет определенным критериям. В случае классификации процесс постепенного разделения набора обучающих данных останавливается, когда подмножества в узлах достигают определенной чистоты. Чистоту узла можно измерить с помощью нескольких различных показателей, которые фиксируют долю точек данных, принадлежащих определенному классу в узле. Самый чистый узел – это узел, который содержит только точки данных, принадлежащие определенному классу.

Поскольку регрессия имеет дело с непрерывными значениями, мы не можем использовать чистоту в качестве меры, чтобы определить, когда прекратить рекурсивное разделение набора данных. Вместо этого у нас есть другая мера, известная как *функция потерь*. В качестве функции потерь для регрессии обычно применяется *среднеквадратичная ошибка* (mean square error). Эта мера показывает, насколько далеки прогнозируемые значения для точек, попадающих в каждый узел, от фактических значений.

Наконец, если набор данных был рекурсивно разделен много раз, мы получим дерево решений, упрощенно представленное на рис. 12.13.

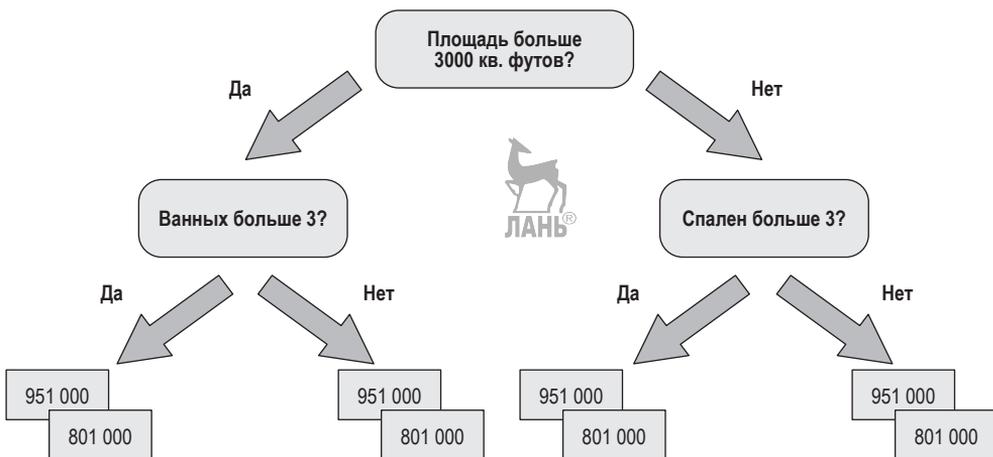


Рис. 12.13 ❖ Упрощенное представление обученного дерева решений. Листовые узлы содержат несколько точек данных

На этом рисунке в конце процесса обучения каждый конечный узел содержит по несколько точек данных.

ЗАКЛЮЧЕНИЕ

Регрессия – это второй вариант применения метода машинного обучения с учителем в Elastic Stack. Цель обучения – взять обучающий набор данных (набор данных, содержащий некоторые признаки и зависимую переменную, которую мы хотим научиться предсказывать) и на его основе получить обученную модель. В регрессии зависимая переменная представляет собой непрерывное значение, что отличает ее от классификации, которая обрабатывает дискретные значения. В этой главе мы использовали возможности машинного обучения Elastic Stack, чтобы применять регрессию для прогнозирования рыночной цены дома на основе ряда атрибутов, таких как местоположение дома и количество спален. Хотя существует множество методов регрессии, Elastic Stack использует для обучения модели дерева решений с градиентным усилением.

В следующей главе мы рассмотрим, как модели машинного обучения с учителем могут применяться вместе с обработчиками вывода и конвейерами входных потоков для создания мощных конвейеров анализа данных на основе машинного обучения.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

Для получения дополнительной информации о том, как вычисляются значения важности признаков, прочитайте публикацию в блоге *Feature importance for data frame analytics with Elastic machine learning* по адресу <https://www.elastic.co/blog/feature-importance-for-data-frame-analytics-with-elastic-machine-learning>.

Если вы хотите более глубоко изучить математический аппарат машинного обучения, включая регрессию, прочтите книгу *Mathematics for Machine Learning* по адресу <https://mml-book.github.io/>.



Логический вывод моделей

В этой главе мы подробно расскажем про увлекательные и полезные вещи, которые вы можете делать с моделями машинного обучения в Elastic Stack. Вы узнаете, как использовать API для просмотра перечня моделей, доступных в вашем кластере, для просмотра подробной информации об отдельных моделях и для экспорта моделей, чтобы их можно было перенести в другие кластеры Elasticsearch. Мы также кратко рассмотрим, как использовать библиотеку `eland` для импорта внешних моделей, например обученных сторонними библиотеками машинного обучения, в Elasticsearch.

Во второй части этой главы вы узнаете, как использовать вывод моделей машинного обучения с учителем в различных контекстах для расширения данных. Мы расскажем об обработчиках логического вывода и конвейерах данных, а также о том, как их можно комбинировать с непрерывными преобразованиями и переиндексированием во время получения, когда используются различные методы поставки данных в Elasticsearch.

В этой главе мы рассмотрим следующие темы:

- поиск, импорт и экспорт обученных моделей с помощью API;
- обработчики логического вывода и конвейеры данных;
- импорт внешних моделей с помощью `eland`.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Для материала данной главы потребуется кластер Elasticsearch версии 7.10 или новее, а также Python версии 3.7 или новее с установленными библиотеками `eland`, `elasticsearch-py` и `scikit-learn`. Подробные инструкции о том, как настроить Python для работы с этой главой, вы найдете в файле README в репозитории GitHub по адресу: <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter13>.

Поиск, импорт и экспорт обученных моделей с помощью API

Допустим, вы подготовили свой набор данных, обучили классификационную или регрессионную модель, оценили ее качество и решили, что хотите использовать ее для обогащения своих производственных наборов данных. Прежде чем вы погрузитесь в работу с конвейерами данных, обработчиками логического вывода и множеством других компонентов, которые вы можете настроить для использования ваших обученных моделей, полезно ознакомиться с API обученных моделей (<https://www.elastic.co/guide/en/elasticsearch/reference/7.10/get-trained-models.html>) – набором конечных точек REST API, которые вы можете использовать для получения информации о ваших моделях и даже для их экспорта в другие кластеры. Давайте познакомимся с этим API, чтобы узнать, что он может рассказать нам о наших моделях.

Обзор API обученных моделей

В этом разделе мы на практике рассмотрим использование Kibana Dev Console для получения информации о моделях машинного обучения с учителем.

1. Начнем с Kibana Dev Console. Вкратце, для тех, кто еще незнаком с этим инструментом, консоль Kibana Dev Console, доступ к которой можно получить, нажав на раскрывающееся меню слева и прокрутив вниз до меню **Management** (Управление), дает опытным пользователям удобную графическую среду, с помощью которой можно выполнять команды REST API. Мы воспользуемся этой функциональностью, чтобы общаться с API Inference и изучать ответы, которые мы получаем от Elasticsearch при запросе различных конечных точек REST API. В Kibana Dev Console выполните следующий запрос REST:

```
GET _ml/trained_models/
```

Ответ, который мы получаем от Elasticsearch, включает сводную информацию о том, сколько обученных моделей в настоящее время имеется в кластере, а также сведения о каждой модели. Точный ответ, который вы получите, конечно же, будет варьироваться от кластера к кластеру в зависимости от того, какие модели вы обучаете, поэтому фрагмент ответа, показанный на рис. 13.1, является лишь примером.

Глядя на ответ на рис. 13.1, вы можете увидеть, что мы получаем от Elasticsearch счетчик – количество обученных моделей, имеющих в кластере, а также список объектов модели, каждый из которых содержит множество атрибутов, описывающих модель.

Хотя большинство полей в объекте модели могут пригодиться в тот или иной момент, на этом этапе важным полем, на которое следует обратить внимание, является `model_id`. Это поле содержит уникальный идентификатор, который назначается каждой модели, хранящейся

в кластере, и используется для ссылки на модель, когда она позже используется в обработчиках вывода и конвейерах данных.

```

1  {
2    "count" : 3,
3    "trained_model_configs" : [
4      {
5        "model_id" : "breast-cancer-wisconsin-classification-1612270856116",
6        "created_by" : "_xpack",
7        "version" : "7.10.1",
8        "create_time" : 1612270856116,
9        "estimated_heap_memory_usage_bytes" : 2272,
10       "estimated_operations" : 17,
11       "license_level" : "platinum",
12       "description" : "",
13       "tags" : [
14         "breast-cancer-wisconsin-classification"
15       ],
16       "metadata" : {
17         "analytics_config" : {
18           "max_num_threads" : 1,
19           "model_memory_limit" : "12mb",
20           "create_time" : 1612270847492,
21           "allow_lazy_start" : false,
22           "description" : "",
23           "analyzed_fields" : {
24             "excludes" : [ ],
25             "includes" : [
26               "Bare_Nuclei",
27               "Bland_Chromatin",
28               "Class",
29               "Clump_Thickness",
30               "Marginal_Adhesion",
31               "Mitoses",
32               "Normal_Nucleoli",
33               "Outlier",

```

Рис. 13.1 ❖ Фрагмент ответа API Inference, показывающий количество обученных моделей в кластере, а также информацию об одной из моделей

Другой фрагмент информации, предоставляемый API обученных моделей, – это `analyzed_fields`, словарь, который включает список включенных или исключенных полей. Рекомендуем проверить его содержимое еще раз, чтобы убедиться, что в обучающие данные были включены только те поля, которые вы намеревались использовать для обучения.

- Наш демонстрационный кластер содержит всего три обученные модели, поэтому объем информации, возвращаемой API, не так уж велик, но если вы работаете в кластере с десятками или сотнями моделей, бывает удобнее просматривать информацию об одной модели за раз, используя вызов API с полным названием модели. Обратите внимание: если вы следуете инструкциям и хотите выполнить вызов API в вашем конкретном экземпляре Kibana, вам нужно будет найти и подставить в команду `model_id` модели, расположенной в вашем кластере:

```
GET _ml/trained_models/breast-cancer-wisconsinclassification-1612270856116
```

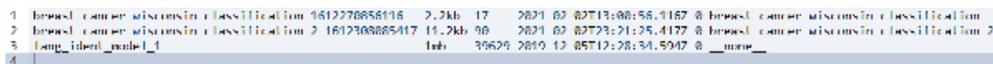
В качестве альтернативы можно использовать вызов API с подстановочным знаком:

```
GET _ml/trained_models/breast-cancer-wisconsinclassification-*
```

Менее подробное резюме доступно через точку _cat API. Вы можете использовать следующий вызов API, чтобы увидеть краткую сводку информации о доступных моделях:

```
GET _cat/ml/trained_models
```

Ответ, который мы получили от нашего кластера, показан на рис. 13.2. Как видите, в нем есть две модели, обученные на наборе данных рака молочной железы, а также третья модель с идентификатором lang_ident_model_1. Это модель идентификации языка, которая по умолчанию поставляется с Elasticsearch и может использоваться для определения языка, на котором написана строка текста. Позже мы расскажем, как работает эта модель идентификации языка и как ее использовать.



1	breast_cancer_wisconsin_classification	1612276858116	2.2kb	17	2021-02-02T13:00:56.116Z	breast_cancer_wisconsin_classification
2	breast_cancer_wisconsin_classification	1612308885417	11.2kb	96	2021-02-02T23:21:25.417Z	breast_cancer_wisconsin_classification
3	lang_ident_model_1		1mb	35629	2019-12-05T12:28:34.594Z	_lang_

Рис. 13.2 ❖ Ответ от _cat API

Теперь, когда вы научились находить и извлекать информацию об обученных моделях, доступных в вашем кластере, давайте более подробно рассмотрим еще одну мощную функцию API обученных моделей – экспорт модели из кластера Elasticsearch. Поскольку эта процедура устроена сложнее, чем предыдущая, мы посвятили ей следующий раздел.

Экспорт и импорт обученных моделей с помощью API и Python

Зачем вам экспортировать модель, обученную в Elasticsearch? Возможно, вы захотите экспортировать свою модель, чтобы сохранить ее на стороне, поделиться ею с коллегами либо импортировать ее позже в другой кластер Elasticsearch. Поскольку обучение моделей бывает весьма ресурсоемким, вы можете выделить один временный (и более мощный) кластер Elasticsearch для обучения, обучить и протестировать модель в этом кластере, а затем экспортировать модель из мощного кластера и импортировать ее в другой, меньший кластер для запуска и получения логического вывода, что является менее ресурсоемкой процедурой.

Чтобы экспортировать модель с помощью Python в соответствии с нашими инструкциями, вам потребуется Python версии 3.7 или новее и библиотека elasticsearch-py версии 7.10.1. Для получения подробных инструкций о том, как настроить среду Python и установить необходимые зависимости, ознакомьтесь с файлом README в репозитории GitHub по адресу: <https://>

github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter13. Все шаги и логика, необходимые для экспорта модели из кластера Elasticsearch, реализованы в скрипте Python `export_model.py` по адресу <https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter13>. В этом разделе мы рассмотрим каждый шаг этого скрипта, чтобы изучить действия, необходимые для экспорта модели. Мы надеемся, что этот обзор послужит руководством по использованию клиента `elasticsearch-py` для построения ваших собственных рабочих процессов машинного обучения на Python.



1. В основе почти всех скриптов Python, взаимодействующих с кластером Elasticsearch, лежит создание клиентского объекта `Elasticsearch`. Сначала необходимо импортировать класс клиентского объекта библиотеки следующим образом:

```
from elasticsearch import Elasticsearch
```

После импорта класса мы можем создать экземпляр объекта и назначить его переменной `es_client`:

```
es_client = Elasticsearch(es_url, http_auth=(ES_USERNAME, ES_PASSWORD))
```

Обратите внимание, что мы передаем в конструктор объекта переменную, которая содержит URL экземпляра `Elasticsearch`. Это может быть что-то вроде `localhost:9200`, если вы запускаете экземпляр `Elasticsearch` на локальном компьютере, или более длинный URL для облачных хостов. Кроме того, мы передаем две переменные, `ES_USERNAME` и `ES_PASSWORD`, которые содержат имя пользователя и пароль нашего экземпляра `Elasticsearch`. Это не требуется, если вы запускаете незащищенный кластер `Elasticsearch` локально в целях разработки, но имейте в виду, что запуск незащищенного кластера `Elasticsearch` в производственной среде чрезвычайно опасен.

2. Для взаимодействия с API машинного обучения, которые нам понадобятся для экспорта нашей модели из кластера, нам необходимо импортировать класс `MlClient`:

```
from elasticsearch.client import MlClient
```

Затем создадим его экземпляр:

```
ml_client = MlClient(es_client)
```



Завершив создание клиентов, мы можем приступить к экспорту нашей модели. Для этого воспользуемся методом `get_trained_models`. Документация по этому методу доступна по адресу <https://elasticsearch-py.readthedocs.io/en/v7.13.0/api.html#x-pack>, и мы советуем постоянно держать эту ссылку под рукой при работе с библиотекой на тот случай, если вам потребуется уточнить значение параметров конфигурации. В метод нужно передать три важных параметра: `model_id`, который указывает имя модели для экспорта, флаг `decompress_definition`, который

должен быть установлен в `False`, и флаг `include_model_definition`, который должен быть установлен в `True`:

```
model_id = breast-cancer-wisconsinclassification-1612270856116
compressed_model = ml.client.get_trained_models(model_id,
decompress_definition=False, include_model_definition=True, for_export=True)
```



Если вы запускаете этот код для кластера Elasticsearch версии 7.11 или новее, установите дополнительный параметр `exclude_generated` в `True`, чтобы убедиться, что ответ, который вы получаете от API, представлен в подходящем формате для импорта в другой кластер Elasticsearch. Если вы используете версию 7.10, применяйте `for_export`, как показано в примере.

После запуска вышеуказанного кода API вернет словарь Python, хранящийся в переменной `compressed_model` и представляющий информацию о нашей модели, а также сжатое определение, как показано во фрагменте на рис. 13.3. На самом деле ответ намного длиннее и не показан полностью для экономии места.

```
{'count': 1,
 'trained model configs': [{'compressed definition': 'H4sIAAAAAAAAA/92XbYvbMAzH3+9j+HUWJNmW5X6JfYbXhFzrtgE3KYmzB4777p
/6ZXL9YMSz0s/TN13LNC9m8/6uMwXuhZhtu0059Lshkwal356z6l+36e-rHPqxv6Ugr1515Y8pNnoxG3Ul2XYD9u+DNPY5f4+55cbMo1xv704mIbD5V
XT-4dUuvL5rOfzBcvfFZHDzqlfqn034RPaVkvG5rgkFwoyrphnGXpknMGZcT/dp7pb+dm5VhoPYmJz6ffehz6t0eauxMTjrrRchR07i493jV/P7RP
05KB6zwejbCMAsHlEga1j1nJb1lHdq2/rG7NK+X3PctoXs9HwqY5t6Vm9aw1F1NLty/d9jjUmdiYueb77TfVqz2oFL+rJezXQ08E5D66ygy+YEsV9N
jmgEpo0Yj1fQqsSzXBG1Y8sWgKx1L15IEULXgrW5QJ7o51ja21ILrLkBi3URrw2jbiEflFioHELkyzCGAF44WCYX1dfl7UYMrTYZAs+e8CPxLW60qP
ZCm4MYMnEstTBItauI8MoY+0K9tULchInrGyyhgT1aEwerBGBxvFvDmKtrVEtVLaWFXJ1XWUj44sqsbAGBhsZriVPzhG61nPDIH3RxFbLZPuRsvMKomv7T
jIVnupCNuga+HWMEILjHrdENIF61g72NwBBMabBqvE4D08KX0pSrxT6MkaFGICkgtSS94TDnA59Sd26LvnAktQ8HYaLDnt032gdapJ1+DtApdX85VHV+r
  'create time': 1612270856116,
  'created by': 'xpack',
  'description': '',
  'estimated heap memory usage bytes': 2272,
  'estimated operations': 17,
  'inference_config': {'classification': {'num_top_classes': 2,
    'num_top_feature importance values': 0,
    'prediction field type': 'number',
    'results field': 'Class prediction',
    'top_classes_results_field': 'top_classes'}}},
  'input': {'field names': ['Bare Nuclei',
    'Bland Chromatin',
    'Clump Thickness',
    'Marginal Adhesion',
    'Mitoses',
    'Normal Nucleoli',
    'Outlier',
    'Single Epithelial Cell Size',
    'Uniformity of Cell Shape',
    'Uniformity of Cell Size']},
  'license level': 'platinum'}
```

Рис. 13.3 ❖ Фрагмент словаря Python, который содержит сжатое определение модели, а также метаданные

- Получив определение модели, сохраненное в переменной `compressed_model` в нашем скрипте Python, мы можем преобразовать словарь в строку в формате JSON и записать ее в файл, который затем можно сохранить в системе контроля версий или импортировать в другой кластер Elasticsearch.
- Чтобы преобразовать словарь в формат JSON, мы должны импортировать встроенную библиотеку `json`

```
import json
```

После этого можем записать экспортированную модель в файл, путь к которому мы сохранили в переменной имени файла:

```
with open(filename, 'w') as handle:
    handle.write(json.dumps(compressed_model))
```

Все вышеперечисленные шаги реализованы в скрипте `export_model.py`, который доступен в репозитории GitHub по адресу https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter13/model_import_export.

Теперь, когда вы умеете экспортировать обученную модель из кластера Elasticsearch в файл, мы покажем, как импортировать модель из файла. Как и раньше, мы разбиваем логику скрипта на этапы, а полный файл скрипта `import_model.py` доступен в репозитории GitHub по адресу https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter13/model_import_export.

1. Многие из шагов этой процедуры похожи на сценарий экспорта, который мы только что прошли. В частности, для создания объектов `Elasticsearch` и `MlClient`, а также для анализа аргументов командной строки выполняются те же шаги, что и в предыдущем сценарии, поэтому мы не будем их подробно объяснять. Первым новым шагом является чтение файла модели и преобразование содержимого строки в словарь Python с использованием метода `loads` из встроенной библиотеки `json`:

```
With open(filename, 'r') as handle:
    model_definition = json.loads(handle.read())
```

2. Когда готовы сжатое определение модели и необходимые метаданные, загруженные в словарь Python, мы можем использовать метод `put_trained_model`, чтобы загрузить его в новый кластер.
3. Теперь можно перейти к экземпляру Kibana нового кластера и использовать API обученных моделей `Trained Models`, чтобы убедиться, что модель действительно была импортирована в ваш кластер.

Сейчас мы готовы перейти к созданию более сложных инфраструктур машинного обучения без моделей. Когда у вас есть обученная модель, возможности для модели практически безграничны – вы можете комбинировать модель с преобразованиями, использовать ее для дополнения данных во время загрузки и делать многое другое. Строительными блоками этой инфраструктуры являются обработчики вывода и конвейеры данных. Мы подробно рассмотрим каждый из этих компонентов в следующем разделе, чтобы подготовить вас к созданию собственной инфраструктуры машинного обучения.

ОБРАБОТЧИКИ ЛОГИЧЕСКОГО ВЫВОДА И КОНВЕЙЕРЫ ДАННЫХ

Допустим, у вас есть обученная модель. Что с ней делать дальше? В главах 11 и 12 мы отмечали, что одна из характерных особенностей моделей машинного обучения заключается в том, что они учатся на размеченном наборе

обучающих данных, а затем, в определенном смысле, кодируют знания, чтобы их можно было использовать для прогнозов, основанных на незнакомых точках данных. В классифицирующих моделях прогнозом является наиболее вероятная метка класса, а в регрессионных моделях – ожидаемое значение зависимой переменной. Этот процесс классификации или прогнозирования значения на основе ранее не встречавшихся данных называется *логическим выводом* (inference).

Как это работает на практике в Elastic Stack?

Существует множество различных архитектур, которые вы можете построить для использования логического вывода в Elastic Stack, но все они основаны на обработчиках логического вывода и конвейерах данных. Они станут предметами нашего внимания в этом разделе.

Конвейер данных (ingest pipeline) – это специальный компонент, который позволяет вам различными способами манипулировать данными и преобразовывать их, прежде чем они будут записаны в хранилище Elasticsearch. Конвейеры данных обычно состоят из различных *обработчиков* (processor), которые являются настраиваемыми задачами и отвечают за какой-то один тип манипуляции или преобразования загружаемых данных. Конвейеры могут состоять из нескольких обработчиков, которые последовательно обрабатывают каждый входящий документ данных по мере его загрузки (отсюда и название «конвейер»).

Например, типичная архитектура конвейера данных может включать в себя обработчик сценариев, который может выполнять сценарий Painless для каждого документа, поступающего в конвейер, за которым следует обработчик вывода, за которым следует другой обработчик сценариев. Для многих приложений машинного обучения, таких как те, которые мы рассмотрим чуть позже в этой главе, это идеальное место для реализации конструирования признаков, преобразования признаков в подходящий формат для использования моделью машинного обучения или для удаления ненужных полей, перед тем как документ будет загружен в Elasticsearch.

Существует множество встроенных обработчиков, которые можно комбинировать и настраивать для создания сложных конвейеров преобразования данных. Например, обработчик **GeoIP** добавляет географическую информацию об IP-адресах, обработчик скриптов позволяет пользователям писать собственный код Painless для выполнения вычислений и манипуляций с существующими полями документа, а обработчик CSV позволяет анализировать и извлекать данные из значений CSV для создания полей. Полный список обработчиков доступен в документации Elasticsearch по адресу <https://www.elastic.co/guide/en/elasticsearch/reference/master/processors.html>. Мы рекомендуем вам посмотреть, какие разновидности архитектуры данных вы можете построить с их помощью.

Для наших целей и с точки зрения изучения машинного обучения наиболее важным обработчиком для первоочередного изучения является *обработчик вывода* (inference processor). Когда документы проходят через этот обработчик, они снабжаются прогнозами модели машинного обучения, на которую ссылается конфигурация обработчика. Давайте посмотрим на практическом примере, как настроить собственный обработчик вывода и использовать его в конвейере данных.

В этом примере мы будем использовать вымышленный набор данных социальных сетей, который впервые рассмотрели в главе 9. На этот раз мы будем использовать модель определения языка, чтобы определить, на каком языке написан текст этих вымышленных сообщений сайта микроблогов. Приступим!

1. Если вы экспериментировали с API обученных моделей, который мы обсуждали в начале этой главы, то могли заметить, что даже если вы не обучили никакие модели в конкретном кластере Elasticsearch, вы все равно увидите одну модель с идентификатором `lang_ident_model_1`. Метаданные, связанные с этой моделью, возвращаемые API обученных моделей, показаны на рис. 13.4.

```
{
  "model_id" : "lang_ident_model_1",
  "created_by" : "_xpack",
  "version" : "7.6.0",
  "create_time" : 1575548914594,
  "estimated_heap_memory_usage_bytes" : 1053992,
  "estimated_operations" : 39629,
  "license_level" : "basic",
  "description" : "Model used for identifying language from arbitrary input text."
,
  "tags" : [
    "lang_ident",
    "prepackaged"
  ],
  "input" : {
    "field_names" : [
      "text"
    ]
  }
}
```

Рис. 13.4 ❖ Модель определения языка `lang_ident_model_1`

Модель по умолчанию поставляется с кластером Elasticsearch и может использоваться в обработчиках вывода и конвейерах данных, как и любая другая модель, которую вы можете обучить самостоятельно в кластере Elasticsearch.

2. Рассмотрим создание конфигурации конвейера данных с обработчиком логического вывода, который ссылается на эту модель определения языка. Напомним, что обработчики – это промежуточные узлы в конвейере приема, которые обрабатывают каждый документ, поступающий в конвейер, до того, как он будет записан в хранилище Elasticsearch. Несмотря на то что вы никогда не сможете использовать обработчик вывода в качестве автономного функционального блока в Elasticsearch – он всегда должен быть частью конвейера, – давайте сначала исследуем конфигурацию обработчика как такового, а затем посмотрим, как он вписывается в конвейер данных. В следующем фрагменте кода показана конфигурация обработчика вывода для нашего конвейера определения языка текстов:

```

{
  "inference": {
    "model_id": " lang_ident_model_1",
    "target_field": "text_language_prediction",
    "field_map": {
      "post": "text"
    },
    "inference_config": { "classification": {} }
  }
}

```

Рассмотрим наиболее важные параметры конфигурации и их значение. Полный справочник по всем доступным параметрам конфигурации для обработчиков вывода представлен в документации Elasticsearch по адресу <https://www.elastic.co/guide/en/elasticsearch/reference/master/inference-processor.html>.

Ключевой частью любого обработчика вывода является обученная модель, которая в данном случае будет использоваться для прогнозирования языка наших текстовых документов. Обработчик вывода узнает, какую модель он должен использовать для классификации входящих документов, через поле конфигурации `model_id`. В нашем случае идентификатор `model_id` (доступный в метаданных модели, которую мы можем просмотреть с помощью API обученных моделей) – это `lang_ident_model_1`.



Вы всегда можете узнать `model_id` вашей модели с помощью API обученных моделей Trained Models.

Следующий параметр конфигурации – `target_field`. Как только обработчик вывода обработает документ, проходящий через конвейер данных, он сгенерирует прогноз для документа на основе вывода (прогноза) обученной модели. Чтобы пользователь мог просмотреть этот прогноз, его следует записать в поле документа. Это поле мы можем, при желании, указать в параметре `target_field`. Если у вас нет особых предпочтений для имени поля, которое будет содержать прогнозы вашей модели, вы можете оставить это поле пустым, и ему будет присвоено имя по умолчанию.

Наконец, прежде чем мы перейдем к реальной конфигурации классификатора, давайте разберемся с параметром конфигурации `field_map`. Вспомните, как в главах 11 и 12 мы говорили о том, что в случае обучения с учителем модель обучается на размеченном наборе данных, который содержит признаки и зависимую переменную. Эти признаки должны нести в себе значимую информацию о зависимой переменной. Поскольку эти признаки важны для обучения модели, они также необходимы, когда мы используем обученную модель для выполнения прогнозов – или вывода – на ранее не встречавшихся точках данных. Однако, поскольку процесс обучения модели отделен от процесса вывода, может возникнуть ситуация, когда имена полей, которые мы использовали для признаков модели в наборе обучающих данных, не

совпадут с именами полей признаков в новых данных, для которых мы хотим получить прогноз.

В данном случае модель определения языка предварительно обучена для пользователя, поэтому возникает естественный вопрос: как конечные пользователи могут узнать названия полей, которые использовались в качестве признаков в модели, и, соответственно, какие названия следует дать полям, которые они собираются использовать для получения прогноза? Ответ на этот вопрос заключается в метаданных модели, возвращаемых API обученных моделей и показанных на рис. 13.4. В нижней части метаданных вы найдете поле ввода и блок конфигурации, вложенный под ним:

```
"input" : {
  "field_names" : [
    "text"
  ]
}
```

Имя поля, указанное ниже `field_names` в метаданных модели, говорит о том, что поле признака, используемого в модели, называется `text`, и наши новые документы должны иметь поле с таким именем, содержащее текст, к которому мы хотим применить определение языка. Поскольку обучение модели не связано с процессом вывода, вполне возможно, что имена полей, выбранные для признаков во время процесса обучения, отсутствуют в данных, которые мы хотим передать через наш обработчик вывода. В таком случае, если невозможно или нежелательно менять имена полей данных, мы можем использовать блок конфигурации `field_map` в нашем обработчике вывода для сопоставления имени поля в данных, которые мы будем использовать для вывода, с именем поля, которое ожидает наша модель. Ниже показан пример сопоставления имени поля сообщения, которое содержит текст из нашего вымышленного набора данных микроблогов, с именем текстового поля, которое ожидает модель:

```
"field_map": {
  "post": "text"
},
```

Мы подошли к завершающей части конфигурации – блоку `inference_config`. Параметры конфигурации для этого блока определяют, что мы используем – классификацию или регрессию. Поскольку в случае определения языка мы работаем с многоклассовой классификацией, мы выберем классификацию и оставим все остальные параметры конфигурации со значениями по умолчанию. Чуть позже в этом разделе мы более подробно рассмотрим доступные поля в `inference_config` и то, как их настройка определяет окончательный формат результатов.

3. Изучив параметры конфигурации обработчика логического вывода, перейдем к настройке конвейера данных. Его можно рассматривать как контейнер верхнего уровня, в котором будет размещаться наш обработчик вывода и, возможно, другие обработчики.

Конфигурация конвейера, содержащего настроенный нами обработчик вывода, выглядит следующим образом:

```
PUT _ingest/pipeline/language-identification-pipeline
{
  "description": "Pipeline for classifying language in
    social media posts",
  "processors": [
    {
      "inference": {
        "model_id": " lang_ident_model_1",
        "target_field": "text_language_prediction",
        "field_map": {
          "post": "text"
        }
      },
      "inference_config": { "classification": {} }
    }
  ]
}
```

- Основная часть конфигурации занята спецификациями обработчика логического вывода, которые мы подробно изучили ранее. Единственными заслуживающими внимания дополнительными параметрами в конфигурации конвейера являются имя конвейера, который является частью конечной точки REST API, и массив `processors` в теле конфигурации. В данном случае мы решили вызвать конвейер `language-identification-pipeline`. Массив `processors` содержит спецификации конфигурации для наших обработчиков. В нашем примере есть только один обработчик. Вы можете скопировать и вставить следующую конфигурацию в Kibana Dev Console или, в качестве альтернативы, использовать мастер **Ingest Node Pipelines**, который доступен на панели **Stack Management** в Kibana, как показано на рис. 13.5.

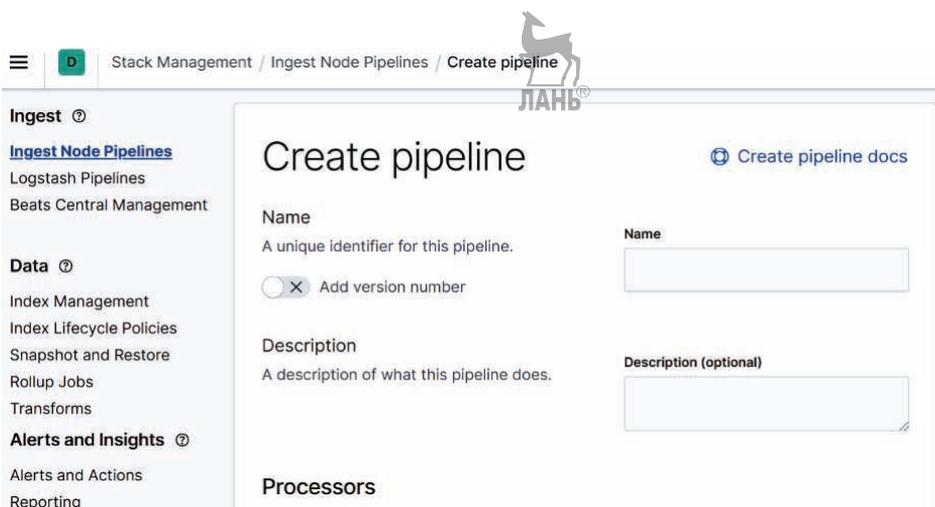


Рис. 13.5 ❖ Мастер создания конвейера

5. После того как мы настроили наш конвейер – либо с помощью консоли разработчика Dev Console, либо с помощью мастера, – он готов к использованию для определения языка сообщений нашей вымышленной платформы микроблогов в социальных сетях.

Хотя обычно мы будем использовать конвейер приема вместе с преобразованием или инструментом `racketbeat`, в этом случае мы собираемся загружать документы в хранилище с помощью консоли `Kibana Dev Console`, так как с ней пример получается более наглядным. Немного позже в данной главе мы рассмотрим более сложные и, следовательно, более реалистичные примеры.

Давайте пропустим через конвейер наш первый документ. Команда REST API, которая делает это, выглядит следующим образом:

```
POST social-media-feed-inference/_doc?pipeline=languageidentification-pipeline
{
  "username": "Sanna",
  "statistics": {
    "likes": 320,
    "shares": 8000
  },
  "timestamp": "2021-01-20T23:19:06",
  "post" : "Terveisiä Suomesta! Täällä olen talvilomalla!"
}
```

Мы отправляем запрос POST на обработку документа в теле запроса и передаем имя конвейера, который создали на предыдущих шагах, в качестве аргумента для необязательного параметра конвейера. В данном случае пользователь `Sanna` написал новый пост (очевидно, на финском языке). Давайте заглянем в хранилище `socialmedia-feed-inference`, чтобы узнать, как выглядит загруженный документ.

Если вы еще этого не сделали, создайте шаблон хранилища для `social-mediafeed-inference` и перейдите к приложению `Discover` в `Kibana`. Теперь хранилище `social-mediafeed-inference` содержит только один документ, который мы обработали с помощью вызова REST API, показанного ранее. Документ представлен на рис. 13.6.

Как видно на рис. 13.6, обработчик вывода добавил четыре новых поля наряду с исходными полями в документе. Все эти поля имеют префикс `text_language_prediction_model`, который мы настроили в нашей конфигурации обработчика вывода. В этих полях записаны `model_id` модели, использованной для прогнозирования, `predicted_value`, которое в данном случае содержит идентификатор языка, предсказанного моделью, `prediction_probability`, а также `prediction_score`. Последние два поля были рассмотрены в главе 11.

Как видите, в данном случае модель правильно определила, что исходный пост был написан на финском языке (`predicted_value: fi`).

t _id	mvVie3gByJotxG6S9hja
t _index	social-media-feed-inference
# _score	-
t _type	_doc
t post	Terveisiä Suomesta! Täällä olen talvilomalla!
# statistics.likes	320
# statistics.shares	8,000
t text_language_prediction.model_id	lang_ident_model_1
t text_language_prediction.predicted_value	fi
# text_language_prediction.prediction_probability	1
# text_language_prediction.prediction_score	1
📅 timestamp	Jan 20, 2021 @ 23:19:06.000
t username	Sanna

Рис. 13.6 ❖ Обработанный документ в хранилище social-mediafeed-inference

6. В предыдущем примере мы создали конфигурацию обработчика вывода и конвейера данных и приступили к обработке документов непосредственно в нашем хранилище через конвейер. Если мы хотим сначала увидеть несколько пробных прогонов нашего конвейера перед развертыванием в производстве, то можем использовать конечную точку `_simulate`:

```
POST _ingest/pipeline/language-identification-pipeline/_
simulate
{
  "docs": [
    {
      "_source": {
        "username": "Sanna",
        "statistics": {
          "likes": 320,
          "shares": 8000
        },
        "timestamp": "2021-01-20T23:19:06",
        "post": "Terveisiä Suomesta! Täällä olen
talvilomalla!"
      }
    }
  ]
}
```

Ответ, возвращаемый API для этого вызова, содержит прогнозы модели, которые вы можете видеть в следующем фрагменте кода:

```

{
  "doc" : {
    "_index" : "_index",
    "_type" : "_doc",
    "_id" : "_id",
    "_source" : {
      "post" : "Terveisiä Suomesta! Täällä olen
talvilomalla!",
      "text_language_prediction" : {
        "prediction_score" : 0.9999995958245499,
        "model_id" : "lang_ident_model_1",
        "prediction_probability" :
0.9999995958245499,
        "predicted_value" : "fi"
      },
      "username" : "Sanna",
      "statistics" : {
        "shares" : 8000,
        "likes" : 320
      },
      "timestamp" : "2021-01-20T23:19:06"
    },
    "_ingest" : {
      "timestamp" : "2021-03-29T01:35:07.492629377Z"
    }
  }
}
]
}

```



- Наконец, также можно использовать пользовательский интерфейс конвейера данных для тестирования документов перед развертыванием в производстве, чтобы убедиться, что все работает должным образом. К сожалению, в пользовательском интерфейсе эту процедуру можно выполнить только во время создания нового конвейера, но невозможно применить к существующему, поэтому для демонстрационных целей вы можете использовать мастер, чтобы начать создание клона конвейера `language-identification-pipeline`, который мы создали ранее. Затем справа от селектора **Processors** на странице мастера, как показано на рис. 13.7, найдите строку **Test pipeline** (Проверить конвейер) и нажмите ссылку **Add documents** (Добавить документы).

В правой части мастера откроется меню, которое позволяет вам либо добавить документ из хранилища, либо вручную указать его в текстовом поле. В данном случае мы собираемся вручную добавить наш тестовый документ для финского языка в текстовое поле, как показано на рис. 13.8.

После того как вы подготовили свои документы, нажмите кнопку **Run the pipeline** (Запустить конвейер), и вы должны увидеть предварительный просмотр ваших документов, после того как они прошли через конвейер вывода.

Processors  Import processorsUse processors to transform data before indexing. [Learn more.](#) **Test pipeline:** [Add documents](#)**Рис. 13.7** ❖ Мастер создания конвейера данных Create Ingest pipeline

Test pipeline

Documents

[Clear all](#)

```
[
  {
    "_source": {
      "username": "Sanna",
      "statistics": {
        "likes": 320,
        "shares": 8000
      },
      "timestamp": "2021-01-20T23:19:06",
      "post": "Terveisiä Suomesta! Täällä olen talvilomalla!"
    }
  }
]
```

Use JSON format: `[{"_index": "index", "_id": "id", "_source": {"foo": "bar"}}]`[▶ Run the pipeline](#)**Рис. 13.8** ❖ Пример документа в конвейере

Обработка отсутствующих или поврежденных данных в конвейерах

Многие (а может быть, и большинство) реальные приложения не имеют безупречных наборов данных. Вместо этого данные могут отсутствовать, иметь неправильную разметку и, возможно, даже быть повреждены. Мы должны разобраться, что происходит в таких случаях с обработчиками логического вывода, чтобы вы могли распознать и сгладить эти проблемы в ваших собственных конвейерах.

1. Продолжим работу с вымышленной платформой микроблогов, которую мы использовали в качестве нашего примера ранее, и предполо-

жим, что из-за ошибки в конфигурации мы переименовали поле `post`, которое содержит текстовую строку для определения языка, в `post_text`, как показано ниже:

```
POST social-media-feed-inference/_doc?pipeline=languageidentification-pipeline
{
  "username": "Sanna",
  "statistics": {
    "likes": 320,
    "shares": 8000
  },
  "timestamp": "2021-01-20T23:19:06",
  "post_text": "Terveisiä Suomesta! Täällä olen talvilomalla!"
}
```

Что произойдет, когда мы пропустим этот текст через конвейер `language-identification-pipeline`? Давайте выполним вызов REST API, показанный ранее, а затем посмотрим документ на вкладке **Discover**, как мы это делали раньше.

2. Как видно из содержимого документа, показанного на рис. 13.9, модель не смогла сделать правильный прогноз относительно языка, на котором был написан текст в поле `post_text`.

<code>r _id</code>	<code>m_UbfXgByJotxG6Smxgu</code>
<code>r _index</code>	<code>social-media-feed-inference</code>
<code># _score</code>	<code>-</code>
<code>r _type</code>	<code>_doc</code>
<code>post_text</code>	<code>Terveisiä Suomesta! Täällä olen talvilomalla!</code>
<code># statistics.likes</code>	<code>320</code>
<code># statistics.shares</code>	<code>8,000</code>
<code>r text_language_prediction.model_id</code>	<code>lang_ident_model_1</code>
<code>text_language_prediction.warning</code>	<code>Model [lang_ident_model_1] could not be inferred as all fields were missing</code>
<code>timestamp</code>	<code>Jan 20, 2021 @ 23:19:06.000</code>
<code>r username</code>	<code>Sanna</code>

Рис. 13.9 ❖ Предупреждающее сообщение в загруженном документе

Реальные наборы данных часто бывают неупорядоченными и содержат недостающие или поврежденные данные, поэтому следите за этим сообщением, чтобы выявить и исправить потенциальные ошибки в настройке вывода!

Первым шагом к устранению причины появления этого сообщения является сравнение полей в документах, которые вы пытаетесь пропустить через конвейер данных, с обучающими полями, хранящимися в метаданных модели. Перечитайте раздел «Экспорт и импорт обученных моделей с помощью

API и Python» в этой главе в качестве руководства о том, как просматривать метаданные модели.

Получение развернутой информации о прогнозах

В предыдущих разделах мы настроили наши обработчики логического вывода таким образом, чтобы документы, поступающие в обработчик, содержали только четыре поля: прогнозируемая метка класса документа, вероятность прогноза, оценка прогноза и идентификатор модели. Но как узнать, что происходит в тех случаях, когда модель делает неверный прогноз, или вы хотите получить дополнительную информацию о вероятностях отнесения к другим потенциальным классам? Это может быть полезно для отладки.

Как настроить обработчик вывода, чтобы он предоставлял нам больше информации? Далее мы рассмотрим возможный вариант конфигурации обработчика.

1. Давайте начнем с того, что вернемся к конфигурации обработчика вывода, которую мы видели в предыдущем разделе, и более подробно разберем блок конфигурации `inference_config`, который мы оставили пустым. В этом случае, поскольку мы хотим увидеть более подробную разбивку различных вероятностей, нам следует добавить в блок конфигурации поле `num_top_classes`. Этот параметр конфигурации определяет количество классов, для которых записываются вероятности. Например, если мы запишем в это поле значение 3, каждый документ будет содержать вероятности для наиболее вероятных классов, к которым он мог бы принадлежать:

```
PUT _ingest/pipeline/language-identification-pipeline-v2
{
  "description": "Pipeline for classifying language in social media posts",
  "processors": [
    {
      "inference": {
        "model_id": " lang_ident_model_1",
        "target_field": "text_language_prediction",
        "field_map": {
          "post": "text"
        },
        "inference_config": { "classification": {"num_top_
classes": 3} }
      }
    }
  ]
}
```

2. Далее загрузим документ через этот новый конвейер `language-identifier-pipeline-v2`, используя следующий вызов REST API:

```
POST social-media-feed-inference/_doc?pipeline=language-identification-pipeline-v2
{
```

```

"username": "Sanna",
  "statistics": {
    "likes": 320,
    "shares": 8000
  },
  "timestamp": "2021-01-20T23:19:06",
  "post" : "Terveisiä Suomesta! Täällä olen
talvilomalla!"
}

```

Мы увидим, что в результате обработчик логических выводов выдает подробную разбивку возможных языков (или классов, если мы используем терминологию классификации), к которым принадлежит сообщение, как показано на рис. 13.10. Возможными кандидатами являются финский язык, обозначаемый ключевым словом `fi`, шведский, обозначаемый ключевым словом `sv`, и эстонский, обозначаемый ключевым словом `eo`.

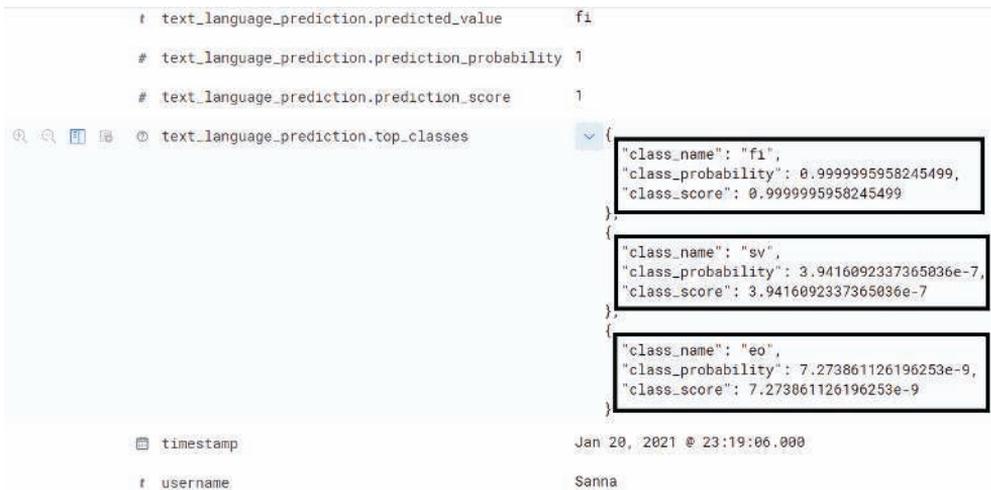


Рис. 13.10 ❖ Подробная разбивка на потенциальные классы, к которым может принадлежать данный документ с указанием соответствующих вероятностей

Теперь перейдем к импорту моделей с помощью `eLand`.

ИМПОРТ ВНЕШНИХ МОДЕЛЕЙ С ПОМОЩЬЮ `eLand`

Предположим, у вас уже есть модель, обученная с использованием какого-то стороннего фреймворка. Можно ли использовать шаги, которые мы обсуждали в предыдущем разделе, для развертывания ваших собственных моделей, обученных извне Elastic ML? Ответ – да, но с некоторыми ограничениями.

В этом разделе мы рассмотрим, как использовать библиотеку eLand вместе с scikit-learn, еще одной библиотекой машинного обучения, для создания и обучения внешних моделей и импорта их в Elasticsearch для развертывания и выполнения.

Кратко о поддержке внешних моделей в eLand

К сожалению, Elastic Stack пока не поддерживает импорт внешних моделей в формате какой-либо сторонней библиотеки (хотя это может произойти в будущем!). Вместо этого документация по eLand (https://eland.readthedocs.io/en/7.10.1b1/reference/api/eland.ml.MLModel.import_model.html#eland.ml.MLModel.import_model) предлагает список сторонних библиотек, которые производят модели, совместимые с Elastic ML. В настоящее время список поддерживаемых типов моделей выглядит следующим образом:

- sklearn.tree.DecisionTreeClassifier;
- sklearn.tree.DecisionTreeRegressor;
- sklearn.ensemble.RandomForestRegressor;
- sklearn.ensemble.RandomForestClassifier;
- lightgbm.LGBMRegressor;
- lightgbm.LGBMClassifier;
- xgboost.XGBClassifier;
- xgboost.XGBRegressor.

! Имейте в виду, что существуют некоторые дополнительные ограничения для моделей, сгенерированных с помощью указанных выше библиотек, которые относятся к типу выбранной целевой функции или типу кодировки, которая должна применяться к признакам, поэтому обязательно проверьте документацию eLand на предмет получения самой свежей информации о поддерживаемых сторонних моделях.

Обучение DecisionTreeClassifier и импорт в Elasticsearch с помощью eLand

Теперь, когда вы знаете, как провести предварительную подготовку, давайте приступим к делу и рассмотрим пример обучения внешней модели с помощью библиотеки scikit-learn. Все примеры кода, используемые в этом пошаговом руководстве, будут доступны в блокноте Jupyter в репозитории GitHub книги по адресу https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter13/external_models.

1. Первым шагом в нашем примере импорта внешней модели в Elasticsearch будет получение обучающих данных и их использование для обучения модели дерева решений. Библиотека scikit-learn имеет отличную коллекцию встроенных наборов данных, которые можно использовать для обучения и быстрого создания прототипов. Мы будем использовать встроенный набор данных по раку молочной железы

в штате Висконсин (который является разновидностью уже знакомого вам набора данных).

Но прежде чем мы начнем, давайте убедимся, что мы импортировали все необходимые функции и библиотеки в наш скрипт Python (или блокнот Jupyter):

```
# импорт варианта набора данных, уже знакомого нам по предыдущим главам
from sklearn.datasets import load_breast_cancer

# импорт функции обучения DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier

# импорт функции для разделения набора на обучающую и тестовую части
from sklearn.model_selection import train_test_split
```

2. Далее загрузим набор данных, вызвав функцию `load_breast_cancer`:

```
# загрузим набор данных и сохраним точки данных в переменной X, а метки классов
# в переменной y
X, y = load_breast_cancer(return_X_y=True)
```

Обратите внимание, что эта функция возвращает два значения, которые мы сохраняем в переменных `X` и `y`. Способ, которым `Elasticsearch` организует свои обучающие данные, отличается от соглашений в `scikit-learn`. В `Elasticsearch` наши обучающие данные хранились в одном хранилище. Каждый документ в хранилище соответствует одной точке данных и представляет собой комбинацию полей, представляющих признаки, и поля, представляющего зависимую переменную (напомним, что в главах 11 и 12 говорилось о зависимых переменных и почему они важны при обучении с учителем).

В отличие от подхода `Elasticsearch`, `scikit-learn` представляет каждую точку данных с помощью вектора, который содержит все значения признаков. Эти векторы составляют матрицу, хранящуюся в переменной `X`. Мы можем использовать синтаксис среза данных Python, чтобы увидеть, как будет выглядеть образец данных. Пример отображения данных показан на рис. 13.11.

```
In [6]: # in contrast with Elasticsearch, features and labels are stored in separate variables not the same document
# a sample entry in the matrix represented by variable X
X[0]
```

```
Out[6]: array([1.799e+01, 1.038e+01, 1.228e+02, 1.001e+03, 1.184e-01, 2.776e-01,
 3.001e-01, 1.471e-01, 2.419e-01, 7.871e-02, 1.095e+00, 9.053e-01,
 8.589e+00, 1.534e+02, 6.399e-03, 4.904e-02, 5.373e-02, 1.587e-02,
 3.003e-02, 6.193e-03, 2.538e+01, 1.733e+01, 1.846e+02, 2.019e+03,
 1.622e-01, 6.656e-01, 7.119e-01, 2.654e-01, 4.601e-01, 1.189e-01])
```

Рис. 13.11 ❖ Точка данных представлена как вектор значений поля

Зависимые переменные хранятся в отдельной переменной `y`. Подобно нашему предыдущему примеру, мы можем использовать синтаксис

среза Python, чтобы увидеть, к какому классу принадлежит точка данных, значения признаков (или вектор признаков) которой изображены выше, как показано на рис. 13.12.

```
In [7]: # a sample entry in the matrix represented by the variable y
        y[0]
Out[7]: 0
```

Рис. 13.12 ❖ Метка класса для первой точки данных – 0

3. Когда мы импортировали наш набор данных и убедились, что он выглядит приемлемым, можно перейти к следующему шагу – обучению нашей модели дерева решений. Elasticsearch автоматически разделяет данные на обучающий и тестовый наборы, но в `scikit-learn` мы должны выполнить этот шаг вручную. Хотя в данном случае это не обязательно, поскольку мы не слишком заинтересованы в систематических измерениях качества нашей модели, мы все равно продемонстрируем эту операцию в качестве примера для заинтересованных читателей, которым нужно будет делать это в собственных проектах:

```
# В отличие от Elasticsearch, в scikit-learn мы должны разделить данные
# на обучающий и тестовый наборы вручную при помощи функции train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=12345)
```

В этом фрагменте кода мы передаем переменную, содержащую векторы признаков для каждой точки данных, и переменную, содержащую значения зависимых переменных или метки классов, в функцию `scikit-learn train_test_split`, и функция возвращает векторы признаков и зависимые переменные, которые принадлежат обучающему набору (`X_train` и `y_train`) и тестовому набору (`X_test`, `y_test`) соответственно.

4. Теперь, когда у нас есть набор данных для обучения и тестирования, мы можем перейти к обучению классификатора дерева решений. Первым шагом является создание экземпляра класса `DecisionTreeClassifier` и согласование его с векторами функций и метками классов из набора обучающих данных:

```
# создаем классификатор на основе дерева решений
dec_tree = DecisionTreeClassifier(random_state=12345).fit(X_train, y_train)
```

На обученную модель ссылается переменная `dec_tree`. Это переменная, которую мы сериализуем и загрузим в Elasticsearch с помощью `eland` чуть позже. Однако сначала давайте проведем быструю проверку нашей модели, попросив ее классифицировать точку из тестового набора данных (которую модель не видела ранее на этапе обучения), как показано на рис. 13.13.

```
In [10]: # we can now use this trained model to predict which class the datapoints in our X_test set belong to
# for example,
dec_tree.predict([X_test[0]])

Out[10]: array([1])
```

Рис. 13.13 ❖ Прогнозы на основе обученной модели дерева решений

Модель предсказывает, что первая точка данных в нашем наборе тестовых данных принадлежит классу 1. Мы можем удостовериться, соответствует ли это фактической метке точки данных, проверив первый элемент в переменной `y_test`, как показано на рис. 13.14.

```
In [11]: # Let's check to see if this matches the actual class label
y_test[0]

Out[11]: 1
```

Рис. 13.14 ❖ Значение зависимой переменной для первой точки данных тестового набора

В данном случае прогноз модели соответствует фактической метке набора данных.

- Теперь подготовимся к загрузке этой модели в кластер Elasticsearch с помощью `eland`. Во-первых, мы должны импортировать требуемый класс `MLModel`, как показано в следующем фрагменте кода:

```
# импорт необходимого класса eland
from eland.ml import MLModel
```

Когда класс импортирован и доступен в скрипте или блокноте Jupyter, можно перейти к следующему шагу, который извлекает имена признаков из исходного набора данных `scikit-learn`. Необходимые для этого шаги показаны в следующем фрагменте кода:

```
data = load_breast_cancer()
feature_names = data.feature_names
```

Заинтересованный читатель может вывести на печать переменную `feature_names` (или взглянуть на блокнот Jupyter, прилагаемый к этому пошаговому руководству), чтобы увидеть, какие признаки включены. В целях экономии места мы опустим список имен признаков.

- Наконец, мы вызываем метод `import_model` в классе `MLModel`, как показано в следующем фрагменте кода:

```
es_model = MLModel.import_model(
    es_client,
    model_id=model_id,
    model=dec_tree,
    feature_names=list(feature_names),
    es_if_exists='replace'
)
```



Как видите, этот метод требует большого количества параметров. Первый параметр `es_client` – это экземпляр клиентского объекта `Elasticsearch`, который указывает, как подключиться к нашему кластеру `Elasticsearch`. Более подробно этот вопрос раскрыт в разделе «Поиск, импорт и экспорт обученных моделей с помощью API» в данной главе. Второй параметр – это `model_id`, идентификатор, который будет использоваться для идентификации модели после ее загрузки в кластер `Elasticsearch`. В нашем случае мы установили переменную `model_id`, как показано в следующем фрагменте кода:

```
model_id = "external-model_breast-cancer-decision-tree"
```

Разумеется, можно указать этот идентификатор по вашему выбору. Наконец, мы передаем имя переменной, которая содержит ссылку на нашу обученную модель, `dec_tree`, список имен признаков, которые мы получили из исходного набора данных, и устанавливаем флаг `es_if_exists` равным `'replace'` – это означает, что если мы запустим фрагмент кода более одного раза, существующая модель с тем же `model_id` будет перезаписана. Бывают случаи, когда это нежелательно, но сейчас, поскольку мы создаем прототип, лучше перезаписывать модель.

7. После выполнения команды из предыдущего раздела мы можем использовать API обученных моделей, чтобы проверить, была ли эта модель успешно импортирована в наш кластер. Для этого запустим следующую команду:

```
GET _ml/trained_models/external-model_breast-cancerdecision-tree
```

Как видно из возвращенного ответа API, наша модель действительно была успешно импортирована в кластер и теперь готова к использованию в конвейере данных.



Все примеры кода, использованные на рисунках, доступны в блокноте Jupyter, расположенном в репозитории GitHub книги по адресу https://github.com/PacktPublishing/Machine-Learning-with-Elastic-Stack-Second-Edition/tree/main/Chapter13/external_models/importing-external-models-into-es-using-eland.ipynb.

ЗАКЛЮЧЕНИЕ

В этой главе мы рассмотрели различные варианты использования моделей, обученных в среде `Elasticsearch` и внешних библиотеках, таких как `scikit-learn`. Мы узнали об API обученных моделей, который полезен при работе с обученными моделями контролируемого обучения в кластере `Elasticsearch`, а также о том, как использовать эти модели для прогнозирования ранее не встречавшихся точек данных с помощью обработчиков вывода и конвейеров данных. В приложении после этой главы мы дадим несколько советов и продемонстрируем приемы, которые упростят работу со стеком `Elastic ML`.



Приложение

Советы по обнаружению аномалий

Завершая работу над этой книгой, мы подумали о том, что у нас осталось множество кратких объяснений, примеров и советов, которые не совсем вписывались в другие главы. Поэтому мы решили разместить их в этом приложении. Наслаждайтесь попури из полезных советов и примеров!

В приложении будут рассмотрены следующие темы:

- роль факторов влияния в разделенных и неразделенных заданиях;
- использование односторонних функций;
- игнорирование определенных периодов времени;
- использование настраиваемых правил и фильтров;
- соображения относительно пропускной способности заданий;
- о вреде излишней сложности сценариев использования;
- использование обнаружения аномалий в полях реального времени.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

В этом приложении мы будем использовать Elastic Stack версии 7.12.

Роль факторов влияния в разделенных и неразделенных заданиях

Возможно, у вас уже возник вопрос, нужно ли разделять анализ по полю или просто надеяться, что использование факторов влияния даст желаемый эффект идентификации «нарушителя».

Вспомните, в чем заключается разница между назначением факторов влияния и назначением разделения задания. Определенный объект идентифицируется заданием по обнаружению аномалий как фактор влияния, если он внес значительный вклад в существование аномалии. Процесс поиска и выявления факторов влияния совершенно не зависит от того, разделено задание или нет. Объект может считаться влияющим на аномалию только в том случае, если эта аномалия обнаружена. Если аномалии не обнаружено, нет необходимости выяснять, существуют ли факторы влияния. Однако задание может обнаружить или не обнаружить аномалию в зависимости от того, разделено ли задание на несколько временных рядов. При разделении задания вы строите модель (создавая отдельный анализ) для каждого объекта поля, которое решили разделить.

Давайте возьмем в качестве примера один из любимых демонстрационных наборов данных команды разработчиков Elastic ML, называемый `faqquote` (доступен в репозитории GitHub для этой книги в виде файла с названием `faqquote-2021.csv` и легко загружается через загрузчик файлов Elastic ML в визуализаторе данных). Этот набор данных был получен от реального клиента, который запустил приложение туристического портала. Журнал доступа к приложению записывал, сколько раз было вызвано промежуточное программное обеспечение, когда оно обращалось к сторонней авиакомпании с запросом цен на авиабилеты. Документы JSON выглядят следующим образом:

```
{
  "@timestamp": "2021-02-11T23:59:54.000Z",
  "responsetime": 251.573,
  "airline": "FFT"
}
```

Количество событий в единицу времени соответствует количеству сделанных запросов, а поле `responsetime` – это время ответа на конкретный запрос к веб-службе расчета стоимости билета этой авиакомпании.

Мы рассмотрим следующие случаи.

- **Случай 1.** Анализ с подсчетом по оси времени, но без разделения по авиакомпаниям, а с использованием авиакомпании в качестве фактора влияния. Вы можете сделать это с помощью мастера `Multi-metric`, настроенного, как показано на рис. П.1.

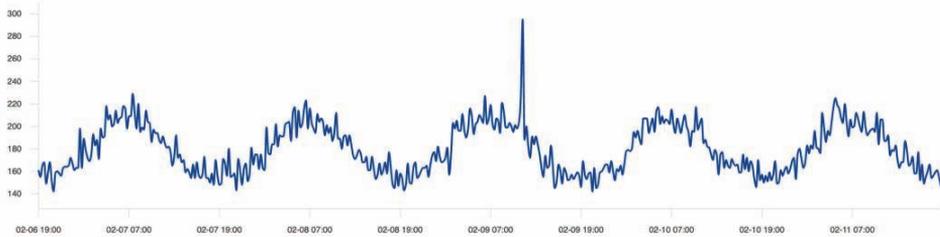
Create job: Multi-metric

Using index pattern farequote



Pick fields

Count(Event rate)



Add metric

Split field

Select a field to partition analysis by. Each value of this field will be modeled independently individually.

Split field

Influencers

Select which categorical fields have influence on the results. Who/what might you 'blame' for an anomaly? Recommend 1-3 influencers.

Influencers

Bucket span

Set the interval for time series analysis, typically between 15m to 1h.

Bucket span

Sparse data

Select if you wish to ignore empty buckets from being considered anomalous. Available for count and sum analysis.

Sparse data

 Sparse data

Рис. П.1 ❖ Задание без разделения, но с факторами влияния

После запуска анализа всплеск, видимый в середине данных (как показано на экране предварительного просмотра конфигурации задания на рис. П.1), действительно помечен как аномальный со скромной оценкой 27, а авиакомпания AAL была указана как фактор влияния (рис. П.2).

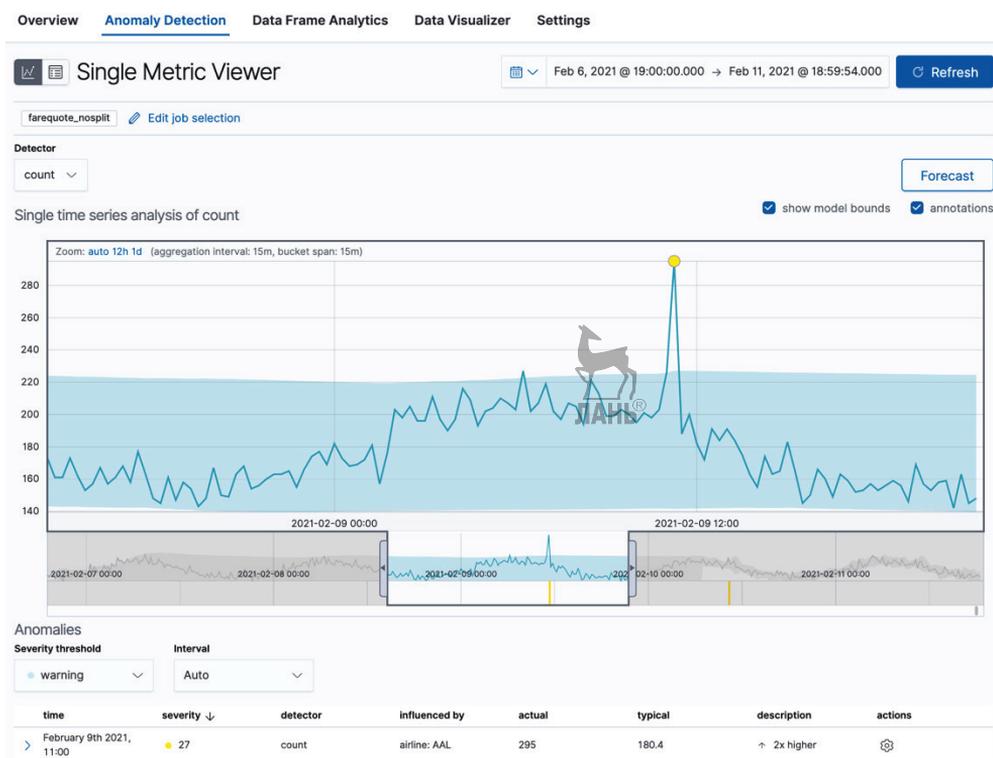


Рис. П.2 ❖ Задание без разделения, найден фактор влияния

- Давайте сравним этот результат с тем, что увидим в следующем случае.
- **Случай 2.** Анализ с подсчетом по оси времени, разделение по авиакомпаниям и использование авиакомпании в качестве фактора влияния. Если мы повторим конфигурацию, показанную на рис. П.1, но на этот раз выберем разделение по полю `airline` (установив `partition_field_name:airline`), то обязательно увидим, что авиакомпания AAL по-прежнему является наиболее необычной и что показатель аномалии для нее намного выше, чем в случае 1 (рис. П.3).

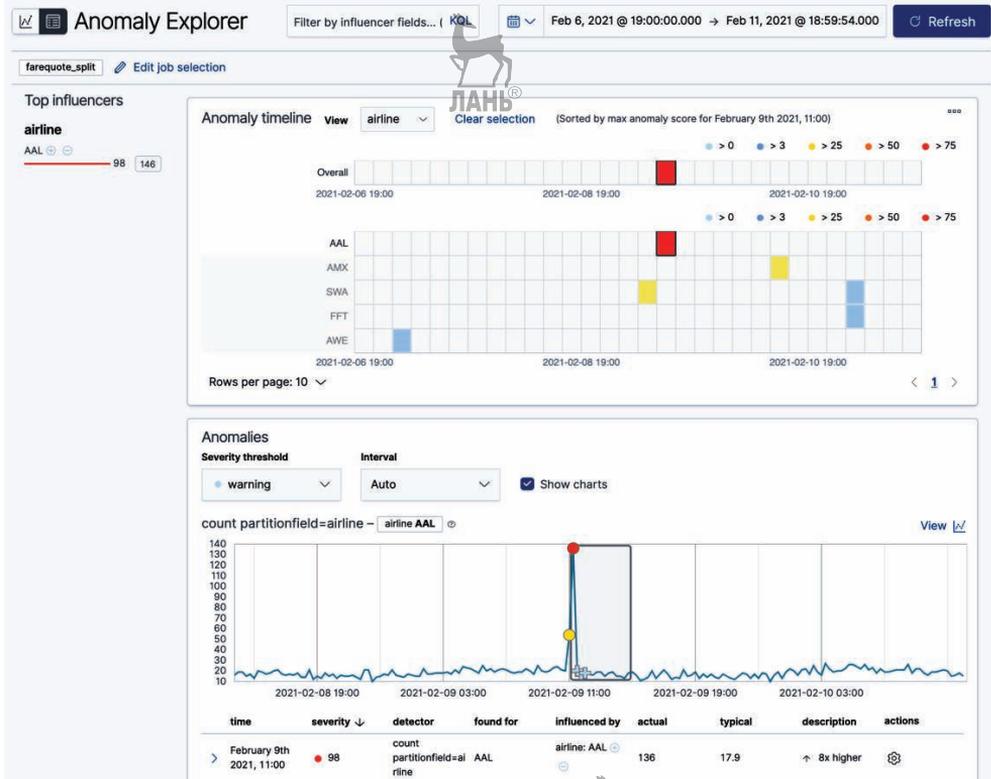


Рис. П.3 ❖ Задание с разделением по полю `airline`, найден фактор влияния

Другими словами, аномальное поведение AAL стало намного более заметным, когда задание было разделено на моделирование каждой авиакомпании по отдельности. В противном случае аномальное поведение AAL было несколько замаскировано за счет суммирования показателей разных авиакомпаний. Разница между разделением и отсутствием деления становится еще более заметной, когда мы смотрим на результаты анализа поля `responsetime` в следующих двух случаях.

- **Случай 3.** Анализ среднего значения поля `responsetime` с разделением по полю `airline`.

Здесь мы видим, что AAL также является самой необычной авиакомпанией с точки зрения анализа поля `responsetime` (рис. П.4).

Теперь сравним этот результат со следующим случаем, в котором мы не будем разделять задание.

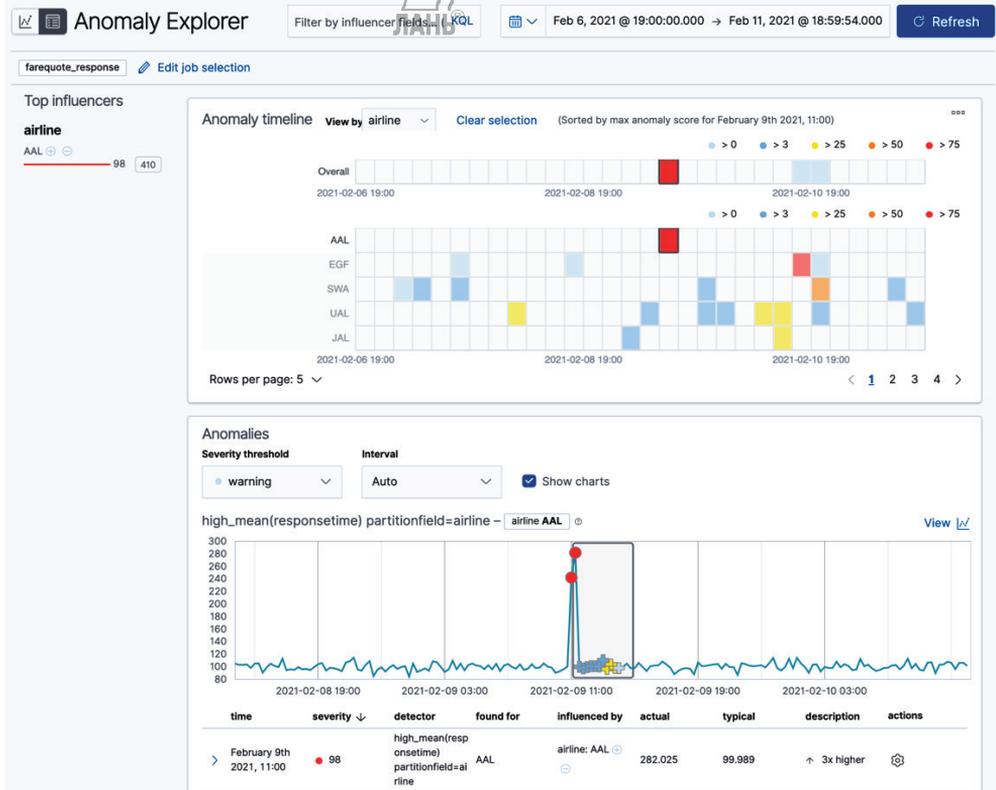


Рис. П.4 ❖ Результат анализа поля responsetime с разделением по airline, найден фактор влияния

- **Случай 4.** Анализ среднего значения поля responsetime без разделения, но продолжаем рассматривать поле airline в качестве фактора влияния.

Результаты для этого варианта показаны на рис. П.5.

Как видите, авиакомпания, которая, как мы знаем, является самой необычной (AAL), больше не найдена. В этом случае все времена отклика всех авиакомпаний усредняются для каждого диапазона периодов, потому что задание не разделено. Теперь видна другая наиболее заметная аномалия (хотя это относительно незначительное отклонение от нормы), и считается, что на нее влияет авиакомпания NKS. Однако это может ввести в заблуждение. Видите ли, авиакомпания NKS имеет очень стабильное время отклика в течение этого периода, но при этом ее нормальный рабочий диапазон намного выше, чем у остальной группы (рис. П.6).

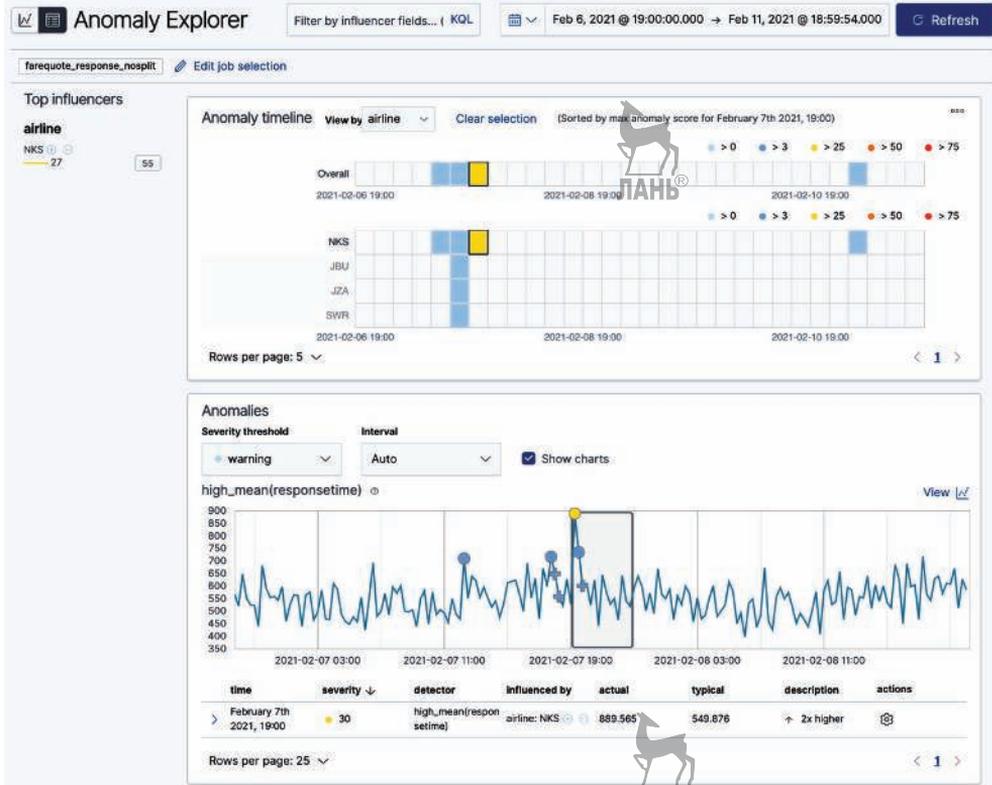


Рис. П.5 ❖ Результат анализа поля responsetime без разделения по airline, найден фактор влияния

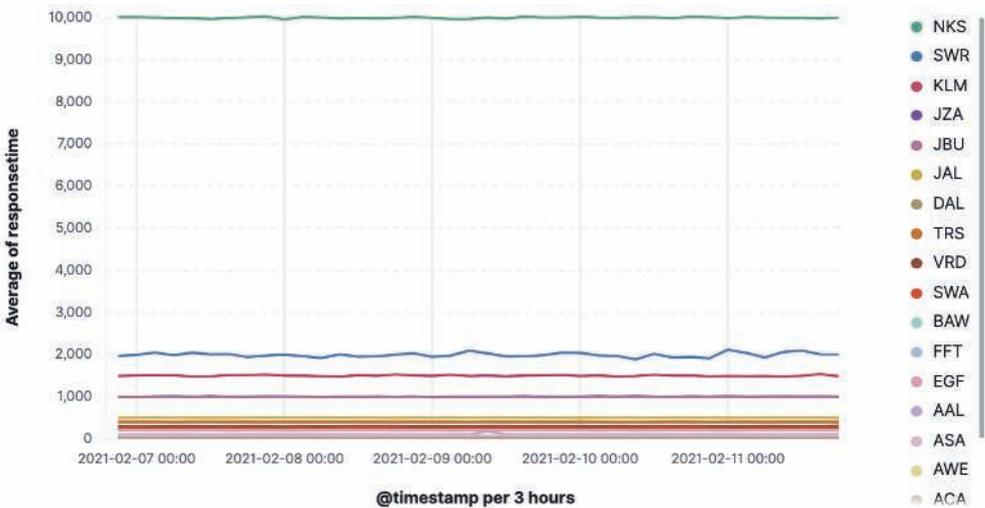


Рис. П.6 ❖ Среднее время отклика каждой авиакомпании

Получается, что вклад NKS в общее время отклика всех авиакомпаний больше, чем у других. Естественно, модель считает NKS наиболее заметным фактором влияния.

Вывод из этого примера прост: вам следует быть внимательными, если вы просто полагаетесь на обнаружение факторов влияния в поиске необычных объектов в наборе данных. Было бы разумнее смоделировать объекты по отдельности!

ИСПОЛЬЗОВАНИЕ ОДНОСТОРОННИХ ФУНКЦИЙ

Наверняка для вас очевидна полезность односторонних функций, таких как `low_count` и `high_mean`, позволяющих обнаруживать только аномально высокие или аномально низкие значения. Они удобны, когда вас беспокоит лишь падение доходов или всплеск времени отклика.

Если же вас интересуют отклонения в обоих направлениях, большинство пользователей склонны использовать только обычную функцию (например, `count` или `mean`). Однако для некоторых наборов данных было бы лучше применять обнаружение пиков и провалов как два отдельных детектора.

Вы можете спросить: почему это так и при каких условиях?

Упомянутый подход имеет смысл, когда динамический диапазон возможных отклонений асимметричен. Другими словами, величина потенциальных всплесков в данных многократно превышает величину потенциальных провалов, возможно потому, что количество или сумма чего-либо не может быть меньше нуля. Давайте рассмотрим снимок экрана на рис. П.7.

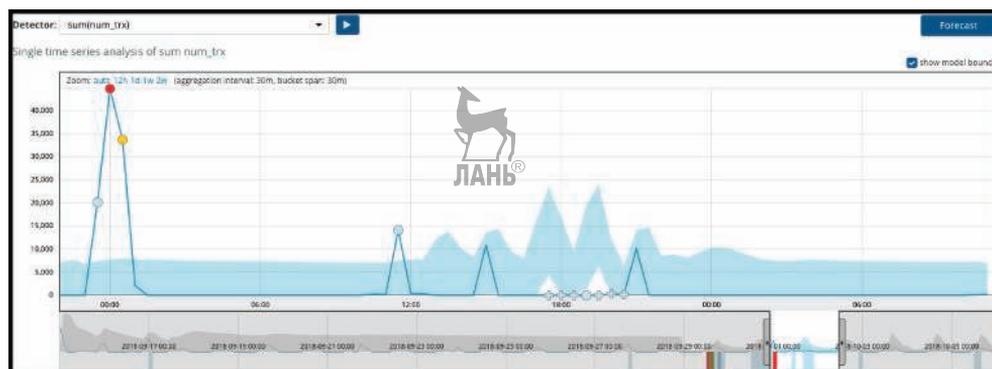


Рис. П.7 ❖ Анализ с использованием двусторонней функции `sum`

Здесь двусторонняя функция `sum` правильно идентифицирует большой всплеск с критической аномалией слева, но отсутствие ожидаемых двойных выпуклостей в середине вызывает только предупреждения об отсутствии ожидаемых аномалий. И снова это связано с тем, что в случае двусторонней функции процесс нормализации ранжирует все аномалии вместе. Величи-

на (и, следовательно, маловероятность) всплеска намного больше, чем отсутствие данных около 18:00 по шкале времени, поэтому оценки аномалии являются относительными.

Однако если набор данных проанализировать двумя отдельными детекторами с использованием расширенного задания, то есть `low_sum(num_trx)` и `high_sum(num_trx)`, то результаты будут совсем другими. Результат анализа пиков показан на рис. П.8.

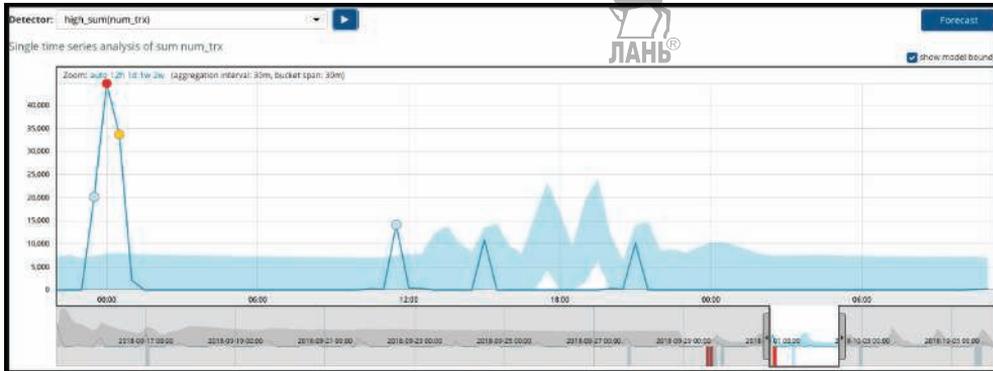


Рис. П.8 ❖ Анализ с использованием односторонней функции `high_sum`

Результат анализа провалов показан на рис. П.9.

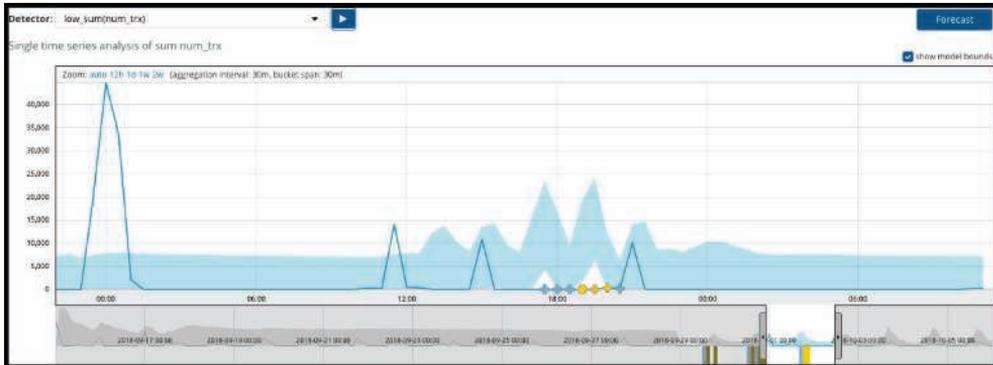


Рис. П.9 ❖ Анализ с использованием односторонней функции `low_sum`

Обратите внимание, что аномалии в середине теперь имеют гораздо более высокий балл (в данном случае максимальный балл 47, отмеченный желтым цветом).

Итак, теперь, когда два односторонних детектора работают вместе в одном задании, динамический диапазон каждого детектора оптимален, поскольку у них есть собственная таблица нормализации!

ИСКЛЮЧЕНИЕ ОПРЕДЕЛЕННЫХ ИНТЕРВАЛОВ ВРЕМЕНИ

Часто люди спрашивают, как заставить ML игнорировать тот факт, что произошло определенное событие. Возможно, это был запланированный период технического обслуживания, или возник сбой в конвейере приема данных, и данные были потеряны на несколько мгновений. Есть несколько способов заставить ML игнорировать заданные периоды времени, и мы условно разделим их на две группы:

- наступающий (известный) интервал времени;
- неожиданный интервал, обнаруживаемый только постфактум.

Для иллюстрации этих вариантов мы будем использовать задание подсчета одной метрики (рис. П.1) из набора данных farequote с аномалией на дату 9 февраля (рис. П.10).

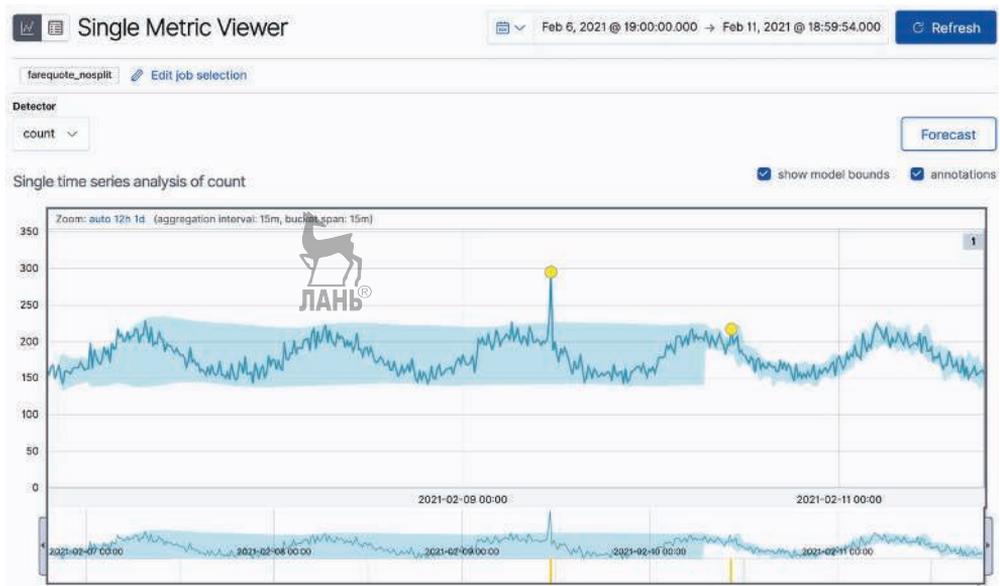


Рис. П.10 ❖ Анализ набора данных farequote с аномалией, которую мы хотели бы игнорировать

Теперь давайте посмотрим, как мы можем игнорировать аномалию 9 февраля в различных ситуациях.

Исключение наступающего (известного) интервала времени

Для исключения наступающего интервала времени можно использовать два метода, которые описаны ниже. Один предусматривает создание специального события календаря, а другой управляет временем, в течение которого работает поток данных.

Создание события календаря

Вы можете легко создать событие календаря, нажав **Settings** (Настройки), а затем **Create** (Создать) в разделе **Calendar** (Календарь). Создадим запись календаря на 9 февраля, как показано на рис. П.11.

Overview Anomaly Detection Data Frame Analytics Data Visualizer **Settings**

Create new calendar

Calendar ID

ignore_feb9

Use lowercase alphanumerics (a-z and 0-9), hyphens or underscores; must start and end with an alphanumeric character

Description

Apply calendar to all jobs

Jobs

Groups

farequote_jobs

Events

Search...

Description ↑	Start	End	
feb9	2021-02-09 00:00:00	2021-02-10 00:00:00	Delete

Rows per page: 5

Рис. П.11 ❖ Создание события календаря для исключения определенного периода времени

Если мы создадим новое задание, принадлежащее к группе `farequote_jobs`, чтобы оно подчинялось этому календарю, то весь день 9 февраля будет полностью проигнорирован (рис. П.12).

Как видите, весь день был замаскирован, включая время аномального всплеска.



Рис. П.12 ❖ Период времени, исключенный событием календаря

Остановка и запуск потока данных в нужное время

Просто остановив и перезапустив поток данных задания по обнаружению аномалий в нужное время, вы можете исключить определенный интервал времени. В нашем примере поток данных был остановлен в полночь 9 февраля и перезапущен в полночь 10 февраля (рис. П.13).

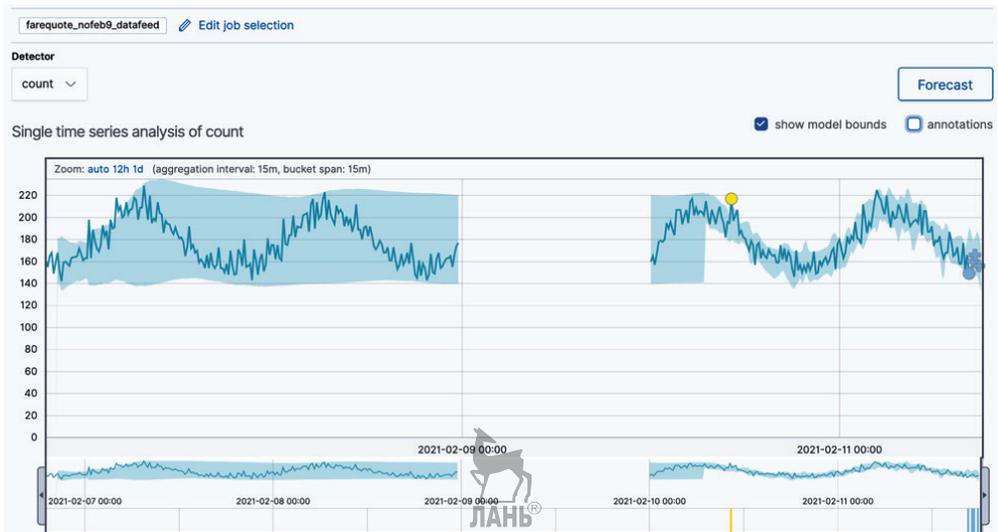


Рис. П.13 ❖ Период времени исключен путем манипуляции с потоком данных

Как будто 9 февраля никогда не было! Теперь давайте обсудим, что вы можете сделать, чтобы исключить интервал времени постфактум – если о нем не было известно заранее.

Исключение интервала времени постфактум

Чтобы вернуться в прошлое и «забыть» определенный интервал времени, мы можем использовать два метода. Первый предусматривает простое клонирование и повторный запуск исторических данных, а второй предполагает использование снимков модели.

Клонирование задания и повторный запуск исторических данных

Взяв за основу метод из предыдущего раздела, вы могли бы создать новое клонированное задание и просто заставить поток данных пропустить интервал времени, который вы хотите игнорировать. Остановите его в начале интервала и возобновите в конце интервала. Этот метод отлично работает, если повторное использование модели на существующих (и все еще доступных) исторических данных не является слишком обременительным. Однако если у вас есть по-настоящему зрелые модели, инкапсулирующие поведение данных, к которым у вас больше нет доступа (потому что они устарели и были удалены из вашего кластера), то вместо этого вам придется использовать описанный ниже метод снимков модели.

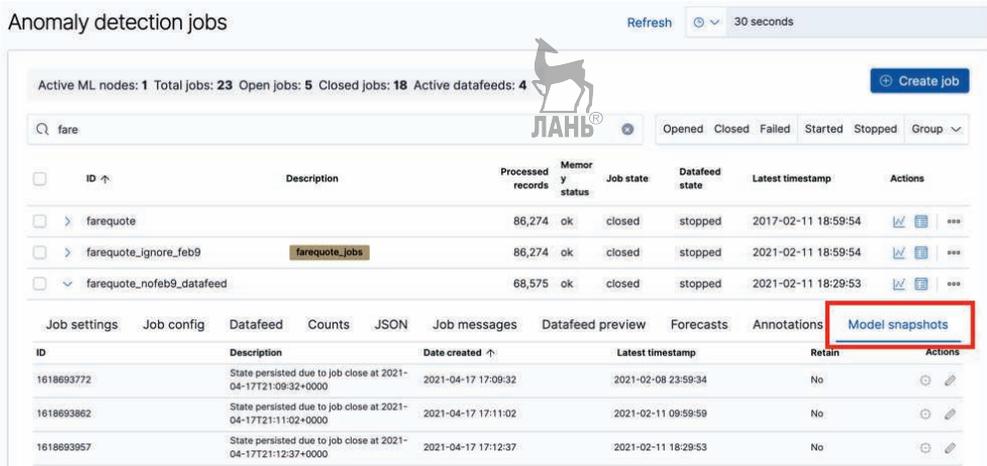
Возврат задания к предыдущему снимку модели

Когда клонирование и повторное обучение модели на существующих исторических данных нежелательно или нецелесообразно, вы можете эффективно удалить временное окно, используя тот факт, что работающее задание периодически создает моментальный снимок модели. По умолчанию снимки создаются примерно каждые 3–4 часа. Вы можете изменить этот интервал (`background_persist_interval`) при создании или обновлении задания.

- ❗ Хранение этих снимков зависит от некоторых других параметров (таких как `daily_model_snapshot_retention_after_days` и `model_snapshot_retention_days`). Обратитесь к документации API обнаружения аномалий по адресу <https://www.elastic.co/guide/en/machine-learning/current/ml-api-quickref.html>.

Процедура возврата задания по обнаружению аномалий к предыдущему моментальному снимку выглядит следующим образом.

1. Остановите поток данных задания, если он запущен.
2. Найдите самый последний снимок модели, сделанный непосредственно перед временным интервалом, который вы хотите стереть, используя вкладку **Model snapshots** (Снимки модели) на странице **Job Management** (Управление заданиями) в Kibana (рис. П.14).



Active ML nodes: 1 Total jobs: 23 Open jobs: 5 Closed jobs: 18 Active datafeeds: 4

Refresh 30 seconds

fare

Opened Closed Failed Started Stopped Group

ID	Description	Processed records	Memory status	Job state	Datafeed state	Latest timestamp	Actions
farequote		86,274	ok	closed	stopped	2017-02-11 18:59:54	🔍 📄 ⋮
farequote_ignore_feb9	farequote_jobs	86,274	ok	closed	stopped	2021-02-11 18:59:54	🔍 📄 ⋮
farequote_nofeb9_datafeed		68,575	ok	closed	stopped	2021-02-11 18:29:53	🔍 📄 ⋮

Job settings Job config Datafeed Counts JSON Job messages Datafeed preview Forecasts Annotations **Model snapshots**

ID	Description	Date created	Latest timestamp	Retain	Actions
1618693772	State persisted due to job close at 2021-04-17T21:09:32+0000	2021-04-17 17:09:32	2021-02-08 23:59:34	No	🔍 📄 ✎
1618693862	State persisted due to job close at 2021-04-17T21:11:02+0000	2021-04-17 17:11:02	2021-02-11 09:59:59	No	🔍 📄 ✎
1618693957	State persisted due to job close at 2021-04-17T21:12:37+0000	2021-04-17 17:12:37	2021-02-11 18:29:53	No	🔍 📄 ✎

Рис. П.14 ❖ Интерфейс управления снимками модели

В качестве альтернативы вы можете использовать вызов API получения снимков, как описано на странице <https://www.elastic.co/guide/en/elasticsearch/reference/current/ml-get-snapshot.html>.

3. Верните задание к этому моментальному снимку, нажав на значок возврата ↺, или, при использовании API, воспользуйтесь командой `_getvert`. В пользовательском интерфейсе Kibana вы увидите пункты меню, позволяющие удалить данные и воспроизвести анализ исторических данных после момента создания моментального снимка, включая возможность замаскировать проблемный период времени с помощью события календаря (рис. П.15).
4. При желании возобновите поток данных в реальном времени после исключенного интервала.

Итак, вы можете с легкостью исключить из анализа ненужный интервал времени и предотвратить загрязнение ваших заданий по обнаружению аномалий операционными проблемами или нежелательными событиями.



The screenshot shows the 'Anomaly Detection' section of a software interface. On the left, there's a list of 'Anomaly detection jobs' with a search bar containing 'fare'. Below the search bar is a table of jobs with columns for ID, Description, and Date created. The table lists three jobs with IDs 1618693772, 1618693862, and 1618693957, all with descriptions related to 'State persisted due to job close at 2021-04-17 17:04-1772112-37+0000'. Below the table are checkboxes for various filters like 'farequote_nospilt' and 'farequote_response'. On the right, a dialog box titled 'Revert to model snapshot 1618693772' is open. It shows a bar chart with a highlighted red area and a yellow warning box stating 'Anomaly data will be deleted'. Below the chart are options for 'Replay analysis', 'Run job in real time', and 'Create calendar to skip a range of time'. At the bottom of the dialog are 'Close' and 'Apply' buttons.

Рис. П.15 ❖ Возврат к предыдущему снимку модели с возможностью замаскировать интервал времени

ИСПОЛЬЗОВАНИЕ НАСТРАИВАЕМЫХ ПРАВИЛ И ФИЛЬТРОВ

Хотя задания по обнаружению аномалий невероятно полезны, они не отражают особенности предметной области и не зависят от актуальности необработанных данных. Другими словами, алгоритмы машинного обучения без учителя не знают, что десятикратное увеличение загрузки ЦП (например, с 1 % до 10 %) фактически не оказывает влияния на правильность работы приложения, даже если оно может быть статистически аномальным или маловероятным в типичном сценарии. Точно так же задания по обнаружению аномалий обрабатывают все анализируемые объекты одинаково, но пользователь может пожелать исключить результаты для определенного IP-адреса или идентификатора пользователя, поскольку он знает, что обнаружение аномалий для этих объектов нежелательно или бесполезно. Использование настраиваемых правил и фильтров позволяет пользователю вводить знания предметной области в конфигурацию задания по обнаружению аномалий, сохраняя полный контроль над тем, что считать аномальным, даже если объект анализа является важной частью процесса моделирования.

Создание собственных правил

Вы можете определить настраиваемое правило либо во время создания задания (но только при использовании API создания задания), либо после того, как задание обнаружит некоторые аномалии, используя параметр меню **Configure rules** (Настроить правила) в меню **Actions** (Действия) в пользовательском интерфейсе Anomaly Explorer (рис. П.16).

The screenshot shows the 'Anomalies' section of the Anomaly Explorer interface. It features a table with columns: time, severity, detector, found for, influenced by, actual, typical, description, and actions. Two anomalies are listed:

time	severity	detector	found for	influenced by	actual	typical	description	actions
February 9th 2017	98	mean(response time) partitionfield=airline	AAL	airline: AAL	282.025	100.025	3x higher	View series
February 10th 2017	68	mean(response time) partitionfield=airline	EGF	airline: EGF	207.765	199.833	Unusually high	Configure rules

Рис. П.16 ❖ Пункт меню **Configure rules** в пользовательском интерфейсе Anomaly Explorer

Процедура определения правила в основном не требует пояснений. В данном случае вы можете решить, что, несмотря на наличие аномалии времени отклика 282,025 мс (показано на рис. П.16), она не представляет особого интереса и что вы хотите игнорировать аномалии, если время отклика меньше 1 секунды (1000 мс). Вы можете определить соответствующее правило, как показано на рис. П.17.

Существуют дополнительные параметры для исключения значения из моделирования, а также для ограничения области действия правила определенным списком фильтров, чтобы правило применялось только к определенным объектам (например, только к серверам, расположенным в определенном месте). Списки фильтров можно определить в разделе **Settings** (Настройки), нажав на **Filter lists** (Списки фильтров) в пользовательском интерфейсе.

Обратите внимание, что определение правила применяется к будущему анализу (с точки зрения определения правила, вперед по оси времени) и не применяется к прошлым аномалиям. Чтобы правило применялось к прошлым аномалиям, вам нужно будет клонировать существующее задание (после того, как правило было определено), а затем повторно запустить анализ исторических данных.

Благодаря наличию правил и фильтров пользователь имеет практически полный контроль над тем, что в конечном итоге будет истолковано как аномалия (и послужит поводом для оповещения). Это позволяет существенно изменить парадигму по сравнению с традиционным подходом, основанным на философии создания предупреждений «снизу вверх», которая существовала в IT на протяжении десятилетий. Альтернативный подход описан в следующем подразделе.

The screenshot displays the 'Create rule' dialog in the Anomaly Explorer. The dialog is divided into several sections:

- Job ID:** farequote
- Detector:** mean(responsetime) partitionfield=airline
- Selected anomaly:** actual 282, typical 100
- Action:** Choose the action(s) to take when the rule matches an anomaly. Options include 'Skip result (recommended)' and 'Skip model update'.
- Conditions:** A checkbox 'Add numeric conditions for when the rule applies. Multiple conditions are combined using AND.' is checked. Below it, a condition is defined: 'WHEN actual IS LESS THAN 1000'.
- Scope:** A checkbox 'Add a filter list to limit where the rule applies.' is unchecked.

The background interface shows a heatmap of anomalies for various airlines (AAL, EGF, AWE, UAL, DAL, ACA, SWR, SWA, JAL, TRS) over time. A table of anomalies is visible below the heatmap:

time	severity	detector	found
> February 9th 2017	96	mean(responsetime) partitionfield=airline	AAL
> February 10th 2017	66	mean(responsetime) partitionfield=airline	EGF
> February 7th 2017	67	mean(responsetime) partitionfield=airline	AWE
> February 10th 2017	61	mean(responsetime) partitionfield=airline	UAL

Рис. П.17 ❖ Пользовательский интерфейс создания правила

Использование настраиваемых правил для оповещения «сверху вниз»

Когда мы спрашиваем: «На какой процент собранных данных вы обращаете внимание?», наиболее частым реалистичным ответом является «менее 10 %», а иногда и «менее 1 %». Причина подобного пренебрежения данными заключается в том, что традиционный подход к упреждающему использованию данных заключается в том, чтобы начать с нуля, а затем постепенно создавать пороговые значения или предупреждения на основе правил. Это может быть сложной и утомительной задачей, требующей предварительных знаний (или, по крайней мере, предположений) относительно того, каким должно быть ожидаемое поведение каждого временного ряда. Затем, как только базовые оповещения будут настроены, обычно начинается долгий процесс точной подгонки критериев, который пытается исправить чувствительность предупреждений с досадными ложными срабатываниями. Кроме

того, могут существовать показатели, необычное поведение которых невозможно обнаружить статическим порогом.

А теперь представьте, что эту задачу нужно масштабировать; если у вас есть 10 показателей на сервер и 100 серверов, т. е. получается 1000 отдельных показателей. Создание индивидуального оповещения для каждого из них нецелесообразно.

Однако единственное задание на обнаружение аномалий, выполняющее всю упомянутую работу, может быть создано на основе этих данных менее чем за 1 минуту. Самообучение Elastic ML на исторических данных, которое также занимает очень мало времени, минимизирует количество ложных срабатываний за счет независимой адаптации к естественным характеристикам каждого временного ряда. Но если при обнаружении аномалии выявляются вещи, о которых нам не важно знать, мы можем просто исключить их с помощью специальных правил.

Этот подход *сверху вниз* (сначала охватываем взглядом всю картину, а затем начинаем исключать ненужные детали) работает быстрее и обеспечивает более широкий упреждающий охват данных, чем подход снизу вверх (создание пороговых оповещений с нуля).

СООБРАЖЕНИЯ ОТНОСИТЕЛЬНО ПРОПУСКНОЙ СПОСОБНОСТИ ЗАДАНИЙ

Elastic ML – потрясающий и, без сомнения, очень быстрый и масштабируемый инструмент, но все же у него есть практическая верхняя граница количества событий в секунду, обрабатываемых любым заданием по обнаружению аномалий, в зависимости от нескольких факторов:

- скорость, с которой данные могут быть доставлены в алгоритмы (то есть быстродействие запроса);
- скорость, с которой алгоритмы могут перебирать данные для анализа.

Для второго пункта быстродействие в основном зависит от следующих факторов:

- функция, выбранная для анализа (например, `count` выполняется быстрее, чем `lat_long`);
- выбранное значение `bucket_span` (более длинные сегменты в целом обрабатываются быстрее, чем короткие, потому что большее количество сегментов, анализируемых в единицу времени, увеличивает накладные расходы на обработку каждого сегмента в связи с необходимостью записи результата и т. д.).

Однако если у вас настроен строго определенный анализ и вы не можете изменить его по каким-либо причинам, то вы мало что сможете сделать, если не проявите творческий подход и не разделите данные на несколько заданий. Это связано с тем, что задания ML (по крайней мере, на данный

момент) в настоящее время выполняются по принципу «один анализ – один процессор». Следовательно, можно было бы разделить данные на несколько отдельных заданий машинного обучения, чтобы использовать хотя бы несколько процессоров одновременно. Но сначала давайте сосредоточимся на первом факторе – быстродействии запроса, поскольку здесь есть множество возможностей для совершенствования:

- избегайте кросс-кластерного поиска, чтобы ограничить передачу данных по сети;
- настройте параметры потока данных для оптимизации быстродействия;
- используйте агрегаты запросов Elasticsearch, чтобы распределить задачу дистрибуции данных на меньший набор алгоритмов машинного обучения.

Первый подход очевиден. Вы сможете повысить быстродействие только в том случае, если приблизите анализ к необработанным данным.

Второй подход может потребовать проведения некоторых экспериментов. Есть такие параметры, как `scroll_size`, управляющие размером каждой прокрутки. Значение по умолчанию – 1000, а для кластеров приличного размера его можно безопасно увеличить до 10 000. Запустите несколько тестов с разными размерами прокрутки и посмотрите, как это влияет на производительность запросов и кластера.

На наш взгляд, последний фактор должен оказать наибольшее влияние на производительность, но очевидно, что он самый сложный в настройке, и пока вы получите оптимальную агрегацию Elasticsearch для правильной работы с ML, можете допустить ошибки, в результате чего потребуются дополнительные итерации, хотя это не очень страшно. Дополнительную информацию вы можете получить в документации по адресу <https://www.elastic.co/guide/en/machine-learning/current/ml-configuring-aggregation.html>. Отрицательный эффект от использования агрегирования с ML, как правило, заключается в том, что вы теряете доступ к другим полям в данных, которые могут быть полезными в качестве факторов влияния.

В любом случае, все перечисленные соображения следует учитывать при оптимизации быстродействия задания машинного обучения.

О ВРЕДЕ ИЗЛИШНЕЙ СЛОЖНОСТИ СЦЕНАРИЕВ

Однажды мы обсуждали с одним из заказчиков различные сценарии применения на практике механизма обнаружения аномалий. В частности, этот заказчик строил защищенный операционный центр как часть своего бизнеса поставщика услуг по управлению безопасностью (managed security service provider, MSSP), поэтому его очень интересовали ситуации, в которых может помочь машинное обучение.

Основная идея большинства вариантов использования заключалась в том, чтобы посмотреть на поведение пользователя и найти какую-либо анома-

лию. Одним из примеров был вход в систему из необычных/редких мест: например, Боб только что вошел в систему из Украины, но обычно он не входит в систему оттуда.

В процессе обсуждения реализации мы говорили о том, что у них есть несколько клиентов, у каждого из которых есть несколько пользователей. Поэтому они думали о способах разделения/разбиения данных таким образом, чтобы отслеживать критерий «необычная страна» для каждого пользователя каждого клиента.

Мы попросили их сделать шаг назад и спросили: «Стоит ли считать аномалией, если кто-то другой входит в систему из Украины, а не только Боб?» – на что был дан ответ «Да».

Таким образом, в этом случае нет смысла разбивать анализ по пользователям; имеет смысл сохранить разделение на уровне клиента и просто объединить все местоположения пользователей от каждого клиента в единый пул наблюдаемых стран. На самом деле это оптимальный сценарий – в нем больше общих данных, и, как мы знаем, функция `geo` работает лучше всего, когда есть много рутинных данных, с которыми можно сопоставить новое наблюдение.

ОБНАРУЖЕНИЕ АНОМАЛИЙ В ВЫЧИСЛЯЕМЫХ ПОЛЯХ

В некоторых случаях может потребоваться проанализировать значение поля, которое не существует в сопоставлениях хранилищ, но может быть вычислено динамически на основе значений других полей. Эта возможность динамически определять значения полей уже довольно давно существует в Elasticsearch в виде *полей сценария* (`script field`), но начиная с версии 7.11 поля сценария заменили обновленной концепцией, известной как *поля времени выполнения* (`runtime fields`). Если коротко, поля времени выполнения обрабатываются как приоритетные в сопоставлении Elasticsearch (если оно там определено) и в конечном итоге позволяют пользователю продвигать поле времени выполнения в хранилище.

Пользователи могут определять поля времени выполнения в сопоставлении или только в поисковом запросе. Стоит отметить, что на момент написания книги не было реализовано определение полей времени выполнения в потоке данных задания обнаружения аномалий. Однако если поля времени выполнения определены в сопоставлениях, то задание обнаружения аномалий может легко ими воспользоваться.

! Дополнительную информацию о полях времени выполнения можно найти в документации Elastic по адресу <https://www.elastic.co/guide/en/elasticsearch/reference/current/runtime.html>.

Хотя более полная информация о полях времени выполнения выходит за рамки этой книги, важно знать, что задания по обнаружению аномалий могут использовать эти динамические поля так же, как если бы они были обычными полями. Рассмотрим интересный, хотя и выдуманный пример.

Давайте вернемся к примеру farequote, показанному на рис. П.1, и в качестве аргумента заявим, что 9 февраля – особый день определенного типа для airline:AAL – возможно, приблизительный эквивалент Черной пятницы или Киберпонедельника, или даже день, когда, как мы точно знаем, определенный показатель будет немного отличаться от нормы на известную величину.

Мы разработаем сценарий, в котором, как мы знаем, AAL будет испытывать *предсказуемо* более высокое время отклика, которое может быть на 20 % больше, чем обычно. Мы не хотим, чтобы 9 февраля было календарным событием для Elastic ML, и не хотим отказываться от данных для AAL. Поэтому давайте поступим иначе и просто в нужный период времени уменьшим время отклика на 20 %, чтобы не нарушать наше обычное моделирование и оповещение. Мы можем сделать это с помощью полей времени выполнения.

1. В первую очередь нужно определить в сопоставлении для хранилища новое поле времени выполнения, которое называется `responsetime_adjusted`:

```
PUT farequote/_mapping
{
  "runtime": {
    "responsetime_adjusted": {
      "type": "double",
      "script": {
        "source": "emit(params._source.responsetime * 1.0)"
      }
    }
  }
}
```

Это поле (на данный момент) будет точно таким же, как поле `responsetime` для всех авиакомпаний, потому что заполняется путем умножения значения исходного поля на константу 1,0.

2. Затем настроим задание на использование детектора `high_mean` в этом новом поле `responsetime_adjusted`, где мы также разделим анализ по полю `airline` (рис. П.18).
3. Мы запустим поток данных до полуночи 9 февраля, но на этом остановим анализ. Чтобы затем уменьшить время отклика для данных AAL на 20 % (но не трогать данные других авиакомпаний), выполним следующую команду:

```
PUT farequote/_mapping
{
  "runtime": {
    "responsetime_adjusted": {
      "type": "double",
      "script": {
        "source": "if(doc['airline'].value.equals('AAL'))
```

```

{emit(params._source.responsetime * 0.8)} else
{emit(params._source.responsetime * 1.0)}"
    }
  }
}
}
}

```

Create job: Multi-metric
Using index pattern farequote

1 Time range 2 Pick fields 3 Job details 4 Validation 5 Summary

Pick fields
Data split by airline

Имя поля времени выполнения

High mesh(responsetime_adjusted)

Add metric

Split field
Select a field to partition analysis by. Each value of this field will be modeled independently individually.

Split field: airline

Bucket span
Set the interval for time series analysis, typically between 15m to 1h.

Bucket span: 15m Estimate bucket span

Influencers
Select which categorical fields have influence on the results. What/what might you 'blame' for an anomaly? Recommend 1-3 influencers.

Influencers: airline

Sparse data
Select if you wish to ignore empty buckets from being considered anomalous. Available for count and sum analysis.

Sparse data: Sparse data

< Previous Next >

Рис. П.18 ❖ Настройка задания для анализа поля времени выполнения

4. Затем возобновим поток данных задания, чтобы проанализировать особый день 9 февраля, но остановимся в полночь в начале 10 февраля.
5. После анализа данных от 9 февраля мы вернем время ответа для данных AAL в норму, повторно вызвав команду, представленную на шаге 1.
6. Продолжим анализ остальных данных в обычном режиме.

Конечным результатом всех этих манипуляций является то, что мы смогли успешно подавить значения времени отклика AAL на 20 % (о чем свидетельствуют заниженные значения между двумя аннотациями), но мы все же смогли выявить значительную аномалию, несмотря на наше специальное отношение к AAL (рис. П.19).

Этот метод может быть полезен для внесения любого количества динамических модификаций в данные на лету для расширенного анализа или для поддержки анализа тех аспектов данных, которые могут быть недоступны в сопоставлениях полей по умолчанию.



Рис. П.19 ❖ Результаты задания, которое проанализировало динамически изменяемое поле времени выполнения

ЗАКЛЮЧЕНИЕ

Elastic ML – это мощный и гибкий, но в то же время простой в использовании инструмент, который дает всю мощь науки о данных в руки специалистов, не являющихся профессионалами в этой области, так что они могут получить представление об огромных объемах данных. На протяжении всей этой книги описывается множество различных способов, с помощью которых пользователи могут применить преимущества технологий машинного обучения для решения реальных задач в IT. Мы надеемся, что вы воспользуетесь знаниями, полученными в этой книге, и реализуете несколько собственных замечательных проектов. Не пытайтесь решить все проблемы в первый день – начните с малого, добейтесь заметных побед и расширяйте масштаб своих проектов по мере того, как обретете больше уверенности. Успех порождает успех!





Предметный указатель

А

Агрегирование, 241
Анализ
 временной, 84
 первопричин, 194
 поведения пользователей, 84
 популяционный, 60, 84
 регрессионный, 318
 фреймов, 239
Аннотация, 50

Б

Бинаризация, 283
 порог, 286

В

Вероятность класса, 314
Влияние характеристики, 276
Выделенный узел, 45

Г

Гиперпараметр, 310
 оптимизация, 310
Градиентное усиление, 309

Д

Дерево решений, 307
Детектор, 57

 функция, 59
 односторонняя, 62
Доверительный интервал, 114

З

Зависимая переменная, 301
Задание, 47
 анализ фреймов данных, 47
 обнаружение аномалий, 47
 с несколькими критериями, 81

И

Избыточное обучение.
См. Переобучение
Историческое обучение, 100

К

Классификация, 33, 239
Ключевой показатель
 эффективности, 20, 194
Конвейер данных, 337
Конечный узел. См. Листовой узел
Конструирование признаков, 298
Конфигурация анализа, 57

Л

Листовой узел, 308
Логический вывод, 337

М

Массив причин, 142
 Матрица неточностей, 285
 Матрица признаков, 33
 Машинное обучение
 без учителя, 23
 с учителем, 24
 Метрическая функция, 76

**Н**

Начало времени, 83
 Нормализация, 129

О

Обзорная полоса, 129
 Обнаружение
 аномалий, 56
 выбросов, 239
 Обработчик, 337
 вывода, 337
 Обучающие данные, 24
 Обучение моделей, 27
 Операционализация, 46
 Оркестровка, 52
 Отклик, 285
 Относительное ранжирование, 131
 Оценка
 аномалии в нескольких
 сегментах, 146
 аномалий, 128
 класса, 314
 на уровне записи, 135
 на уровне сегмента, 129
 на уровне фактора влияния, 131
 Ошибка
 обобщения, 300
 обучения, 299

П

Переобучение, 310
 Период сегмента, 48
 Подсчет
 ненулевой, 73
 раздельный, 74
 Поле, 59
 времени выполнения, 373

сценария, 373
 by, 60
 over, 60
 partition, 59
 Поток данных, 49, 57
 Преобразование
 непрерывное, 249
 пакетное, 248
 Признак
 важность, 34, 315
 вектор, 34
 Прогнозирование, 98
 нескольких временных рядов, 119
 одиночного временного ряда, 103
 просмотр результатов, 114
 Прокрутка, 50

Р

Рабочая панель, 159
 Разделение, 203
 Распознаватель данных, 222
 Расстояние Левенштейна, 89
 Регрессия, 33, 239, 327
 Решающая граница, 296

С

Сводная таблица, 241
 Сегментирование входных
 данных, 48
 Скриптовое поле, 83
 Среднеквадратичная ошибка, 328

Т

Точность, 285

У

Уникальность, 74

Ф

Фактор
 влияния, 129, 203
 внешний, 99
 неизвестный, 99
 Факторная мера, 275
 Формула детектора, 60
 Фрейм данных, 33

Функция

времени, 80
геолокации, 79
метрическая, 76
потерь, 328
count, 61
freq_rare, 79
info_content, 79
rare, 78

Х

Хранилище, 49
системное, 51
скрытое, 51

Ч

Чистое разделение, 308
Чистота узла, 308



Книги издательства «ДМК ПРЕСС»
можно купить оптом и в розницу
в книготорговой компании «Галактика»
(представляет интересы издательств
«ДМК ПРЕСС», «СОЛОН ПРЕСС», «КТК Галактика»).

Адрес: г. Москва, пр. Андропова, 38;
тел.: **(499) 782-38-89**, электронная почта: **books@aliants-kniga.ru**.

При оформлении заказа следует указать адрес (полностью),
по которому должны быть высланы книги;
фамилию, имя и отчество получателя.

Желательно также указать свой телефон и электронный адрес.
Эти книги вы можете заказать и в интернет-магазине: **www.a-planeta.ru**.



Рич Кольер, Камилла Монтонен, Бахаалдин Азарми

Машинное обучение в Elastic Stack

Главный редактор	<i>Мовчан Д. А.</i> dmkpress@gmail.com
Зам. главного редактора	<i>Сенченкова Е. А.</i>
Перевод	<i>Яценков В. С.</i>
Корректор	<i>Синяева Г. И.</i>
Верстка	<i>Чаннова А. А.</i>
Дизайн обложки	<i>Мовчан А. Г.</i>

Гарнитура PT Serif. Печать цифровая.
Усл. печ. л. 30,88. Тираж 200 экз.

Веб-сайт издательства: **www.dmkpress.com**

Elastic Stack – это комплексное решение для анализа журналов, которое помогает пользователям эффективно получать, обрабатывать и анализировать данные поиска. Книга содержит всесторонний обзор функций машинного обучения Elastic Stack (Elastic ML) как для анализа данных временных рядов, так и для классификации, регрессии и обнаружения выбросов.

Концепции машинного обучения объясняются понятным и доступным языком. Рассмотрен анализ временных рядов для различных типов данных, таких как файлы журналов, сетевые потоки, показатели приложений и финансовые данные. Описано использование Elastic ML для ведения журнала, обеспечения безопасности и отслеживания показателей.

После прочтения вы приобретете практический опыт совместного использования технологии машинного обучения и Elastic Stack, а также знания, необходимые для включения машинного обучения в вашу платформу распределенного поиска и анализа данных.

Вы узнаете:

- как Elastic ML применяется для обнаружения различных типов аномалий и составления прогнозов;
- как использовать обнаружение аномалий в ИТ-деятельности, аналитике безопасности и других областях;
- как задействовать результаты анализа Elastic ML в настраиваемых представлениях, панелях мониторинга и прогнозных оповещениях;
- как обучать и развертывать модели машинного обучения с учителем для работы в реальном времени;
- как анализ фреймов открывает доступ к новым сценариям использования данных;
- как включить платную функциональность машинного обучения в Elastic Stack.

Книга содержит различные примеры, советы и приемы, которые помогут максимально эффективно использовать возможности Elastic ML



Интернет-магазин:
www.dmkpress.com

Оптовая продажа:
КТК «Галактика»
e-mail: books@aliens-kniga.ru

Ракт

DMK
ИЗДАТЕЛЬСТВО
www.dmk.pf

ISBN 978-5-93700-107-8



9 785937 001078 >