



П. Мишра

Объяснимые модели искусственного интеллекта на Python

Прадипта Мишра

Объяснимые модели искусственного интеллекта на Python

Модель искусственного интеллекта.
Объяснения с использованием библиотек,
расширений и фреймворков
на основе языка Python

Practical Explainable ИИ Using Python

Artificial Intelligence Model Explanations
Using Python-based Libraries, Extensions,
and Frameworks

Pradeepta Mishra

Apress®

Объяснимые модели искусственного интеллекта на Python

Модель искусственного интеллекта.
Объяснения с использованием библиотек,
расширений и фреймворков
на основе языка Python

Прадипта Мишра



Москва, 2022

УДК 004.438Python

ББК 32.973.22

M71

Прадипта Мишра

- M71 Объяснимые модели искусственного интеллекта на Python. Модель искусственного интеллекта. Объяснения с использованием библиотек, расширений и фреймворков на основе языка Python / пер. с англ. С. В. Минца. – М.: ДМК Пресс, 2022. – 298 с.: ил.

ISBN 978-5-93700-124-5

В книге рассматриваются так называемые модели «черного ящика» для повышения адаптивности, интерпретируемости и объяснимости решений, принимаемых алгоритмами искусственного интеллекта (ИИ) с использованием библиотек Python XAI, TensorFlow 2.0+, Keras, а также пользовательских фреймворков с использованием Python Wrappers.

Издание предназначено специалистам по анализу данных, инженерам по внедрению моделей ИИ, а также может быть полезно бизнес-пользователям и руководителям проектов, использующих результаты работы решений ИИ в своей деятельности.

УДК 004.438Python

ББК 32.973.22

First published in English under the title Practical Explainable ИИ Using Python
This edition has been translated and published under licence from APress Media, LLC, part of Springer Nature.

APress Media, LLC, part of Springer Nature takes no responsibility and shall not be made liable for the accuracy of the translation.

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.

*Я посвящаю эту книгу моему покойному отцу, который всегда вдохновлял меня стремиться к следующему уровню, никогда не останавливаться на достигнутом и продолжать двигаться вперед.
Люблю тебя, папа, ты бы гордился этой книгой.*

*Трем женщинам в моей жизни – моей жене Праджне и моим дочерям, Аарии и Аадии, которые всегда поддерживали и любили меня.
Завершение этой книги было бы невозможно без их поддержки.*

Оглавление

Об авторе	10
О рецензентах	11
Благодарности	12
Введение	13
Глава 1. Объяснимость и интерпретируемость модели.....	15
Создание основ	15
Искусственный интеллект	16
Необходимость XAI.....	17
Сравнение объяснимости с интерпретируемостью.....	20
Типы объяснимости	22
Инструменты для объяснимости моделей.....	22
SHAP	23
LIME	23
ELI5	24
Skater	25
Skope_rules.....	26
Методы XAI для ML.....	27
Совместимые с XAI модели	28
Объяснимый ИИ удовлетворяет требованиям ответственного ИИ	29
Оценка XAI	31
Заключение	33
Глава 2. Этика, предвзятость и надежность ИИ	34
Основы этики ИИ	34
Предвзятость в ИИ.....	37
Предвзятость данных	37
Алгоритмическая предвзятость	37
Процесс снижения предвзятости	38
Предвзятость интерпретации.....	38
Предвзятость при обучении	39
Надежность в ИИ	42
Заключение	44
ГЛАВА 3. Объяснимость для линейных моделей.....	45
Линейные модели	45
Линейная регрессия	45

VIF и проблемы, которые он может породить.....	53
Окончательная модель	57
Объяснимость модели	57
Доверие к модели ML: SHAP	59
Локальное объяснение и индивидуальные прогнозы в модели ML	62
Глобальное объяснение и общие прогнозы в модели ML.....	65
Объяснение LIME и модель ML	69
Объяснение Skater и модель ML	73
Объяснение ELI5 и модель ML	75
Логистическая регрессия	76
Интерпретация	85
Вывод LIME	86
Заключение	92
ГЛАВА 4. Объяснимость для нелинейных моделей	93
Нелинейные модели	93
Объяснение дерева решений	95
Подготовка данных для модели дерева решений	97
Создание модели	99
Дерево решений – SHAP	106
График частичной зависимости	106
PDP с использованием Scikit-Learn	115
Объяснение нелинейной модели – LIME	118
Нелинейное объяснение – Skope-Rules	121
Заключение	123
ГЛАВА 5. Объяснимость для ансамблевых моделей	124
Ансамблевые модели	124
Типы ансамблевых моделей	125
Почему ансамблевые модели?	125
Использование SHAP для ансамблевых моделей	128
Использование интерпретации, объясняющей модель повышения	133
Модель классификации ансамблей: SHAP	139
Использование SHAP для объяснения категориальных моделей повышения.....	146
Использование многоклассовой категориальной модели повышения SHAP	152
Использование SHAP для объяснения модели легкой GBM	154
Заключение	157
ГЛАВА 6. Объяснимость для моделей временных рядов	159
Модели временных рядов	159
Выбор подходящей модели.....	161
Стратегия прогнозирования.....	162
Доверительный интервал прогнозов	162
Что происходит с доверием?	163

Временные ряды: LIME	175
Заключение	178
ГЛАВА 7. Объяснимость для NLP.....	179
Задачи обработки естественного языка	179
Объяснимость для классификации текстов	180
Набор данных для классификации текста	180
Объяснение с помощью ELI5	182
Вычисление весов характеристик для локального объяснения	183
Локальное объяснение. Пример 1	184
Локальное объяснение. Пример 2	184
Локальное объяснение. Пример 3	185
Объяснение после удаления стоп-слов	185
Классификация текста на основе N-грамм.....	189
Объяснимость многоклассовой классификации текста по меткам	193
Локальное объяснение. Пример 1	198
Локальное объяснение. Пример 2	199
Локальное объяснение. Пример 3	201
Заключение	209
ГЛАВА 8. Справедливость модели ИИ, использующей сценарий «что, если»	210
Что такое WIT?	210
Установка WIT	211
Метрика оценки.....	220
Заключение	221
ГЛАВА 9. Объяснимость для моделей глубокого обучения....	222
Объяснение моделей DL.....	222
Использование SHAP с DL.....	225
Использование Deep SHAP	225
Использование Alibi	225
Объяснитель SHAP для глубокого обучения	229
Еще один пример классификации изображений	231
Использование SHAP	234
Deep Explainer для табличных данных.....	237
Заключение	239
ГЛАВА 10. Контрфактуальные объяснения для моделей XAI	240
Что такое CFE?	240
Применение CFE	240
CFE с помощью Alibi	241
Контрфактуал для задач регрессии	248
Заключение	251

ГЛАВА 11. Контрастные объяснения для машинного обучения.....	252
Что такое CE для ML?	252
СЕМ, использующие Alibi.....	253
Сравнение оригинального изображения и изображения, сгенерированного автокодировщиком.....	258
Объяснения СЕМ для табличных данных	262
Заключение	267
ГЛАВА 12. Модельно независимые объяснения путем определения инвариантности прогноза	268
Что такое независимость от модели?	268
Что такое якорь?	268
Объяснения якорей с помощью Alibi	269
Якорь текста для классификации текста.....	273
Якорь изображения для классификации изображений	277
Заключение	280
ГЛАВА 13. Объяснимость модели для экспертных систем, основанных на правилах.....	281
Что такое экспертная система?	281
Прямая и обратная цепочки	282
Извлечение правил с помощью Scikit-Learn	283
Потребность в системе, основанной на правилах.....	289
Проблемы экспертной системы	290
Заключение	290
ГЛАВА 14. Объяснимость моделей для компьютерного зрения.....	291
Почему объяснимость для данных изображений?	291
Якорь изображения с помощью Alibi	292
Метод интегрированных градиентов	292
Заключение	297

Об авторе



Прадипта Мишра (Pradeepta Mishra) – руководитель отдела искусственного интеллекта в компании L&T Infotech (LTI), возглавляет группу специалистов по анализу данных, вычислительной лингвистике, машинному и глубокому обучению, участвующих в создании функций искусственного интеллекта для обработки данных. Он был два года подряд (2019, 2020) награжден премией «40 лучших специалистов по обработке данных Индии в возрасте до 40 лет», по версии журнала Analytics India. Как изобретатель, подал пять патентов, которые в настоящее время находятся на рассмотрении в разных странах мира. Является автором четырех книг, опубликованных на разных языках, включая английский, китайский и испанский. Его первая книга была рекомендована центром HSLS при Питтсбургском университете, штат Пенсильвания, США. Последняя его книга «PyTorch. Рецепты» была опубликована издательством Apress. Он выступил с основным докладом на конференции 2018 Global Data Science Conference, Калифорния, США. Выступил с более чем 500 докладами по анализу данных, машинному обучению, глубокому обучению, обработке естественного языка и искусенному интеллекту в различных университетах, на встречах, в технических институтах и на форумах, организованных сообществом. Является приглашенным преподавателем по курсам искусственного интеллекта, машинного обучения и кибербезопасности в магистратуре университета Рева, Бангалор, Индия, а также в других университетах. За последние девять лет обучал более 2 000 специалистов по анализу данных и инженеров по искусственному интеллекту.

О рецензентах



Абхишек Виджайваргия (Abhishek Vijayvargia) работает специалистом по данным и прикладным наукам в компании Microsoft. Ранее он работал во многих стартапах, связанных с машинным обучением и интернетом вещей. Разрабатывал алгоритмы искусственного интеллекта для решения различных проблем в области кибербезопасности, интернета вещей, производства, оптимизации перевозок и логистики. Он также является экспертом и активным исследователем в области объяснимого искусственного интеллекта и проектирования систем машинного обучения. Является автором работ по анализу данных, автором и рецензентом научных статей, представил свои идеи на многих конференциях по искусственному интеллекту.



Бхаратх (Bharath) имеет более чем десятилетний опыт работы, в настоящее время он старший инженер-консультант по науке о данных в компании Verizon, Бенгалуру. Имеет диплом аспиранта по анализу данных от бизнес-школы Praxis и степень магистра наук о жизни от Университета штата Миссисипи, США. Работал исследователем данных в университете Джорджии, Университете Эмори и компании Eurofins LLC. В компании Happiest Minds он работал над продуктами цифрового маркетинга на основе искусственного интеллекта и решениями на основе обработки естественного языка в сфере образования. Наряду с повседневными обязанностями является наставником и активным исследователем. Его особенно интересуют архитектуры самостоятельного, полусамостоятельного и эффективного глубокого обучения в обработке естественного языка и компьютерном зрении.

Благодарности

Эта книга основана на исследовании объяснимости моделей искусственного интеллекта на основе «черного ящика» и преобразовании решений, принимаемых моделями искусственного интеллекта, в прозрачные. Я благодарен моим друзьям и семье за то, что они побудили меня начать эту работу, упорно продолжать ее и дойти до последнего шага – опубликовать ее.

Я благодарю свою жену Праджну (Prajna) за ее постоянное ободрение и поддержку в завершении работы над книгой, помочь в расстановке приоритетов между книгой и отпуском, заботу о детях и за то, что давала мне достаточно времени для завершения книги.

Я благодарю моего редактора Дивию (Divya), которая оказывала мне постоянную поддержку на протяжении всей работы над книгой, проявляя гибкость в отношении сроков и давая мне достаточно времени, чтобы завершить начатое.

Я благодарю редакционную коллегию и Суреша (Suresh) за веру в то, что смогу написать о сложной теме объяснимости моделей, и предоставление возможности написать на эту тему.

Наконец, я благодарю своих дочерей Аарью (Aarya) и Аадию (Aadya) за то, что они любят меня и поддерживали в завершении работы над этой книгой.

Введение

Объяснимый искусственный интеллект (explainable artificial intelligent – XAI) является актуальной потребностью, поскольку все больше и больше моделей искусственного интеллекта (ИИ) используется для выработки бизнес-решений. Таким образом, эти решения также воздействуют на многих пользователей. При этом каждый пользователь может получить благоприятное или неблагоприятное воздействие. Поэтому важно знать ключевые особенности, приводящие к принятию этих решений. Часто утверждают, что модели ИИ являются по своей природе «черным ящиком», поскольку решения модели невозможно объяснить. Поэтому в промышленности они внедряются довольно медленно. Эта книга представляет собой попытку раскрыть так называемые модели «черного ящика», чтобы повысить адаптивность, интерпретируемость и объяснимость решений, принимаемых алгоритмами ИИ с использованием таких фреймворков, как библиотеки Python XAI, TensorFlow 2.0+, Keras, а также пользовательских фреймворков с использованием декораторов Python (Python Wrappers). Цель этой книги – объяснить модели ИИ на простом языке с использованием вышеупомянутых фреймворков.

Интерпретируемость и объяснимость модели – ключевые моменты этой книги. Существуют математические формулы и методы, которые обычно используются для объяснения решения, принятого моделью ИИ. Вам будут предоставлены методы, классы, фреймворки и функции программных библиотек, а также их использование для объяснения модели, прозрачности, надежности, этики, предвзятости и интерпретируемости. Если человек может понять причины решения, принятого моделью ИИ, это даст пользователю гораздо больше возможностей для внесения поправок и рекомендаций. Существует два различных типа пользователей – бизнес-пользователи и практикующие специалисты. Бизнес-пользователь хочет получить объяснение на простом языке без каких-либо статистических или математических терминов. Практик хочет знать объяснимость с точки зрения вычислений. Эта книга – попытка сбалансировать обе потребности при создании объяснений.

Эта книга начинается с введения в основы объясимости и интерпретируемости модели, этических соображений при применении ИИ и предвзятости прогнозов, генерируемых моделями ИИ. Затем вы узнаете о надежности моделей искусственного интеллекта при создании прогнозов в различных случаях использования, изучите методы и системы интерпретации линейных моделей, нелинейных моделей и моделей временных рядов, используемых в ИИ. Далее узнаете о наиболее сложных ансамблевых моделях, объясимости и интерпретируемости с использованием таких фреймворков, как Lime, SHAP, Skater, ELI5 и Alibi. Затем вы узнаете об объясимости моделей для неструктурированных данных и обработки естественного языка, связанной с задачами классификации текстов. Изучение справедливости моделей также требует моделирова-

ния сценариев «что, если» с использованием результатов прогнозирования. Об этом вы узнаете далее. Затем вы прочитаете о контрафактных и контрастных объяснениях для моделей ИИ. Вы изучите объяснимость моделей для глубокого обучения, экспертных систем на основе правил, а также объяснения, не зависящие от модели, для инвариантности предсказаний и для задач компьютерного зрения, использующих различные фреймворки ХАІ.

Сегодня у нас есть инженеры по ИИ и специалисты по анализу данных, которые обучают или создают эти модели; разработчики программного обеспечения, которые вводят эти модели в производство и в эксплуатацию; бизнес-пользователи, которые потребляют конечный результат или результат, созданный с помощью моделей; и лица, принимающие решения, которые обдумывают решения, принятые с помощью моделей. Руководители, занимающиеся продвижением проектов/продуктов ИИ, думают: «Есть ли способ добиться ясности вокруг моделей и разработчиков прогнозирующих моделей?» Био-статистики, конечно, думают, как объяснить предсказания модели и т. д. Ожидается, что будет разработана система объяснения, которая отвечает потребностям всех заинтересованных сторон, вовлеченных в процесс внедрения ИИ в реальную жизнь. В этой книге соблюден баланс между несколькими заинтересованными сторонами. Предпочтение предоставляется специалистам по анализу данных (data scientists), поскольку, если они убеждены в объясимости моделей, то смогут разъяснить далее заинтересованным сторонам бизнеса.

Чтобы сделать модели ИИ понятными для бизнес-пользователей на простом, доступном языке, потребуется некоторое время. Возможно, для решения этой задачи появится новая система. На данный момент проблема заключается в том, что специалист по анализу данных, построивший модель, не имеет полной ясности о поведении модели, и не хватает ясности в ее объяснении. Новоиспеченные специалисты по анализу данных или выпускники вузов получат огромную пользу от этой книги. Аналогичным образом эта книга будет полезна и другим инженерам по ИИ. Это развивающаяся область, объяснения в данной книге были актуальны на июль 2021 года.

ГЛАВА 1

Объяснимость и интерпретируемость модели

Мы начнем эту книгу с введения в основы объяснимости и интерпретируемости моделей, этических аспектов применения ИИ и предвзятости прогнозов, генерируемых моделями ИИ, рассмотрим надежность моделей ИИ при создании прогнозов в различных случаях использования. Затем изучим методы и системы для интерпретации линейных, нелинейных моделей и моделей временных рядов, используемых в ИИ. Далее разберем наиболее сложные ансамблевые модели, объяснимость и интерпретируемость с использованием таких фреймворков, как Lime, SHAP, Skater, ELI5 и т. д. Затем обсудим объяснимость моделей для неструктурированных данных и задач, связанных с обработкой естественного языка.

Создание основ

За последние несколько лет был достигнут огромный прогресс в области машинного обучения и глубокого обучения при создании решений на основе искусственного интеллекта (ИИ) в различных областях, в том числе в розничной торговле, банковском деле, финансовых услугах, страховании, здравоохранении, производстве и отраслях, основанных на интернете вещей. ИИ является основой многих продуктов и решений, которые появляются в связи с быстрой цифровизацией различных бизнес-функций. Причина того, что ИИ лежит в основе этих продуктов и решений, заключена в том, что интеллектуальные машины в настоящее время обладают способностями к обучению, рассуждению и адаптации. Опыта мало. Если мы сможем использовать богатый опыт, накопленный умными людьми, и отразить его с помощью применения обучения и рассуждений в компьютерах, это может значительно повысить эффективность обучения. В силу этих возможностей современные модели машинного обучения и глубокого обучения способны достичь беспрецедентных уровней производительности при решении сложных бизнес-задач, тем самым повышая эффективность бизнеса.

За последние два года появилось множество инструментов автоматического машинного обучения (AutoML – Automatic Machine Learning), фреймворков, инструментов с низким содержанием кода и без кода (с минимальным вмешательством человека), что является еще одним уровнем сложности, ко-

торого достигли системы, поддерживаемые искусственным интеллектом. Это воплощение практически нулевого вмешательства человека, необходимого с точки зрения разработки, поставки и развертывания решений. Когда решения полностью принимаются машинами, а люди всегда находятся на стороне получателя, возникает острая необходимость понять, как машины пришли к этим решениям. Модели, на которых основаны системы ИИ, часто называют моделями «черного ящика». Следовательно, существует необходимость в объяснимости и интерпретируемости моделей для того, чтобы объяснить сделанные ими прогнозы.

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Искусственный интеллект означает систему в виде компьютерной программы, которая автоматически принимает решения от имени человека в отношении некоторой задачи, без явного программирования. Рисунок 1.1 объясняет взаимосвязь между машинным обучением (machine learning – ML), глубоким обучением (deep learning – DL) и искусственным интеллектом.

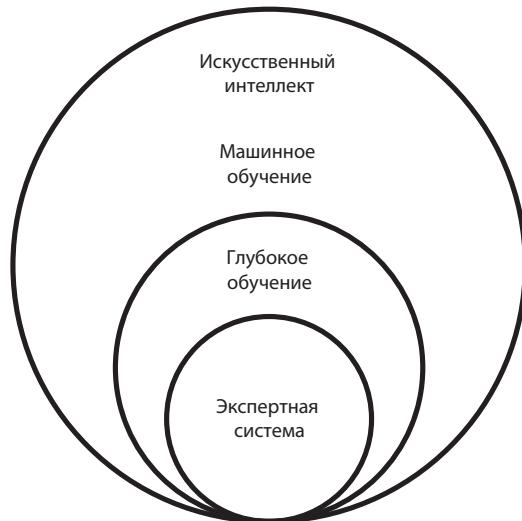


Рис. 1.1. Взаимосвязь между ML, DL и ИИ

Искусственный интеллект – это система, созданная с помощью компьютерной программы, в которой интеллектуальные выводы могут быть сделаны в отношении постановки задачи. В процессе получения вывода могут быть использованы алгоритмы машинного обучения или глубокого обучения. Алгоритмы машинного обучения – это простые математические функции, используемые в процессе оптимизации с использованием комбинаций входных и выходных данных. Кроме того, эти функции могут быть использованы для предсказания неизвестного выхода с использованием новых входных данных. Для структурированных данных мы можем использовать алгоритмы машинного обучения, но, когда размеры и объем данных, таких как изображения, аудиоданные, текстовые и видеоданные, увеличиваются, модели машинного

обучения не могут хорошо работать, поэтому требуется модель глубокого обучения. Экспертная система разработана как система, основанная на правилах, которая помогает в получении выводов. Это требуется, когда нет достаточного количества обучающих данных для обучения моделей машинного или глубокого обучения. В целом создание системы искусственного интеллекта требует для создания выводов сочетания экспертных систем, алгоритмов машинного обучения и алгоритмов глубокого обучения.

Машинное обучение можно определить как систему, в которой алгоритм обучается на примерах в отношении некоторой задачи, определенной ранее, и эффективность обучения увеличивается по мере ввода в систему все большего количества данных. Задачи могут быть определены как контролируемые, где выход/результат известен заранее; неконтролируемые, где выход/результат не известен заранее; и с подкреплением, где действия/результаты всегда определяются средой обратной связи, а обратная связь может быть вознаграждением или штрафом. Что касается алгоритмов обучения, то их можно разделить на следующие категории – линейные, детерминированные, аддитивные и мультиплективные, алгоритмы на основе деревьев, ансамблевые алгоритмы и на основе графов. Критерии эффективности могут быть определены в соответствии с выбором алгоритма. Объяснение решений модели ИИ называется объяснимым ИИ (explainable ИИ – XAI).

Необходимость XAI

Рассмотрим причину, по которой модели ИИ называют моделями «черного ящика». Рисунок 1.2 объясняет классический сценарий моделирования, когда набор независимых переменных передается через функцию, которая предопределена для получения выхода. Полученный результат сравнивается с истинным результатом, чтобы оценить, соответствует ли функция данным или нет. Если функция плохо подходит, то необходимо либо преобразовать данные, либо рассмотреть возможность использования другой функции, чтобы она подходила к данным. Но эксперимент ручной, и каждый раз, когда происходит обновление данных, статистикам или специалистам по моделированию приходится заново калибровать модели и снова проверять, соответствуют ли данные модели. Вот почему классический способ создания прогностических моделей для обработки выводов зависит от человека и всегда подлежит более чем одной интерпретации. Временами заинтересованным сторонам трудно доверять модели, так как все варианты, предложенные многими экспертами, могут звучать в определенном смысле хорошо, но нет обобщения. Таким образом, классическую систему разработки моделей, показанную на рис. 1.2, сложно реализовать в мире систем искусственного интеллекта, где данные постоянно меняются. Зависимость калибровки от человека является узким местом, поэтому существует необходимость в современной системе генерации выводов с использованием динамических алгоритмов.

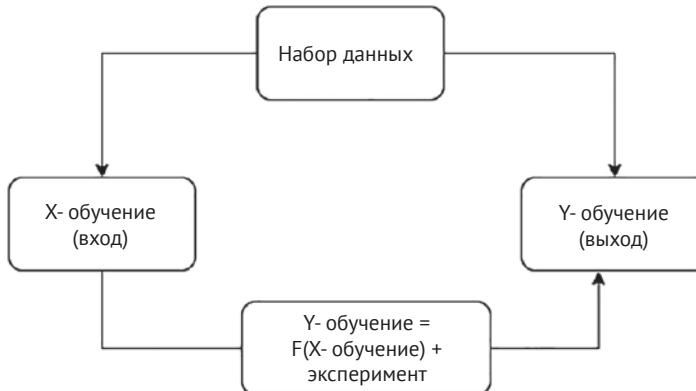


Рис. 1.2. Классическая система обучения модели

На рис. 1.2 показан классический сценарий разработки модели, когда модель может быть представлена через уравнение и это уравнение легко интерпретировать и просто объяснить кому угодно, но существование интерпретации на основе формул не всегда возможно в мире ИИ. На рис. 1.3 показана структура поиска наилучших возможных функций, которые производят выход, используя входные данные. Здесь нет ограничений модели конкретной функцией, например линейной или нелинейной. В этой структуре обучение происходит через множество итераций, и с помощью перекрестной проверки определяется лучшая модель. Проблема с моделями ИИ заключается в интерпретируемости и объяснимости, так как многие алгоритмы сложны, и поэтому нелегко объяснить прогнозы каждому.

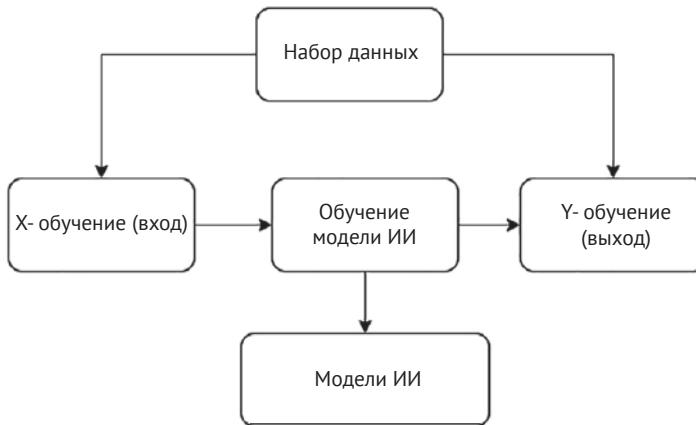


Рис. 1.3. Процесс обучения модели ИИ

С развитием компьютерных программ и алгоритмов разработчикам стало очень трудно искать различные виды линейных и нелинейных функций, и оценка таких функций также стала чрезвычайно сложной. Модели машин-

ного или глубокого обучения берут на себя поиск функций, которые хорошо подходят к обучающим данным. Рисунок 1.3 поясняет, что машина определяет окончательную модель, обеспечивающую лучшую эффективность не только с точки зрения точности, но также стабильности и надежности при генерировании прогнозов. Когда функциональная связь между входом и выходом четко определена, возникает меньше двусмысленности и прогнозы становятся прозрачными. Однако, когда модели ИИ делают выбор сложной функциональной связи, это очень трудно понять конечному пользователю. Поэтому модели ИИ считаются «черным ящиком». В этой книге мы хотим сделать модель «черного ящика» интерпретируемой, чтобы решения ИИ становились все более развернутыми и адаптируемыми.

Повседневное использование моделей ИИ для принятия решений требует прозрачности, непредвзятости и этики. Существуют различные сценарии, в которых в настоящее время не хватает объяснности:

- кто-то подает заявку на получение кредитной карты, и модель ИИ отклоняет его заявку. Важно объяснить, почему заявка была отклонена и какие корректирующие действия может предпринять заявитель, чтобы изменить свое поведение;
- в медицинской диагностике, основанной на образе жизни и жизненно важных параметрах, модель ИИ предсказывает, будет ли у человека диабет, или нет. Здесь если модель предсказывает, что у человека может развиться диабет, то она также должна объяснить, почему и каковы факторы, способствующие развитию заболевания в будущем;
- автономные транспортные средства идентифицируют объекты на дороге и принимают четкие решения. В этом случае также необходимо четкое объяснение того, почему они принимают эти решения.

Существует множество других примеров использования, где объяснения, подтверждающие вывод модели, имеют решающее значение. Человеку свойственно не принимать то, что он не может интерпретировать или понять. Поэтому снижается фактор доверия к прогнозам модели ИИ. Мы используем модели ИИ для устранения предвзятости при принятии решений человеком. Однако решения будут опасными, если результат не будет оправданным, правомерным и прозрачным. С другой стороны, можно утверждать, что если мы не можем интерпретировать и объяснить решения моделей ИИ, то зачем их использовать. Причинами их использования являются точность и эффективность моделей. Всегда будет существовать компромисс между эффективностью модели и ее объяснностью. Рисунок 1.4 объясняет компромисс между этими двумя понятиями.

На рис. 1.4 горизонтальная ось показывает производительность или точность модели, а вертикальная – интерпретацию и объяснение модели. Система, основанная на правилах, находится в позиции, где эффективность не является оптимальной, однако интерпретируемость хороша. Напротив, модели на основе глубокого обучения обеспечивают превосходную эффективность и хорошую точность при меньшей интерпретируемости и объяснности.

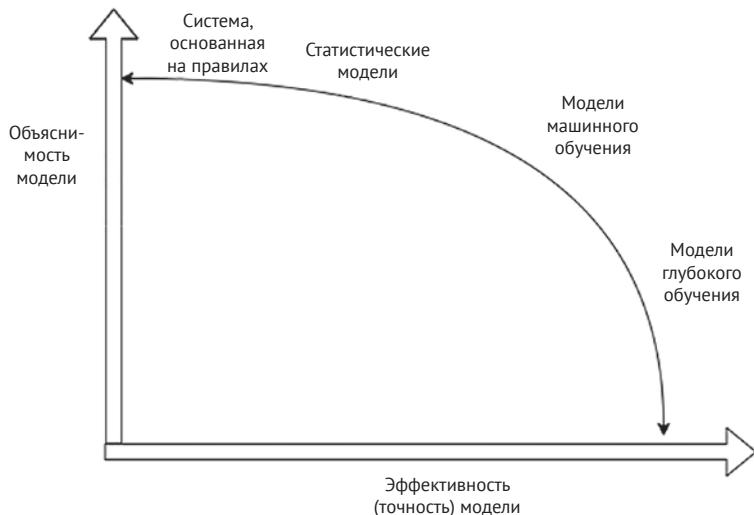


Рис. 1.4. Компромисс между объяснимостью и эффективностью (точностью) модели

СРАВНЕНИЕ ОБЪЯСНИМОСТИ С ИНТЕРПРЕТИРУЕМОСТЬЮ

Существует разница между интерпретируемостью и объяснимостью модели. Интерпретация – это смысл прогнозов. Объяснимость – это то, почему модель предсказывает что-либо и почему кто-то должен доверять модели. Чтобы лучше понять разницу, давайте рассмотрим реальный пример прогноза продаж, где факторами, которые помогают в прогнозировании, являются расходы на рекламу, качество продукции, источники для создания рекламы, размер рекламы и т. д. Каждый фактор имеет коэффициент после выполнения регрессионного моделирования. Коэффициенты могут быть интерпретированы как приращение продаж в результате дельта-изменения одного фактора, например расходов на рекламу. Однако, если вы прогнозируете, что продажи в следующем месяце составят 20 000 долл., когда среднемесячные продажи исторически были меньше или равны 15 000 долл., это требует объяснения. В рамках объяснимости модели нам необходимо следующее:

- интерпретируемость модели должна обеспечить естественность принятия решений и отсутствие предвзятости в прогнозе;
- дифференциация ложной причинности от истинной причинности, которая помогает сделать прогнозы прозрачными;
- необходимость создания объяснимых моделей без ущерба для высокой производительности опыта обучения и итераций;
- возможность лицам, принимающим решения, доверять моделям ИИ.

«Приложения и продукты XAI станут новым трендом в 2021 году, поскольку спрос на интерпретируемость, доверие и этику в ИИ громкий и ясный». – Прадипта Мишра (Pradeeptha Mishra; источник – различные научно-исследовательские отчеты).

Вокруг XAI постоянно ведутся исследования, которые делают все возможное, чтобы объяснить конечным пользователям модели ИИ и их поведение с целью повысить уровень принятия моделей. Теперь возникает вопрос – кто является конечными пользователями XAI? Ими являются:

- кредитные инспекторы, которые оценивают заявки на получение кредитов, кредитные запросы и многое другое. Если они понимают, какие решения принимаются, то могут помочь обучить клиентов корректировать свое поведение;
- специалисты по анализу данных, оценивающие собственные решения и обеспечивают введение улучшений в модели. Является ли это лучшей моделью, которую можно сделать, используя имеющийся набор данных?
- старшие менеджеры, которым необходимо соответствовать нормативным требованиям на высоком уровне;
- руководители предприятий, кому необходимо доверять решениям «черного ящика» ИИ и кто ищет любые исторические свидетельства, на которые они могут положиться;
- руководители службы поддержки клиентов, кому необходимо отвечать на жалобы и объяснять решения;
- внутренние аудиторы и регуляторы, обязанные обеспечить прозрачность процесса, основанного на данных.

Целью XAI является достижение следующих показателей:

- **доверия:** точность прогноза является четкой функцией качества данных, истинности причинно-следственной связи и выбора подходящего алгоритма. Однако модели могут генерировать ложные срабатывания в процессе прогнозирования. Если модели генерируют много ложных срабатываний, то конечный пользователь потеряет доверие к модели. Таким образом, важно передать доверие к модели конечному пользователю;
- **ассоциаций:** модели ML или DL учатся делать прогнозы на основе ассоциаций между различными признаками. Ассоциации могут быть корреляциями или просто ассоциациями. Необъяснимые корреляции являются ложными корреляциями, которые делают модель невозможной для интерпретации. Следовательно, важно уловить истинные корреляции;
- **надежности:** уверенность в модели, стабильность модели в прогнозах, а также устойчивость модели также очень важны. Это необходимо для того, чтобы модели ИИ вызывали больше доверия, и для того, чтобы конечный пользователь был достаточно уверен в прогнозах модели. Если этого нет, то ни один пользователь не будет доверять моделям;
- **справедливости:** модели ИИ должны быть справедливыми и соответствовать этическим нормам. Они не должны дискриминировать религию, пол, класс и расу при генерировании прогнозов;
- **идентичности:** модели ИИ должны быть способны сохранять соображения конфиденциальности без раскрытия личности человека. Конфиденциальность и управление идентификацией при создании XAI очень важны.

Типы объяснимости

Интерпретируемость машинного обучения является неотъемлемой частью объяснимости модели. Существуют различные классификации интерпретаций моделей:

- **внутреннее объяснение:** в эту категорию попадают простые модели, такие как простые линейные регрессионные модели и модели на основе дерева решений, где простое условие «если/иначе» (if/else) может объяснить предсказания. Это означает, что XAI присущ самой модели, и нет необходимости проводить какой-либо постанализ;
- **объяснение post-hoc:** сложные модели, такие как нелинейные, ансамблевые древовидные, стохастическая древовидная модель с усилением градиента и стековые модели, где необходимо уделить больше внимания созданию объяснимости;
- **конкретная модель:** существует набор объяснений, которые могут быть получены из конкретного типа модели, не более того. Например, модель линейной регрессии не обеспечивает значимость признаков. Однако коэффициенты линейной регрессионной модели кто-то может использовать в качестве косвенного показателя;
- **не зависящие от модели:** эти объяснения кто-то может интерпретировать, глядя на пару комбинаций обучающих входных данных и обучающих выходных данных. В этой книге мы рассмотрим объяснения, не зависящие от модели, в последующих главах;
- **локальная интерпретация:** дает представление об отдельных предсказаниях, что является интерпретацией одной точки данных. Например, если заемщик предсказан моделью как склонный к дефолту, то почему это так? Это локальная интерпретация;
- **глобальная интерпретация:** дает представление о глобальном понимании прогнозов для всех точек данных, общем поведении модели и многом другом;
- **сублокальная интерпретация:** объясняет локальные интерпретации для группы точек данных, а не всех точек данных. Это отличается от локальной интерпретации;
- **текстовые объяснения:** включают числовую часть, а также язык для передачи значения определенных параметров модели;
- **визуальные объяснения:** визуальные объяснения хороши, но иногда они недостаточно интуитивны для объяснения прогнозов, поэтому очень необходимы визуальные объяснения, сопровождаемые текстовой интерпретацией.

Инструменты для объяснимости моделей

Существуют различные инструменты и механизмы для создания объяснимости из ML- и DL-моделей. Библиотеки Python с открытым исходным кодом имеют некоторые преимущества и недостатки. В примерах на протяжении всей этой

книги мы будем использовать сочетание библиотек Python с открытым исходным кодом и общедоступных наборов данных с различных веб-сайтов. Далее перечислены инструменты, которые необходимо установить, и среда, которую необходимо настроить.

SHAP

Библиотека SHAP (SHapley Additive exPlanations) – это унифицированный подход на базе Python для объяснения результатов любой модели машинного обучения. Библиотека SHAP Python основана на теории игр с локальными объяснениями. Подход теории игр – это способ получить прогнозы при наличии одного фактора по сравнению с его отсутствием. Если происходит значительное изменение в ожидаемом результате, значит, фактор очень важен для целевой переменной. Этот метод объединяет несколько предыдущих методов для объяснения результатов, генерируемых моделями машинного обучения. Фреймворк SHAP может быть использован для различных типов моделей, за исключением моделей на основе временных рядов (см. рис. 1.5). Библиотека SHAP может быть использована для осмысливания моделей.

Для установки SHAP можно использовать следующие методы:

<pre>! Pip install shap conda install -c conda-forge shap ! pip3 install shap</pre>	(из ноутбука Jupyter) (с помощью терминала)
---------------------------------------------------------------------------------------------	------------------------------------------------

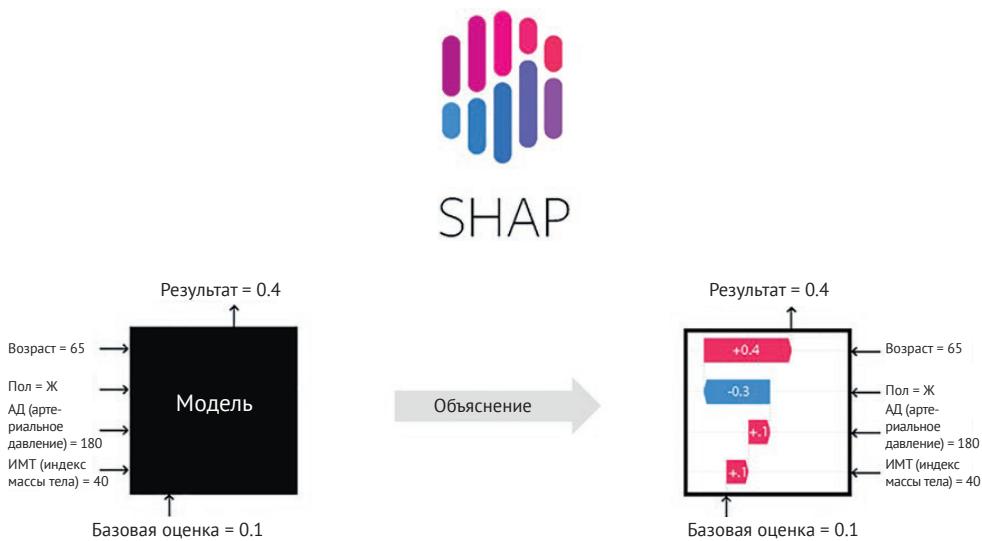


Рис. 1.5. Значения Шепли и пример объяснения изображения

LIME

LIME расшифровывается как локальные интерпретируемые не зависящие от модели объяснения (Local Interpretable Model-Agnostic Explanations). Локальное относится к объяснению локальности класса, который был предсказан мо-

делью. Поведение классификатора при локальности дает хорошее понимание прогноза. Интерпретируемость означает, что если предсказание не может быть интерпретировано человеком, то в нем нет смысла. Следовательно, предсказания классов должны быть интерпретируемыми. Независимость от модели подразумевает, что вместо понимания конкретного типа модели система и метод должны быть способны генерировать интерпретации.

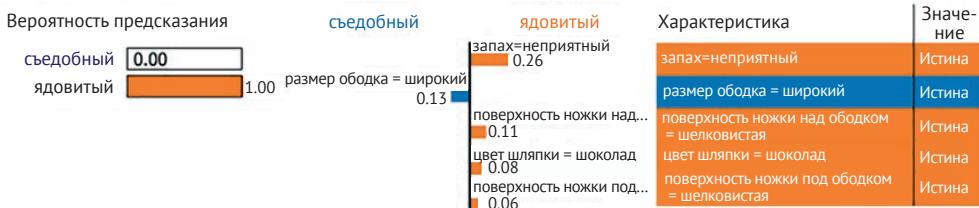


Рис. 1.6. Пример классификации грибов

Проблема классификации текста (например, анализ настроения) – это когда на вход подаются предложения в виде документов, а на выходе получается класс (см. рис. 1.6). Когда модель предсказывает положительное настроение для предложения, нам необходимо знать, какие слова заставили модель предсказать класс как положительный. Эти векторы слов иногда очень просты, например отдельные слова. Иногда они сложны (например, вкрапления слов), и в этом случае нужно знать, как модель интерпретировала вкрапления слов и как это влияет на классификацию. В этих сценариях LIME чрезвычайно полезна для понимания смысла моделей машинного обучения и глубокого обучения. LIME – это библиотека на базе Python, которую можно использовать для демонстрации ее работы. Для ее установки необходимо выполнить следующее:

```
! pip install lime
```

ELI5

ELI5 – это библиотека на базе Python, предназначенная для создания объяснимого конвейера ИИ, который позволяет визуализировать и отлаживать различные модели машинного обучения с помощью унифицированного API. Она имеет встроенную поддержку нескольких ML-фреймворков и предоставляет возможность объяснять «черные ящики» моделей. Цель библиотеки – сделать объяснения простыми для всех видов моделей «черного ящика» (см. рис. 1.7).

Вес	Характеристика
0.3717	взаимосвязь
0.1298	семейное положение
0.1247	длительность обучения
0.1108	прирост капитала
0.0611	потеря капитала
0.0362	возраст
0.0307	занятость
0.0298	пол
0.0289	количество рабочих часов в неделю
0.0188	рабочий класс
0.0161	родная страна
0.0160	раса
0.0132	вес выборки ¹
0.0123	образование

Рис. 1.7. Пример изображения в ELI5

Рисунок 1.7 из ELI5 показывает важность факторов в прогнозировании дохода класса в примере использования классификации доходов, который мы рассмотрим в последующих главах. Установка ELI5 на Python может быть выполнена с помощью следующего синтаксиса:

```
!pip install eli5
```

Это требует обновления многих библиотек на базе Python, и вам, возможно, придется подождать некоторое время, пока это произойдет.

SKATER

Skater – это унифицированный фреймворк с открытым исходным кодом, позволяющий интерпретировать модели для всех форм моделей, чтобы помочь построить интерпретируемую систему машинного обучения, что часто необходимо для использования в реальном мире. Skater поддерживает алгоритмы для прояснения изученных структур модели «черного ящика» как глобально (вывод на основе полного набора данных), так и локально (вывод на основе индивидуального прогноза).

¹ Примерная оценка количества людей, которое представляет каждая строка данных

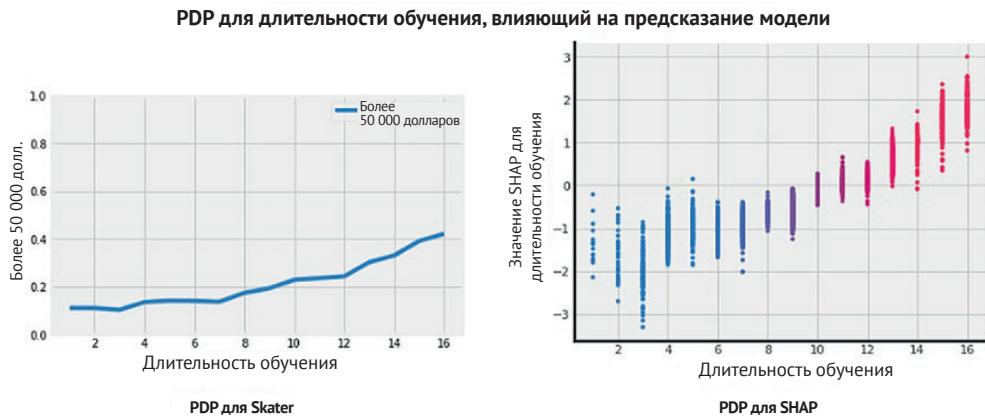


Рис. 1.8. Пример изображения, показывающего PDP² с использованием SHAP и Skater

Skater позволяет реализовать это видение, предоставляя по мере необходимости возможность выводить и отлаживать политики принятия решений модели, т. е. обеспечивая «участие человека в процессе» (см. рис. 1.8). Чтобы установить библиотеку Skater, можно использовать следующую команду:

```
!pip install skater
```

SKOPE_RULES

Skope-rules нацелен на изучение логических, интерпретируемых правил для «обследования» целевого класса (т. е. обнаружения с высокой точностью экземпляров этого класса). Skope-rules – это компромисс между интерпретируемостью дерева решений и моделирующей способностью случайного леса (см. рис. 1.9).

Вышеупомянутые библиотеки на базе Python в основном имеют открытый исходный код и бесплатны для использования и интеграции в любое программное приложение. Однако существует множество корпоративных инструментов и фреймворков, таких как H2O.ai. Область XAI является относительно новой, так как еще ведутся исследования в направлении упрощения интерпретации моделей. Инструменты и фреймворки выходят на передний план, чтобы сделать этот процесс доступным для применения в промышленности.

² PDP (Partial Dependence Plot) – график частичной зависимости, показывает краевой эффект одного или двух признаков на прогнозируемый результат модели машинного обучения. – Прим. перев.

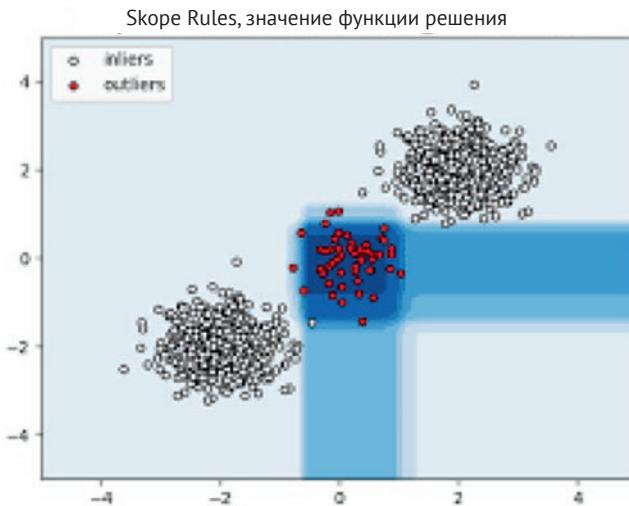


Рис. 1.9. Пример изображения Skope-rules

Методы XAI для ML

Уровни прозрачности моделей машинного обучения можно разделить на три группы – прозрачность алгоритмов, декомпозицию параметров и гиперпараметров и воспроизводимость одного и того же результата в аналогичных ситуациях. Некоторые модели ML по своей конструкции интерпретируемые, а некоторые требуют набора других программ, чтобы сделать их объяснимыми. Рисунок 1.10 объясняет методы объяснимости моделей.

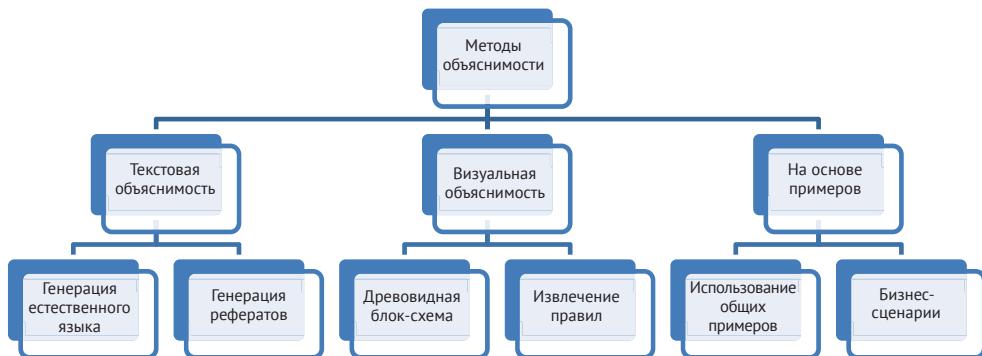


Рис. 1.10. Методы XAI

Существует три метода объяснения модели:

- текстовое объяснение требует уточнения значения математической формулы, параметра модели или метрик, определенных моделью. Интерпретации могут быть разработаны на основе определенного шаблона, где сюжетная линия должна быть подготовлена заранее и только параметры должны быть введены в шаблон. Существует два различных

подхода для достижения этой цели – использование методов генерации естественного языка (Natural Language Generation – NLG), для чего необходимо собрать текст и составить предложение, описывающее объект, и использование генерации реферата;

- визуальное объяснение может быть предоставлено с помощью пользовательских графиков и диаграмм. Древовидные графики довольно просты и понятны для конечных пользователей. Каждый древовидный метод опирается на набор правил. Если эти правила могут быть показаны пользователю в виде простых инструкций «если/иначе» (*if/else*), то это будет гораздо более мощным;
- метод, основанный на примерах, гарантирует, что мы можем взять обычные повседневные примеры для объяснения модели путем проведения параллелей. Также для объяснения моделей можно использовать обычный бизнес-сценарий.

Совместимые с XAI модели

Давайте посмотрим на текущее состояние моделей, их характер, насколько они совместимы с XAI, и нужны ли этим моделям для объяснимости дополнительные фреймворки:

- **линейные модели** – модели линейной регрессии или логистической регрессии легко интерпретировать, анализируя значение их коэффициента, который представляет собой число. Эти значения очень легко интерпретировать. Однако, если мы распространим это на регуляризованные регрессионные семейства, это становится очень сложным для объяснения. Обычно мы придаём большее значение отдельным признакам. Мы не учтываем особенности взаимодействия. Сложность модели возрастает, если мы включаем взаимодействия, такие как аддитивные, мультиплективные, полиномиальные взаимодействия второй или третьей степени. В этих сложных сценариях математический результат требует более простой интерпретации;
- **модели прогнозирования временных рядов** – это также очень простые модели, они следуют регрессионному типу сценария, который легко объяснить с помощью параметрического подхода;
- **модели на основе деревьев** более просты для анализа и также очень интуитивно понятны для интерпретации человеком. Однако эти модели часто не обеспечивают лучшей точности и производительности. Им также не хватает устойчивости, и они имеют присущие им проблемы предвзятости и чрезмерной подгонки. Поскольку недостатков так много, их интерпретация не имеет смысла для конечного пользователя;
- **ансамблевые модели** – существуют три различных типа ансамблевых моделей: упаковка, повышение и укладка. Все три типа не обладают достаточной объяснимостью. Необходимо упрощенное описание, чтобы передать результаты модели. Важность признаков также должна быть упрощена;

- **математические модели** – машины, поддерживающие векторную математику, используются для задач, основанных на регрессии и классификации. Эти модели довольно сложны для объяснения, поэтому очень важно упрощение модели;
- **модели глубокого обучения** – модели глубоких нейронных сетей (Deep Neural Network – DNN) обычно имеют более трех скрытых слоев. Помимо слоев модели глубокого обучения, существуют различные параметры настройки модели, такие как веса, типы регуляризации, сила регуляризации, типы функций активации для различных слоев, типы функций потерь, используемые в модели, и алгоритмы оптимизации, включающие скорость обучения и параметры импульса. Все это очень сложно по своей природе и требует упрощенной структуры для интерпретации;
- **конволюционная нейронная сеть** (Convolutional Neural Network – CNN) – это еще один тип нейросетевой модели, которая обычно применяется для обнаружения объектов и задач, связанных с классификацией изображений. Она рассматривается как полная модель «черного ящика». В ней есть слои свертки, максимальное или среднее число слоев объединения и многое другое. Если кто-то спросит, почему эта модель предсказала кошку как собаку, можем ли мы объяснить, что пошло не так? В настоящее время ответ отрицательный. Требуется большая работа по объяснению этой модели конечному пользователю;
- **рекуррентные нейронные сети** (Recurrent Neural Networks – RNNs) – рекуррентные нейронные модели обычно применяются для классификации текстов и предсказаний текста. Существуют различные варианты, такие как сеть с долговременной краткосрочной памятью (long short term memory – LSTM) и двунаправленные LSTM, очень сложные для объяснения. Есть постоянная потребность в более совершенных структурах и методах, которые можно использовать для объяснения таких моделей;
- **модели, основанные на правилах**, – это очень простые модели, поскольку нам нужны только условия if/else для создания таких моделей.

Объяснимый удовлетворяет требованиям ответственного ИИ

Ответственный искусственный интеллект – это структура, в которой объяснимость, прозрачность, этика и подотчетность обеспечиваются в различных программных приложениях, цифровых решениях и продуктах. Развитие искусственного интеллекта стремительно создает множество возможностей в различных областях, где эти технологии затрагивают жизнь простых людей, поэтому искусственный интеллект должен быть ответственным, а решения – объяснимыми.

Семь основных принципов ответственного искусственного интеллекта являются критически важной частью объясимости (рис. 1.11). Давайте рассмотрим каждый из них.

- **Справедливость.** Предсказания, генерируемые системами ИИ, не должны приводить к дискриминации людей по их касте, вероисповеданию, религии, полу, политическим убеждениям, этнической принадлежности и т. д., поэтому требуется большая степень справедливости.



Рис. 1.11. Основные принципы ответственного ИИ

- **Этика.** В стремлении к построению интеллектуальных систем мы не должны забывать об этике при сборе данных.
- **Прозрачность.** Прогнозы моделей и методы их генерации должны быть прозрачными.
- **Конфиденциальность.** При разработке систем ИИ должны быть защищены персональные данные (персонализированная идентифицируемая информация – personalized identifiable information – PII).
- **Информационная безопасность.** Интеллектуальные системы должны быть безопасными.
- **Ответственность.** В случае ошибочного прогноза модель ИИ должна быть способна взять на себя ответственность за устранение проблемы.
- **Безопасность.** Когда модели ИИ принимают решения по навигации самодвижущихся автомобилей, роботизированной стоматологической хирургии и медицинской диагностике, любой неверный прогноз может привести к опасным последствиям.

Многие организации находятся в процессе подготовки руководящих принципов и стандартов для использования ИИ в своих решениях, чтобы избежать непреднамеренных негативных последствий в будущем. Возьмем организацию А. Она использует ИИ для прогнозирования объема продаж. ИИ предсказывает, что объем продаж будет на 30 % выше среднего, поэтому предприятие делает запасы продукта и мобилизует рабочую силу для поддержки продаж. Но если фактические продажи окажутся на уровне среднего исторического уровня продаж, то создание этих запасов было напрасным. Здесь прогноз ИИ оказался неверным. С помощью объяснимости модели эта ситуация могла бы быть проанализирована, и, возможно, модель могла быть исправлена.

ОЦЕНКА XAI

Не существует единого стандарта для оценки различных объяснений, генерируемых библиотеками на основе Python через интернет. Процесс XAI должен следовать следующим шагам при оценке объяснений:

- каждая страта должна иметь отдельное объяснение. Если у нас есть набор данных, который не может быть использован для обучения модели из-за большого объема, мы обычно делаем выборку из этого набора данных. Если мы используем стратифицированную выборку, то каждая страта должна иметь отдельное объяснение;

- **ограничения по времени.** Мы знаем, что реальные наборы данных достаточно велики. Даже если мы работаем с распределенными вычислительными системами, объяснения, генерируемые библиотеками XAI, как правило, не должны занимать много времени;
- **инвариантность экземпляра.** Если точки данных идентичны по своим атрибутам, они должны быть частью одной и той же группы и, следовательно, должны давать схожие интерпретации.

В различных проектах и инициативах в области ИИ, когда мы делаем прогнозы, часто возникает вопрос, почему кто-то должен доверять нашей модели. В предиктивной аналитике, машинном или глубоком обучении существует компромисс между тем, что было предсказано, и тем, почему это было предсказано. Если предсказание соответствует ожиданиям человека, то это хорошо. Если оно выходит за рамки человеческих ожиданий, то нам нужно знать, почему модель приняла такое решение. Прогнозирование и отклонение от ожиданий – это нормально, если речь идет о сценарии с низким уровнем риска, например о таргетировании клиентов, цифровом маркетинге или рекомендации контента. Однако в условиях высокого риска, таких как клинические испытания или система тестирования лекарств, незначительное расхождение между прогнозом и ожиданиями имеет большое значение, и возникнет множество вопросов о том, почему модель сделала такой прогноз. Как человеческие существа, мы считаем себя выше всех. Возникает любопытство, как модель пришла к такому прогнозу и почему этого не сделал человек. Система XAI – это отличный инструмент для выявления предвзятости, присущей процессу машинного обучения. XAI помогает нам выяснить, где именно появляется предвзятость.

Поскольку многие люди не в состоянии объяснить результаты работы модели машинного обучения, они не могут обосновать решения модели и поэтому не готовы к использованию моделей ИИ (рис. 1.12). Эта книга является попыткой популяризировать концепцию фреймворков XAI для отладки моделей машинного и глубокого обучения, чтобы повысить уровень внедрения ИИ в промышленности. В средах с высоким уровнем риска при использовании существуют нормативные требования и требования аудита для обоснования решения модели. Книга организована таким образом, чтобы обеспечить практическое выполнение фреймворков XAI для задач, связанных с контролируемой регрессией, классификацией, обучением без учителя, кластеризацией и сегментацией. Кроме того, некоторые модели прогнозирования временных рядов нуждаются в XAI для описания прогнозов. Далее, фреймворки XAI могут быть использованы для решения задач классификации неструктурированного текста. Один фреймворк XAI не подходит для всех типов моделей, поэтому мы собираемся обсудить различные типы библиотек Python с открытым исходным кодом и их использование для создания объяснений на основе XAI.

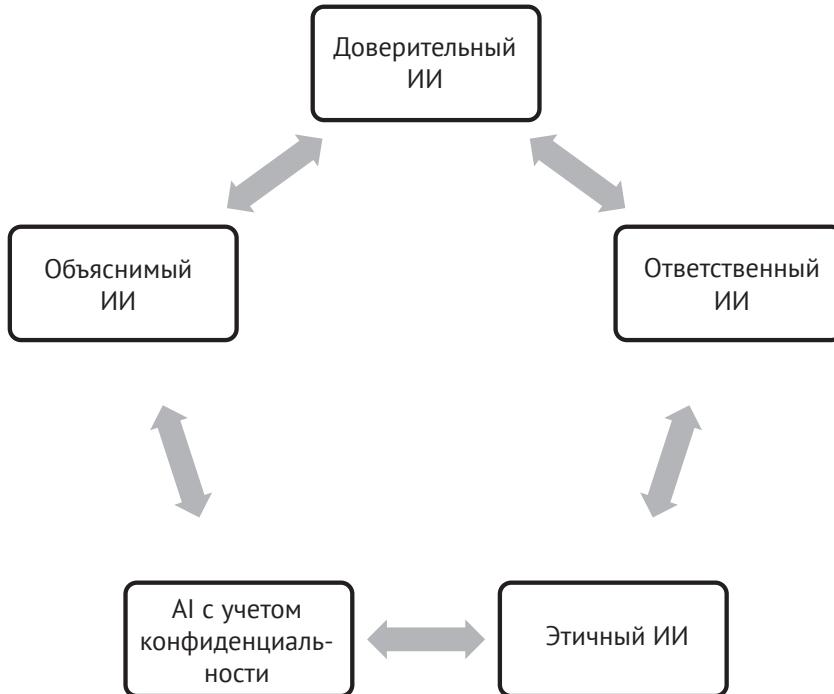


Рис. 1.12. Что необходимо для повышения уровня внедрения ИИ

Изучение справедливости модели также требует моделирования сценариев «что, если» (what-if) с использованием результатов прогнозов. Мы рассмотрим и это. Затем обсудим контрафактные и контрастные объяснения для моделей ИИ. Мы рассмотрим объяснимость для моделей глубокого обучения, экспертных систем, основанных на правилах, а также объяснения, не зависящие от модели, для инвариантности предсказаний и для задач компьютерного зрения, использующих различные фреймворки XAI. Интерпретируемость и объяснимость моделей – ключевые темы этой книги. Существуют математические формулы и методы, которые обычно используются для объяснения решений, принимаемых моделями ИИ. Читателям предоставляются методы программных библиотек, классы, фреймворки и функции, а также методы их использования для объяснения моделей, прозрачности, надежности, этики, предвзятости и интерпретируемости. Если человек может понять причины решения, принятого моделью ИИ, это даст пользователю гораздо больше возможностей для внесения поправок и рекомендаций.

ЗАКЛЮЧЕНИЕ

Интерпретируемость и объяснимость модели необходимы для всех процессов, использующих ИИ для прогнозирования чего-либо, потому что нам нужно знать причины, стоящие за прогнозом. В этой главе вы узнали следующее:

- основы объяснимости и интерпретируемости моделей;

-
- этические соображения при применении ИИ и предвзятость прогнозов, генерируемых моделями ИИ;
 - надежность моделей ИИ при создании прогнозов в различных случаях использования;
 - методы и системы для интерпретации линейных моделей, которые используются в ИИ, нелинейные модели и модели временных рядов, используемые в ИИ;
 - наиболее сложные ансамблевые модели, объяснимость и интерпретируемость с использованием таких фреймворков, как Lime, SHAP, Skater, ELI5 и др.;
 - объяснимость моделей для неструктурированных данных и задач, связанных с обработкой естественного языка.

ГЛАВА 2

Этика, предвзятость и надежность ИИ

В этой главе рассматриваются различные фреймворки, использующие библиотеки Python по объяснимому искусственному интеллекту (Explainable Artificial Intelligence – XAI) для контроля предвзятости, выполнения принципов надежности и поддержания этики при создании прогнозов. По мере того как цифровизация затрагивает различные отрасли, открываются совершенно новые возможности для применения искусственного интеллекта и решений, связанных с машинным обучением. Основными проблемами с принятием этих технологий ИИ являются этика, предвзятость, надежность системы и прозрачность процессов. Системы ИИ поднимают фундаментальные вопросы. Могу ли я доверять прогнозам, сделанным системой ИИ? Могу ли я считать, что она непредвзята? Какой риск она несет? Существует ли процесс, в котором я могу контролировать системы ИИ и их прогнозы на будущее? Этика ИИ больше сосредоточена на законном использовании ИИ для человеческого рода и его благополучия. ИИ не должен использоваться для уничтожения человеческой цивилизации. В этой главе мы обсудим различные аспекты ИИ.

Основы этики ИИ

ИИ – относительно новая область, и она все больше совершенствуется различными правительствами по всему миру, поскольку они видят свидетельства несоответствий. ИИ широко известен как интеллектуальное поведение, демонстрируемое компьютерами для достижения конечных целей. Цели всегда определяются бизнесом, например в контексте логистики, для организации склада или определения робота и его действий по подъему и перемещению упаковок на складе. Или это может быть робот, который способен стрелять, наблюдая угрозу со стороны человека в ситуации самообороны. Эти сценарии совершенно разные. В первом использование искусственного интеллекта является допустимым. Во втором сценарии это не лучшее из возможных применений ИИ. Системы ИИ могут рассуждать, думать, воспринимать действия и реакции и, следовательно, могут быть обучены действиям. Главная цель ИИ – создание машин, которые могут помочь в восприятии, логических рассуждениях, играх, принятии решений с помощью методов мо-

делирования, понимании обработки естественного языка, генерации текста, подобно людям, и многом другом.

Существует огромное количество дебатов на тему этики ИИ в академических кругах, между политиками в различных публичных выступлениях, а также практиками. Есть что сказать, но мало оснований для выработки определенной политики в отношении этики ИИ. Это происходит потому, что это трудно сделать. ИИ развивается, и поэтому технологическую политику трудно планировать и проводить в жизнь.

Очень трудно заранее оценить влияние технологии ИИ на жизнь людей, угрозу применения ИИ в армии и других областях. Поэтому выработка политики в отношении использования ИИ с этической точки зрения порой носит итеративный характер. Поскольку различные организации используют ИИ в разных целях, среди представителей деловых кругов нет единого мнения о том, как следует использовать ИИ и что правильно с этической точки зрения. Каждое правительство мира хочет доминировать и быть могущественным с помощью технологии ИИ, поэтому нет консенсуса в отношении построения общей политики по этическому использованию ИИ (рис. 2.1). Это повторяет путь ядерных технологий в прошлом.

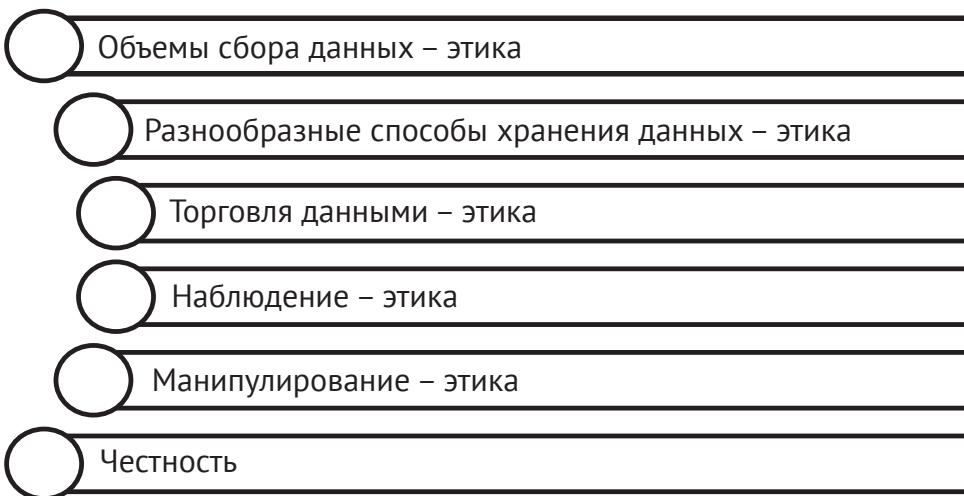


Рис. 2.1. Этика в ИИ

- Объем данных (неанонимный). Процесс сбора данных и системы хранения данных преимущественно являются облачными и полностью цифровыми. Данные, собранные у пользователей, включают их персонализированную информацию, а также их поведение в сети. Эти данные больше не являются анонимными, хотя компании используют эту информацию на агрегированном уровне, при этом персонализированная идентифицируемая информация (Personalized Identifiable Information – PIs) в основном маскируется. Тем не менее кампании проводятся на индивидуальном уровне. Следовательно, данные больше не анонимны, они могут быть отнесены к отдельным лицам. Все присутствие челове-

ка в интернете хранится, анализируется и направляется для продажи любых продуктов или услуг. В настоящее время многие организации соблюдают правила, установленные их клиентами в отношении использования данных. По-прежнему существует потребность в политике, позволяющей сделать данные анонимными.

- Разнообразие данных. Сегодня каждое действие, выполняемое машинами, фиксируется и хранится для того, чтобы улучшить производительность машины, оптимизировать затраты на обслуживание, предотвратить сбои в производственном процессе и многое другое. Существует большое количество данных, генерируемых приложениями и системами интернета вещей (Internet of Things – IoT), такими как системы промышленного производства, телематические данные с транспортных средств на дорогах и многое другое.
- Торговля данными (намеренная и не преднамеренная). Данные, собранные для одной цели, не должны использоваться для другой цели, намеренно или непреднамеренно. Проблема торговли данными возникает, когда мы перепрофилируем данные для использования в других целях. Эта проблема была решена в соответствии с регламентом GDPR³, и подобные законы вводятся в действие различным компаниями и органами власти, чтобы гарантировать отсутствие торговли данными. Если мы создадим систему искусственного интеллекта с торговлей данными, такая система станет неэтичной.
- Наблюдение (целенаправленное и нецеленаправленное). Этика ИИ также относится к поголовному наблюдению с любой целью за всеми группами населения независимо от того, является ли оно целенаправленным или не целевым. Этические споры возникают по поводу того, когда и как следует применять систему наблюдения. Да, биометрическая система аутентификации без дополнительных мер предосторожности вызывает беспокойство, поэтому во всем мире постоянно ведется кампания в пользу остановки систем распознавания лиц.
- Манипулирование вмешивается в рациональный выбор. ИИ с этической точки зрения должен быть рациональным. Должен существовать законный процесс, а системы ИИ должны быть прозрачными.
- Честность. Она лежит в основе определения организационной прозрачности. С точки зрения этики для организаций важно придерживаться принципа объяснимости и обеспечивать техническую прозрачность основных продуктов и приложений ИИ. Это относится к источнику обучающих данных, выбору алгоритмов, тому, как именно обучается система, как генерируются прогнозы и многому другому.

³ Генеральный регламент о защите данных (General Data Protection Regulation – GDPR) регулирует сбор и обработку информации о физических лицах – гражданах Европейской экономической зоны. Он призван усилить защиту конфиденциальных данных и сделать прозрачными все элементы сбора, хранения и обработки информации в интернете. <https://yandex.ru/support/metrica/general/gdpr.html> – Прим. перев.

ПРЕДВЗЯТОСТЬ В ИИ

Многие системы ИИ используют модели ML и DL как конкретные и прогнозистические. Предвзятость в ИИ означает предвзятость в прогнозах. Почему возникают погрешности в прогнозах? Как контролировать предвзятые предсказания – вопрос, который сегодня обсуждается. Этические последствия алгоритмического принятия решений системами ИИ вызывают серьезную озабоченность. Появление предвзятости в процессе принятия решений под руководством ИИ серьезно повлияло на внедрение ИИ. Чтобы построить непредвзятую систему, необходимо иметь сильное чувство справедливости, чтобы помочь лицам, принимающим решения, действовать справедливо, без предрассудков и фаворитизма.

ПРЕДВЗЯТОСТЬ ДАННЫХ

Предвзятость можно разделить на предвзятость данных и алгоритмическую предвзятость. Первая возникает, когда мы рассматриваем ограниченную выборку для маркировки данных под определенным углом и увеличиваем присутствие основной группы. Это приводит к получению необъективного набора данных. Этот процесс может быть улучшен с помощью связи с естественным источником сбора данных. Это показано на рис. 2.2. Давайте поймем предвзятость данных на примере. Если мы посмотрим на тенденцию выручки или прибыли компании электронной коммерции за последние 15 лет, то определенно прослеживается тенденция к росту, и каждые пять лет наблюдается перелом тенденции. Он связан с увеличением цены единицы товара. Если мы хотим построить модель машинного обучения для прогнозирования выручки на следующие два года, мы не можем включить данные за последние 15 лет. Если мы выберем случайные точки данных для обучения модели ML, то возникнет предвзятость данных, которая может привести к неверным прогнозам.

АЛГОРИТМИЧЕСКАЯ ПРЕДВЗЯТОСТЬ

Алгоритмическая предвзятость в некоторой степени обусловлена предвзятостью данных, поскольку последняя не может быть полностью устранена в процессе обучения. Следовательно, обучается неправильная модель, и это приводит к необъективному прогнозу. Чтобы уменьшить предвзятость в процессе обучения и предвзятость данных, необходимо сгенерировать правильное объяснение прогнозов (рис. 2.2). На глобальном уровне, как и на локальном, результаты прогнозирования модели должны быть объяснимы для всех заинтересованных сторон. Таким образом, существует постоянная потребность в системе XAI. Платформа и структура объяснимого ИИ могут предоставить необходимые инструменты и структуру для проработки предвзятости в алгоритме и данных, а также помочь информировать лицо, принимающее решение, о существовании предвзятости.

Системы искусственного интеллекта демонстрируют разумное поведение, которое может обеспечить значительную эффективность в производственных системах, и помогают создавать интеллектуальные приложения с умными реше-

ниями. ИИ, как правило, труден для понимания заинтересованными сторонами бизнеса и пользователями. Если на прикладном уровне любого программного обеспечения используется модель ИИ, становится трудно объяснить решения, принятые системой ИИ, регулирующим органам, руководящим правоохранительным органам и др. Предвзятость данных приводит к предвзятости решений ИИ и может привести организацию к потере репутации. Иногда системы ИИ выдают результаты, которые неудобны или неблагоприятны для организации. Кроме того, эти прогнозы находятся вне контроля организации. В типичном сценарии разработки программного обеспечения мы знаем, при каких обстоятельствах программное обеспечение будет работать и когда оно работать не будет. Однако в системе принятия решений, управляемой ИИ, мы не уверены в условиях, при которых ИИ не будет работать. Это очень трудно предсказать.

ПРОЦЕСС СНИЖЕНИЯ ПРЕДВЗЯТОСТИ

Для снижения предвзятости и улучшения этических стандартов важную роль играет управление. Управление ИИ подразумевает следование набору правил, рекомендаций, стандартов, практик и процессов, с помощью которых можно контролировать и управлять системами принятия решений на основе ИИ. Предвзятость данных можно уменьшить путем установления стандартов управления, таких как оценка данных, тщательное тестирование приложения и т. д.



Рис. 2.2. Предвзятость в системе принятия решений на основе ИИ

ПРЕДВЗЯТОСТЬ ИНТЕРПРЕТАЦИИ

Если прогнозы формируются не в соответствии с ожидаемой линией мышления, то некоторые специалисты используют ту же метрику и математику, чтобы изменить изложение результатов модели. Это еще больше запутывает конечного пользователя или бизнес-пользователя. Предвзятость интерпретации известна как предвзятость в использовании прогностических моделей. Предположим, мы обучаем модель ML, используя популяцию A, и получаем требу-

емый результат, но применяем модель к популяции В, что в некотором смысле называется трансфером обучения в машинном обучении, чтобы избежать дальнейшего обучения. Это классический пример предвзятости интерпретации. Это происходит потому, что прогнозы могут быть предвзятыми, поскольку модель обучена на другой популяции, которая может иметь отличающиеся особенности или характеристики. Предвзятость в процессе алгоритмического обучения возникает из-за постоянной потребности в повышении точности модели. Обычно мы используем методы сглаживания и преобразования признаков, такие как логарифмическое и квадратичное преобразование. Иногда для ограничения чрезмерной подгонки на этапе обучения и тестирования мы также прибегаем к регуляризации. Этот процесс, заключающийся в обрезке коэффициентов модели и связанных с ним шагах, также известен как алгоритмическая предвзятость на этапе обучения модели.

ПРЕДВЗЯТОСТЬ ПРИ ОБУЧЕНИИ

Предвзятость при обучении в системе ИИ возникает, если мы либо выбираем неправильный набор гиперпараметров, либо неправильно выбираем тип модели, либо переобучаем модель, стремясь к достижению более высокой точности выполнения задачи. При разработке модели машинного обучения настройка гиперпараметров и перекрестная проверка играют важную роль в стабильности модели. Чтобы оценить, свободен алгоритм от предвзятости или нет, необходимо рассмотреть собранные данные, процесс обучения модели и допущения процесса моделирования. Рассмотрим пример готовности платить за конкретную услугу ОТТ-платформы⁴, основанную на демографических характеристиках людей и их прошлых расходах. Может ли какая-либо система ИИ предсказать готовность людей платить за подписку на ОТТ-платформу, и сколько они могут заплатить за месячную подписку? Существует ли предвзятость в процессе прогнозирования или обучения модели?

Таблица 2.1. Различия между процедурным и реляционным подходом к измерению предвзятости

Процедурный	Реляционный
Это специфические алгоритмы	Это специфические данные
Фокусируется больше на методах	Сравнивает различные наборы данных
Типы задач известны	Задачи неизвестны

Существует два различных подхода для измерения предвзятости – процедурный и реляционный (см. табл. 2.1). Если мы собираем данные в мировом

⁴ ОТТ (от англ. *Over the Top*) – метод (формат), с помощью которого информация, набор данных (цифровой контент, файлы) разбивается на IP-пакеты и доставляется по неуправляемой сети интернет (по сетям сторонних операторов связи) от источника к получателю. Принципиальное отличие ОТТ от IPTV заключается в том, что интернет-провайдер не контролирует ОТТ-сервис, а ОТТ-сервис не контролирует сеть (и не гарантирует качество сигнала). <https://telesputnik.ru/materials/video-v-internete/article/ott-terminy-i-ponyatiya/> – Прим. перев.

масштабе, в одинаковых пропорциях от разных групп, разных стран, разного возраста, пола и расы, то можно сказать, что в собранных для целей прогнозирования данных нет предвзятости. Реляционный подход помогает найти предвзятость в наборе данных. Процедурный подход фокусируется на алгоритмическом процессе обучения при составлении прогнозов. Возможно, нам придется обучать разные модели для разных возрастных групп, так как атрибуты и отношения в этих группах могут быть различными.

Таблица 2.2. Метрики предвзятости

Статистические показатели	Метрики однородности	Метрики причинно-следственных связей
Управляемая метрика	Выглядят как метрики из набора параметров	Это похоже на условия «если/иначе» (if/else)
Иногда не имеют смысла	Имеют смысл для всех	Чрезвычайно полезны
Оценка невозможна	Оценка возможна	Оценка возможна

Три наиболее часто используемые метрики предвзятости – это статистические показатели, показатели, основанные на однородности и на причинно-следственных связях (см. рис. 2.2 и 2.3). Статистические метрики сосредоточены на схожих прогнозах, основанных на демографических характеристиках различных групп. Если результат прогноза отличается от одной группы к другой и существует разница в точности для разных групп при использовании аналогичной модели, то мы можем измерить эту погрешность в статистических терминах. Статистические меры довольно популярны. Однако их недостаточно для определенной группы алгоритмов.

В качестве альтернативного способа мы можем рассмотреть меры сходства. Если два клиента совершенно одинаковы с точки зрения характеристик, прогнозируемый для них результат должен быть одинаков. Если он отличается, значит, в алгоритме существует предвзятость в отношении к разным клиентам. Здесь под клиентом понимается одна запись из обучающего набора данных, если мы говорим о классификации оттока, кредитном скоринге или типе использования заявки на получение кредита. Чтобы этот метод был успешным, необходимы метрики сходства для оценки того, насколько похожи две записи в обучающем наборе данных. Если они точно скопированы, мы можем назвать это идеальным сходством. Однако, если одна незначительная или основная характеристика отличается, то каков будет процент сходства? Когда мы распространим этот метод на n признаков, насколько это будет полезно? Этот метод выявления предвзятости также не свободен от ограничений. Здесь успех метода заключается в мере сходства. Чем надежнее метрика сходства, тем лучше результат.

Третий важный метод оценки предвзятости – это причинно-следственные рассуждения, которые могут быть осуществлены путем создания чего-то вроде условий «если/иначе» (if/else). Мы понимаем условия if/else гораздо лучше, поэтому причинно-следственные рассуждения для классификации записи в бинарный класс дадут дополнительное представление о предвзятости алгоритма. Условие «если/иначе/тогда» (if/else/then) можно применить, принимая во внимание все признаки, которые присутствуют в обучающем наборе данных.

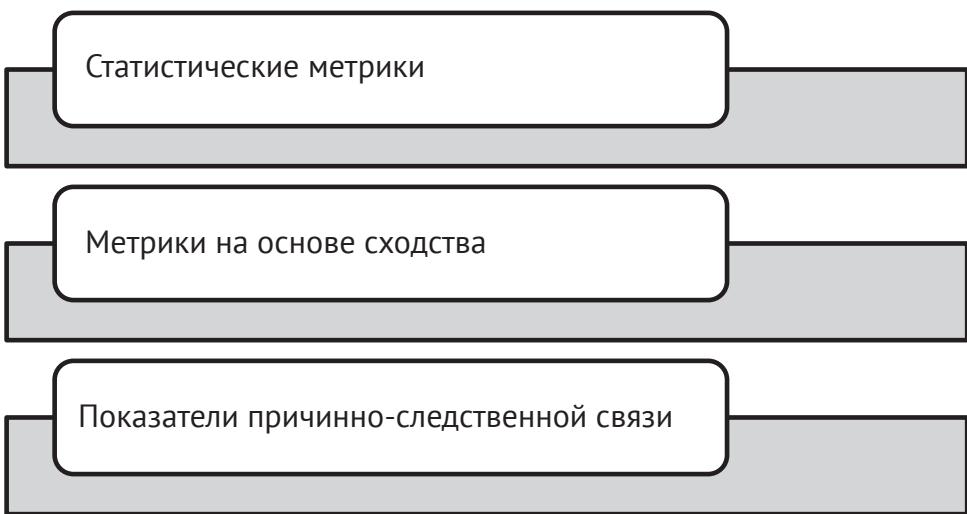


Рис. 2.3. Метрики для измерения предвзятости

Вопросы этики в системах ИИ могут решаться путем обеспечения качества системы, задаваемого следующим образом: корректность процесса, эффективность прогнозов, надежность прогнозов, объяснимость решений на индивидуальном уровне, обеспечение безопасности и конфиденциальности, а также создание прозрачной архитектуры с тем, чтобы все заинтересованные стороны были осведомлены о системе принятия решений ИИ и ее внутренних процессах. Хорошая система ИИ должна нести ответственность за решения, которые она генерирует, поэтому решения должны быть справедливыми.

Текущая форма управления ИИ и связанные с ней политики ограничивают только повторное использование данных для других целей, но не использование трансфера обучения в машинном обучении. Модели ML и DL, обученные на одном сценарии, могут быть повторно использованы в другом сценарии внутри или за пределами организации. Однако эти ограничения не распространяются на модели. Модели обладают критической информацией об обучающих данных, которая может быть переработана любым разработчиком для получения дополнительной информации и повторного использования в других целях. Например, для виртуальной примерки очков мы загружаем изображение, а система относит лицо к определенной категории, которую выдает обученная модель, например овальное, квадратное или круглое. Соответственно, система отображает оправы, предназначенные для данной классификации лица. Эта система может быть использована другой компанией-конкурентом для получения собственных обучающих данных и создания параллельной системы. Генеральный регламент о защите данных (General Data Protection Regulation – GDPR) был принят в 2018 году, чтобы обеспечить право человека на любые решения на основе ИИ, основанные на его данных. Если это персональные данные, то они обязательны. Однако если это не персональные данные, то положения GDPR не применяются.

Высокоуровневый процесс удаления предвзятости из прогнозных моделей показан на рис. 2.4. Предсказание 1 и предсказание 2 должны совпадать. Если

это так, то точка данных считается непредвзятой, в противном случае – наоборот. Если точка данных является предвзятой, то она должна быть исключена из процесса обучения модели, чтобы получить модель без предвзятости.

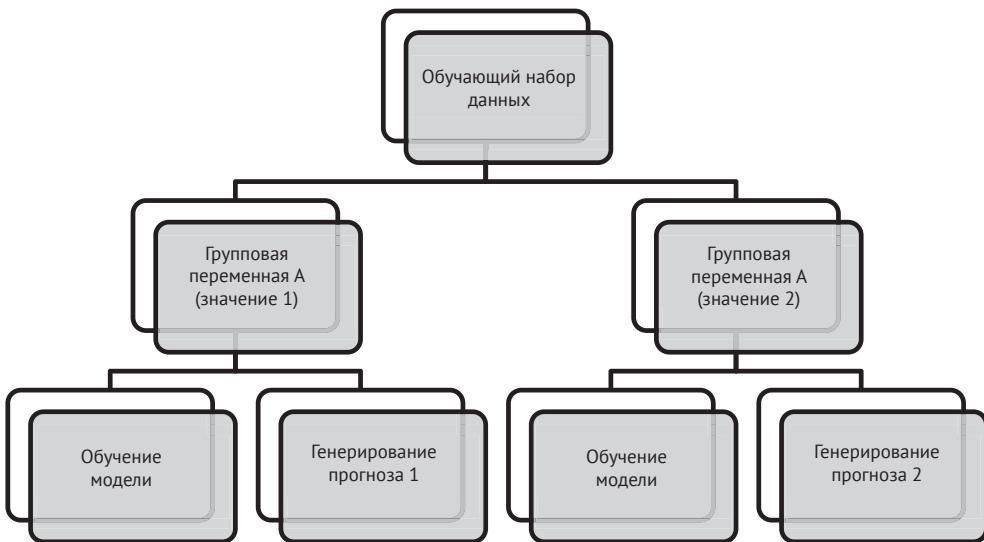


Рис. 2.4. Процесс удаления предвзятости из обучения модели

Для уменьшения предвзятости в алгоритме модели обязательным условием является объяснимость. Модели машинного обучения обычно оцениваются перед выпуском на предмет интерпретируемости, надежности, справедливости, безопасности и конфиденциальности, а также на предмет того, изучила ли обученная модель шаблон, недообучена она или переобучена. Справедливость модели ML должна в идеале оцениваться по двум конкретным признакам:

- прогнозам модели (т. е. решениям, генерируемым моделью);
- влиянию предвзятости данных на справедливость модели.

Надежность в ИИ

Развитие технологии ИИ привело к значительным изменениям в области медицинской визуализации. Трехмерные изображения виртуальной и дополненной реальности передают реальные впечатления от жизни в программной среде. Система ИИ является надежной, если она выдает этические заключения без каких-либо предубеждений и предрассудков. Известны случаи, когда алгоритмические предубеждения имели расовую, политическую и гендерную направленность, что проявлялось в прогнозах. Чтобы придать надежность системе ИИ, необходимо привнести в нее интерпретируемость, конструктивные соображения модели и управление ML. Надежность системы ИИ для автоматизированной поддержки принятия решений на основе данных зависит от беспристрастности алгоритмов.

Алгоритмическая дискриминация является одним из негативных последствий предвзятости в процессе принятия решений. Она приводит к несправед-

ливому отношению к людям на основе их касты, вероисповедания, религии, пола и этнической принадлежности. Надежность автоматизированной системы поддержки принятия решений повышается по мере того, как решение становится менее предвзятым и практически не подвергается дискриминации. Рассмотрим систему кредитной оценки, где система ИИ делает оценку того, кому следует предоставить кредит, а кому нет. Решение, принятое алгоритмом, может быть рациональным, но иногда мы должны сделать его очень прозрачным, чтобы определенная группа людей не восприняла его как решение, направленное против их группы. Предвзятость и дискриминация в автоматизированных системах принятия решений – это две разные вещи.

Дискриминация – это несправедливое и неравнное отношение к группам людей или слоям населения на основе их демографических характеристик или признаков. Такой вид дискриминации встречается в системах искусственного интеллекта, принимающих решения при утверждении займов и кредитных заявок от пользователей. По мере того как мы движемся к большей автоматизации и использованию ИИ ML в интеллектуальных системах, возрастает риск обнаружения дискриминации при принятии решений. Следовательно, система должна быть свободной от дискриминации, объясняя решения и делая процесс очень прозрачным. Дискриминация преднамеренна. А предвзятость является непреднамеренной и присуща процессу обучения модели. Следовательно, надежность и внедрение систем ИИ возрастут только тогда, когда мы создадим систему, свободную как от дискриминации, так и от предвзятости.

Чтобы повысить доверие и надежность моделей ИИ и услуг, разработанных на их основе, необходимо объяснить, как принимается решение. Надежность системы ИИ должна быть совместной ответственностью всех заинтересованных сторон, участвующих в разработке решения ИИ для клиента. Как правило, клиентом является владелец бизнеса, внутренний или внешний по отношению к организации, предоставляющий высокоуровневую цель системы и функции, которые должны быть разработаны.

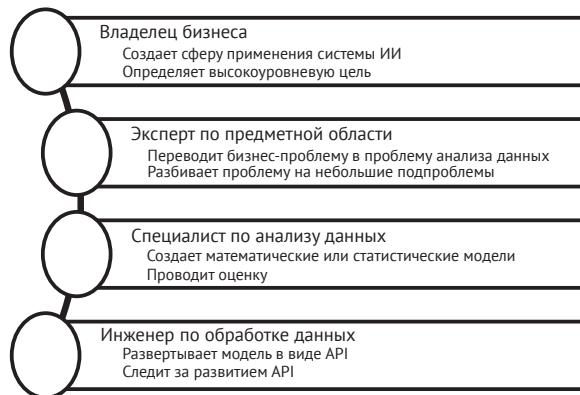


Рис. 2.5. Путь доверия и надежности

Роль всех заинтересованных сторон заключается в том, чтобы обеспечить доверие и надежность при создании и развертывании моделей ИИ в производственной системе (рис. 2.5). Чтобы заслужить доверие, при разработке системы

необходима надлежащая документация, которая должна отвечать на определенные вопросы. Какие данные используются для обучения такой системы? Можно ли использовать эту модель в другой области и в другом сценарии использования? В каких сценариях модель будет работать лучше? Известны ли нам случаи, когда модель работает плохо? Если мы знаем, что такое возможно, как мы можем избавиться от такого рода предвзятости системы? Существует ли какой-либо механизм для объяснения прогнозов, генерируемых системой ИИ? Эти вопросы должны быть задокументированы с примерами и названиями доменов и тематических исследований.

ЗАКЛЮЧЕНИЕ

В этой главе вы узнали об этике при проектировании систем искусственного интеллекта, алгоритмических предвзятостях и надежности моделей. Мы, люди, всегда находим способы сделать наше понимание правильным и совершенным. Системы искусственного интеллекта разрабатываются и используются в другом мире, где катастрофические вещи вполне вероятны и происходят очень часто. Мы учимся на своих ошибках и создаем новые правила, чтобы избежать их в будущем. Из трех вышеперечисленных концепций важно выдвинуть на первый план объяснимый ИИ, чтобы обеспечить по крайней мере интерпретируемость, доверие и надежность в отношении решений, принимаемых системой ИИ, и поддерживать набор этических стандартов.

ГЛАВА 3

Объяснимость для линейных моделей

В данной главе рассматривается использование библиотек SHAP, LIME, SKATER и ELI5 для объяснения решений, принимаемых линейными моделями для задач контролируемого обучения на структурированных данных. В этой главе вы познакомитесь с различными способами объяснения линейных моделей и их решений. В задачах контролируемого машинного обучения существует целевая переменная, которая также известна как зависимая переменная, и набор независимых переменных. Цель состоит в том, чтобы предсказать зависимую переменную как взвешенную сумму входных переменных или независимых переменных.

Линейные модели

Линейные модели, такие как линейная регрессия для прогнозирования реальной величины выхода или модель логистической регрессии для предсказания класса и соответствующих ему вероятностей, являются алгоритмами контролируемого обучения. Эти линейные модели для контролируемого машинного обучения очень просты для интерпретации. Их также легко объяснить заинтересованным лицам. Давайте начнем с пояснения для линейных моделей для полноты модуля.

Линейная регрессия

Линейная регрессия используется для прогнозирования количественного результата целевой переменной, учитывая набор предикторов. Формула моделирования обычно выглядит следующим образом:

$$Y = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p + \varepsilon.$$

Коэффициенты бета известны как параметры, а член эпсилон известен как член ошибки. Член ошибки можно рассматривать как сводную метрику, которая отражает неспособность модели предсказывать. В реальном мире мы не можем предсказывать с точностью 100 %, поскольку вариации в данных – это реальность. Данные постоянно меняются. Цель разработки модели – предсказать с максимально возможной точностью и стабильностью. Целевая переменная принимает значение члена перехвата, когда независимые переменные

принимают нулевые значения. Вы собираетесь использовать доступный в интернете набор данных automobile.csv, чтобы создать модель линейной регрессии для прогнозирования цены автомобиля, учитывая его характеристики.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import datasets, linear_model
from scipy import linalg
df = pd.read_csv('automobile.csv')
df.info()
df.head()
```

В этом наборе данных имеется 6019 записей и 11 характеристик, которые являются базовыми. Словарь данных показан в табл. 3.1.

Таблица 3.1. Словарь данных для характеристик

№ п/п	Название характеристики	Описание
0	Цена	Цена в индийских рупиях
1	Марка	Производитель
2	Местоположение	Город, в котором находится автомобиль
3	Возраст	Сколько лет автомобилю
4	Одометр	Пройденный километраж
5	Топливо Тип	Тип топлива
6	Трансмиссия	Тип трансмиссии
7	Владелец Тип	Число владельцев
8	Пробег	Пробег на литр
9	Двигатель, см ³	Рабочий объем двигателя, см ³
10	Мощность, л. с.	Мощность автомобиля, л. с.

После очистки данных и преобразования характеристик, что является основным шагом перед переходом к этапу разработки модели, набор данных будет состоять из 11 характеристик с одинаковым количеством записей в наборе данных. На следующем графике показана корреляция между различными характеристиками. Это важно для того, чтобы понять связь между различными характеристиками и зависимой переменной. Результаты отображены на рис. 3.1 с помощью парной диаграммы рассеяния.

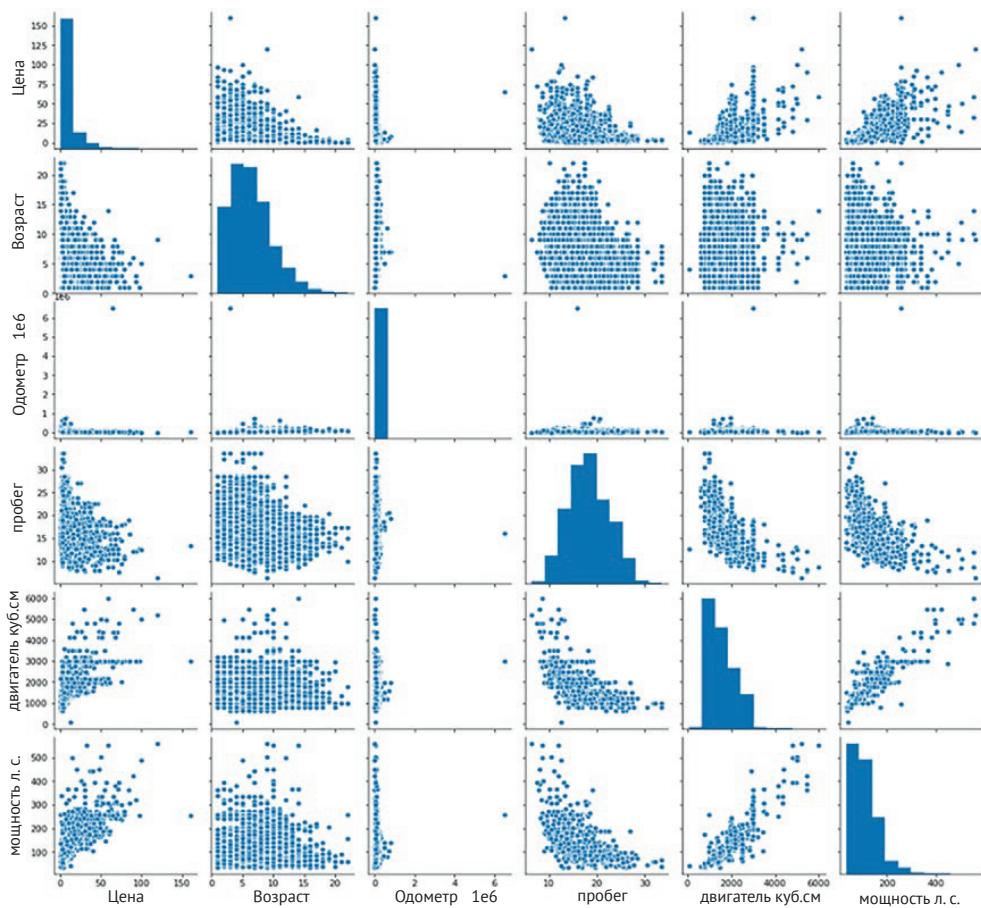


Рис. 3.1. Корреляция между зависимыми и независимыми переменными

```
import seaborn as sns
sns.pairplot(df[['Price','Age','Odometer','mileage','engineCC','powerBhp']])
```

Чтобы получить точную корреляцию между различными характеристиками, необходимо вычислить таблицу корреляций, и следующий скрипт делает это (см. табл. 3.2).

```
corr1 = (df[['Price','Age','Odometer','mileage','engineCC','powerBhp']]).corr()
corr1
```

Таблица 3.2. Коэффициент корреляции между переменными

	Цена	Возраст	Одометр	Пробег	Двигатель	Мощность
Цена	1.000000	-0.305327	-0.011493	-0.334989	0.659230	0.771140
Возраст	-0.305327	1.000000	0.173048	-0.295045	0.050181	-0.028722
Одометр	-0.011493	0.173048	1.000000	-0.065223	0.090721	0.031543
Пробег	-0.334989	-0.295045	-0.065223	1.000000	-0.641136	-0.545009
Двигатель	0.659230	0.050181	0.090721	-0.641136	1.000000	0.863728
Мощность	0.771140	-0.028722	0.031543	-0.545009	0.863728	1.000000

Чтобы сравнить положительную и отрицательную корреляцию на одной и той же таблице, вы можете использовать градиентный график (см. табл. 3.3).

```
corr.style.background_gradient(cmap='coolwarm')
```

Таблица 3.3. Сопоставление положительной и отрицательной корреляции

	Цена	Возраст	Одометр	Пробег	Двигатель	Мощность
Цена	1.000000	-0.305327	-0.011493	-0.334989	0.659230	0.771140
Возраст	-0.305327	1.000000	0.173048	-0.295045	0.050181	-0.028722
Одометр	-0.011493	0.173048	1.000000	-0.065223	0.090721	0.031543
Пробег	-0.334989	-0.295045	-0.065223	1.000000	-0.641136	-0.545009
Двигатель	0.659230	0.050181	0.090721	-0.641136	1.000000	0.863728
Мощность	0.771140	-0.028722	0.031543	-0.545009	0.863728	1.000000

Иногда таблица может показывать и ложные корреляции. Чтобы проверить это, необходимо использовать статистическую значимость каждого коэффициента корреляции между различными числовыми характеристиками и целевой переменной:

```
np.where((df[['Price','Age','Odometer','mileage','engineCC','powerBhp']]).corr()>0.6,'Yes','No')
array([['Yes', 'No', 'No', 'No', 'No', 'Yes', 'Yes'],
       ['No', 'Yes', 'No', 'No', 'No', 'No', 'No'],
       ['No', 'No', 'Yes', 'No', 'No', 'No'],
       ['No', 'No', 'No', 'Yes', 'No', 'No'],
       ['Yes', 'No', 'No', 'No', 'Yes', 'Yes'],
       ['Yes', 'No', 'No', 'No', 'Yes', 'Yes']], dtype='|U3')
```

Из таблицы видно, что мощность имеет высокую положительную корреляцию с ценой, а рабочий объем двигателя также сильно корелирует с ценой. Есть четыре категориальные переменные, для которых необходимо ввести фиктивные переменные, чтобы выполнить матричное умножение. Вы не можете использовать строку в том виде, в котором она есть, в процессе вычислений. В машинном обучении это называется *горячим кодировщиком*, который необходимо приме-

нить к категориальным столбцам для генерирования флагов, соответствующих каждой категории, чтобы можно было ввести информацию в модель. В рамках статистического моделирования эта же техника называется *созданием фиктивных переменных*. Переменные, для которых создаются фиктивные переменные, следующие: местоположение, тип топлива, тип трансмиссии и число владельцев. Расчет фиктивных переменных выполняет следующая программа:

```
Location_dummy = pd.get_dummies(df.Location,prefix='Location',drop_first=True)
FuelType_dummy = pd.get_dummies(df.FuelType,prefix='FuelType',drop_first=True)
Transmission_dummy = pd.get_dummies(df.Transmission,prefix='Transmission',
drop_first=True)
OwnerType_dummy = pd.get_dummies(df.OwnerType,prefix='OwnerType',drop_
first=True)
combine_all_dummy = pd.concat([df,Location_dummy,FuelType_dummy,Transmission_
dummy,OwnerType_dummy],axis=1)

combine_all_dummy.head()
combine_all_dummy.columns
Index(['Price', 'Make', 'Location', 'Age', 'Odometer', 'FuelType',
       'Transmission', 'OwnerType', 'Mileage', 'EngineCC', 'PowerBhp',
       'mileage', 'engineCC', 'powerBhp', 'Location_Bangalore',
       'Location_Chennai', 'Location_Coimbatore', 'Location_Delhi',
       'Location_Hyderabad', 'Location_Jaipur', 'Location_Kochi',
       'Location_Kolkata', 'Location_Mumbai', 'Location_Pune',
       'FuelType_Diesel', 'FuelType_Electric', 'FuelType_LPG',
       'FuelType_Petrol', 'Transmission_Manual',
       'OwnerType_Fourth +ACY- Above', 'OwnerType_Second', 'OwnerType_Third'],
      dtype='object')
clean_df = combine_all_dummy.drop(columns=['Make','Location','FuelType',
                                             'Transmission','OwnerType',
                                             'Mileage', 'EngineCC','PowerBhp'])
clean_df.columns
Index(['Price', 'Age', 'Odometer', 'mileage', 'engineCC', 'powerBhp',
       'Location_Bangalore', 'Location_Chennai', 'Location_Coimbatore',
       'Location_Delhi', 'Location_Hyderabad', 'Location_Jaipur',
       'Location_Kochi', 'Location_Kolkata', 'Location_Mumbai',
       'Location_Pune', 'FuelType_Diesel', 'FuelType_Electric',
       'FuelType_LPG', 'FuelType_Petrol', 'Transmission_Manual',
       'OwnerType_Fourth +ACY- Above', 'OwnerType_Second',
       'OwnerType_Third'],      dtype='object')
```

Перед созданием модели линейной регрессии необходимо проверить предположения модели, которые приведены в блокноте сценариев. После необходимого преобразования характеристик, такого как нормализация столбцов и управление выбросами, вы разделяете набор данных на 75 % для обучения и 25 % для тестирования или оценки модели. Вы используете API машинного обучения sklearn на языке Python.

```
# разделите набор данных на тренировочный и тестовый
data_train, data_test = train_test_split(clean_df,test_size=0.25,
random_state=1234)

data_train.shape,data_test.shape

XTrain = np.array(data_train.iloc[:,0:(clean_df.shape[1]-1)])
YTrain = np.array(data_train['Price'])

XTest = np.array(data_test.iloc[:,0:(clean_df.shape[1]-1)])
YTest = np.array(data_test['Price'])

XTrain.shape, XTest.shape
```

После завершения обучения модели вы получите точность обучения и точность тестирования. Обе точности равны 100 %. Когда вы смотрите на коэффициенты, то обнаруживаете, что все коэффициенты равны 0, а член перевхата – 1. Что-то пошло не так. Здесь вступает в силу роль объяснимого ИИ в прояснении того, что произошло.

```
#модель множественной линейной регрессии
reg = linear_model.LinearRegression()
reg

reg.fit(XTrain,YTrain) #обучение модели

print('Coefficients: \n', np.round(reg.coef_,4))
print('Intercept: \n', np.round(reg.intercept_,0))

reg.score(XTrain,YTrain) # значение R-квадрат для обученной модели
reg.score(XTest,YTest) # значение R-квадрат для тестового набора
```

Вы также можете использовать API из статистических моделей, чтобы понять, есть ли разница в результатах. В табл. 3.4 показаны результаты его использования.

```
from scipy import stats

# Использование статистического API

import statsmodels.api as sm

y = np.array(clean_df['Price'])
xx = np.array(clean_df[['Price', 'Age', 'Odometer', 'mileage', 'engineCC',
'powerBhp',
'Location_Bangalore', 'Location_Chennai', 'Location_Coimbatore',
'Location_Delhi', 'Location_Hyderabad', 'Location_Jaipur',
'Location_Kochi', 'Location_Kolkata', 'Location_Mumbai',
'Location_Pune', 'FuelType_Diesel', 'FuelType_Electric', 'FuelType_LPG',
'FuelType_Petrol', 'Transmission_Manual',
'OwnerType_Fourth +ACY- Above', 'OwnerType_Second', 'OwnerType_Third']])
```

```

y
mod = sm.OLS(y, xx)

results = mod.fit()
print(results.summary())

```

Из таблицы результатов регрессии обычным методом наименьших квадратов (Ordinary Least Squares – OLS) видно, что результат одинаков. Различий нет (см. табл. 3.4).

Таблица 3.4. Результаты OLS-регрессии

Зависимая переменная:	y	R2 (некентрированн.):	1.000
Модель:	OLS	Приращение R2 (некентрированн.):	1.000
Метод:	наименьших квадратов	F-статистика:	4.310e+28
Дата:	Суббота, 19.12.2020	Вероятность (F-статистика):	0.00
Время:	23:01:57	Логарифмическая вероятность:	1.5711e+05
Количество наблюдений:	6019	ИИС :	-3.142e+05
Остатки Df:	5995	BIC:	-3.140e+05
Df модели:	24		
Тип ковариации:	неустойчивая		

Сводка результатов регрессии показывает ошибку в процессе создания модели. Это может быть связано с сильной мультиколлинеарностью. Значение R^2 равно 1.0, что означает, что 100 % дисперсии зависимой переменной может быть объяснено с помощью независимых переменных. В этой модели нет ошибки, во что совершенно невозможно поверить. Высокая степень мультиколлинеарности между различными переменными может быть объяснена коэффициентом инфляции дисперсии (Variance Inflation Factor – VIF). VIF для любого предиктора должен быть меньше 10. В любой ситуации он не может быть больше 10.

```

# Это может указывать на наличие сильной мультиколлинеарности

print('Parameters: ', results.params)
print('R2: ', results.rsquared)

print('Parameters: ', results.params)
print('Standard errors: ', results.bse)
print('Predicted values: ', results.predict())

```

Следующий скрипт показывает, как рассчитать VIF для всех переменных, отсортировать их и показать пять переменных с наиболее высоким значением VIF (см. табл. 3.5):

```
infl = results.get_influence()
print(infl.summary_frame().filter(regex="dfb"))
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):
    # Расчет VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in
    range(X.shape[1])]

    return(vif)

X = clean_df.drop('Price',axis=1)
vif_df = calc_vif(X)
vif_df.sort_values(by='VIF', ascending=False).head()
X = clean_df.drop(['Price','engineCC'],axis=1)
vif_df = calc_vif(X)

vif_df.sort_values(by='VIF', ascending=False).head()

X = clean_df.drop(['Price','engineCC','FuelType_Diesel'],axis=1)
vif_df = calc_vif(X)

vif_df.sort_values(by='VIF', ascending=False).head()

X = clean_df.drop(['Price','engineCC','FuelType_Diesel','mileage'],axis=1)
vif_df = calc_vif(X)

vif_df.sort_values(by='VIF', ascending=False).head()

# VIF менее 10 является приемлемым
# Чем больше VIF, тем менее надежными будут результаты регрессии.
# В целом VIF выше 10 указывает на высокую корреляцию и является поводом для беспокойства.

X = clean_df.drop(['Price','engineCC','mileage'],axis=1)
vif_df = calc_vif(X)

vif_df.sort_values(by='VIF', ascending=False).head()

X = clean_df.drop(['Price','engineCC','FuelType_Diesel','mileage'],axis=1)
vif_df = calc_vif(X)

vif_df.sort_values(by='VIF', ascending=False).head()
X = clean_df.drop(['Price','engineCC','FuelType_Diesel','mileage','powerBhp'],axis=1)
vif_df = calc_vif(X)

vif_df.sort_values(by='VIF', ascending=False).head()
```

Таблица 3.5. Переменные с наиболее высоким значением VIF

Переменные	VIF	
0	Возраст	6.322638
15	Трансмиссия_ручная	3.482934
14	Тип топлива_бензин	1.997771
6	Местоположение_Хайдерабад	1.838072
11	Местоположение_Пуна	1.760061

VIF И ПРОБЛЕМЫ, КОТОРЫЕ ОН МОЖЕТ ПОРОДИТЬ

VIF (variance inflation factor) – это коэффициент инфляции дисперсии. Эта метрика количественно определяет степень мультиколлинеарности, которая существует в модели. Мультиколлинеарность можно определить как существование высокой корреляции между более чем двумя независимыми переменными. Стандартная индустриальная практика – следовать правилу $VIF \leq 10$ для выявления мультиколлинеарности в модели. Связанная с VIF проблема может быть объяснена на примере. Возьмем два признака, X_1 и X_2 . Оба могут быть использованы для прогнозирования зависимой переменной Y . Коэффициент X_1 равен 0.20, и его можно определить так: если X_1 изменится на одну единицу, то зависимая переменная, по прогнозу, изменится в 0.20 раза, при этом все остальные переменные в модели остаются неизменными. Когда X_1 и X_2 сильно коррелируют, предположение о сохранении всех других переменных неизменными нарушается. Следовательно, мультиколлинеарность должна быть удалена из модели для того, чтобы создать правильную интерпретацию значения коэффициента, соответствующего каждой предикторной переменной.

Значение VIF менее 10 является приемлемым. Чем больше VIF, тем менее надежными будут ваши результаты регрессии. В целом VIF выше 10 указывает на высокую корреляцию и является поводом для беспокойства. Следующий скрипт показывает VIF после удаления мультиколлинеарных переменных. Модель переобучается на уточненном наборе переменных. Оценка обучения при этом достигает 70 %, а оценка теста – 69 %, следовательно, это хорошая модель. После того как модель будет окончательно признана хорошей, можно использовать объясняющие пакеты ИИ Python для объяснения компонентов модели.

```
y = clean_df['Price']
x = clean_df.drop(['Price','engineCC','FuelType_Diesel','mileage'],axis=1)
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.25,random_
state=1234)

xtrain.shape,ytrain.shape,xtest.shape,ytest.shape

new_model = LinearRegression()

new_model.fit(xtrain,ytrain)

print(new_model.score(xtrain,ytrain))
```

```
print(new_model.score(xtest,ytest))  
0.7000714797069869  
0.6902967954209108
```

Таблица коэффициентов (табл. 3.6) показывает имена переменных, значения их коэффициентов и их последовательность в наборе данных.

```
resultsDF = pd.DataFrame()  
resultsDF['Variables'] = pd.Series(xtrain.columns)  
resultsDF['coefficients'] = pd.Series(np.round(new_model.coef_,2))  
resultsDF.sort_values(by='coefficients',ascending=False)
```

Таблица 3.6. Таблица коэффициентов

Имя переменной	Коэффициенты
13	Тип топлива_электрический
17	Тип владельца_четвертый и выше
5	Местоположение_Коимбатор
3	Местоположение_Бангалор
7	Местоположение_Хайдерабад
19	Тип владельца_третий
14	Тип топлива_LPG
4	Местоположение_Ченнаи
8	Местоположение_Джайпур
12	Местоположение_Пуна
2	мощность л.с.
1	Одометр
9	Местоположение_Кочи
6	Местоположение_Дели
18	Тип владельца_второй
11	Местоположение_Мумбаи
0	Возраст
10	Местоположение_Колката
15	Тип топлива_бензин
16	Трансмиссия_ручная

Правильность соответствия регрессионной модели определяется по скорректированному значению R^2 . Значение R^2 может быть высоким из-за добавления любых избыточных переменных в наборе данных / процессе обучения. Однако скорректированное значение R^2 учитывает влияние дополнительных переменных в процессе обучения модели.

```

#скорректированный R квадрат
def AdjustedRSquare(model,X,Y):
    YHat = model.predict(X)
    n,k = X.shape
    sse = np.sum(np.square(YHat-Y),axis=0) #sum of square error
    sst = np.sum(np.square(Y-np.mean(Y)),axis=0) # sum of square total
    R2 = 1- sse/sst          #объяснение суммы квадратов
    adjR2 = R2-(1-R2)*(float(k)/(n-k-1))
    return adjR2, R2

from scipy import stats

def ReturnPValue(model,X,Y):
    YHat = model.predict(X)
    n,k = X.shape
    sse = np.sum(np.square(YHat-Y),axis=0)
    x = np.hstack((np.ones((n,1)),np.matrix(X)))
    df = float(n-k-1)
    sampleVar = sse/df
    sampleVarianceX = x.T*x
    covarianceMatrix = linalg.sqrtm(sampleVar*sampleVarianceX.I)
    se = covarianceMatrix.diagonal()[1:]
    betasTstat = np.zeros(len(se))
    for i in range(len(se)):
        betasTstat[i] = model.coef_[i]/se[i]
    betasPvalue = 1- stats.t.cdf(abs(betasTstat),df)
    return betasPvalue

resultsDF['p_value'] = pd.Series(np.round(ReturnPValue(new_model,xtrain,ytrain),2))
resultsDF.sort_values(by='coefficients',ascending=False)

```

Таблица 3.7. Значения коэффициентов с соответствующими вероятностями

Имя переменной	Коэффициенты		p
13	Тип топлива_электрический	9.02	0.02
17	Тип владельца_четвертый и выше	4.70	0.03
5	Местоположение_Коимбатор	2.35	0.00
3	Местоположение_Бангалор	1.96	0.00
7	Местоположение_Хайдерабад	1.92	0.00
19	Тип владельца_третий	1.66	0.01
14	Тип топлива_LPG	1.50	0.29
4	Местоположение_Ченнаи	1.05	0.01
8	Местоположение_Джайпур	0.65	0.08

Окончание табл. 3.7

Имя переменной	Коэффициенты		p
12	Местоположение_Пуна	0.21	0.31
2	Мощность л.с.	0.14	0.00
1	Одометр	0.00	0.04
9	Местоположение_Кочи	-0.06	0.44
6	Местоположение_Дели	-0.12	0.38
18	Тип владельца_второй	-0.53	0.02
11	Местоположение_Мумбаи	-0.60	0.06
0	Возраст	-0.93	0.00
10	Местоположение_Колката	-0.97	0.01
15	Тип топлива_бензин	-1.31	0.00
16	Трансмиссия_ручная	-2.68	0.00

```

reg.adjR2, reg.R2 = AdjustedRSquare(new_model,xtrain,ytrain)
print (reg.adjR2, reg.R2)

0.6987363872507527 0.7000714797069869

def ErrorMetric(model,X,Y):
    Yhat = model.predict(X)
    MAPE = np.mean(abs(Y-Yhat)/Y)*100
    MSSE = np.mean(np.square(Y-Yhat))
    Errgor = sns.distplot(Y-Yhat)
    return MAPE, MSSE, Errgor

resultsDF.sort_values(by='p_value',ascending=False)

```

р – значение вероятности коэффициента β – показывает статистическую значимость предикторов в сценарии линейной регрессии. Пороговым значением р считается 0.05, т. е. 5%-ный уровень значимости для статистического теста. Если для какого-либо предиктора значение р меньше 0.05, то предиктор является значимым, в противном случае – нет. Если значение р больше 0.05, то значение коэффициента β будет близким к нулю. В этой модели шесть переменных имеют р больше 0.05. На рис. 3.2 показана корреляция между фактической переменной Y, или целевой переменной, и прогнозируемой целевой переменной. Вы можете видеть хорошую корреляцию между фактической и прогнозируемой переменными Y, которая составляет 0.83, что означает, что это хорошая модель. В табл. 3.7 показаны коэффициенты, отсортированные в порядке убывания, с соответствующими значениями р. Любой предиктор со значением р более 0.05 может быть удален из модели итерационным способом.

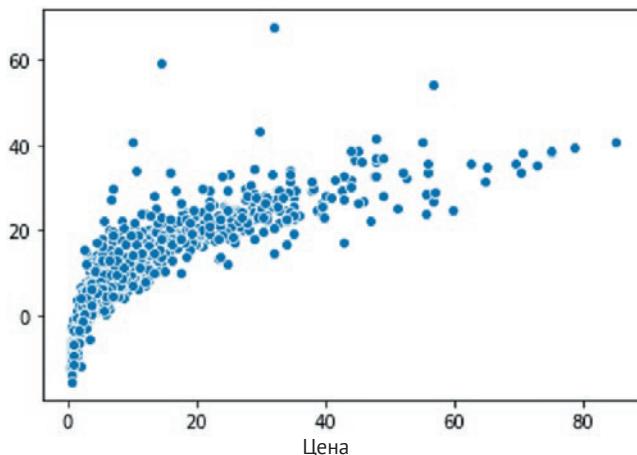


Рис. 3.2. Корреляция между фактическим Y и прогнозируемым Y

ОКОНЧАТЕЛЬНАЯ МОДЕЛЬ

После удаления сильно мультиколлинеарных переменных и избыточных переменных, имеющих статистическую незначительность, точность модели остается близкой к 70 % на обучающем множестве и 69 % на тестовом множестве.

```
y = clean_df['Price']
x = clean_df.drop(['Price','engineCC','FuelType_Diesel','mileage','Location_Kochi'],axis=1)

xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.25,random_state=1234)
xtrain.shape,ytrain.shape,xtest.shape,ytest.shape

new_model = LinearRegression()

new_model.fit(xtrain,ytrain)

print(new_model.score(xtrain,ytrain))
print(new_model.score(xtest,ytest))
```

ОБЪЯСНИМОСТЬ МОДЕЛИ

Интерпретацию модели можно сделать, посмотрев на коэффициенты β модели. Самыми большими преимуществами линейной регрессионной модели являются простота и линейность, что облегчает ее интерпретацию. Линейная регрессия также заставляет прогноз быть линейной комбинацией признаков. Доверительный интервал – это диапазон значений, в пределах которого будет с некоторой вероятностью находиться истинное значение прогноза. 95%-ный доверительный интервал означает, что в 95 % случаев истинное значение прогноза попадет в этот диапазон. В данном примере вы видите 95%-ный доверительный интервал. Уровень значимости означает, что при двухвостой

проверке гипотезы с 5%-ным уровнем значимости интерпретация может быть объяснена в табл. 3.8. С точки зрения интерпретации значение члена перехвата не играет никакой роли в моделируемом сценарии. Если все числовые признаки в обучающих данных нулевые, а все бинарные признаки находятся в своей эталонной категории, тогда прогноз, генерируемый моделью, будет эквивалентен члену перехвата. Это исключительный сценарий, но когда все числовые признаки присутствуют в обучающем наборе данных и они стандартизированы со средним значением нуль и стандартным отклонением 1, тогда член перехвата отражает прогнозируемый результат для экземпляра, где все признаки имеют среднее значение.

```
resultsDF = pd.DataFrame()
resultsDF['Variables'] = pd.Series(xtrain.columns)
resultsDF['coefficients'] = pd.Series(np.round(new_model.coef_,2))
resultsDF['p_value'] = pd.Series(np.round(ReturnPValue(new_
model,xtrain,ytrain),2))
resultsDF.sort_values(by='p_value',ascending=False)
```

Таблица 3.8. Интерпретация параметров линейной модели

Интерпретация коэффициентов регрессии	Целевой столбец – это цена автомобилей, которая является функцией различных переменных, как числовых, так и категориальных
Возраст = -0.911	Увеличение возраста автомобиля на один год приводит к предполагаемому снижению цен на автомобили на 0.911 единицы. Если предположить, что цена выражена в тысячах долларов, то снижение цены автомобиля составит 911.00 долл. с каждым годом эксплуатации при сохранении всех остальных характеристик постоянными
Мощность л. с. = 0.141	Увеличение мощности двигателя автомобиля на одну единицу приводит к предполагаемому увеличению цен на автомобили на 0.141 единицы. Чем выше мощность, тем выше цена автомобиля при условии, что все остальные характеристики одинаковы
Местоположение_Джайпур = 0.663	Предполагаемые цены на автомобиль увеличиваются на 663 долл., если базовое местоположение автомобиля – Джайпур, по сравнению с другим местом, при условии, что все остальные характеристики неизменны
Тип топлива_электрический = 8.972	Расчетная цена автомобиля увеличится на 8972 долл., если автомобиль является электромобилем, по сравнению с любым другим видом топлива, при условии, что другие характеристики модели не изменятся
Тип владельца_второй = -0.54	Расчетная цена автомобиля снизится на 540 долл., если владелец – второй, по сравнению с любой другой формой собственности, при условии, что другие характеристики модели не изменились
Трансмиссия_ручная = -2.671	Цена автомобиля снизится на 2671 долл., если тип трансмиссии будет ручной, по сравнению со всеми другими типами, при неизменности всех остальных факторов

ДОВЕРИЕ К МОДЕЛИ ML: SHAP

Чтобы доверять модели машинного обучения на основе линейной регрессии, необходимо понять значение R^2 , полученное с помощью модели. Значение R^2 указывает на степень соответствия регрессионной модели, что означает долю дисперсии целевой переменной, объясненную всеми признаками вместе. Значение R^2 находится в диапазоне от 0.0 до 1.0. Если оно равно нулю, то модель показывает отсутствие корреляции между зависимой и независимой переменными. Если значение равно 1, то это говорит о том, что признаки очень сильно коррелируют. Для того чтобы модель считалась заслуживающей доверия, ее значение R^2 должно составлять 0.80 или более. В приведенном выше примере с автомобилем значение составляет 0.70, что очень близко к стандартной модели, которой можно доверять. Вместо R^2 чаще всего применяют скорректированное значение R^2 (adjusted R square), поскольку оно учитывает количество признаков, используемых в модели. Связь между R^2 и скорректированным R^2 показана ниже с помощью формулы. В следующей формуле N означает общее количество обучающих примеров, а p – общее количество признаков, используемых в модели:

$$\text{Adjusted } R^2 = 1 - (1 - R^2) \times (N - 1)/(N - p - 1).$$

Значение R^2 может увеличиваться по мере добавления в модель избыточных переменных, но скорректированный R^2 останется неизменным. Он будет увеличиваться только в том случае, если признак вносит какой-либо вклад в общую объяснимость модели.

Чтобы создать дополнительную объясимость и глубже понять, как работает модель, вам могут помочь дополнительные библиотеки на базе Python. Значения Шепли (Shapley values) – это широко используемый подход из теории кооперативных игр, который обладает желательными свойствами. Из коэффициентов модели вы понимаете, как оценивается изменение результата при изменении входных параметров. Однако это не говорит вам, какие параметры важны. Значение каждого коэффициента зависит от значений входных характеристик. Например, возраст автомобиля может изменяться от 0 до 15 лет. Однако мощность может варьироваться от 34.20 до 560.00 в приведенном выше наборе данных. Следовательно, величина коэффициента модели не обязательно является хорошим показателем важности характеристики в линейной регрессионной модели.

Некоторые специалисты по анализу данных используют абсолютное значение t-статистики в качестве меры важности характеристик в модели линейной регрессии:

$$t_{\widehat{\beta_j}} = \frac{\widehat{\beta_j}}{SE(\widehat{\beta_j})}.$$

Один из способов понять важность характеристики – посмотреть на график частичной зависимости характеристики от выходных данных модели.

```
! pip install shap
```

После успешной установки SHAP вы можете использовать эту библиотеку для построения графика частичной зависимости, показанного на рис. 3.3.

```
import shap
shap.plots.partial_dependence("Age", new_model.predict,xtrain, ice=False,
model_expected_value=True, feature_expected_value=True)
```

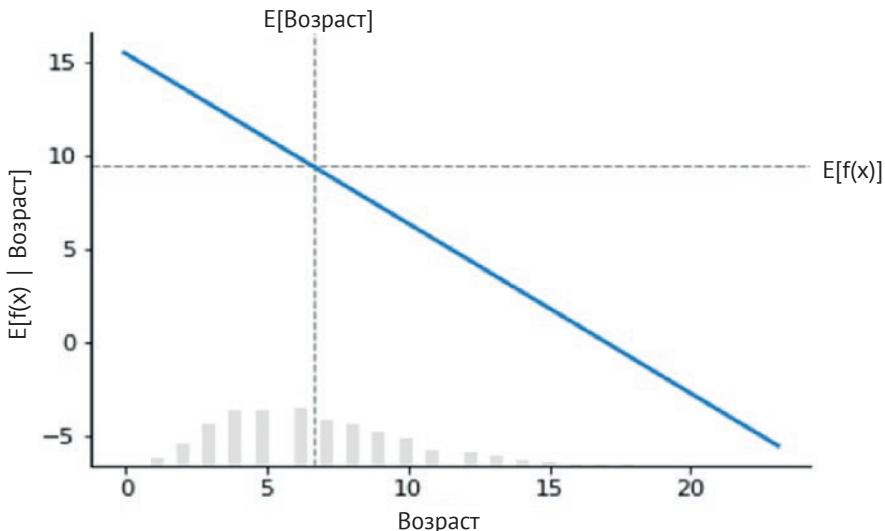


Рис. 3.3. График частичной зависимости

Пунктирная горизонтальная линия показывает ожидаемое значение выхода модели при применении к набору данных, вертикальная пунктирная линия показывает среднее значение возраста. Синяя линия графика частичной зависимости, которая представляет собой среднее значение выхода модели при фиксации возраста на заданном значении, всегда проходит через точку пересечения двух серых линий ожидаемого значения. Эту точку пересечения можно рассматривать как «центр» графика частичной зависимости относительно распределения данных. Вертикальные серые квадраты на горизонтальной оси показывают, что распределение возраста немного перекошено вправо.

Основная идея, лежащая в основе объяснения модели, основанной на значении Шепли, заключается в использовании распределения результатов теории кооперативных игр для распределения вознаграждения за выход модели (x) между ее входными характеристиками. Чтобы связать теорию игр с моделями машинного обучения, необходимо сопоставить входные характеристики модели с игроками в игре, а также соотнести функцию модели с правилами игры. В теории игр игрок имеет возможность присоединиться или не присоединиться к игре, подобно тому, как характеристика может «присоединиться» или «не присоединиться» к модели.

Что такое значение SHAP, и как оно вычисляется? Значение Шепли для каждого значения характеристики вычисляется по следующей формуле:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!} \left[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \right].$$

Следующие пункты поясняют, как это работает:

- чтобы вычислить вклад каждой характеристики, SHAP требует переобучения модели на всех подмножествах S;
- в приведенной выше формуле i – это отдельный признак;
- F – множество всех признаков;
- S – подмножество признаков из множества F;
- для любого признака i создаются две модели – модель 1 с признаком i и модель 2 без признака i. Затем вычисляется разница между прогнозами;
- влияние одного признака на модель зависит от того, как ведут себя другие признаки в модели;
- разница в прогнозах вычисляется для всех возможных подмножеств S, и берутся их средние значения;
- средневзвешенное значение всех возможных различий используется для определения важности признака.

Самый общий способ определить, что значит для характеристики «присоединиться» к модели, – это сказать, что характеристика «присоединилась к модели», когда мы знаем значение этой характеристики и она «не присоединилась к модели», когда мы не знаем значения этой характеристики. Это происходит, когда вес/коэффициент характеристики в модели принимает значение 0.000, – тогда считаем, что характеристика не вступила в игру. Если ее коэффициент не равен 0.000, то считаем эту характеристику вступившей в игру. Давайте рассмотрим значения SHAP для линейной регрессионной модели.

```
# вычислим значения SHAP для линейной модели
background = shap.maskers.Independent(xtrain, max_samples=2000)
background
xtrain.shape
```

Функция `shap.maskers.Independent` маскирует табличные характеристики путем интегрирования по заданному фоновому набору данных. Здесь фоновый набор данных содержит 4500 записей из обучающего набора данных, что является максимальным количеством образцов, которые можно использовать из переданных фоновых данных. Если данные имеют больше, чем `max_samples`, то для подвыборки набора данных используется `shap.utils.sample`. Количество образцов, выходящих из маскировщика (которые будут интегрированы), совпадает с количеством образцов в фоновом наборе данных. Увеличение фонового набора данных приводит к увеличению времени работы программы. Обычно 1, 10, 100 или 1000 фоновых образцов являются разумными вариантами.

```
explainer = shap.Explainer(new_model, background)
explainer
shap_values = explainer(xtrain)
shap_values
```

Приведенный выше скрипт показывает вывод значений SHAP, образцов фоновых данных и базовых значений из модели. Следующий скрипт показывает стандартный график частичной зависимости, учитывающий возраст и функцию предсказания модели, для образца записи № 23 из набора данных. Значение SHAP для конкретной характеристики i – это просто разность между ожидаемым результатом модели и графиком частичной зависимости при значении характеристики x_i с учетом аддитивной природы значений Шепли. Одним из фундаментальных свойств значений Шепли является то, что они всегда суммируют разность между результатами игры, когда присутствуют все игроки и когда не присутствует ни один игрок. Для моделей машинного обучения это означает, что значения SHAP для всех входных характеристик всегда будут равны разнице между базовым (ожидаемым) результатом модели и текущим результатом модели для объясняемого прогноза. Проще всего увидеть это на диаграмме водопада, который начинается с базового предварительного ожидания для цены $[(X)]$, а затем добавляются характеристики по одной за раз, пока вы не достигнете текущего результата модели (x) . Соответственно, для возраста 10 лет вертикальная разница между синей линией и серой пунктирной линией – это значение SHAP для характеристики.

ЛОКАЛЬНОЕ ОБЪЯСНЕНИЕ И ИНДИВИДУАЛЬНЫЕ ПРОГНОЗЫ В МОДЕЛИ ML

График частичной зависимости SHAP (рис. 3.4) – это хороший способ объяснить отдельные прогнозы, получить о них дополнительную информацию и объяснить, какие характеристики имеют значение для данного конкретного индивидуального прогноза.

```
# построение стандартного графика частичной зависимости
sample_ind = 23
fig,ax = shap.partial_dependence_plot(
    "Age", new_model.predict, xtrain, model_expected_value=True, feature_
expected_value=True, show=False, ice=False,
    shap_values=shap_values[sample_ind:sample_ind+1,:],
    shap_value_features=X.iloc[sample_ind:sample_ind+1,:]
)
```

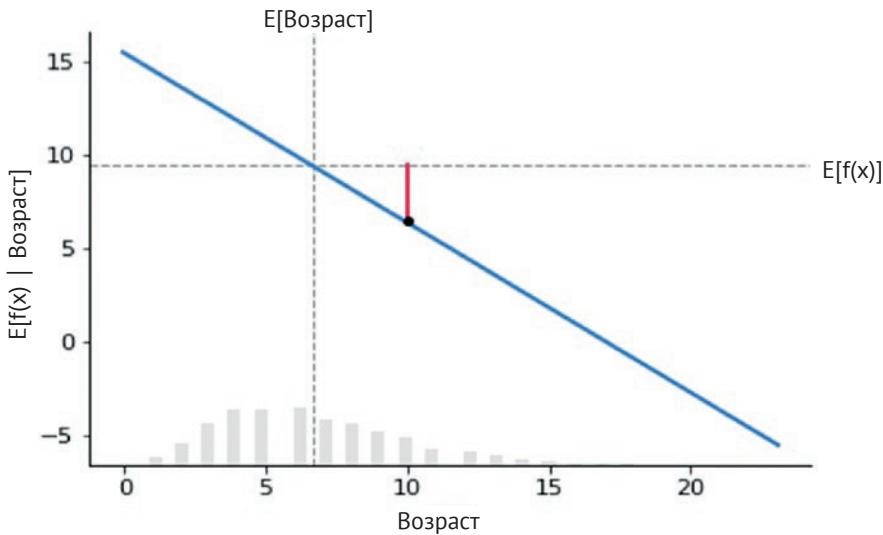


Рис. 3.4. График частичной зависимости возраста и ожидаемого значения прогноза

Рисунок 3.5 поясняет корреляцию в виде диаграммы рассеяния между возрастом и значением SHAP возраста.

```
shap.plots.scatter(shap_values[:, "Age"])
```

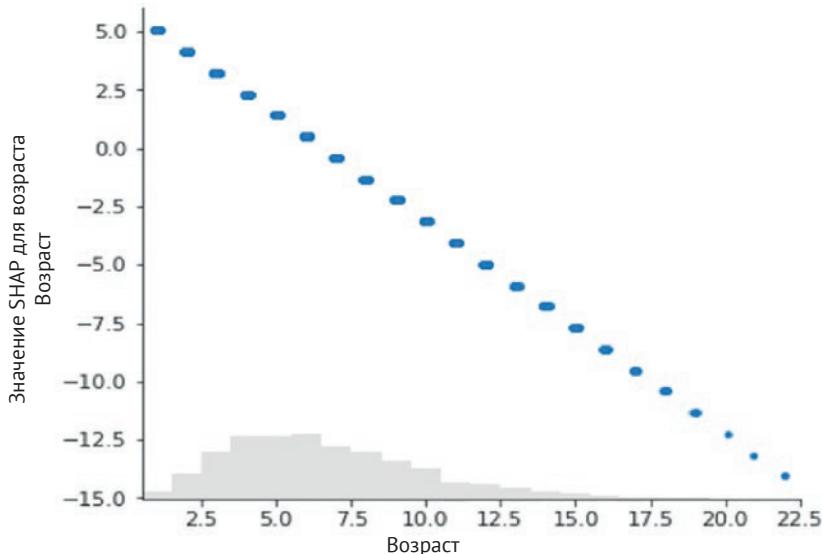


Рис. 3.5. Связь между возрастом и значением SHAP для возраста

Получение значений SHAP для образца строки из набора данных можно объяснить с помощью диаграммы водопада (рис. 3.6). Это часть локального объяснения. Диаграммы водопада предназначены для отображения объяснений отдельных прогнозов, поэтому они ожидают одну строку из объяснения в качестве входных данных. Нижняя часть диаграммы водопада начинается с ожидаемого значения результата модели, а затем каждая строка показывает, как положительный (красный) или отрицательный (синий) вклад каждой характеристики перемещает значение от ожидаемого результата модели для фонового набора данных к результату для данного прогноза.

Рассмотрим пример для понимания значений SHAP – запись № 60 из обучающего набора данных, которая является 2966-й строкой в общем наборе данных. Используем ее в качестве новой точки данных и сделаем прогноз с помощью обученной модели. Прогнозируемое значение равно 4.7656. Однако фактическое значение цены для этой же записи составляет 4.85. Диаграмма водопада из библиотеки SHAP показывает, как прогнозируемое значение 4.766 получается из базовых значений SHAP для функции.

```
train[60:61]
new_model.predict(xtrain[60:61])
ytrain[60:61]

# диаграмма водопада показывает, как мы переходим от shap_values.base_values к
# model.predict(X)[sample_ind]
shap.plots.waterfall(shap_values[60])
```



Рис. 3.6. Диаграмма водопада, показывающая положительные и отрицательные значения SHAP

Из приведенной выше диаграммы водопада видно, что для записи № 60 наиболее значительными предикторами являются мощность, возраст, тип топлива бензин и механическая трансмиссия. Есть пять наименее важных характеристик, которые объединены вместе, и их совместный вклад в прогноз составляет +0.09. Если снять это ограничение на объединение в одну характеристику, вы можете расширить параметр максимального отображения. Следующий скрипт показывает модифицированную версию диаграммы водопада из значений SHAP для локальной интерпретации прогноза (рис. 3.7):

```
shap.plots.waterfall(shap_values[60], max_display=30)
```

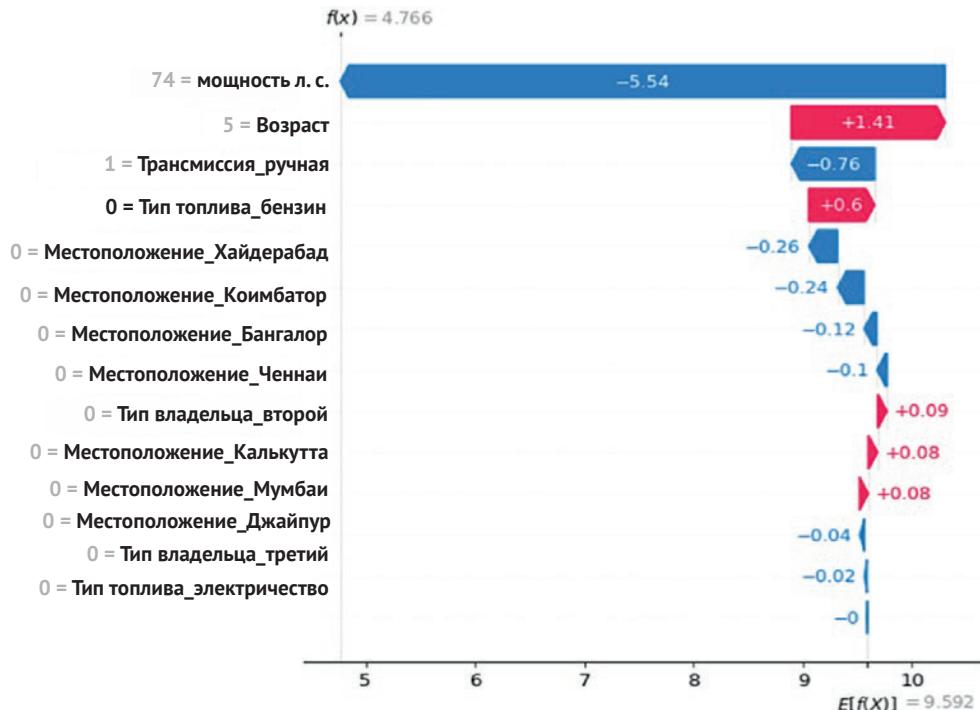


Рис. 3.7. Расширенная форма, показывающая все характеристики

ГЛОБАЛЬНОЕ ОБЪЯСНЕНИЕ И ОБЩИЕ ПРОГНОЗЫ В МОДЕЛИ ML

График пчелиного роя – это визуализация, предназначенная для отображения информации о том, как основные характеристики в наборе данных влияют на результат модели. Каждый экземпляр данного объяснения представлен одной точкой в каждом ряду характеристик. Положение центра определяется значением SHAP этой характеристики, и для отображения плотности точки «нагромождаются» вдоль каждого ряда характеристик. Цвет используется для отображения исходного значения характеристики. На рис. 3.8 видно, что возраст и мощность являются в среднем наиболее важными характеристиками.

- Чем выше мощность, тем большее влияние она оказывает на результат модели, поскольку известно, что с ростом мощности повышается цена автомобиля.
- Аналогичным образом возраст отрицательно связан с прогнозируемой ценой автомобиля. Следовательно, чем моложе автомобиль, тем больше результат модели, как показано на диаграмме ниже.
- По умолчанию на графике пчелиного роя отображается 10 характеристик. Вы можете изменить этот параметр, изменив параметр максимального отображения.
- По умолчанию характеристики сортируются с помощью `shap_values.abs().mean(0)`, которое является средним абсолютным значением SHAP для каждой характеристики. Такой порядок, однако, делает больший акцент на среднее воздействие и меньший – на редкие, но большие по величине воздействия.
- Если вы хотите найти характеристики с высоким воздействием для отдельных людей, можете отсортировать их по максимальному абсолютному значению.

```
model.predict(X)[sample_ind]
shap.plots.beeswarm(shap_values)
```

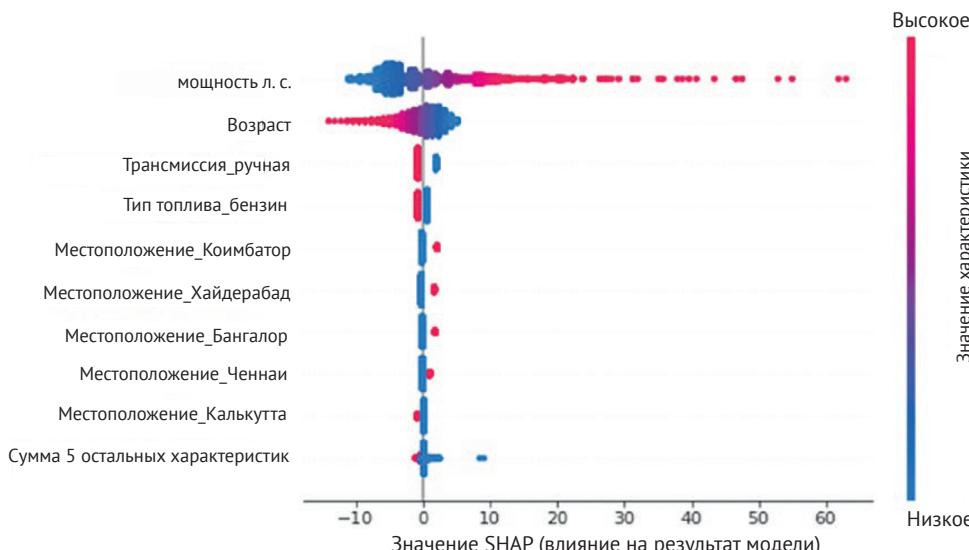


Рис. 3.8. График пчелиного роя, показывающий положительные и отрицательные значения SHAP

По умолчанию в графике пчелиного роя используются красные и синие цвета. Вы можете их настроить и изменить (рис. 3.9).

```
import matplotlib.pyplot as plt
shap.plots.beeswarm(shap_values, color=plt.get_cmap("cool"))
```

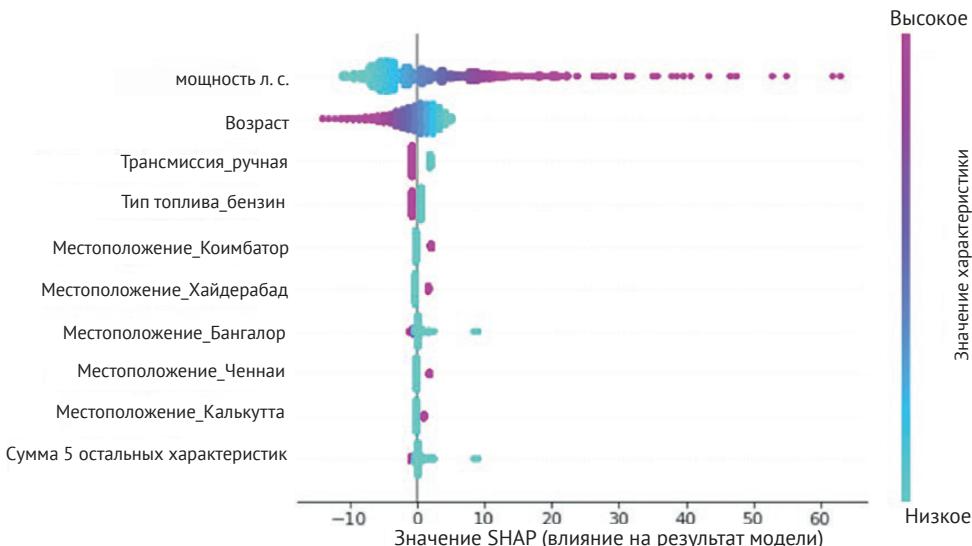


Рис. 3.9. График пчелиного роя с использованием другой цветовой палитры

Существует вариант отображения значений SHAP с помощью гистограммы, где учитывается среднее абсолютное значение SHAP. Горизонтальная ось на рис. 3.10 показывает среднее абсолютное значение SHAP, а вертикальная ось показывает характеристики. Пять наименее важных характеристик объединяются вместе. Кроме того, есть возможность посмотреть максимальное абсолютное значение всех характеристик. Максимальные абсолютные значения SHAP указывают на наблюдения в обучающем наборе данных, которые повлияли на прогнозы. Передача матрицы значений SHAP в функцию построения тепловой карты создает диаграмму с экземплярами на оси x, входами модели на оси y и значениями SHAP, закодированными в цвете. По умолчанию образцы сортируются с помощью функции shap.order.hclust, которая упорядочивает образцы на основе иерархической кластеризации по сходству их объяснений (см. также рис. 3.11 и 3.12).

```
shap.plots.bar(shap_values)
```

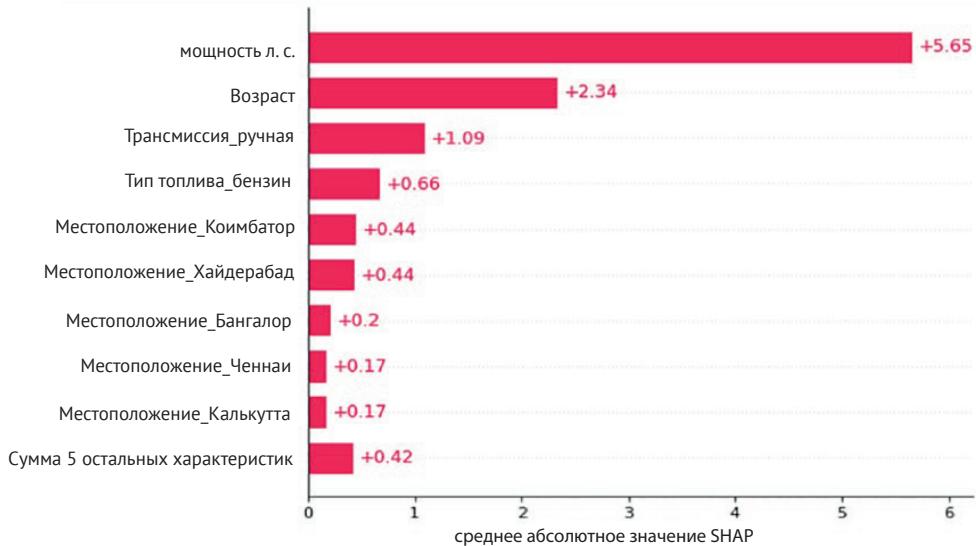


Рис. 3.10. Средние абсолютные значения SHAP

```
shap.plots.bar(shap_values.abs.max(0))
```



Рис. 3.11. Максимальные абсолютные значения SHAP

```
shap.plots.heatmap(shap_values[:1000])
```

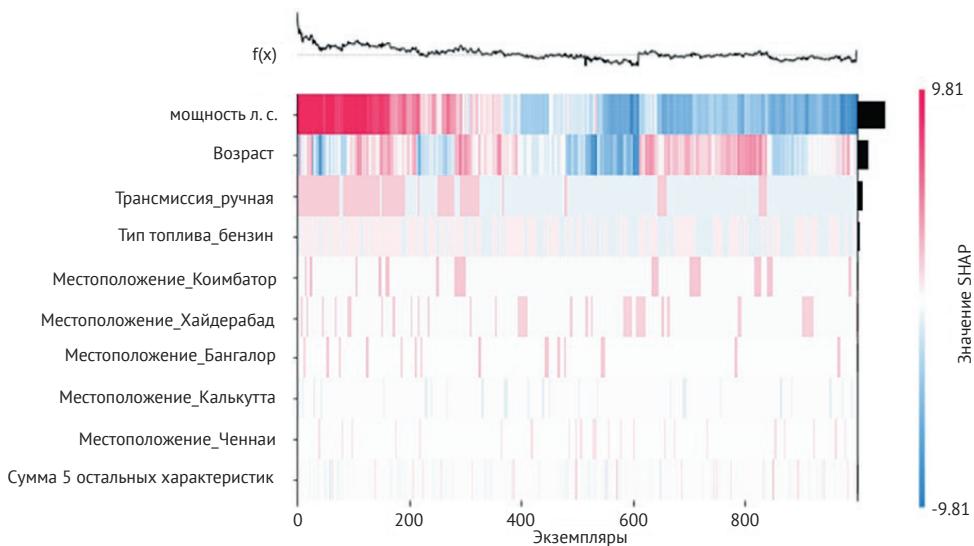


Рис. 3.12. Связь экземпляров характеристик со значениями SHAP

Это приводит к тому, что образцы, имеющие одинаковый выход модели по одной и той же причине, группируются вместе (например, люди с высоким влиянием мощности и возраста). Результат модели показан над матрицей тепловой карты на рис. 3.12 (с центром вокруг explanation's .base_value), а глобальная важность каждого входного параметра модели показана в виде столбчатой диаграммы в правой части графика (по умолчанию это shap.order.abs.mean – мера общей важности).

Объяснение LIME и модель ML

LIME, или Local Interpretable Model-Agnostic Explanations, – это алгоритм, который может достоверно объяснять прогнозы любого классификатора или регрессора путем аппроксимации его локальной интерпретируемой моделью. Он изменяет один образец данных путем изменения значений характеристик и наблюдает за тем, как это влияет на результат. Он выполняет роль «объяснителя» для объяснения прогнозов на основе каждого образца данных. Результатом работы LIME является набор объяснений, представляющих вклад каждой характеристики в прогноз для одного образца, что является формой локальной интерпретируемости. Интерпретируемыми моделями в LIME могут быть, например, линейная регрессия или деревья решений, которые обучаются на небольших возмущениях (например, добавление шума, удаление слов или скрытие частей изображения) исходной модели, чтобы получить хорошее локальное приближение. LIME можно установить с помощью команды `ripl`.

Вы можете взять обучающий набор данных для модели с целевым столбцом и обучить регрессионную модель заново вместо объекта предыдущей модели `new_model`, потому что LIME – это метод, не зависящий от модели, и он переобучает модели во время генерации объяснения. LIME локализует проблему и объясняет модель на локальном уровне, а не на глобальном.

```
!pip install lime

import lime
import lime.lime_tabular

explainer = lime.lime_tabular.LimeTabularExplainer(np.array(xtrain),
                                                    mode='regression',
                                                    feature_names=xtrain.columns,
                                                    class_names=['price'],
                                                    verbose=True)
```

Табличный объяснитель LIME требует массив NumPy в качестве входных данных, поэтому формат обучающих данных меняется. Режим выбран как регрессия после возмущения, а целевой столбец – цена. После разработки объяснителя дополнительно могут быть созданы подробные локальные объяснения. Опция частот характеристик предоставляет распределение характеристик и сколько раз они участвовали в процессе возмущения.

```
explainer.feature_selection
explainer.feature_frequencies
```

Если вы хотите использовать ранее обученный объект модели, то можете применить опцию объяснения экземпляра (explain instance option). Для этого требуется тестовый набор данных, объект модели и количество характеристик, которые можно использовать. Возьмем ту же 60-ю запись из тестового набора данных. Вы получите результат как член перехвата, локальное значение предсказания и right, которое является глобальным значением прогноза. Для 60-й записи из тестового набора данных если вы используете функцию predict, то получите предсказанное значение 27.5854, которое равно правому значению из экземпляра explain. Локальная мода прогноза составляет 28.54, что ближе к фактическому предсказанному значению 35.0.

```
# запрос на объяснение модели LIME
i = 60
exp = explainer.explain_instance(np.array(xtest)[i], new_model.predict,
                                  num_features=14)

new_model.predict(xtest)[60]
ytest[60:61]

Intercept 18.186435664326485
Prediction_local [28.15539047]
Right: 27.58547373488966
exp.show_in_notebook(show_table=True)
exp.as_list()
```

Вы можете показать объяснитель в табличном формате с предсказанным значением, положительными и отрицательными значениями характеристик, а также общими характеристиками и их значением в таблице. Прогнозируемое значение равно 46.62. На второй диаграмме для одного и того же экземпляра, № 60, ото-

брожаются положительные и отрицательные значения характеристик. Горизонтальные полосы на второй диаграмме показывают важность характеристики для данной записи. В третьей таблице показано значение LIME, соответствующее каждой характеристики. Методология и локальные объяснения достаточно понятны интуитивно. Это может быть объяснено любому бизнес-пользователю. Локальность образца 60-й записи выбирает единственную точку данных равномерно и случайным образом, создавая возмущенные точки данных и соответствующее предсказанное значение из модели. По умолчанию метод выбора характеристики – автоматический. LIME фокусируется на подгонке интерпретируемой модели на перемешанном наборе данных (возмущенных) с применением весов образца и дает локальные объяснения, используя новую обученную модель (см. рис. 3.13).

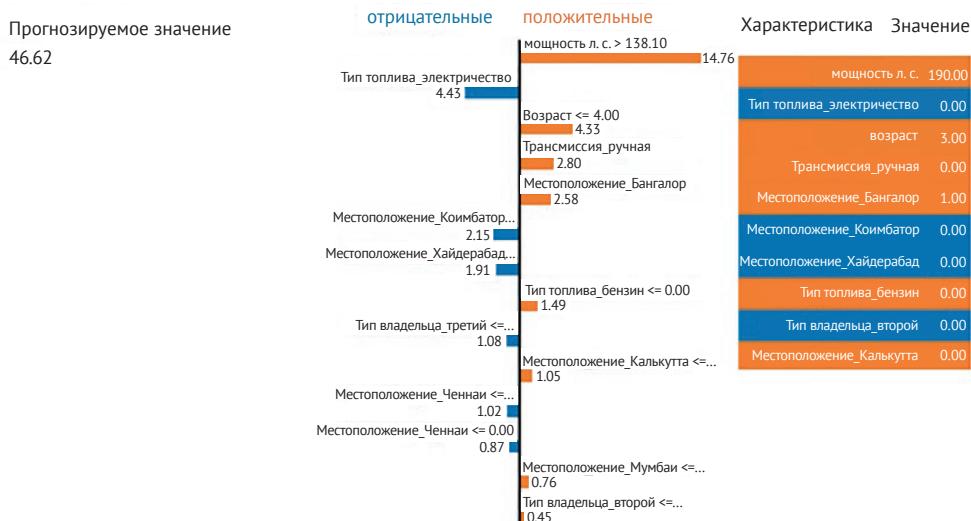


Рис. 3.13. Вклад положительных и отрицательных значений различных характеристик в прогнозируемое значение

Объяснятель для 60-й записи также может быть отображен в виде списка, как указано выше. Существует еще одна функция класса, называемая субмодульной выборкой, для создания границы глобального решения.

```
# Код для SP-LIME
import warnings
from lime import submodular_pick

# Не забудьте преобразовать кадры датафреймы в матричные значения
# SP-LIME возвращает объяснения набора образцов для обеспечения неизбыточной
# границы глобального решения исходной модели
sp_obj = submodular_pick.SubmodularPick(explainer, np.array(xtrain),
                                           new_model.predict,
                                           num_features=14,
                                           num_exps_desired=10)
[exp.show_in_notebook() for exp in sp_obj.sp_explanations ]
```

Здесь количество характеристик, используемых для создания границы глобального решения, равно 14, а желаемое количество экспериментов – 10. Эта часть скрипта займет некоторое время для завершения, поскольку вы генерируете несколько итераций (см. рис. 3.14).

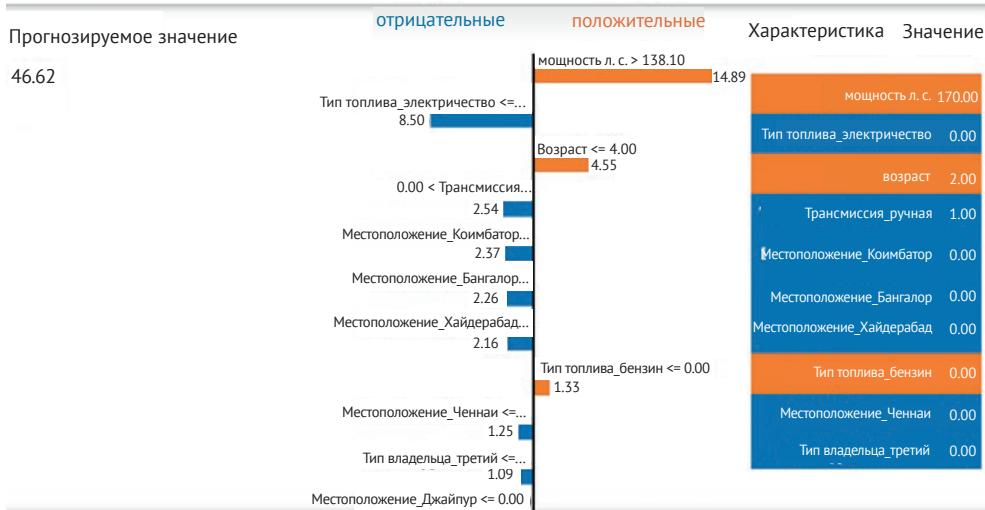


Рис. 3.14. Объяснения субмодульной выборки

LIME пытается найти компромисс между верностью и интерпретируемостью. Верность означает, что модель должна быть способна воспроизвести поведение в локальности экземпляра, используемого для прогноза. Интерпретируемость, как вы уже знаете, – это ясность, чтобы люди могли понять результат модели. Следующее уравнение показывает взаимосвязь между этими двумя параметрами:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \quad L(f, g, \pi_x) + \Omega(g).$$

Давайте разберемся в каждом из символов, используемых в приведенной выше формуле:

- f – исходный предиктор;
- g – объяснение модели;
- исходные характеристики – это x ;
- Π от x – это мера близости, определяющая локальность вокруг x ;
- Ω от g – это мера сложности модели для объяснения.

Одним из ограничений LIME является неточное определение соседства и близости. При выборке записей в окрестности используется гауссово распределение, которое не учитывает взаимосвязь между различными характеристиками. Если связь между зависимыми и независимыми переменными нелинейна, то объяснения не будут точными. Еще одним ограничением является субмодульная выборка, представляющая собой набор из n образцов, взятых из

набора данных, который лучше всего объясняет модель. Субмодульная выборка генерирует большое количество результатов, которые иногда трудно интерпретировать. Несмотря на ограничения, LIME часто используется для создания локальных интерпретаций отдельных прогнозов, не зависящих от модели. Он довольно популярен из-за простоты его вывода и легкости объяснения.

Объяснение Skater и модель ML

Skater – это унифицированный фреймворк с открытым исходным кодом, позволяющий интерпретировать модели всех форм и построить интерпретируемую систему машинного обучения, часто необходимую для реального мира. Skater поддерживает алгоритмы для раскрытия изученных структур модели «черного ящика» как глобально (вывод на основе полного набора данных), так и локально (вывод на основе отдельного прогноза). Изначально пакет был разработан Аароном Крамером (Aaron Kramer), Прамитом Чоудхари (Pramit Choudhary) и остальными членами команды DataScience.com, чтобы помочь аналитикам данных и энтузиастам получить более глубокое понимание моделей. Skater реализует это видение, предоставляя возможность выводить и отлаживать политики принятия решений модели по мере необходимости; таким образом, «человек включается в процесс».

Библиотека Skater может быть установлена с помощью скриптов установки pip или conda. Ее также легко установить в среде JuPyter. Объект модели может быть реализован двумя классами – моделью в памяти и развернутой моделью. Модели, которые являются вызываемыми функциями, могут быть использованы через модель в памяти; модели, которые развернуты и могут быть вызваны через Rest API, открываются через объект развернутой модели.

```
!pip install skater
import skater
from skater import Interpretation
# Интерпретация потребляет набор данных и, по желанию, некоторые метаданные,
# такие как имена
# характеристики и идентификаторы строк
intergrete = Interpretation(xtrain, feature_names=xtrain.columns)
intergrete
from skater.model import InMemoryModel
model = InMemoryModel(new_model.predict, examples = xtrain[:10])
model
```

Модуль интерпретации потребляет набор данных. Из интерпретатора можно извлечь значимость характеристики объекта регрессии. Для модели в памяти требуется функция predict и образец данных, для которого необходимо объяснить прогноз.

```
intergrete.feature_importance.feature_importance(model)
#Вычисляет важность всех характеристик, связанных с экземпляром модели.
#Поддерживает регрессию.
```

Из приведенной выше таблицы важности характеристик следует, что двумя важными характеристиками являются мощность и возраст, они в совокупности дают 75 % важности. Остальные 12 характеристик вносят 25 % важности. Функция отчета о модели показывает подробные метаданные, такие как тип обученной модели, тип выходной переменной, форма выхода, форма входа и вероятностный статус модели.

```
model.model_report(examples=xtrain)
```

Чтобы сгенерировать график частичной зависимости или двусторонний график частичной зависимости, интерпретатор требует повторной загрузки обучающего набора данных, а также имен характеристик.

```
# Skater интуитивно разработан для поддержки
# - модели в памяти: модель, которая в настоящее время строится, и экземпляр
# оценщика находится # в области видимости;
# - развернутой модели: модель, которая введена в эксплуатацию, или сторонняя
# модель.
deployed model
interpreter.load_data(xtrain, feature_names=xtrain.columns)
print("2-way partial dependence plots")
# Характеристики могут быть переданы в виде кортежа для построения графика
двусторонней
# частичной зависимости
pdp_features = [('Age', 'powerBhp')].
interpreter.partial_dependence.plot_partial_dependence(pdp_features,
    model, grid_resolution=10
)
```

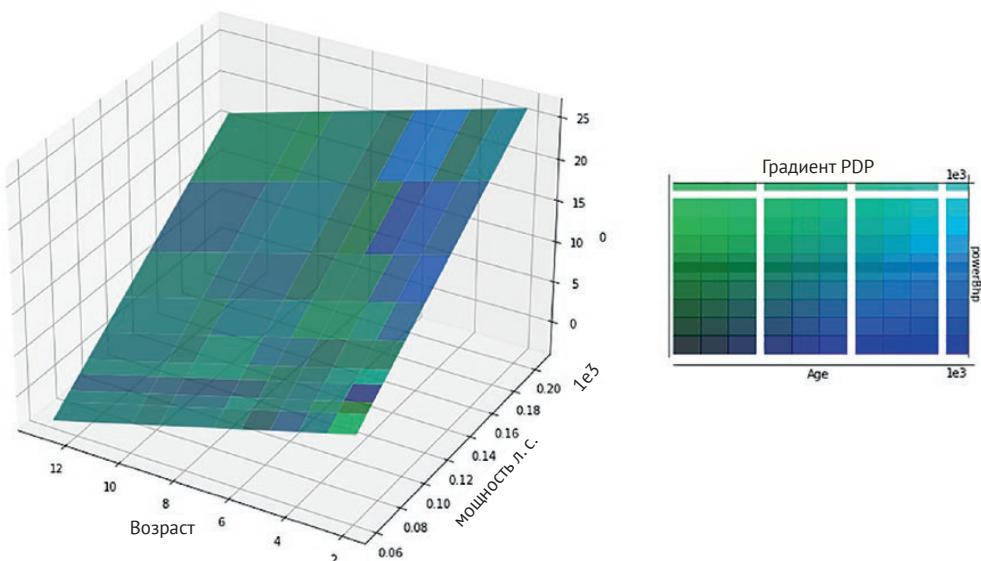


Рис. 3.15. График двусторонней частичной зависимости

График двусторонней частичной зависимости, изображенный на рис. 3.15, построен с использованием двух мощных характеристик и зависимой переменной. Цвета на графике показывают просто ассоциации и не должны интерпретироваться как причинно-следственные связи. На втором градиентном графике более высокий вклад в прогнозируемое значение показан зеленым цветом, а более низкий вклад в прогнозируемое значение этих двух характеристик, возраста и мощности совместно, – темно-синим цветом. Светло-зеленый цвет показывает более высокое предсказанное значение, а синий – более низкое. Аналогично можно построить график односторонней частичной зависимости. График PDP с доверительным интервалом может быть построен на том же графике односторонней частичной зависимости.

```
print("1-way partial dependence plots")
# или как независимые характеристики для односторонних графиков
pdp_features = ['powerBhp', 'Age']
interpreter.partial_dependence.plot_partial_dependence(
    pdp_features, model, grid_resolution=30
)
# Частичный график с эффектом дисперсии
interpreter.partial_dependence.plot_partial_dependence(
    pdp_features, model, grid_resolution=30, with_variance=True
)
```

Графики односторонней частичной зависимости здесь не приведены, так как они не представляют особой ценности. Вы можете построить столько графиков PDP из набора данных, сколько захотите. Соответствующие графики дают важное представление о характеристиках.

Объяснение ELI5 и модель ML

ELI5 – это пакет Python, который помогает отлаживать классификаторы машинного обучения и объяснять их прогнозы. Он обеспечивает поддержку фреймворка машинного обучения и пакета scikit-learn. В настоящее время ELI5 может объяснить веса и прогнозы линейных классификаторов и регрессоров scikit-learn, выводить деревья решений в виде текста или SVG, показывать важность характеристик и объяснять предсказания деревьев решений и ансамблей на основе деревьев. ELI5 понимает утилиты обработки текста из scikit-learn и может выделять текстовые данные соответствующим образом. Он также позволяет отлаживать конвейеры scikit-learn, содержащие HashingVectorizer, отменяя хеширование.

Веса характеристик также имеют свойство смешения. Это не что иное, как член перехвата для модели линейной регрессии. Характеристики перечисляются в порядке убывания их веса. ELI5 означает «Объясни, как будто мне пять» (explain like I am five). Он поддерживает все алгоритмы scikit-learn. Он также имеет глобальную интерпретацию с помощью функции `show_weights()` и локальную с помощью функции `show_prediction()`. В библиотеке ELI5 есть модель перестановки. Она работает только для глобальной интерпретации. Вот как она работает:

- сначала берет базовую модель из обучающего набора данных и вычисляет ошибку;
- перетасовывает значения характеристик, переобучает модель и вычисляет ошибку;
- сравнивает уменьшение ошибки после перестановки и до нее.

Характеристика считается важной, если после перетасовки дельта ошибки становится очень большой, и неважной, если после перетасовки уровень ошибок остается неизменным. Результат показывает среднее значение важности и стандартное отклонение характеристик при нескольких шагах перетасовки. Это не способ определить, какая характеристика влияет на производительность модели. Он говорит нам только о величине изменений в весах, поэтому мы не можем рассматривать веса как важность характеристик. Вышеописанный процесс продемонстрирован с помощью следующего скрипта:

```
pip install eli5
import eli5
eli5.show_weights(new_model, feature_names=list(xtrain.columns))
eli5.explain_weights(new_model, feature_names=list(xtrain.columns))
eli5.explain_prediction(new_model,xtrain.iloc[60])
from eli5.sklearn import PermutationImportance
perm = PermutationImportance(new_model)
perm.fit(xtest, ytest)
eli5.show_weights(perm,feature_names=list(xtrain.columns))
```

Объяснение индивидуального прогноза предоставляет локальную интерпретацию записи. Также оно сообщает о трех главных характеристиках, которые имеют наибольший вес при составлении прогноза. Из диаграммы важности перестановки видно, что мощность и возраст являются наиболее важными характеристиками. Это согласуется с аналогичными результатами из других библиотек XAI языка Python, которые вы реализовали до сих пор, принимая во внимание файл automobile.csv.

Теперь в качестве следующего шага давайте рассмотрим линейную модель классификации, которая также известна как модель логистической регрессии, чтобы понять различные аспекты XAI на ее основе. График частичной зависимости изображает взаимосвязь между целевой характеристикой, или зависимой переменной, и характеристикой, известной как независимая переменная. Взаимосвязь может быть линейной, нелинейной, криволинейной или более сложной, такой как кольцевое или циклическое монотонное отношение.

ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

Модель линейной регрессии применима, когда целевая характеристика является непрерывной, но, если последняя является бинарной, например 0 или 1, истина или ложь, принять или отвергнуть, линейная модель регрессии неприменима. Это связано с тем, что прогнозируемое значение целевой характеристики может выходить за пределы диапазона между 0 и 1, но ожидание заклю-

чается в том, чтобы ограничить результат двумя классами, так как вам нужно предсказать два класса по отдельности. Вот почему вам нужна модель логистической регрессии, которая использует двоичные значения для вычисления логарифмической вероятности, известной как отношение шансов. Оно линейно связано с характеристиками. Вот почему логистическая регрессионная модель известна как линейная модель.

Логистическая регрессия обычно используется, когда необходимо смоделировать вероятности и результат является двоичным или многочленным по своей природе. Здесь нельзя применить модель линейной регрессии, потому что в сценарии логистической регрессии переменная результата равна либо 0, либо 1 (а иногда она может быть и многочленной, когда возможны более двух исходов). Если применить линейную регрессию, то диапазон прогноза может выйти за пределы диапазона от 0 до 1. Это не даст вероятностей ни для одного из классов. В многочленной модели классификации разделение классов было бы большой проблемой. Линейная регрессионная модель, основанная на обычном методе наименьших квадратов, предполагает, что связь между зависимой и независимыми переменными является линейной, а логистическая регрессионная модель – что связь между ними является логарифмической.

Существует множество реальных сценариев, в которых интересующая нас переменная будет категориальной по своей природе, например покупать продукт или нет, одобрять кредитную карту или нет, опухоль является раковой или нет. Логистическая регрессия предсказывает не только класс зависимой переменной, но и вероятность принадлежности случая к тому или иному уровню зависимой переменной. Независимые переменные не обязательно должны быть нормально распределены и иметь одинаковую дисперсию. Логистическая регрессия относится к семейству нелинейных регрессий. Если зависимая переменная имеет два уровня, можно применить логистическую регрессию, но если более двух уровней, например высокий, средний и низкий, то можно применить модель полиномиальной регрессии. Независимые переменные могут быть непрерывными, категориальными или номинальными.

Модель логистической регрессии можно объяснить с помощью следующего уравнения:

$$\text{Sigmoid}(t) = 1/(1 + \exp(-t)).$$

Приведенная выше функция также известна как сигмоидная функция.

$$\ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k.$$

$\ln(P/1-P)$ – логарифмическая вероятность результата. Бета-коэффициенты, указанные в вышеприведенном уравнении, объясняют, как вероятность переменной результата увеличивается или уменьшается на каждую единицу увеличения или уменьшения объясняющей переменной. На рис. 3.16 показана форма сигмоидной функции. Она выглядит как s-образная кривая. Интерпретация модели логистической регрессии значительно отличается от модели линейной регрессии. Взвешенная сумма из правой части уравнения преобразуется в значение вероятности. Левая часть уравнения называется логарифмической ве-

роятностью, так как это логарифм отношения вероятности того, что событие произойдет, к вероятности того, что не произойдет. Более подробную интерпретацию логарифмической вероятности можно понять на примере, который мы рассмотрим далее.



Рис. 3.16. Логистическая/сигмовидная функция, по оси X показаны характеристики, по оси Y – вероятности

Чтобы объяснить модель логистической регрессии и то, как происходит принятие решения, необходимо понять, что такое вероятность и отношение шансов. Вы собираетесь использовать файл `churndata.csv`, который принадлежит домену `telecom` и содержит почти 3 333 записи и 18 характеристик. Вы собираетесь предсказать, будет ли клиент, скорее всего, изменять или не изменять заданные значения характеристик.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.metrics import confusion_matrix, classification_report
df_train = pd.read_csv('ChurnData.csv')
```

В качестве первого шага вы получаете данные, преобразуете определенные характеристики, которые уже находятся в строковом формате, и применяете кодировщик меток для выполнения преобразования. После преобразования вы разделяете данные на 80 % для обучения и 20 % для тестирования. При создании разделения на обучение и тестирование вы сохраняете пропорцию случаев оттока клиентов и случаев без оттока, чтобы сохранить баланс между классами. Затем вы обучаете модель, применяете ее к тестовым данным и сравниваете точность обучающего и тестового наборов данных.

```

del df_train['Unnamed: 0']

df_train.shape
df_train.head()

from sklearn.preprocessing import LabelEncoder
tras = LabelEncoder()

df_train['area_code_tr'] = tras.fit_transform(df_train['area_code'])

df_train.columns

del df_train['area_code']

df_train.columns

df_train['target_churn_dum'] = pd.get_dummies(df_train,
churn,prefix='churn',drop_first=True)
df_train.columns

del df_train['international_plan']
del df_train['voice_mail_plan']
del df_train['churn']
df_train.info()

df_train.columns

from sklearn.model_selection import train_test_split

df_train.columns

X = df_train[['account_length', 'number_vmail_messages', 'total_day_minutes',
'total_day_calls', 'total_day_charge', 'total_eve_minutes',
'total_eve_calls', 'total_eve_charge', 'total_night_minutes',
'total_night_calls', 'total_night_charge', 'total_intl_minutes',
'total_intl_calls', 'total_intl_charge',
'number_customer_service_calls', 'area_code_tr']]
Y = df_train['target_churn_dum']

```

Преобразуется только переменная кода региона. Остальные характеристики являются либо целыми числами, либо числами с плавающей точкой, что вполне позволяет продолжить обучение модели.

Теперь вы можете рассмотреть распределение вероятностей, логарифмические вероятности и отношения шансов, а также параметры модели для того, чтобы понять, как принимается решение при прогнозировании. Если взять ссылку на значения SHAP для объяснения вероятности модели логистической регрессии, то можно увидеть сильные эффекты взаимодействия. Это связано с неаддитивным поведением модели логистической регрессии в пространстве вероятностей. Если использовать логарифмическую вероятность модели в качестве результата, можно увидеть сильную корреляцию или идеальную линейную зависимость между входом и выходом модели.

```
xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=0.20,stratify=Y)
log_model = LogisticRegression()

log_model.fit(xtrain,ytrain)

print("training accuracy:", log_model.score(xtrain,ytrain)) # точность обучения
print("test accuracy:", log_model.score(xtest,ytest)) # точность тестирования
```

Рассматривая точность, можно сделать вывод, что это хорошая модель и, возможно, нет проблемы чрезмерной подгонки, поскольку нет никаких расхождений в точности обучения и тестирования.

```
np.round(log_model.coef_,2)
```

```
log_model.intercept_
```

```
X.columns
```

Существуют две служебные функции для получения требуемого результата, которые можно использовать в графическом представлении значений SHAP.

```
# Предоставление вероятности в качестве выходных данных
```

```
def model_churn_proba(x):
    return log_model.predict_proba(x)[:,1]
```

```
# Предоставление логарифмических вероятностей в качестве выходных данных
```

```
def model_churn_log_odds(x):
    p = log_model.predict_log_proba(x).
    return p[:,1] - p[:,0]
```

Поскольку вы уже рассмотрели интерпретацию графика зависимости в разделе этой главы о регрессии, аналогичную интерпретацию можно сделать и для модели логистической регрессии. График частичной зависимости берет в качестве примера запись из набора данных, поскольку она является локальным объяснением.

```
# построение стандартного графика частичной зависимости
sample_ind = 25
fig,ax = shap.partial_dependence_plot(
    "total_day_minutes", model_churn_proba, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)
```

Для записи № 25 график частичной зависимости характеристики «общая длительность звонков в дневное время (мин)» на рис. 3.17 показывает, что связь между значением вероятности или ожидаемым значением функции и характеристикой положительна, но не линейна.

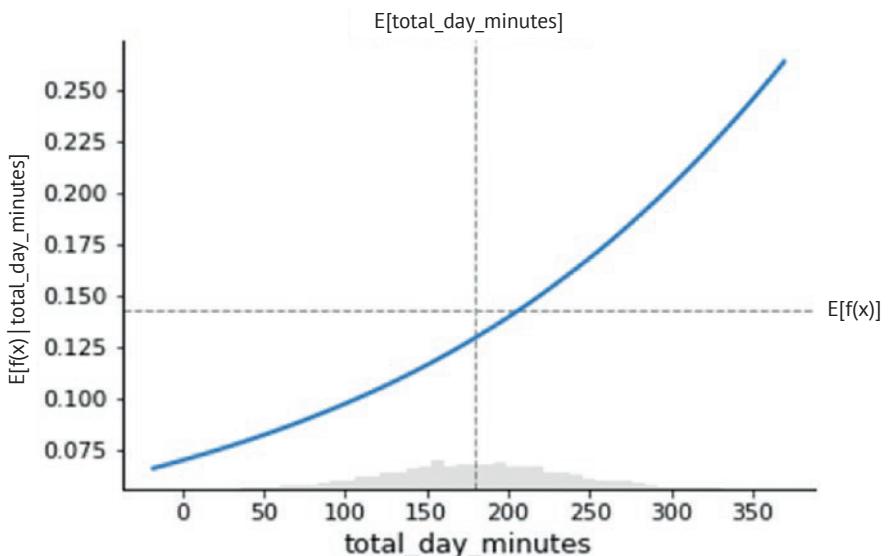


Рис. 3.17. График частичной зависимости между общей длительностью звонков в дневное время (мин) и вероятностью оттока кадров

```
# вычисление значения SHAP для линейной модели
background_churn = shap.maskers.Independent(X, max_samples=1000)
explainer = shap.Explainer(log_model, background_churn, feature_names=list(X.columns))
shap_values_churn = explainer(X)
shap_values = pd.DataFrame(shap_values_churn.values)
shap_values.columns = list(X.columns)
shap_values
```

Модуль `shap.Explainer` имеет важные параметры, перечисленные в табл. 3.9.

Таблица 3.9. Параметры модуля Explainer из библиотеки SHAP

Параметр	Описание
Модель (Model)	Имя объекта модели
Маскировщик (Masker)	Это функция для маскировки скрытых характеристик
Ссылка (Link)	Функция используется для сопоставления между выходными единицами модели и единицами значения SHAP
Алгоритм (Algorithm)	Он используется для оценки значений Шепли, которые называются auto, permutation, partition, tree, kernel, sampling, linear, deep и gradient. По умолчанию используется auto

Если посмотреть на диаграмму зависимости между абонентским стажем в телефонной компании и значениями SHAP для этой характеристики (рис. 3.18), наблюдается идеальная линейная зависимость.

```
shap.plots.scatter(shap_values_churn[:, 'account_length'])
```

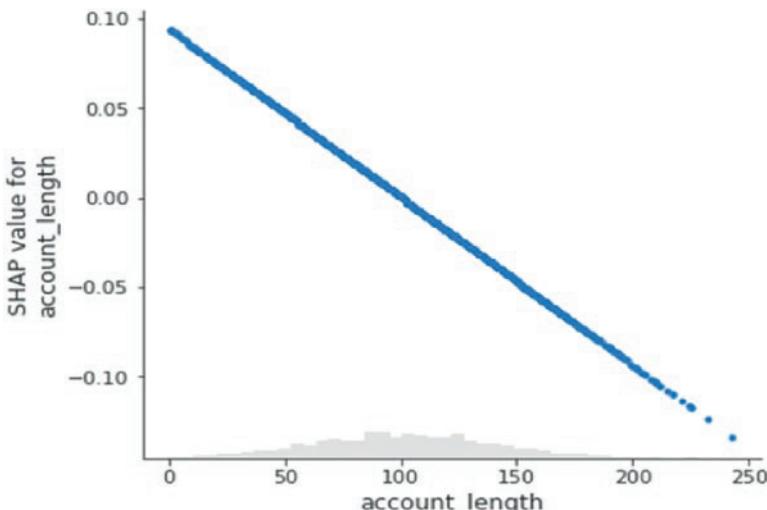


Рис. 3.18. Связь между характеристикой абонентского стажа и значениями SHAP

Значения SHAP и средние абсолютные значения для соответствующих характеристик представлены на следующем графике. На нем показано, какая характеристика имеет больший вес в задаче классификации. Число обращений в службу поддержки клиентов является хорошим фактором для отсева, так как люди с большим количеством жалоб чаще звонят в службу поддержки, и, следовательно, это «сидящие на заборе», они могут уйти в любой момент. Вторым по важности фактором является длительность вызовов за день в минутах, а третьим – число голосовых сообщений по электронной почте. Далее вы можете увидеть, что семь наименее важных характеристик объединены вместе. Существует и другой способ представления значений SHAP, в котором представлено максимальное абсолютное значение SHAP для каждой характеристики, но существенной разницы между двумя этими графиками нет. Существует еще один вид, называемый графиком пчелиного роя, который показывает значение SHAP и его влияние на результат модели. Тепловая карта значений SHAP для тысяч записей показывает плотность значений SHAP по отношению к характеристикам модели. Высокое значение SHAP наблюдается для лучшей характеристики, и постепенно важность характеристик уменьшается, а также уменьшается значение SHAP (см. рис. 3.19–3.22).

```
# построение стандартного графика частичной зависимости
sample_ind = 25
fig,ax = shap.partial_dependence_plot(
    "number_vmail_messages", model_churn_proba, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)
shap_values_churn.feature_names

# вычисление значений SHAP для линейной модели
explainer_log_odds = shap.Explainer(log_model, background_churn,
feature_names=list(X.columns))
```

```
shap_values_churn_log_odds = explainer_log_odds(X)
shap_values_churn_log_odds
shap.plots.bar(shap_values_churn_log_odds)
```

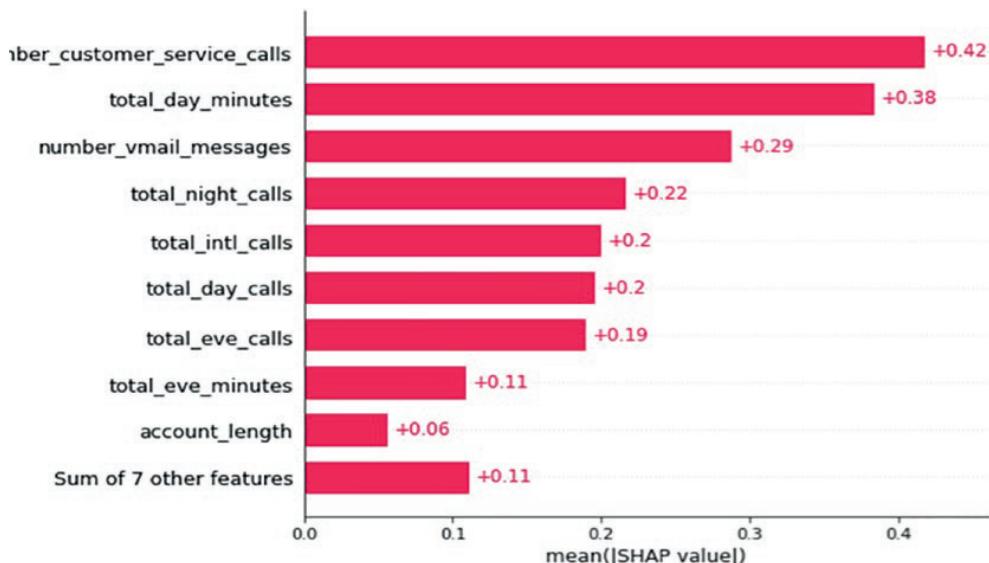


Рис. 3.19. Средние абсолютные значения SHAP, вносимые каждой характеристикой

```
shap.plots.bar(shap_values_churn_log_odds.abs.max(0))
```

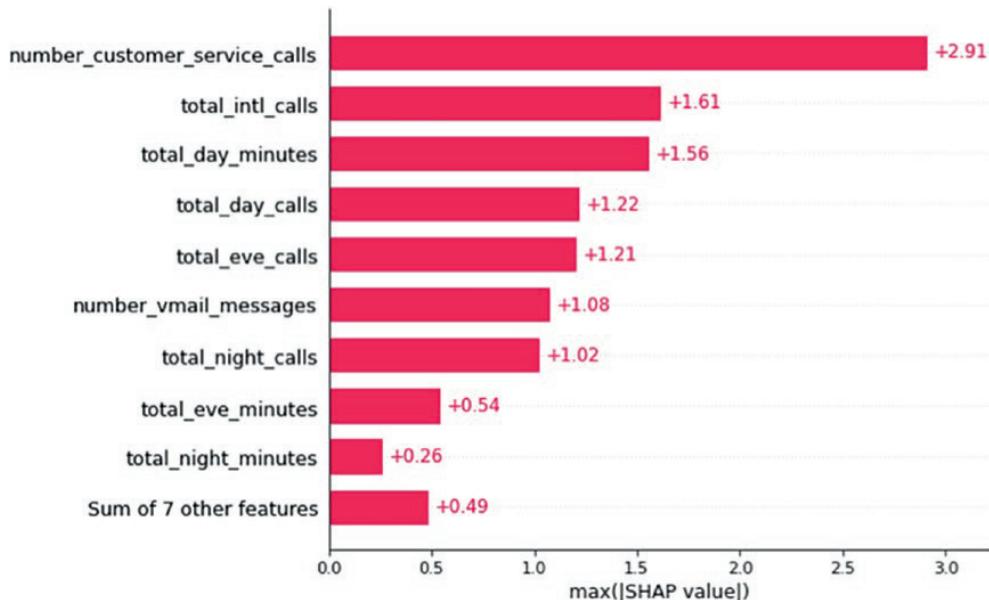


Рис. 3.20. Максимальное абсолютное значение SHAP, вносимое каждой характеристикой

```
shap.plots.beeswarm(shap_values_churn_log_odds)
```

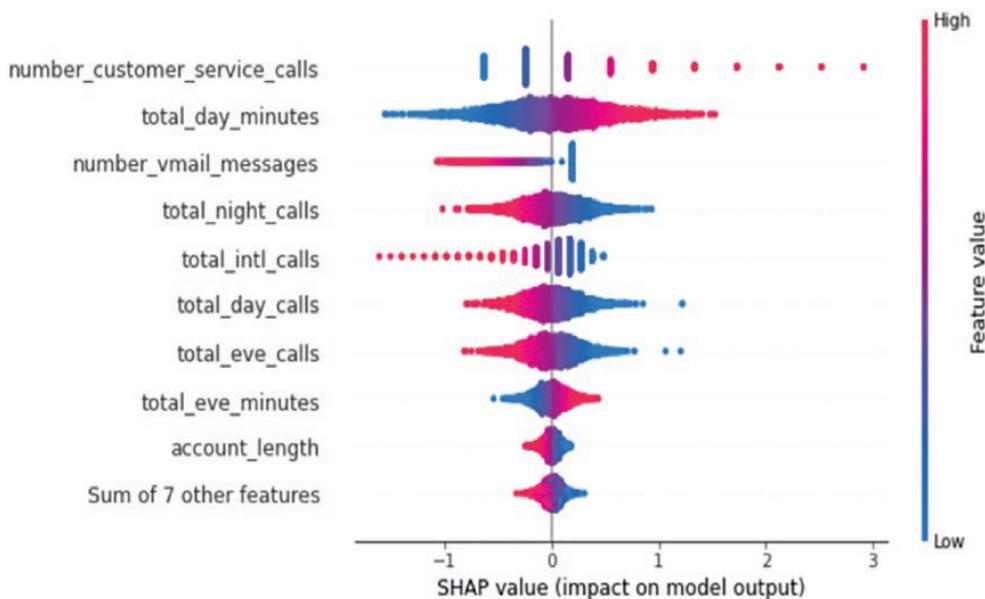


Рис. 3.21. Положительный и отрицательный вклад характеристик в значение SHAP

```
shap.plots.heatmap(shap_values_churn_log_odds[:1000])
```

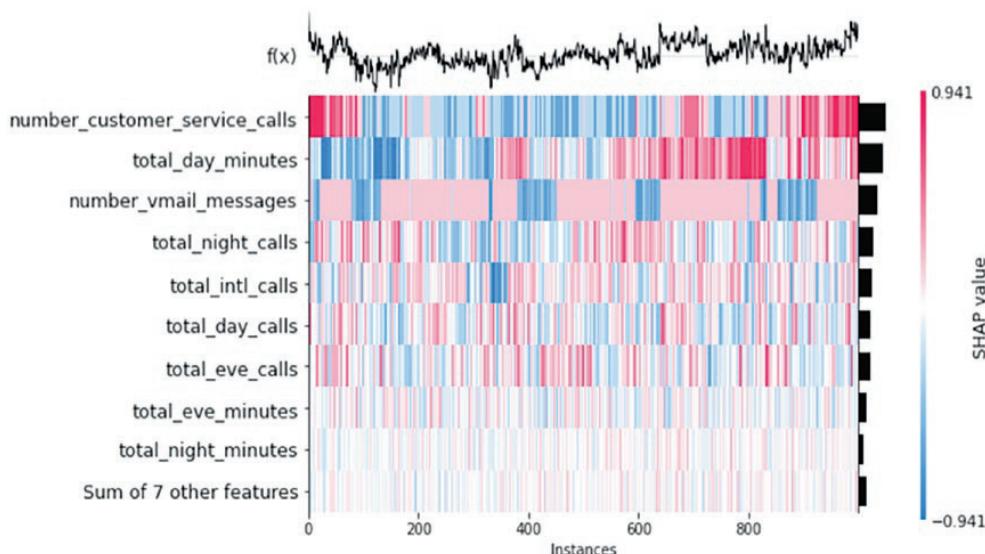


Рис. 3.22. Доля значений SHAP, вносимых каждой характеристикой, для экземпляров, используемых в процессе обучения

```
temp_df = pd.DataFrame()
temp_df['Feature Name'] = pd.Series(X.columns)
temp_df['Coefficients'] = pd.Series(log_model.coef_.flatten())
temp_df.sort_values(by='Coefficients', ascending=False)
```

Таблица 3.10. Значения весовых коэффициентов характеристик

Код характеристики	Название характеристики	Коэффициент
14	Число обращений в службу поддержки клиентов	0.383573
2	Всего минут вызовов за день	0.008251
4	Оплата за день	0.001378
5	Всего минут дневных вызовов	0.000947
7	Оплата за вечер	0.000098
10	Оплата за ночь	-0.000048
13	Оплата международных вызовов	-0.000196
11	Всего минут международных вызовов	-0.000464
0	Абонентский стаж	-0.000573
8	Всего минут ночных вызовов	-0.001730
3	Всего дневных вызовов	-0.009254
9	Всего ночных вызовов	-0.010050
6	Всего вечерних вызовов	-0.012706
1	Число голосовых сообщений по электронной почте	-0.019944
15	Код региона	-0.033119
12	Всего международных вызовов	-0.097870

ИНТЕРПРЕТАЦИЯ

Когда вы увеличиваете значение одной характеристики на единицу, уравнение модели дает два коэффициента: один – базовый, а другой – приращение значения характеристики. Цель состоит в том, чтобы посмотреть на соотношение шансов при каждом увеличении или уменьшении значения. Изменение на единицу приводит к изменению отношения шансов за счет соответствующих экспоненциальных бета-коэффициентов. Это можно объяснить с помощью следующего уравнения, где β_0 – член перехвата, от β_1 до β_k – параметры модели, от x_1 до x_k – независимые предикторы для модели:

$$\frac{P(y=1)}{1-P(y=1)} = odds = \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p).$$

Назовем правую часть уравнения $\exp(a)$, где a – уравнение, представляющее концепцию линейной регрессии. Если увеличить любой параметр модели, то уравнение изменится на одну единицу, поэтому назовем его b , и тогда правая часть уравнения станет равной $\exp(b)$. Отношение шансов в зависимости от

изменения значения предиктора на одну единицу будет иметь вид: $\text{odds_new}/\text{odd_old} = \exp(a-b)$. В таком формате можно интерпретировать все числовые характеристики. Это представление также можно использовать для всех категориальных или бинарных характеристик.

Вывод LIME

Для объяснения модели логистической регрессии можно использовать значения SHAP. Однако сложность заключается во времени. Если у вас есть миллион записей и взята достаточно большая выборка для генерации всех перестановок и комбинаций, чтобы прийти к глобальному уровню для объяснения локальной точности, вам потребуется больше времени. Чтобы избежать этого узкого места при обработке больших наборов данных, LIME предлагает скорость при генерации объяснений. Чтобы объяснить табличные матричные данные, которые являются структурированными, необходимо использовать Lime Tabular Explainer. Для числовых характеристик внесите в них возмущение, используя выборку из стандартного нормального распределения ($\text{Normal}(0.1)$) и выполнив обратную операцию среднего центрирования и масштабирования, в соответствии со средним значением и стандартным отклонением в обучающих данных. Для категориальных характеристик возмутите их путем выборки в соответствии с обучающим распределением и создайте бинарную характеристику, которая равна 1, если значение совпадает с объясняемым экземпляром.

При создании объяснителя LIME необходимо передать данные в виде массива, предоставить список имен столбцов, указать имя целевого столбца, задать режим регрессии и классификацию на основе задачи машинного обучения, которую вы планируете использовать. Опция `verbose` предназначена для включения прогнозов модели.

```
import lime
import lime.lime_tabular
explainer = lime.lime_tabular.LimeTabularExplainer(np.array(xtrain),
    feature_names=list(xtrain.columns),
    class_names=['target_churn_dum'],
    verbose=True, mode='classification')
# эта запись представляет собой сценарий без оттока клиентов
exp = explainer.explain_instance(xtest.iloc[0], log_model.predict_proba, num_
features=16)
exp.as_list()
```

Для создания объяснений после создания объекта модели можно проверить наличие индивидуальных и глобальных прогнозов. В классификации, где у вас есть два или несколько классов, можете создать важность характеристик для каждого класса в столбце характеристик. В данном примере рассматриваются две записи: запись 1, где модель правильно предсказывает результат, и запись 20, где неверно предсказывает исход. Вы объясните для обоих сценариев, почему такое решение было принято моделью. Характеристики,

имеющие положительную связь с целевым классом, имеют положительное число, а класс с отрицательной связью имеет отрицательный знак. Вы можете показать результаты в виде таблиц. Также можете ограничить представление, выделив в нем наиболее важные характеристики.

```
pd.DataFrame(exp.as_list())
```

Весовые коэффициенты характеристик в представлении датафрейма показаны в табл. 3.11.

Таблица 3.11. Характеристики с порогом и их весовые коэффициенты для значения прогноза

0	1
0	Число обращений в службу поддержки клиентов <= 1.00
1	Всего минут вызовов за день <= 143.70
2	Число голосовых сообщений по электронной почте <= 0.00
3	Всего вечерних вызовов > 114.00
4	101.00 < всего ночных вызовов <= 114.00
5	Всего минут вечерних вызовов > 235.07
6	Абонентский стаж > 126.00
7	1.00 < код региона <= 2.00
8	2.27 < оплата международных вызовов <= 2.75
9	87.00 < всего вызовов за день <= 101.00
10	166.93 < всего минут ночных вызовов <= 200.40
11	Оплата дневных вызовов <= 24.43
12	3.00 < всего международных вызовов <= 4.00
13	7.51 < оплата ночных вызовов <= 9.02
14	Оплата вечерних вызовов > 19.98
15	8.40 < всего минут международных вызовов <= 10.20

В соответствии с записью № 1 может быть представлен следующий рисунок – рис. 3.23. Член перехвата равен 0.126. локальная предсказанная вероятность оттока с помощью LIME равна 0.18. и предсказанная вероятность оттока по модели логистической регрессии – 0.15. По сути, это член перехвата плюс все веса различных характеристик. Поскольку вероятность оттока меньше, прогноз классифицируется как сценарий без оттока. Синяя полоса на рис. 3.23 показывает вероятность отсутствия оттока 0.84 и вероятность оттока 0.16. Общая важность характеристик по их весам указана в правой части рисунка. В средней части даны веса по значению характеристик.

```
exp.show_in_notebook(show_table=True)
```

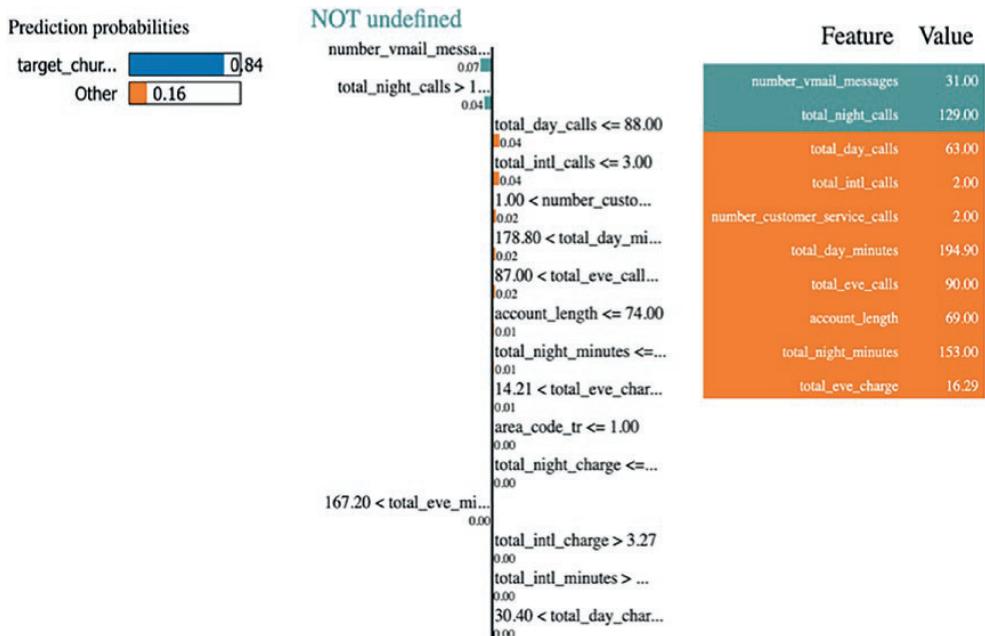


Рис. 3.23. Сводка и локальная интерпретация для записи № 1 из тестового набора

Для записи 20 из тестового набора модель предсказывает один результат, а фактический тест – другой. Это сценарий, в котором прогноз модели отличается от реальности, и поэтому модель должна объяснить, почему это произошло. Вы можете получить лучшее представление, используя локальный экземпляр LIME.

```
# Это сценарий оттока
exp = explainer.explain_instance(xtest.iloc[20], log_model.predict_proba,
num_features=16)
exp.as_list()

exp.show_in_notebook(show_table=True)
xtest.iloc[20]
```

На рис. 3.24 вероятность прогноза имеет две полосы: синяя показывает вероятность отсутствия оттока, а оранжевая – вероятность оттока. На рисунке также показаны характеристики и их весовые коэффициенты, вносящие вклад в оранжевую полосу.

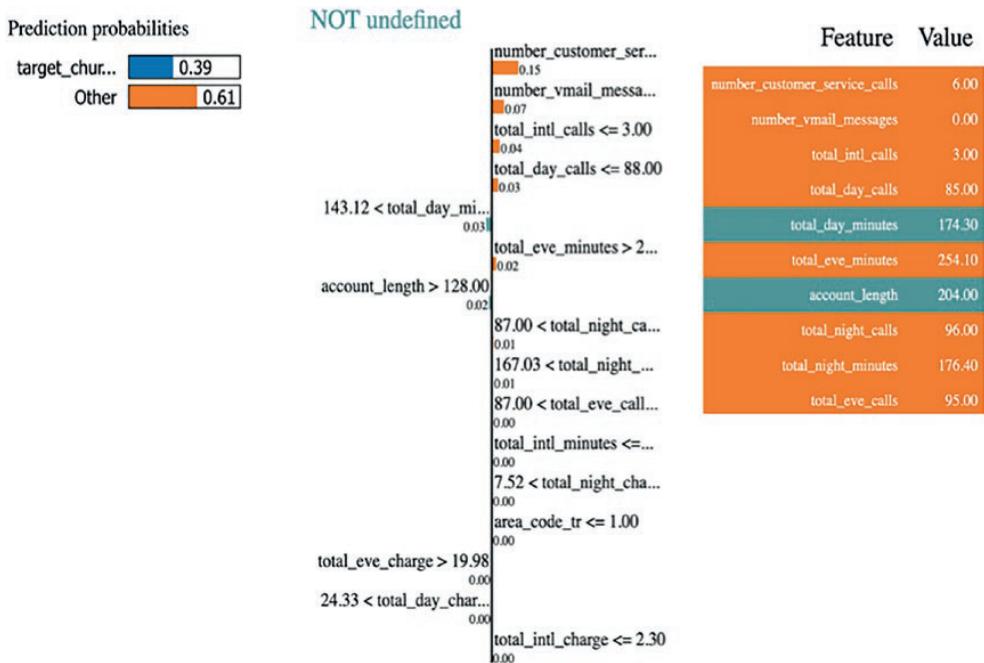


Рис. 3.24. Локальная интерпретация для записи 20 из тестового набора

За исключением двух характеристик, общего количества минут вызовов в день и абонентского стажа, все остальные характеристики вносят вклад в вероятность оттока, следовательно, модель правильно предсказывает результат и предсказание объяснено. Пороговые значения для каждой из характеристик приведены вместе с их весами. Это дает лучшую картину для деловых людей, чтобы понять поведение прогностической модели.

```
explainer = lime.lime_tabular.LimeTabularExplainer(np.array(xtrain),
    feature_names=list(xtrain.columns),
    class_names=['target_churn_dum'],
    verbose=True, mode='classification')

# Код для SP-LIME импортирует предупреждения из субмодульного выбора LIME
# SP-LIME возвращает объяснения для выборочного набора, чтобы обеспечить
неизбыточную
# глобальную границу принятия решений исходной модели
sp_obj = submodular_pick.SubmodularPick(explainer, np.array(xtrain),
log_model.predict_proba,
num_features=14,
num_exps_desired=10)
```

Опция субмодульного выбора обеспечивает глобальную границу принятия решений исходной модели. Вы можете использовать объект объяснителя, обучающий набор данных и извлеченные вероятности из обученной модели,

а затем указать количество характеристик, которые должны присутствовать в описании, и желаемое количество выражений (см. рис. 3.25–3.28).

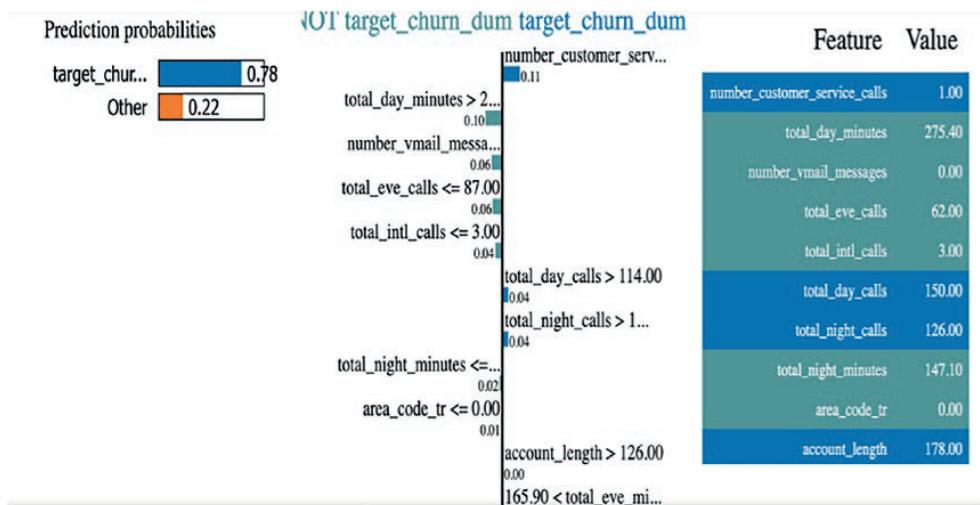


Рис. 3.25. Локальная интерпретация для первой из 10 записей

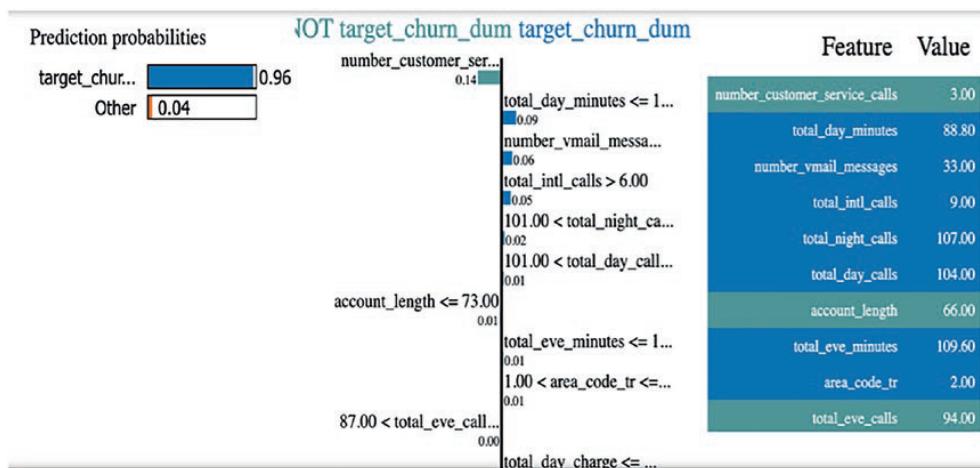


Рис. 3.26. Локальная интерпретация для второй из 10 записей

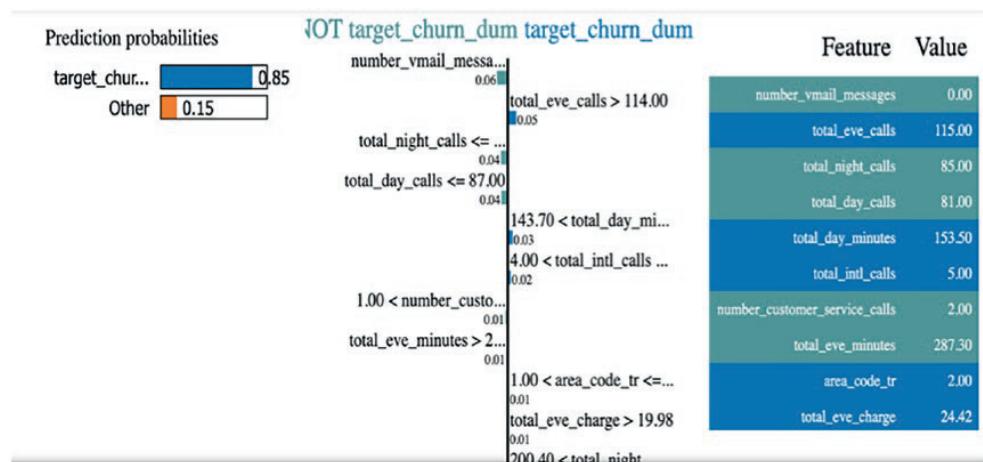


Рис. 3.27. Локальная интерпретация для третьей из 10 записей

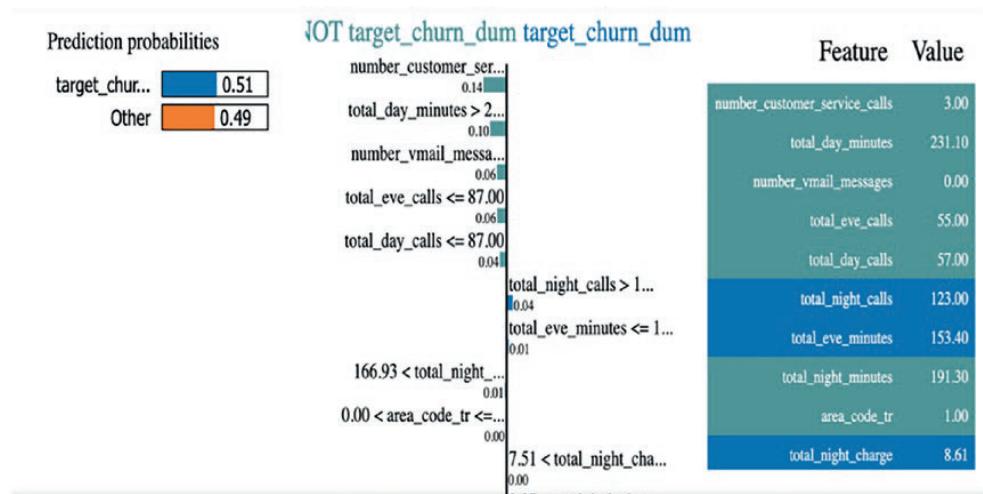


Рис. 3.28. Локальная интерпретация для четвертой из 10 записей

Skater генерирует те же результаты, что и библиотека LIME, поэтому для него описания не будут отличаться от рассмотренных на примере библиотеки LIME. ELI5 в основном используется для классификации текстов, поэтому она не применима для объяснения линейных или логистических регрессионных моделей (см. табл. 3.12 для получения дополнительной информации).

Таблица 3.12. Использование библиотек

Имя библиотеки	Определение	Когда использовать	Преимущества	Ограничения
SHAP	Использует значения Шепли для объяснения любой модели машинного обучения	Табличные данные, изображения	Лучшее объяснение с большим количеством метрик и статистики	Не очевидны
LIME	Локальное интерпретируемое объяснение модели (LIME)	Для локальной интерпретации, табличные данные	Хорошо подходит для объяснения отдельных экземпляров	Глобальное объяснение не интуитивное. Не подходит для изображений
Skater	Общий рабочий процесс в рамках пакета Skater заключается в создании интерпретации, создания модели и выполнении алгоритмов интерпретации	Табличные данные, доступные в двух модулях - in-memory и развернутая модель	Обучение модели и интерпретация должны быть запущены один раз. Нет необходимости запускать как отдельный процесс	Только несколько моделей поддерживаются. Не полный охват всех типов моделей
ELI5	Пакет Python, помогающий отлаживать классификаторы машинного обучения и объяснять их предсказания	Модели Scikit-learn, классификация текстов, объяснение модели Keras	Хорошо подходит для классификации текстов	Не является зрелой библиотекой для всех остальных задач. Объяснения очень простые

ЗАКЛЮЧЕНИЕ

В этой главе вы узнали, как интерпретировать линейные модели, модели линейной регрессии для прогнозирования и логистические регрессии для бинарной классификации. Аналогичным образом модель логистической регрессии также может быть расширена на полиномиальную классификацию. Линейные модели более просты для интерпретации, и все хорошо понимают, как эти модели работают. Поэтому к ним всегда существует высокая степень доверия. В этой главе вы рассмотрели различные аспекты создания представлений для линейных моделей с использованием объясняемых библиотек ИИ, таких как LIME и SHAP. В следующей главе вы узнаете об объяснимости для нелинейных моделей.

ГЛАВА 4

Объяснимость для нелинейных моделей

В этой главе рассматривается использование объясняемых библиотек языка Python LIME, SHAP и Skope-rules, основанных на ИИ, для объяснения решений, принимаемых нелинейными моделями в задачах контролируемого обучения со структуризованными данными. Вы познакомитесь с различными способами объяснения нелинейных и древовидных моделей и их решений при прогнозировании зависимой переменной. В задаче контролируемого машинного обучения есть целевая (зависимая) и набор независимых переменных. Задача состоит в том, чтобы предсказать зависимую переменную как взвешенную сумму входных или независимых переменных, где существует высокая степень взаимодействия характеристик и сложная нелинейная взаимосвязь.

Нелинейные модели

Дерево решений – это нелинейная модель, которая связывает независимую переменную с зависимой. На локальном уровне это можно рассматривать, как кусочно-линейную регрессию, но на глобальном уровне это нелинейная модель, так как нет однозначной связи между зависимой и независимыми переменными. В отличие от линейной регрессионной модели не существует математического уравнения, показывающего взаимосвязь между входными и выходными переменными. Если мы сохраним параметр максимальной глубины дерева на бесконечном уровне, то дерево решений может идеально подходить к данным, что является классическим сценарием избыточной подгонки модели. Независимо от того, является ли обучающий набор данных линейно разделяемым или нет, деревья решений склонны к чрезмерной подгонке. С этим необходимо бороться. Обычно прибегают к обрезке деревьев, чтобы получить наиболее подходящую модель. Можно рассматривать дерево решений как последовательность условных утверждений, результатом которых является значение или класс в выходном столбце. Например, если человеку 45 лет, он работает в частном секторе и имеет степень доктора философии, то он определенно зарабатывает более 50 тыс. долл. в год. Деревья решений – это алгоритм контролируемого обучения, который применяется, когда существует

бесконечное количество возможных комбинаций характеристик, способных повлиять на целевой столбец. В дереве решений мы разбиваем генеральную совокупность на две или более подгрупп на основе наиболее значимого разделятеля или дифференциатора во входных переменных.

Преимущественно алгоритм дерева решений следует алгоритму ID3 (iterative dichotomiser 3 – итеративный дихотомайзер 3), хотя существуют и другие, такие как C4.5, CART, MARS и СНИД. Он основывается на таких моментах, как:

- определение лучшего атрибута или характеристики на основе информационного значения (имеющего высокую предсказательную ценность) из набора данных и размещение его в корне дерева;
- разделение обучающего набора данных на поднаборы таким образом, чтобы каждый поднабор имел одинаковое значение для атрибута;
- повторение двух вышеуказанных шагов до тех пор, пока каждому классу не будет сопоставлен один узел или минимальное количество образцов в узле.

В моделях дерева решений, чтобы предсказать метку класса для записи, мы начинаем с корня дерева. Мы сравниваем корневые атрибуты, используем лучший атрибут в начале и в дальнейшем продолжаем развивать дерево решений. Способность дерева решений объяснять прогнозы довольно хороша. Проще говоря, дерево решений предоставляет правила, которые могут быть непосредственно использованы любым приложением. Правила в основном представляют собой набор операторов «если/иначе» (if/else). В ситуациях, когда существует взаимосвязь между характеристиками, например взаимодействие между ними, квадраты или кубы характеристик связаны с зависимой переменной и т. д. В таких ситуациях линейная и логистическая регрессия обязательно потерпят неудачу. Это происходит, потому что количество взаимодействий может быть большим. Модели на основе деревьев учитывают значения характеристик, определяют пороговое значение, разбивают характеристику на две части и продолжают отращивать ветви дерева. Процесс разбиения данных на несколько подгрупп помогает уловить нелинейность, существующую в наборе данных. Аналогичным образом квадратичные и кубические характеристики также следуют правилам «если/иначе», как указано в модели.

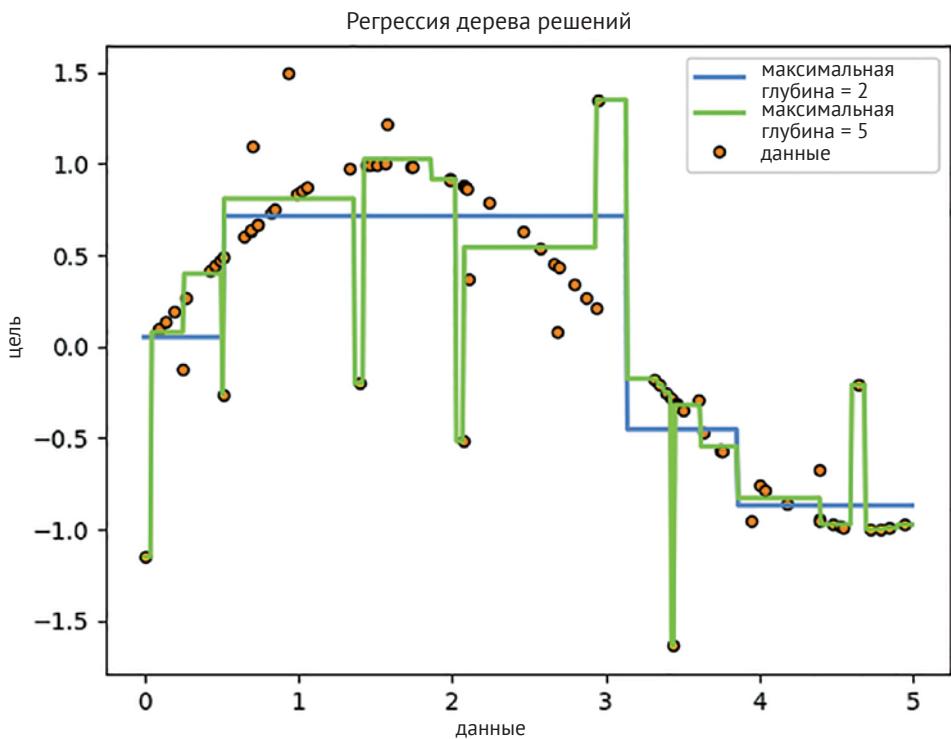


Рис. 4.1. Регрессия дерева решений учитывает нелинейность

Как показано на рис. 4.1 (источник: <https://scikit-learn.org/>), взаимосвязь между данными и целью является нелинейной. Нелинейность аппроксимируется моделью дерева решений путем увеличения параметра максимальной глубины (maximum depth). По мере увеличения такого параметра с 2 до 5 все точки вне кривой также становятся частью модели, поэтому они появляются в наборе правил, генерируемых моделью дерева решений.

ОБЪЯСНЕНИЕ ДЕРЕВА РЕШЕНИЙ

На рис. 4.2 изображено начало дерева. Это происходит с помощью корневой характеристики. Логика ветвления основана на наилучшей возможной характеристике, которая создает разветвление.

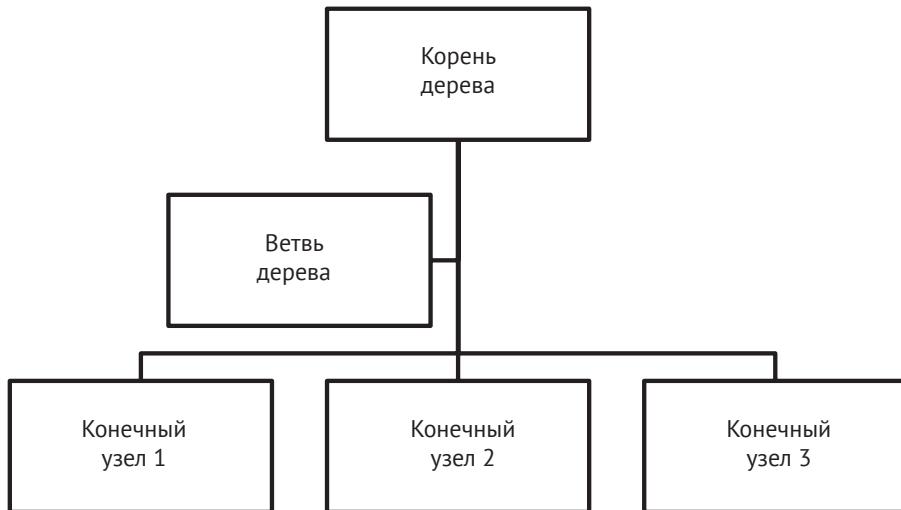


Рис. 4.2. Анатомия дерева решений

Терминальные узлы – это конечные узлы, на которых заканчивается построение дерева. Для прогнозирования результата конкретной записи используется средний результат по всем данным.

Как показано на рис. 4.3, объяснимость модели может быть достигнута двумя способами – с помощью библиотеки XAI и основной библиотеки ML.

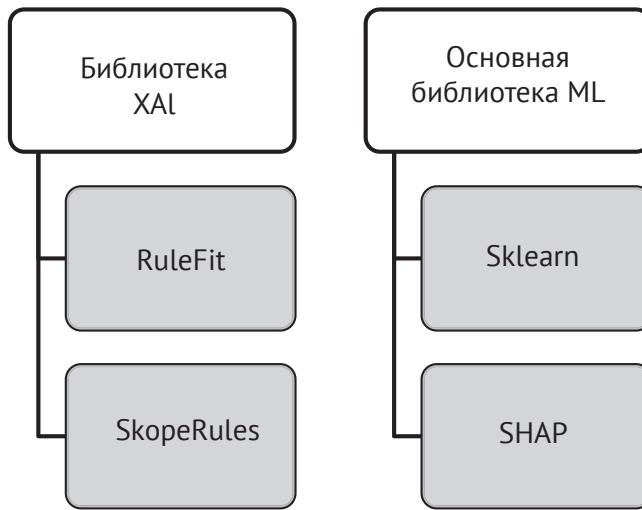


Рис. 4.3. Два способа объяснения древовидных моделей

Разница в том, что при использовании родительской библиотеки ML возможно применять уже обученную модель. Если же мы берем библиотеку XAI, то, возможно, придется обучать модель ML заново. Теперь вернемся к объяснению результатов моделирования. Если мы повторно обучаем модель, это добавляет

накладные расходы на ее переобучение. Настройка гиперпараметров и выбор лучшей модели – это разные процессы, и они занимают много времени.

Однако, если у нас есть легкодоступная модель и мы можем просто создать ее объяснения, это выгодно для конечного пользователя. В следующем коде показаны библиотеки, необходимые для создания модели дерева решений. Вы собираетесь использовать данные об оттоке, которые использовали в главе 3. Этот набор данных содержит 20 характеристик и 3 333 записи. Для некоторых категориальных переменных, таких как код региона, необходимо преобразовать переменную и выполнить кодирование меток.

Подготовка данных для модели дерева решений

В следующем скрипте показаны необходимые библиотеки для подготовки модели дерева решений. Вы также импортируете необходимые данные для разработки модели в формате CSV.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np, pandas as pd, matplotlib.pyplot as plt
from sklearn import tree, metrics, model_selection, preprocessing
from IPython.display import Image, display
from sklearn.metrics import confusion_matrix, classification_report

df_train = pd.read_csv('ChurnData.csv')
del df_train['Unnamed: 0']
df_train.shape
df_train.head()
```

Дополнительный столбец, присутствующий в данных, удаляется. Функция `shape` дает представление о количестве строк и столбцов, присутствующих в наборе данных. Кроме того, функция `head` предоставляет первые пять записей из набора данных.

```
from sklearn.preprocessing import LabelEncoder

tras = LabelEncoder()

df_train['area_code_tr'] = tras.fit_transform(df_train['area_code'])

df_train.columns

del df_train['area_code']

df_train.columns
```

В данных присутствуют некоторые строковые колонки, например код региона. Эти данные необходимо преобразовать в цифровой формат с помощью кодировщика меток (label encoder). Это нужно для обучения модели, поскольку

строковые переменные нельзя использовать в том виде, в котором они представлены.

```
df_train['target_churn_dum'] = pd.get_dummies(df_train.  
churn,prefix='churn',drop_first=True)  
df_train.columns  
del df_train['international_plan']  
del df_train['voice_mail_plan']  
del df_train['churn']  
df_train.info()  
df_train.columns
```

Следующим шагом будет разделение набора данных на обучающий и тестовый наборы. Обучающий набор предназначен для разработки модели, а тестовый – для получения выводов или прогнозов.

```
from sklearn.model_selection import train_test_split  
  
df_train.columns  
  
X = df_train[['account_length', 'number_vmail_messages', 'total_day_minutes',  
    'total_day_calls', 'total_day_charge', 'total_eve_minutes',  
    'total_eve_calls', 'total_eve_charge', 'total_night_minutes',  
    'total_night_calls', 'total_night_charge', 'total_intl_minutes',  
    'total_intl_calls', 'total_intl_charge',  
    'number_customer_service_calls', 'area_code_tr']]  
Y = df_train['target_churn_dum']  
  
xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=0.20,stratify=Y)  
  
tree.DecisionTreeClassifier() # обычная модель дерева  
  
# гиперпараметры по умолчанию для классификатора дерева решений  
class_weight=None,  
criterion='gini',  
max_depth=None,  
max_features=None,  
max_leaf_nodes=None,  
min_impurity_decrease=0.0,  
min_impurity_split=None,  
min_samples_leaf=1,  
min_samples_split=2,  
min_weight_fraction_leaf=0.0,  
presort=False,  
random_state=None,  
splitter='best'  
  
dt1 = tree.DecisionTreeClassifier()  
dt1.fit(xtrain,ytrain)
```

```
print(dt1.score(xtrain,ytrain))
print(dt1.score(xtest,ytest))
```

Таблица 4.1. Объяснение гиперпараметров для дерева решений

Параметры	Объяснение
Вес класса (Class_weight)	Веса, связанные с классами
Критерий (Criterion)	Функция для измерения качества разбиения, коэффициента Джини и энтропии
max_depth	Максимальная глубина дерева
max_features	Число характеристик, которые следует учитывать при поиске наилучшего разбиения
max_leaf_nodes	Выращивание дерева с максимальным количеством листовых узлов по принципу «первый – лучший» (best-first)
min_samples_leaf	Минимальное количество образцов, необходимое для нахождения в листовом узле
min_samples_split	Минимальное количество образцов, необходимое для разделения внутреннего узла

В модели дерева решений существует множество гиперпараметров, самые важные из которых упомянуты в табл. 4.1. Некоторые используются для контроля переподгонки модели, а другие – для повышения ее точности.

СОЗДАНИЕ МОДЕЛИ

Следующим шагом является создание модели и проверка соответствия точности обучающей и тестовой модели.

Есть две модели, dt1 и dt2: одна имеет ограничение на максимальную глубину, а другая – нет.

Разница в том, что dt1 – это модель с переподгонкой, а dt2 – модель, в которой произошла обрезка деревьев, а переподгонки модели не произошло.

```
dt1 = tree.DecisionTreeClassifier()
dt1.fit(xtrain,ytrain)
print(dt1.score(xtrain,ytrain))
print(dt1.score(xtest,ytest))

1.0
0.8590704647676162

dt2 = tree.DecisionTreeClassifier(max_depth=3)
dt2.fit(xtrain,ytrain)
print(dt2.score(xtrain,ytrain))
print(dt2.score(xtest,ytest))

0.9021005251312828
0.8875562218890555
```

Неудобство неограниченной модели с переподгонкой заключается в том, что она создает большое дерево решений, и правил для получения прогноза будет много. Все они могут не являться важными, и по мере роста дерева некоторые избыточные правила могут принимать участие в процессе его построения.

```
!pip install pydotplus
!pip install graphviz
import pydotplus
dot_data = tree.export_graphviz(dt1, out_file=None, filled=True, rounded=True,
                                feature_names=list(xtrain.columns),
                                class_names=['yes','no'])
graph = pydotplus.graph_from_dot_data(dot_data)
display(Image(graph.create_png()))
```

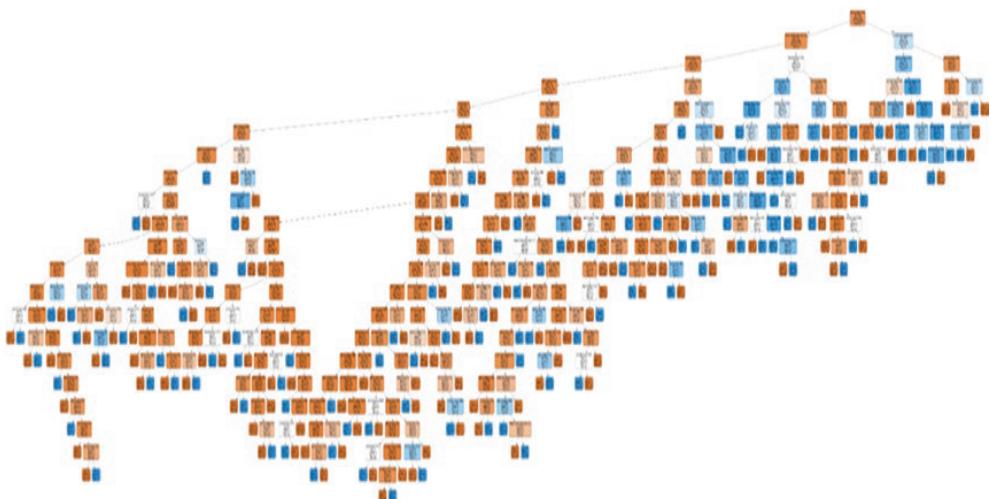


Рис. 4.4. Визуализация дерева решений с моделью по умолчанию с помощью GraphViz

Дерево решений, обученное со всеми гиперпараметрами по умолчанию, даст большое дерево, а также приведет к созданию множества правил, что не только трудно интерпретировать, но и трудно осуществить в реальном сценарии проекта. Наибольшее возможное дерево решений, как показано на рис. 4.4, является результатом установки максимальной глубины дерева `None`, что означает, что вы просите дерева решений продолжать отращивать ветви.

```
dot_data = tree.export_graphviz(dt2, out_file=None, filled=True, rounded=True,
                                feature_names=list(xtrain.columns),
                                class_names=['yes','no'])
graph = pydotplus.graph_from_dot_data(dot_data)
display(Image(graph.create_png()))
```

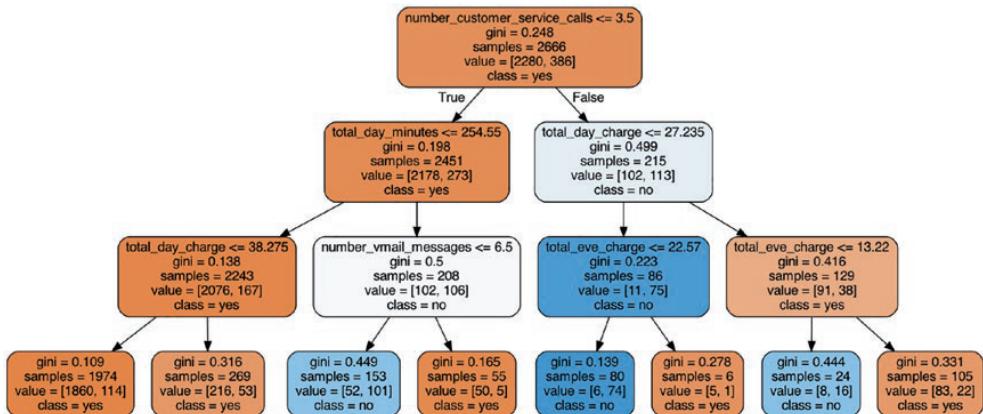


Рис. 4.5. Обрезанная версия той же модели дерева решений

```
tree.plot_tree(dt1)
tree.plot_tree(dt2)
```



Рис. 4.6. Визуализация дерева решений с помощью метода plot_tree

Та же модель, показанная на рис. 4.4, воспроизведена на рис. 4.5 и 4.6. Последний воспроизведен с использованием встроенной в библиотеку Sklearn Python функции `plot_tree`. Различий между ними нет, это просто вопрос использования той или иной библиотеки. Если есть проблема с установкой библиотек GraphViz или Pydotplus в производственной среде, вы можете переключиться на функцию `plot_tree`.

Дерево модели `dt1` настолько велико, что становится трудно ориентироваться в нем и интерпретировать результаты. После применения метода обрезки путем ограничения параметра максимальной глубины до 3 можно увидеть дерево гораздо меньшего размера, в котором отражены только соответствующие характеристики, участвующие в построении дерева (рис. 4.7).

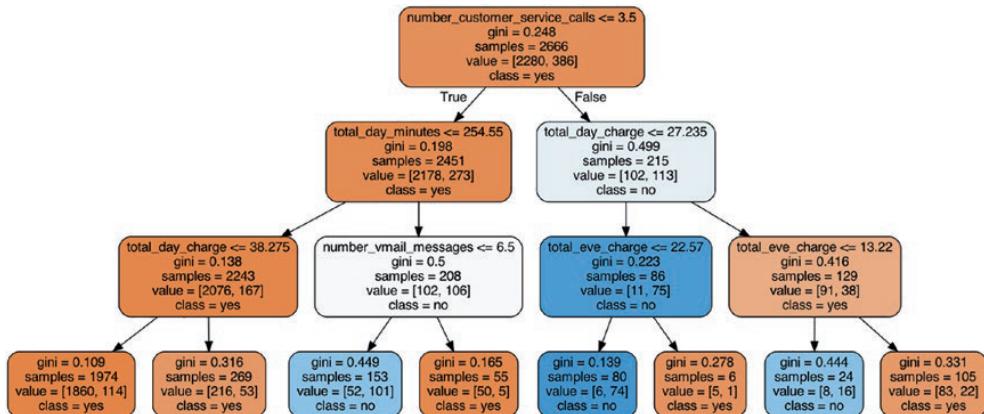


Рис. 4.7. Обрезанная версия модели дерева решений, показанного на рис. 4.6

Обрезанная версия объекта модели дерева решений `dt2` использует параметр максимальной глубины 3, что означает, что после третьего уровня ветвления дерево должно прекратить дальнейшее расширение. Эта версия модели создает более компактное дерево с жесткими правилами в виде условий «если/иначе», которые необходимо использовать. Это наиболее консервативный подход к моделированию дерева решений, свободный от чрезмерной подгонки.

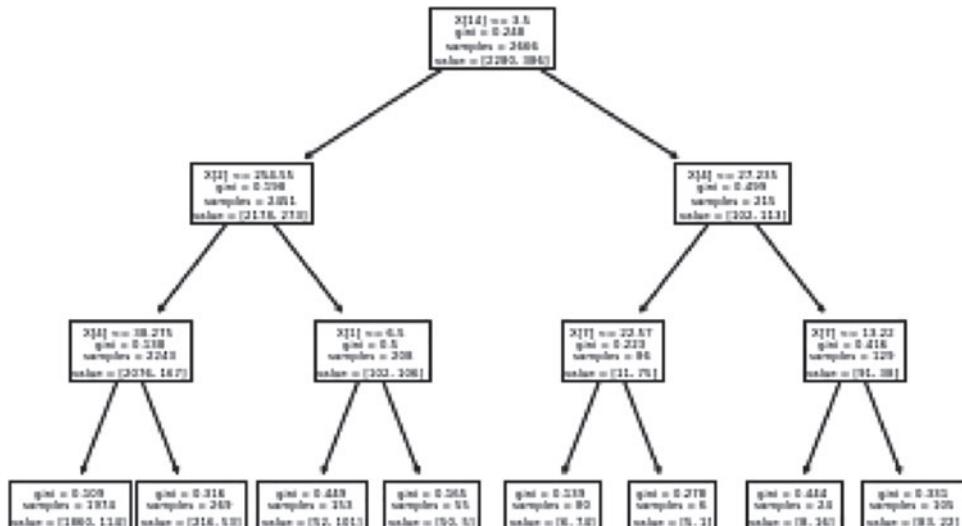


Рис. 4.8. Визуализация обрезанного дерева с помощью функции `plot_tree`

Версия дерева решений, показанная на рис. 4.8, использует меньшее количество правил, что облегчает понимание того, как принимается решение, и позволяет легче внедрить его в любую рабочую систему.

Если хотите интерпретировать, внедрить или встроить правила, созданные моделью дерева решений, в другое внешнее приложение, вы можете сделать это с помощью экспортации текстов правил.

```
from sklearn.tree import export_text
r = export_text(dt1, feature_names=list(xtrain.columns))
print(r)
```

Из модели `dt1` генерируется множество правил, которые трудно интерпретировать. Давайте рассмотрим правило, сгенерированное моделью `dt2`. Класс 0 – это сценарий отсутствия оттока, а класс 1 – индикатор возможного оттока клиентов. Текст правила показывает, что если оплата дневных вызовов (`total_day_charge`) меньше или равна 38.27, количество звонков в службу поддержки клиентов (`number_customer_service_calls`) меньше или равно 3.5, а общее количество минут дневных вызовов (`total_day_minutes`) меньше или равно 254.55, то это случай отсутствия оттока. Аналогично обратный сценарий может быть интерпретирован как сценарий оттока. Вы можете распечатать и просмотреть все правила с помощью скрипта, приведенного ниже.

```
from sklearn.tree import export_text
r = export_text(dt2, feature_names=list(xtrain.columns))
print(r)
|--- number_customer_service_calls <= 3.50
| |--- total_day_minutes <= 254.55
| | |--- total_day_charge <= 38.27
| | | |--- class: 0
| | | |--- total_day_charge > 38.27
| | | |--- class: 1
| | |--- total_day_minutes > 254.55
| | |--- number_vmail_messages <= 6.50
| | | |--- class: 1
| | | |--- number_vmail_messages > 6.50
| | | |--- class: 0
|--- number_customer_service_calls > 3.50
| |--- total_day_charge <= 27.24
| | |--- total_eve_charge <= 22.57
| | | |--- class: 1
| | | |--- total_eve_charge > 22.57
| | | |--- class: 0
| |--- total_day_charge > 27.24
| | |--- total_eve_charge <= 13.22
| | | |--- class: 1
| | | |--- total_eve_charge > 13.22
| | | |--- class: 0
```

Значимость характеристик из модели dt1 приведена ниже.

```
list(zip(dt1.feature_importances_,xtrain.columns))
[(0.04051943775304626, 'account_length'),
(0.08298083105277364, 'number_vmail_messages'),
(0.0644144400251063, 'total_day_minutes'),
(0.028172622004021135, 'total_day_calls'),
(0.20486110565087778, 'total_day_charge'),
(0.10259170929879882, 'total_eve_minutes'),
(0.03586253729017199, 'total_eve_calls'),
(0.0673761405897894, 'total_eve_charge'),
(0.0613662104733965, 'total_night_minutes'),
(0.05654698887273517, 'total_night_calls'),
(0.03894924950827072, 'total_night_charge'),
(0.01615654226052593, 'total_intl_minutes'),
(0.039418913794511345, 'total_intl_calls'),
(0.02842685405881307, 'total_intl_charge'),
(0.11203068621155501, 'number_customer_service_calls'),
(0.020325731155606916, 'area_code_tr')]
# извлечение массивов, определяющих дерево
children_left = dt2.tree_.children_left
children_right = dt2.tree_.children_right
children_default = children_right.copy() # так как sklearn не использует
                                         недостающие значения
features = dt2.tree_.feature
thresholds = dt2.tree_.threshold
values = dt2.tree_.value.reshape(dt2.tree_.value.shape[0], 2)
node_sample_weight = dt2.tree_.weighted_n_node_samples
print(" children_left", children_left)
# обратите внимание, что отрицательные дочерние значения указывают, что это
конечный узел
print(" children_right", children_right)
print(" children_default", children_default)
print(" features", features)
print(" thresholds", thresholds.round(3))
print(" values", values.round(3))
print("node_sample_weight", node_sample_weight)
```

Модель классификатора дерева решений имеет атрибут `tree_`, который позволяет вам получить подробное объяснение об объекте моделирования. В нем хранится вся бинарная структура дерева в виде параллельных массивов. Узел 0 является корневым узлом, а остальные параметры объяснены в табл. 4.2. Идентификатор левого дочернего или правого дочернего узла, когда он принимает отрицательное значение, например `-1`, означает, что это конечный узел, на котором дерево решений заканчивается.

Таблица 4.2. Низкоуровневые атрибуты для дерева решений

Параметры	Объяснение
<code>dt2.tree_node_count</code>	Общее количество узлов в дереве
<code>tree_children_left[i]</code>	Идентификатор i-го левого дочернего узла
<code>tree_children_right[i]</code>	Идентификатор i-го правого дочернего узла
<code>tree_feature[i]</code>	Характеристика, используемая для разделения i-го узла
<code>tree_threshold[i]</code>	Пороговое значение для i-го узла
<code>tree_n_node_samples[i]</code>	Количество обучающих образцов, достигающих i-го узла
<code>tree_impurity[i]</code>	Загрязнение в i-м узле
<code>tree_weighted_n_node_samples</code>	<code>n_node_samples</code> – количество фактических образцов набора данных в каждом узле. <code>weighted_n_node_samples</code> – тоже самое, взвешенное по весу класса и/или весу образца

```
# определение пользовательской модели дерева
tree_dict = {
    "children_left": children_left,
    "children_right": children_right,
    "children_default": children_default,
    "features": features,
    "thresholds": thresholds,
    "values": values,
    "node_sample_weight": node_sample_weight
}
model = {
    "trees": [tree_dict]
}

import shap
explainer = shap.TreeExplainer(model)

# Предоставление вероятности в качестве результата
def model_churn_proba(x):
    return dt2.predict_proba(x)[:,1]

# Предоставление логарифмической вероятности в качестве результата
def model_churn_log_odds(x):
    p = dt2.predict_log_proba(x)
    return p[:,1] - p[:,0]

# построение стандартного графика частичной зависимости
sample_ind = 25
fig,ax = shap.partial_dependence_plot(
    "total_day_minutes", model_churn_proba, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)
```

ДЕРЕВО РЕШЕНИЙ – SHAP

Библиотека Python SHAP может использоваться для объяснения дерева решений. Библиотека SHAP имеет полезные функции, которые дают дополнительную информацию о возможности объяснения модели.

```
import shap
explainer = shap.TreeExplainer(model)
# Предоставление вероятности в качестве результата
def model_churn_prgoba(x):
    return dt2.predict_proba(x)[:,1]

# Предоставление логарифмической вероятности в качестве результата
def model_churn_log_odds(x):
    p = dt2.predict_log_proba(x)
    return p[:,1] - p[:,0]

# построение стандартного графика частичной зависимости
sample_ind = 25
fig,ax = shap.partial_dependence_plot(
    "total_day_minutes", model_churn_prgoba, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)
```

ГРАФИК ЧАСТИЧНОЙ ЗАВИСИМОСТИ

График частичной зависимости (PDP) используется для визуализации взаимодействия между характеристикой и целевым столбцом, или столбцом ответа. Целевой столбец имеет два значения – 0 для случаев отсутствия оттока и 1 для случаев оттока. При использовании функции прогнозирования вероятности можно создать вероятность для класса 0 и класса 1.

```
pd.DataFrame (dt2.predict_proba (X)) # 0 - без оттока, 1 - отток
model_churn_prgoba (X) .max ()
model_churn_prgoba (X) .mean ()
model_churn_prgoba (X) .min ()
```

В первой строке приведенного выше скрипта содержится функция выбора столбца 1, который является вероятностью оттока. Например, первая запись показывает вероятность 0.090909, что означает, что вероятность оттока составляет менее 10 %. Вы можете сделать вывод, что это – экземпляр без оттока. В следующей строке `model_churn_prgoba` показывает только вероятность оттока для второго столбца.

Перед построением графика вероятности оттока по характеристике «число минут дневных вызовов» (total day minutes), который показан на рис. 4.10, следует взглянуть на распределение вероятности оттока, показанное на рис. 4.9. Это поможет получить представление об интерпретации графика частичной зависимости.

```
import seaborn as sns
sns.distplot(model_churn_proba(X))
pd.DataFrame(model_churn_proba(X))
from sklearn.inspection import plot_partial_dependence
xtrain.columns
plot_partial_dependence(dt2, X, ['account_length', 'number_vmail_messages',
'total_day_minutes'])
plot_partial_dependence(dt2, X, [
    'total_day_calls', 'total_day_charge', 'total_eve_minutes',
])
plot_partial_dependence(dt2, X, [
    'total_eve_calls', 'total_eve_charge', 'total_night_minutes'])
```

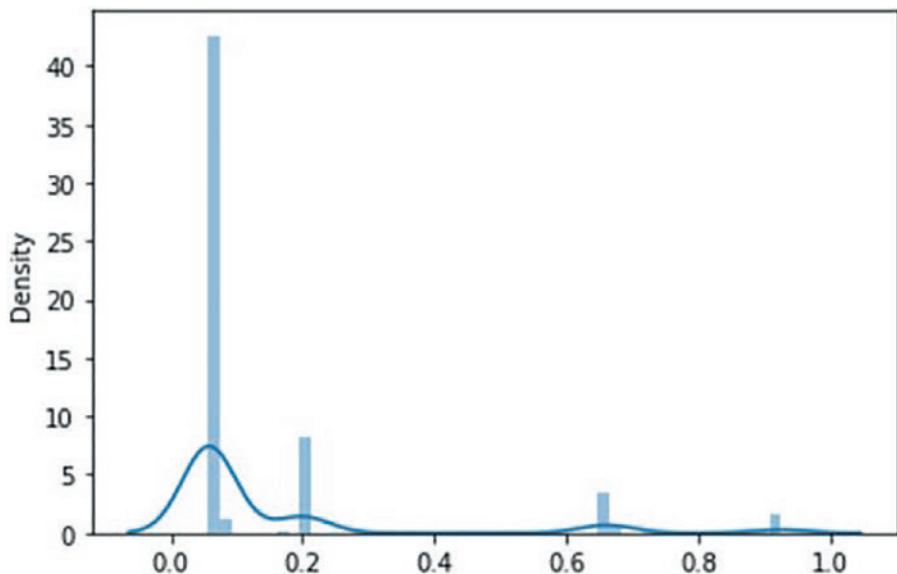


Рис. 4.9. Распределение вероятности оттока из модели дерева решений

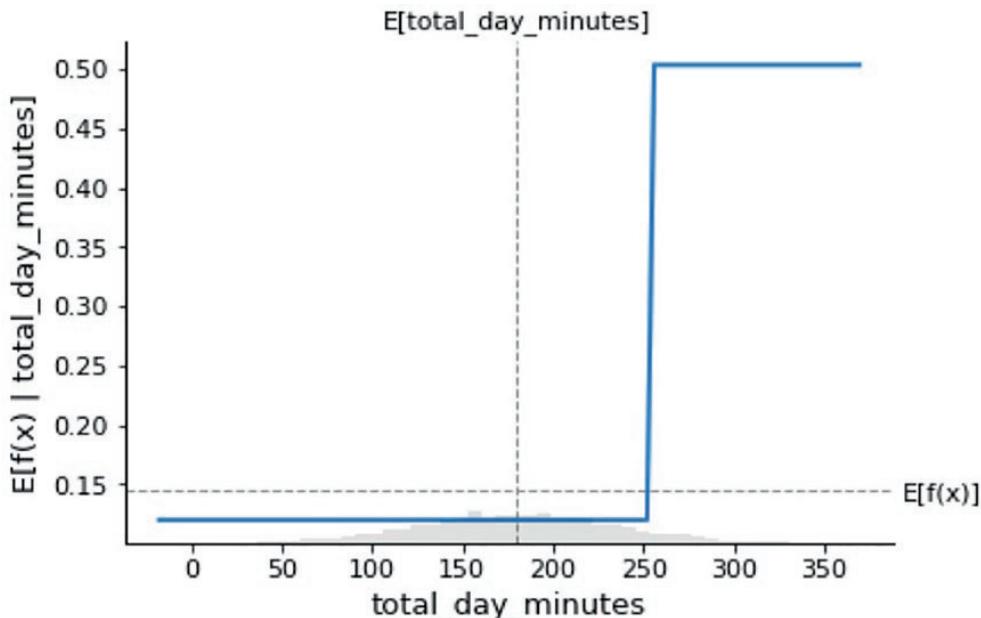


Рис. 4.10. График частичной зависимости между вероятностью оттока и числом минут дневных вызовов

Ожидаемое значение числа минут дневных вызовов и число этих минут, представленное на графике на рис. 4.10, являются кусочно-линейными, но синяя линия в целом линейной не является. Это локальная интерпретация для образца № 25. Число минут дневных вызовов имеет числовой формат и рассматривается как столбец в обучающем наборе данных. Для того же образца наблюдения № 25 число голосовых сообщений по электронной почте (voice mail message) незначительно влияет на предсказание оттока (см. рис. 4.11). На рис. 4.10 если пользователь тратит до 225 мин дневных вызовов, то вероятность оттока будет меньше или равна средней вероятности оттока (0.1425). Даже в тех случаях, когда их число больше 225 мин, вероятность оттока меньше 25 %.

```
# создание стандартного графика частичной зависимости
sample_ind = 25
fig,ax = shap.partial_dependence_plot(
    "number_vmail_messages", model_churn_proba, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)
```

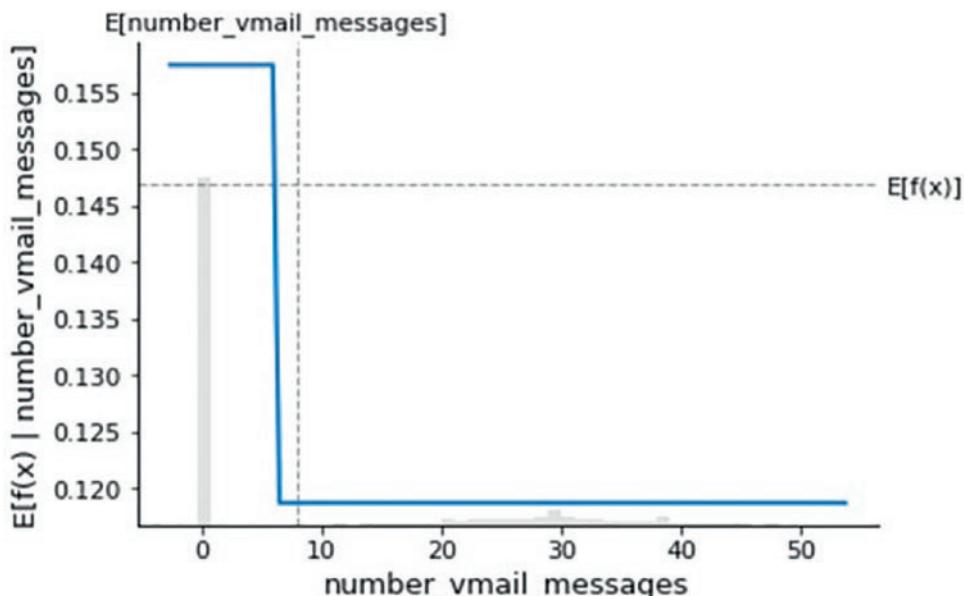


Рис. 4.11. PDP для числа голосовых сообщений по электронной почте и вероятности оттока

При рассмотрении связи между вкладом абонентского стажа (account length) в прогноз оттока и абонентским стажем можно увидеть синюю горизонтальную линию (рис. 4.12). Это означает, что, независимо от абонентского стажа, вклад остается постоянным, т. е. эта характеристика не имеет прогнозирующего значения и не играет роли в модели прогнозирования оттока.

```
# создание стандартного графика частичной зависимости
sample_ind = 25
fig,ax = shap.partial_dependence_plot(
    "account_length", model_churn_proba, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)
```

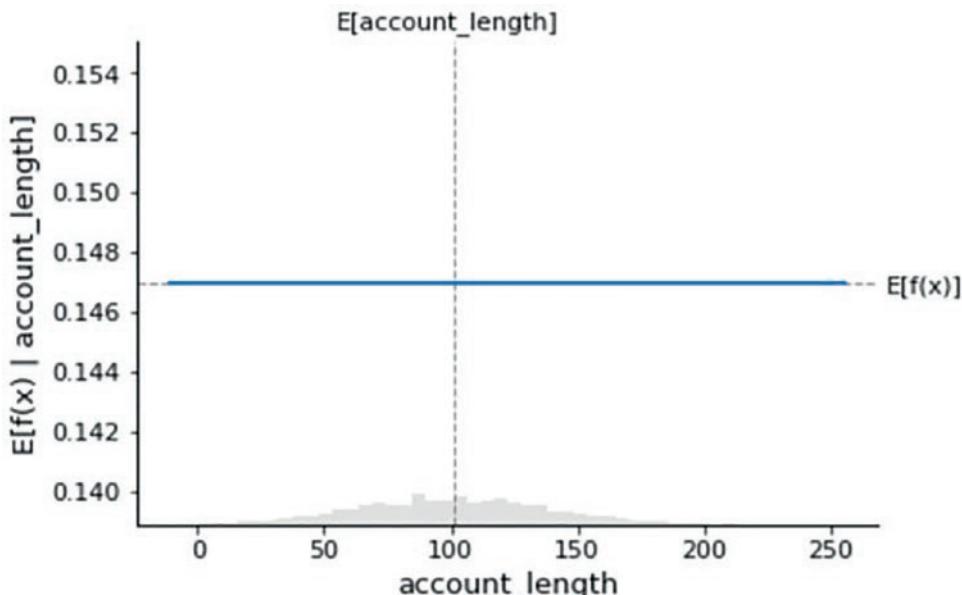


Рис. 4.12. PDP для абонентского стажа

PDP для абонентского стажа не влияет на вероятность оттока, что ясно видно по синей прямой на рис. 4.12. Синяя линия соответствует средней вероятности оттока.

```
# создание стандартного графика частичной зависимости
sample_ind = 25
fig,ax = shap.partial_dependence_plot(
    "number_customer_service_calls", model_churn_proba, X, model_expected_
    value=True,
    feature_expected_value=True, show=False, ice=False
)
```

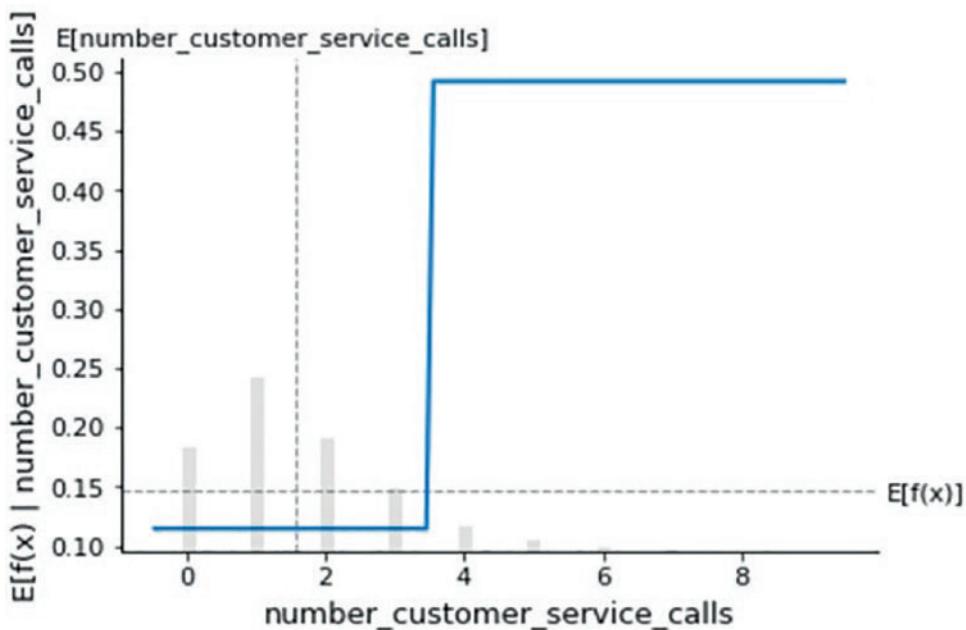


Рис. 4.13. PDP для числа обращений в службу поддержки клиентов

Важно отметить, что чем больше количество обращений в службу поддержки клиентов, тем выше вероятность оттока. Это связано с тем, что кто-то сталкивается с проблемами, поэтому количество обращений в службу поддержки клиентов увеличивается. Такую же картину можно увидеть на рис. 4.13 – более четырех вызовов увеличивают вероятность оттока.

```
# создание стандартного графика частичной зависимости
sample_ind = 25
fig,ax = shap.partial_dependence_plot(
    "total_day_charge", model_churn_proba, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)
```

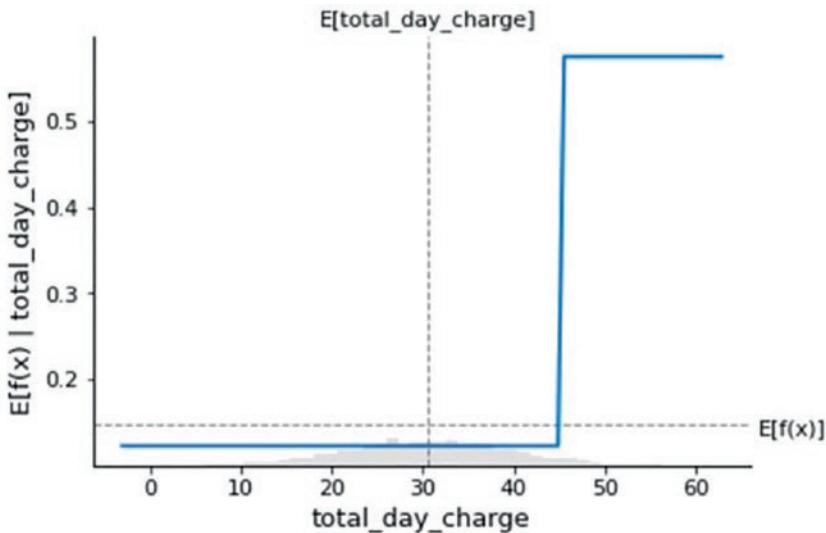


Рис. 4.14. PDP для оплаты дневных вызовов

Если оплата дневных вызовов превышает 45 долл., то вероятность оттока будет выше (рис. 4.14). Это связано с более высокой оплатой, пользователь будет вынужден выбрать альтернативных провайдеров.

```
# создание стандартного графика частичной зависимости
sample_ind = 25
fig,ax = shap.partial_dependence_plot(
    "total_eve_minutes", model_churn_proba, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)
```

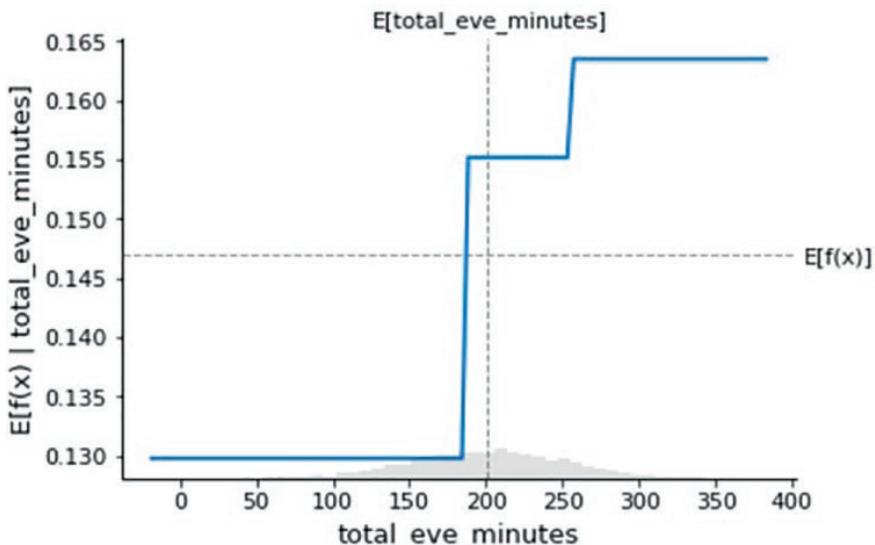


Рис. 4.15. PDP для числа минут вечерних вызовов

Число минут вечерних вызовов оказывает некоторое влияние на вероятность оттока (рис. 4-15). Вероятность оттока остается постоянной и очень низкой до 180 мин, составляя 13 %. При переходе через 180 мин вероятность оттока немного возрастает – до 15.5 %. При 250 мин и более она возрастает до 16.5 %, т. е. является относительно низкой.

```
# создание стандартного графика частичной зависимости
sample_ind = 25
fig,ax = shap.partial_dependence_plot(
    "total_eve_calls", model_churn_proba, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)
```

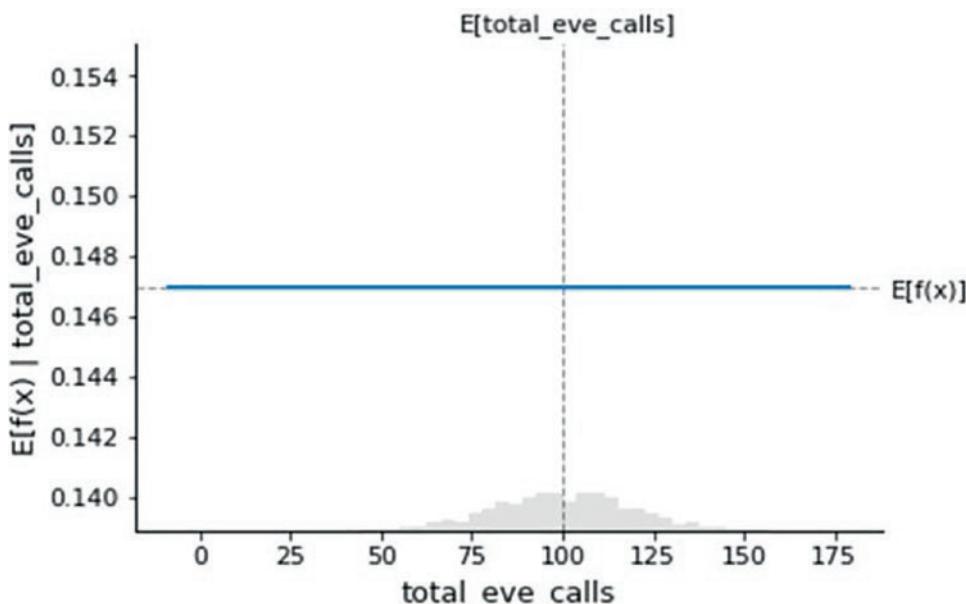


Рис. 4.16. PDP для числа вечерних вызовов

Число вечерних вызовов не влияет на вероятность оттока, так как она остается равной среднему значению вероятности независимо от увеличения числа вечерних вызовов (рис. 4.16). В табл. 4.3 показана оценка важности каждой характеристики.

```
shap_values_churn.feature_names
temp_df = pd.DataFrame()
temp_df['Feature Name'] = pd.Series(X.columns)
temp_df[ 'Score' ] = pd.Series(dt2.feature_importances_.flatten())
temp_df.sort_values(by='Score', ascending=False)
```

Таблица 4.3. Оценка важности характеристик

Имя характеристики	Оценка
4	Оплата дневных вызовов
14	Число обращений в службу поддержки клиентов
2	Число минут вызовов за день
1	Число голосовых сообщений по электронной почте
5	Число минут вечерних вызовов
0	Абонентский стаж
3	Число вызовов за день
6	Число вечерних вызовов

Окончание табл. 4.3

Имя характеристики	Оценка
7	Оплата вечерних вызовов
8	Число минут ночных вызовов
9	Число ночных вызовов
10	Оплата ночных вызовов
11	Число минут международных вызовов
12	Число международных вызовов
13	Оплата международных вызовов
15	Код региона

На приведенных выше рисунках показаны графики частичной зависимости, построенные на основе библиотеки SHAP. То же самое можно создать с помощью модуля обследования библиотеки Scikit-learn.

PDP с использованием Scikit-Learn

В наборе модулей библиотеки Scikit-learn есть новый модуль, который поможет создать графики частичной зависимости. Это модуль обследования (см. рис. 4.17–4.22).

```
from sklearn.inspection import plot_partial_dependence
xtrain.columns
plot_partial_dependence(dt2, X, ['account_length', 'number_vmail_messages',
'total_day_minutes'])
])
```

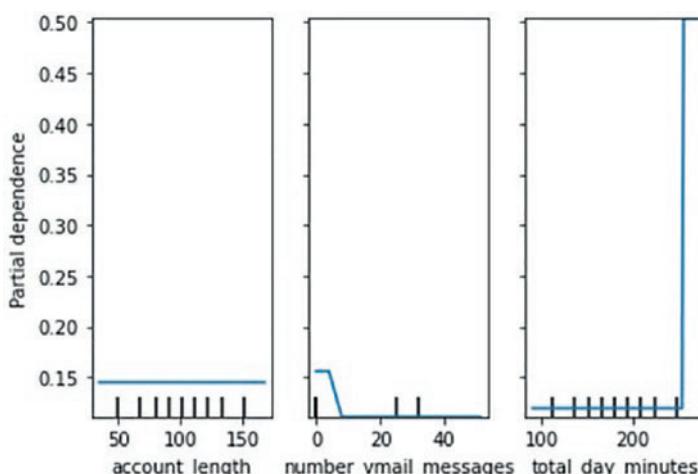


Рис. 4.17. PDP для трех переменных вместе

```
plot_partial_dependence(dt2, X, [
    'total_day_calls', 'total_day_charge', 'total_eve_minutes',
])
```

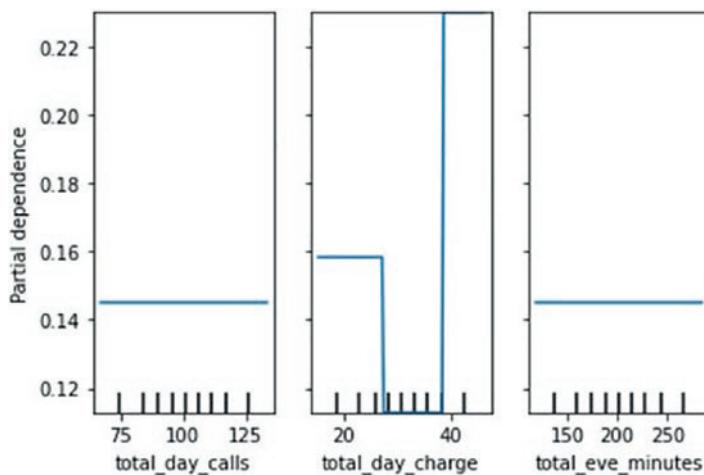


Рис. 4.18. PDP для следующего набора из трех характеристик

```
plot_partial_dependence(dt2, X, [
    'total_eve_calls', 'total_eve_charge', 'total_night_minutes'
])
```

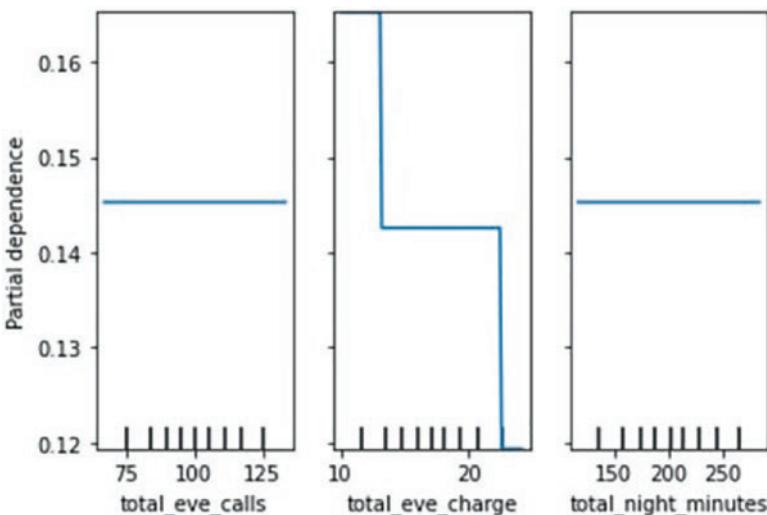


Рис. 4.19. PDP для еще трех переменных

```
plot_partial_dependence(dt2, X, [
    'total_night_calls', 'total_night_charge', 'total_intl_minutes'])
```

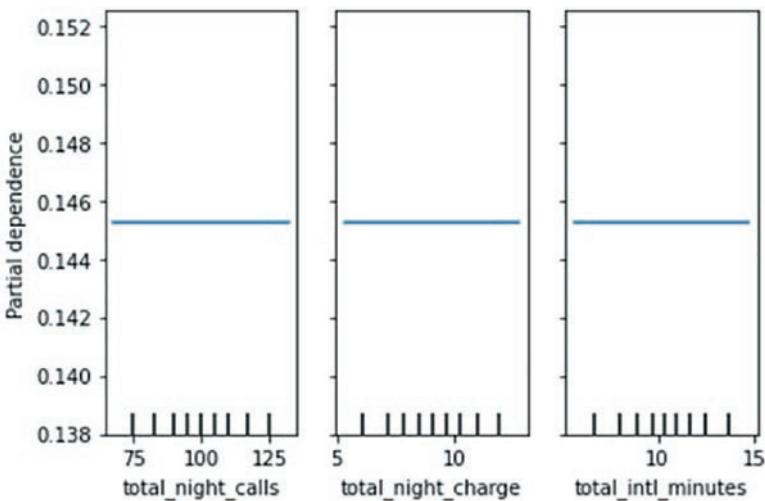


Рис. 4.20. PDP для трех характеристик, которые не влияют на вероятность оттока

```
plot_partial_dependence(dt2, X, [  
    'total_intl_calls', 'total_intl_charge'])
```

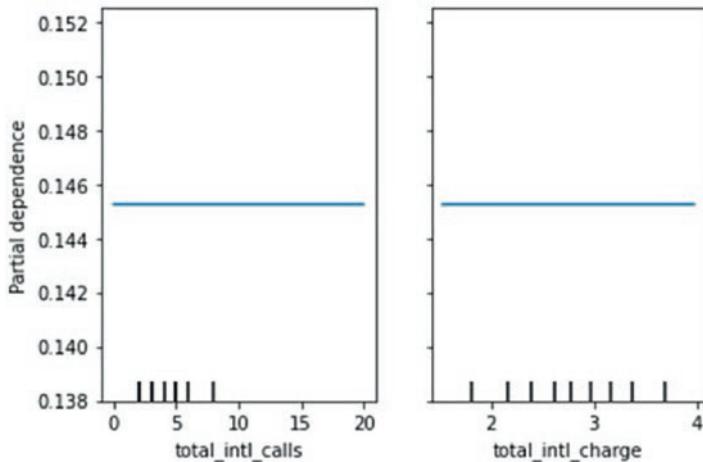


Рис. 4.21. PDP для числа международных вызовов и их оплат

```
plot_partial_dependence(dt2,X, [  
    'number_customer_service_calls', 'area_code_tr'])
```

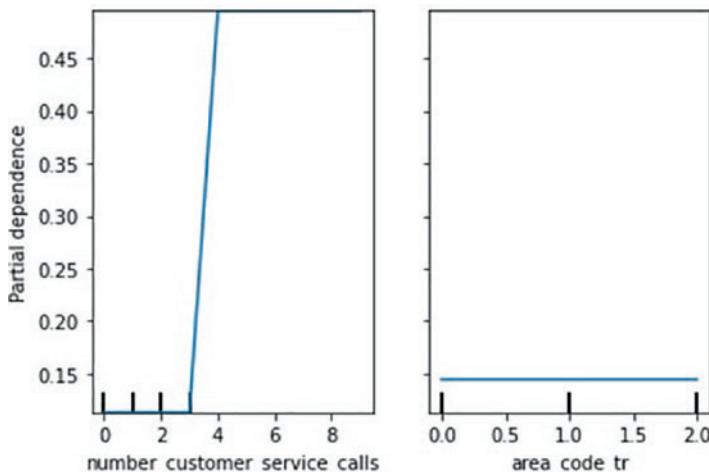


Рис. 4.22. PDP для числа обращений в службу поддержки клиентов и кода региона

Рисунки 4.17–4.22 генерируются посредством функции, которая является частью библиотеки scikit-learn, дающей объяснение, аналогично объяснению библиотеки SHAP. Аналогичным образом могут быть получены и интерпретации.

Объяснение нелинейной модели – LIME

Наиболее важными характеристиками, которые играют роль в процессе прогнозирования оттока, являются оплата дневных вызовов, число обращений в службу поддержки клиентов, число минут дневных вызовов, число голосовых сообщений по электронной почте и число минут вечерних вызовов. Другие характеристики не играют никакой роли, поэтому графики частичной зависимости для несвязанных характеристик являются горизонтальными линиями.

Можно также использовать некоторые функции из библиотеки LIME Python для объяснения решений, принятых моделью дерева решений.

```
import lime
import lime.lime_tabular

explainer = lime.lime_tabular.LimeTabularExplainer(np.array(xtrain),
                                                    feature_names=list(xtrain.columns),
                                                    class_names=['target_churn_dum'],
                                                    verbose=True, mode='classification')

# эта запись является сценарием отсутствия оттока
exp = explainer.explain_instance(xtest.iloc[0], dt2.predict_proba,
                                   num_features=16)

exp.as_list()
pd.DataFrame(exp.as_list())
exp.show_in_notebook(show_table=True)
```

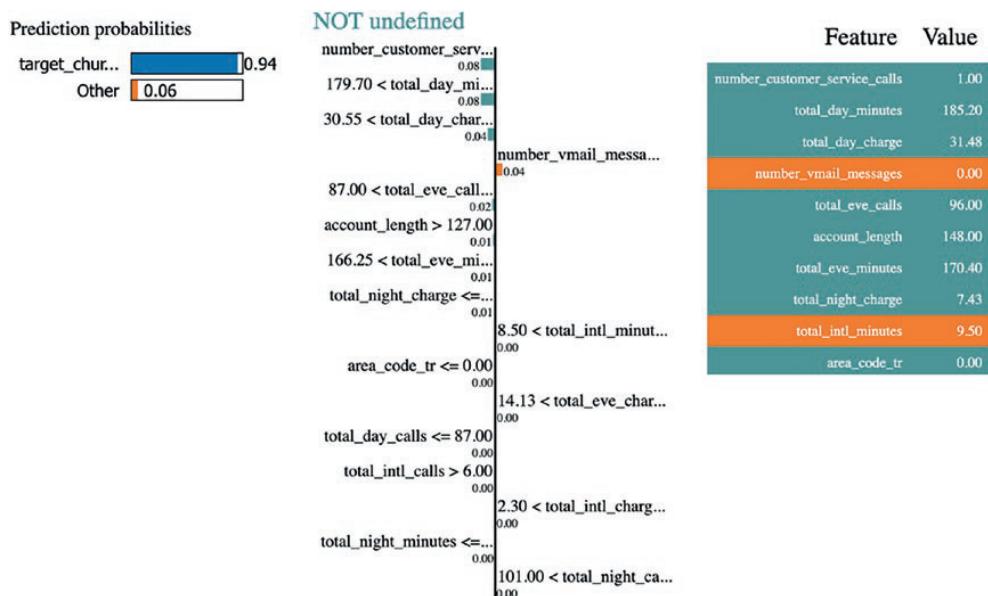


Рис. 4.23. Важность характеристики, положительный и отрицательный вклад в вероятность прогноза для записи 0

На рис. 4.23 показано объяснение модели локальной интерпретации для первой записи из тестового набора `xtest` [0]. На средней диаграмме на рис. 4.23 синий цвет отображает вклад в вероятность прогнозирования целевого оттока, а оранжевый – вклад в другой класс, который является сценарием отсутствия оттока. Дерево решений создает характеристики в виде диапазона значений, таких как число минут дневных вызовов между 179.90 и 216.20, поскольку эти значения изменяют вероятность оттока на 0.08 (8 %). Если удалить две характеристики из модели – число обращений в службу поддержки клиентов ≤ 1.00 и число минут дневных вызовов > 179.90 и ≤ 216.20 , – то целевая вероятность предсказания оттока снизится на 0.16 (16 %), т. е. $(0.94 - 0.08 - 0.08)$, что составляет 0.78 (78 %). С другой стороны, если удалить число голосовых сообщений по электронной почте ≤ 0.00 , то целевая вероятность оттока возрастает на 4 % (0.04). Третья таблица на рис. 4.23 показывает значение вклада каждой характеристики в прогноз. Подобный вид анализа и интерпретации может быть выполнен еще для нескольких записей, как показано на рис. 4.24–4.26.

```
# Это сценарий оттока
exp = explainer.explain_instance(xtest.iloc[20], dt2.predict_proba,
num_features=16)
exp.as_list()
exp.show_in_notebook(show_table=True)
```

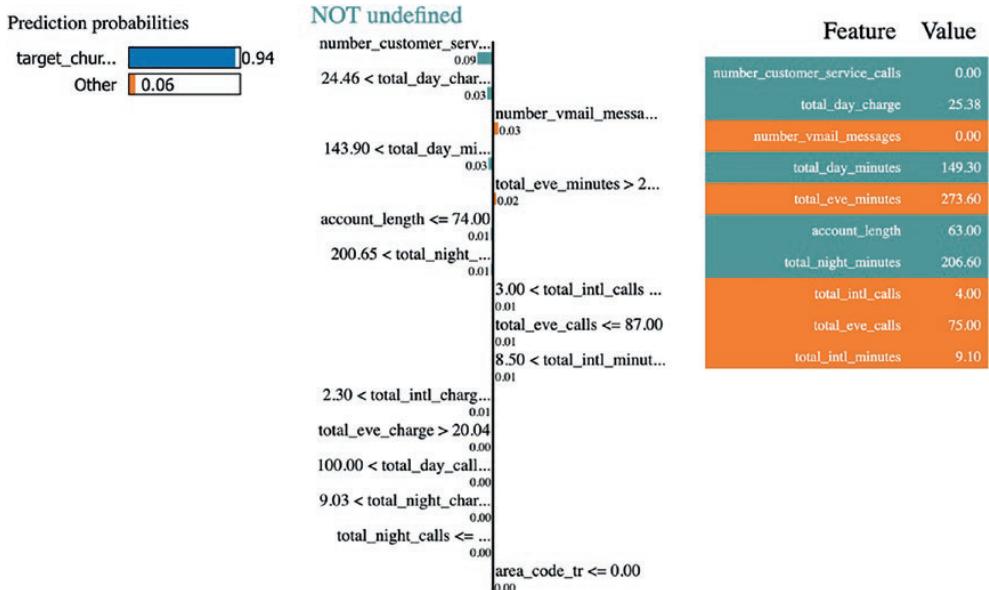


Рис. 4.24. Важность характеристик, положительный и отрицательный вклад в вероятность прогноза для записи № 20

```

xtest.iloc[20]
ytest.iloc[20]
dt2.predict(xtest)[20]
explainer = lime.lime_tabular.LimeTabularExplainer(np.array(xtrain),
    feature_names=list(xtrain.columns),
    class_names=['target_churn_dum'],
    verbose=True, mode='classification')

# Код для SP-LIME импортирует предупреждения из субмодульного выбора LIME

# SP-LIME возвращает объяснения для выборочного набора, чтобы обеспечить
неизбыточную
# глобальную границу принятия решений исходной модели
sp_obj = submodular_pick.SubmodularPick(explainer, np.array(xtrain),
    dt2.predict_proba,
    num_features=14,
    num_exps_desired=10)

```

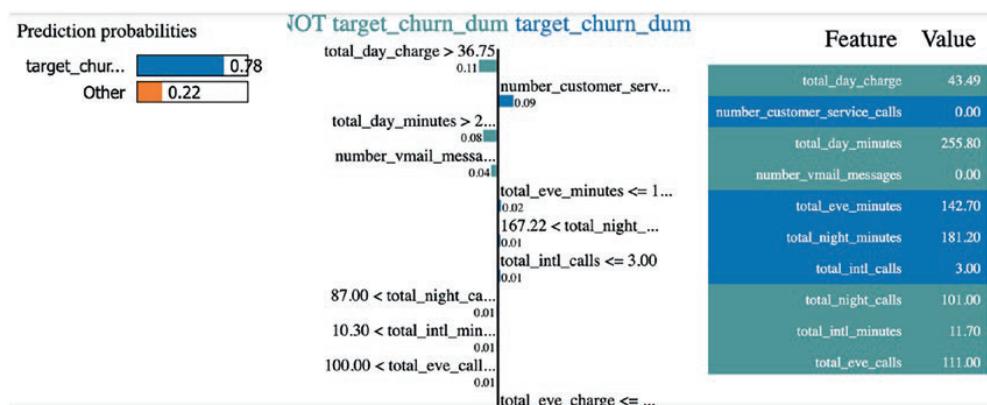


Рис. 4.25. Важность характеристик, положительный и отрицательный вклад в вероятность прогноза для всех записей

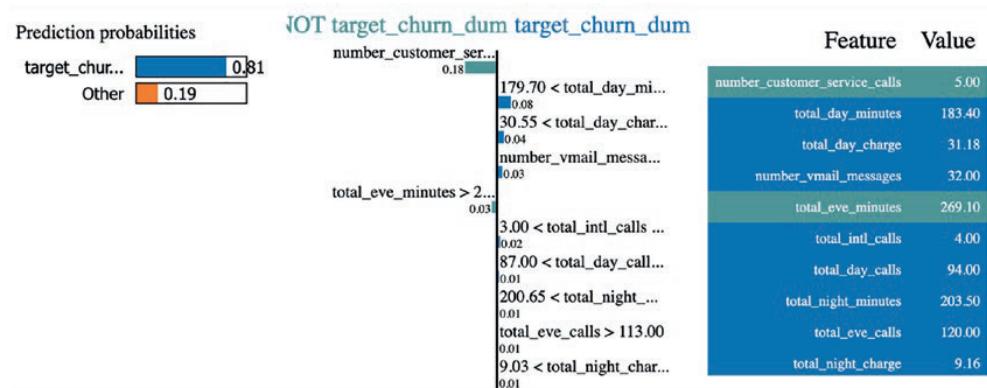


Рис. 4.26. Важность характеристик, положительный и отрицательный вклад в вероятность прогноза всех записей

Нелинейное объяснение – Skope-Rules

Существует другая библиотека объяснений с именем Skope-rules, которая может использоваться для создания правил из обучающей модели, а правила – для прогнозирования любого нового набора данных.

Следующий код можно использовать для установки библиотеки на основе Python. Параметры приведены в табл. 4.4.

```
!pip install skope-rules
import six
import sys
sys.modules['sklearn.externals.six'] = six
import skrules
```

```

from skrules import SkopeRules
clf = SkopeRules(max_depth_duplication=2,
                  n_estimators=30,
                  precision_min=0.3,
                  recall_min=0.1,
                  feature_names=list(xtrain.columns))

clf
clf.fit(xtrain,ytrain)
print('The 5 most precise rules are the following:')
for rule in clf.rules_[:5]:
    print(rule[0])

```

Таблица 4.4. Объяснение параметров Skope-Rules

Параметр	Объяснение
feature_names	Имя каждой функции, используемой для возврата правил в формате строк
precision_min	Минимальная точность выбираемого правила
recall_min	Минимальный отзыв выбираемого правила
n_estimators	Количество базовых оценщиков (правил), используемых для прогнозирования
max_depth_duplication	Максимальная глубина дерева решений для дедупликации правил
max_depth	Максимальная глубина деревьев решений
max_samples	Количество образцов для извлечения из X для обучения каждого дерева решений, из которого генерируются и выбираются правила

С помощью метода `clf.fit()` можно обучить модель дерева решений и составить правила, которые могут быть отфильтрованы или обработаны с использованием параметров точности (`precision`) и отзыва (`recall`). Пороговое значение, если оно уменьшено до более низкого уровня, означает, что будет сгенерировано множество правил, которые не используются пользователем, поскольку в бизнес-приложение будут введены ложные положительные правила. Более высокий порог уменьшит количество правил, и в бизнес-приложениях будут использоваться только наиболее значимые из них.

Ниже приведены пять наиболее точных правил:

```

number_vmail_messages <= 5.5 and number_customer_service_calls <= 3.5 and
total_day_minutes > 249.70000457763672

number_vmail_messages <= 6.5 and total_day_charge > 44.989999771118164 and
total_eve_minutes > 145.39999389648438

number_customer_service_calls > 3.5 and total_day_minutes <= 263.5500030517578 and
total_day_charge <= 29.514999389648438

```

```
number_customer_service_calls <= 3.5 and total_day_minutes > 245.0999984741211 and  
total_eve_minutes > 204.20000457763672  
  
number_customer_service_calls > 3.5 and total_day_charge <=  
30.65999984741211  
  
clf.predict(xtest)  
clf.predict_top_rules(xtest,5)  
clf.score_top_rules(xtest)
```

Функция `predict` использует все правила для прогнозирования целевого класса оттока: 0 не является оттоком, а 1 является.

Функция `predict_top_rules` используется для использования пяти правил для прогнозирования результата. Пять – это количество правил, используемых для прогнозирования. Если активировано одно из пяти наиболее эффективных правил, прогноз равен 1. Для каждого наблюдения функция указывает, следует ли (1 или 0) рассматривать его как отклонение от выбранных правил.

```
clf.decision_function(xtrain)  
clf.decision_function(xtest)
```

Функция `score_top_rules` представляет упорядочение базовых классификаторов (правил). Ее значение велико, когда экземпляр обнаружен выполняющимся правилом. Положительные значения представляют отклонения, а нулевые оценки представляют соответствие.

`decision_function` представляет собой оценку аномалии входной выборки и вычисляется как взвешенная сумма выходов двоичных правил, причем вес является соответствующей точностью каждого правила. Для оценки аномалии входных выборок чем она выше, тем ненормальнее. Положительные оценки представляют отклонения, а нулевые оценки представляют соответствие.

ЗАКЛЮЧЕНИЕ

В этой главе вы научились интерпретировать нелинейные модели, в частности модели дерева решений вместо логистической регрессии для бинарной классификации. Аналогичным образом модель дерева решений может использоваться для регрессионной модели, а также может быть расширена до полиномиальной классификации. Нелинейные модели проще интерпретировать, и все понимают, как эти модели работают, используя простые правила «если/тогда иначе» (if/then else). Поэтому существует высокая степень доверия к деревовидным нелинейным моделям. В этой главе рассмотрены различные аспекты создания представлений нелинейных моделей с использованием библиотек объяснимого ИИ, таких как LIME, SHAP и Skope-rules. В следующей главе вы узнаете об объяснимости ансамблевых моделей.

ГЛАВА 5

Объяснимость для ансамблевых моделей

Ансамблевые модели – это группа моделей, где прогнозы объединяются с использованием некоторой метрики для генерации окончательного прогноза. Например, группа древовидных моделей может быть разработана для прогнозирования реального результата как регрессионная модель. Прогнозы от всех деревьев усредняются и принимаются за конечный результат. Аналогичным образом классификационные модели генерируют индивидуальные прогнозы классов, а затем применяют правило голосования. В качестве конечного результата принимается класс с наибольшим числом прогнозов. Ансамблевые модели не только трудно интерпретировать, но и трудно объяснить конечному пользователю. Рассмотрим сценарий, в котором четыре дерева предсказывают «да», а шесть деревьев – «нет». Мы будем рассматривать «нет» в качестве окончательного результата на основе правила большинства 6/4. Здесь трудно объяснить конечному пользователю, почему некоторые модели предсказали «да». Отсюда очень важно объяснить модели ансамбля. В этой главе для объяснения прогнозов модели ансамбля будет использован преимущественно SHAP.

Ансамблевые модели

Ансамблевые модели – это наиболее сложный набор моделей, которые нуждаются в подробном объяснении, поскольку выходные данные являются совокупным результатом множественных прогнозов. Ансамбль просто подразумевает группировку. Что важно в случае ансамблевых моделей – так это как объяснить прогнозы, какой вариант модели на самом деле произвел прогноз и как прочитать индивидуальный вклад характеристик в процесс окончательного прогноза.

Преимущество модели дерева решений заключается в том, что она учитывает потенциальную нелинейность, существующую в наборе данных. Переменные взаимодействия вступают в игру при генерации прогнозов модели. Однако ограничением этой модели является то, что она склонна к смещению, поскольку мощные или более сильные характеристики принимают участие в процессе построения дерева, а слабые не способны войти в процесс ветвления дерева, поскольку у них отсутствует прогнозирующая сила. Следовательно, модель становится смещенной к нескольким избирательным и более сильным характеристикам из набора данных. Это также иногда приводит к перепод-

гонке модели. Чтобы сбалансировать влияние сильных характеристик, важно регулировать их ввод в древовидные модели. Если исключить сильные характеристики из шага создания модели и включить в создание дерева только слабые, вы по-прежнему сможете генерировать прогнозы, но это снова будет смещенная модель. Следовательно, обработка смещения модели и в то же время управление переподгонкой могут быть выполнены только при использовании комбинации сильных и слабых характеристик в процессе построения дерева. Комбинация может быть выполнена исключительно на основе метода начальной загрузки, и количество деревьев может быть увеличено в достаточной степени для усреднения прогнозов.

Типы ансамблевых моделей

Ансамблевые модели бывают трех типов: модели упаковки (*bagging*, также известные как модели агрегирования начальной загрузки), модели повышения (*boosting*) и модели укладки (*stacking*) (рис. 5.1). Модели укладки могут быть двух видов: групповой укладки (*same group stacking*) и компоновки различных вариантов (*different variant model stacking*). Групповая укладка включает в себя однородные типы моделей, такие как только включение древовидных моделей и сопоставление каждого результата модели с другими моделями. Гетерогенная модель укладки означает наложение древовидных и недревовидных моделей друг на друга и объединение их прогнозов. Объяснимость моделей укладки совсем не трудна, поскольку мы можем идентифицировать конкретную модель и объяснить прогнозы и параметры. С моделями упаковки и повышения немного сложнее. Модель случайного леса является примером модели упаковки, где выращивается много деревьев, и их прогнозы объединяются для достижения конечного результата. Модели повышения берут базовую модель, обучаются на ее результатах и пытаются улучшить модель итеративным способом.



Рис. 5.1. Три типа ансамблевых моделей

ПОЧЕМУ АНСАМБЛЕВЫЕ МОДЕЛИ?

Когда общее число характеристик в наборе данных увеличивается, скажем до более чем 50, его нельзя разместить в одном дереве решений по следующим причинам:

- мощные характеристики займут шаг создания ветви в построении дерева решений, следовательно, слабые характеристики не будут принимать участие в формировании дерева;
- модели в сценарии дерева решений становятся более предвзятыми, поскольку прогнозом управляет несколько мощных характеристик;
- выбор гиперпараметров ограничен несколькими мощными характеристиками;

- эта проблема становится еще больше, когда вы думаете, что из 100 с лишним характеристик 20 % мощных из них управляют прогнозом. Остальные 80 % слабых характеристик не используются;
- из-за одного дерева не все характеристики получают справедливый шанс в прогнозе;
- компенсация смещения в сценарии дерева решений в основном выполняется путем обрезки.

Чтобы устраниТЬ вышеуказанные ограничения модели одного дерева решений, необходимо иметь группу деревьев, отсюда и название – ансамбли деревьев. Модели упаковки, повышения и укладки могут применяться как к проблемам регрессии, так и к проблемам классификации. На рис. 5.2 представлено графическое представление, позволяющее различать три типа ансамблевых моделей.

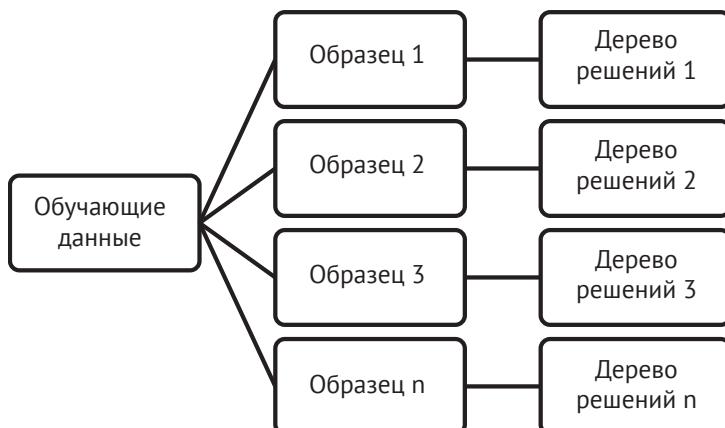


Рис. 5.2. Модель упаковки

Упаковка (также известная как агрегирование начальной загрузки) берет различные образцы характеристик и записей из обучающего набора данных и пытается обучить дерево решений. На рис. 5.2 можно создать n деревьев, где n может быть 100, 500 или 1000. По мере увеличения числа деревьев точность модели повышается, когда в обучающем наборе данных появляется больше характеристик. Если у вас меньше характеристик и образцов и если увеличить количество деревьев, то вы сможете получить больше точности, но при этом создадите много дубликатов деревьев. Они являются дубликатами, поскольку в каждом дереве появляется один и тот же набор характеристик. Упаковка будет параллельной по своей природе, так как вы можете обучать несколько деревьев параллельно как для классификации, так и для регрессионных задач. Поэтому в соответствии с отраслевой практикой рекомендуется использовать ансамбль, если в обучающем наборе данных имеется более 50 характеристик и более 50 000 записей.

Повышение, с другой стороны, является последовательным процессом, в котором базовый классификатор обучается классификации или прогнозированию целевого столбца. Затем правильно спрогнозированные случаи от-

деляются в сценарии классификации, и распределение ошибок выбирается в сценарии на основе регрессии, чтобы переобучить модель в той же конфигурации или настройке. Это повторяют несколько раз до тех пор, пока не исчезнет возможность дальнейшего повышения точности. Данный процесс имеет последовательный характер, поскольку вторая модель работает на результатах первой модели. Это мощная техника получения ансамблевых моделей. Процесс описан на рис. 5.3.



Рис. 5.3. Процесс обучения модели повышения

Третий вариант ансамблевой модели – укладка. Существует два способа суммирования прогнозов модели – однородная групповая и гетерогенная групповая укладка (см. табл. 5.1).

Таблица 5.1. Однородная групповая укладка в задаче регрессии

Тестовые данные	Модели						
	DT	RF	Адаптивное повышение	Повышение градиента	XGBoost	Прогноз	
1	25.3	25.2	24.3	26.1	24.5	25.08	
2	12.5	13.21	14.1	11.9	12.2	12.782	
3	17.8	15.3	16.3	16.7	17.1	16.64	

Последний столбец «Прогноз» представляет собой среднее арифметическое значений, прогнозируемых различными моделями. Мы рассмотрели только три записи из набора тестовых данных. Теперь рассмотрим табл. 5.2.

Таблица 5.2. Однородная групповая укладка в задаче классификации

Модели						
Данные	DT	RF	Адаптивное повышение	Повышение градиента	XGBoost	Прогноз
1	Да	Нет	Да	Да	Да	Да
2	Нет	Нет	Нет	Да	Нет	Нет
3	Да	Да	Нет	Нет	Нет	Нет

В ее первой строке есть четыре ответа «да» и один «нет», следовательно, окончательный прогноз – «да». Аналогичным образом во втором и третьем рядах окончательный прогноз определяется большинством голосов. В гетерогенном сценарии укладки могут принимать участие не только древовидные модели, но и другие, такие как модели логистической регрессии, опорных векторов (support vector machine – SVM) и т. д. В табл. 5.3 показана регрессионная модель гетерогенной групповой укладки, а в табл. 5.4 – модель классификации с гетерогенной групповой укладкой.

Таблица 5.3. Гетерогенная групповая укладка в задаче регрессии

Модели									
Данные	DT	RF	Адаптивное повышение	Повышение градиента	XGBoost	SVM	LR	Прогноз	
1	25.3	25.2	24.3	26.1	24.5	24.9	24.6	24.98571429	
2	12.5	13.21	14.1	11.9	12.2	12.6	11.9	12.63	
3	17.8	15.3	16.3	16.7	17.1	16.8	17.1	16.72857143	

Таблица 5.4. Гетерогенная групповая укладка в задаче классификации

Модели									
Данные	DT	RF	Адаптивное повышение	Повышение градиента	XGBoost	SVM	LR	Прогноз	
1	Да	Нет	Да	Да	Да	Нет	Нет	Да	
2	Нет	Нет	Нет	Да	Нет	Да	Да	Нет	
3	Да	Да	Нет	Нет	Нет	Нет	Да	Нет	

ИСПОЛЬЗОВАНИЕ SHAP ДЛЯ АНСАМБЛЕВЫХ МОДЕЛЕЙ

При этом учитываются два различных набора данных. Вы будете использовать популярный набор данных цен на жилье в Бостоне, чтобы объяснить прогнозы модели в сценарии регрессионного использования, а набор данных о взрослых – для объяснения сценария классификации. Ниже приведены переменные из набора данных цен на жилье в Бостоне:

- CRIM – уровень преступности на душу населения в разбивке по городам;
- ZN – доля жилой земли, зонированной под участки площадью более 25 000 фут²;
- INDUS – доля акров, не связанных с розничной торговлей, на город;
- CHAS – фиктивная переменная Charles River (1, если тракт граничит с рекой, 0 в противном случае);
- NOX – концентрация оксида азота (частей на 10 млн);
- RM – среднее количество комнат на одно жилище;
- AGE – доля единиц, занятых владельцами, построенных до 1940 года;
- DIS – взвешенные расстояния до пяти бостонских центров занятости;
- RAD – индекс доступности радиальных магистралей;
- TAX – ставка налога на имущество полной стоимости за 10 000 долл.;
- PTRATIO – соотношение учеников и учителей по городам;
- B – 1000 (Bk – 0.63)², где Bk – доля чернокожих по городам;
- LSTAT – на % ниже статус населения;
- MEDV – медианная стоимость занятых владельцами домов в тысячах долларов.

```
import pandas as pd
import shap
import sklearn

# Прогноз цен на жилье в Бостоне
X,y = shap.datasets.boston()
X100 = shap.utils.sample(X, 1000) # 1000 экземпляров для использования
в качестве фонового

# распределения
# простая линейная модель
model = sklearn.linear_model.LinearRegression()
model.fit(X, y)
```

Набор данных о ценах на жилье в Бостоне теперь является частью библиотеки SHAP. Расчет базовой модели выполняется с использованием модели линейной регрессии, чтобы можно было выполнить ансамблевую модель для этого набора данных и сравнить результаты.

```
print("Model coefficients:\n")
for i in range(X.shape[1]):
    print(X.columns[i], "=", model.coef_[i].round(4))
```

Таблица 5.5. Коэффициенты для модели линейной регрессии**Коэффициенты модели:**

CRIM = -0.108

ZN = 0.0464

INDUS = 0.0206

CHAS = 2.6867

NOX = -17.7666

RM = 3.8099

AGE = 0.0007

DIS = -1.4756

RAD = 0.306

TAX = -0.0123

PTRATIO = -0.9527

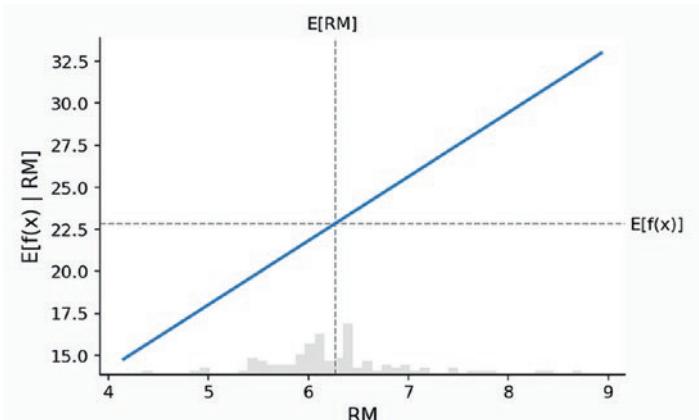
B = 0.0093

LSTAT = -0.5248

Коэффициенты базовой модели являются начальной точкой (табл. 5.5). Впоследствии вы рассмотрите сложные ансамблевые модели и сопоставите коэффициенты с базовой линейной моделью. Кроме того, можно сравнить объяснения. Чем лучше прогноз, тем лучше объяснимость.

```
shap.plots.partial_dependence(
    "RM", model.predict, X100, ice=False,
    model_expected_value=True, feature_expected_value=True
)
```

Горизонтальная пунктирная линия $E[f(x)]$ на рис. 5.4 представляет собой не что иное, как прогнозируемое медианное значение средней цены на жилье.

**Рис. 5.4.** PDP для RM и прогнозируемой цены жилья с использованием образца поднабора X100

Если взять среднее значение результата прогноза, оно составит 22.84. Существует линейная зависимость между характеристикой RM и прогнозируемым результатом модели, как показано на рис. 5.4.

```
# вычисление значений SHAP для линейной модели
explainer = shap.Explainer(model.predict, X100)
shap_values = explainer(X)

# создание стандартного графика частичной зависимости
sample_ind = 18
shap.partial_dependence_plot(
    "RM", model.predict, X100, model_expected_value=True,
    feature_expected_value=True, ice=False,
    shap_values=shap_values[sample_ind:sample_ind+1,:]
)
```

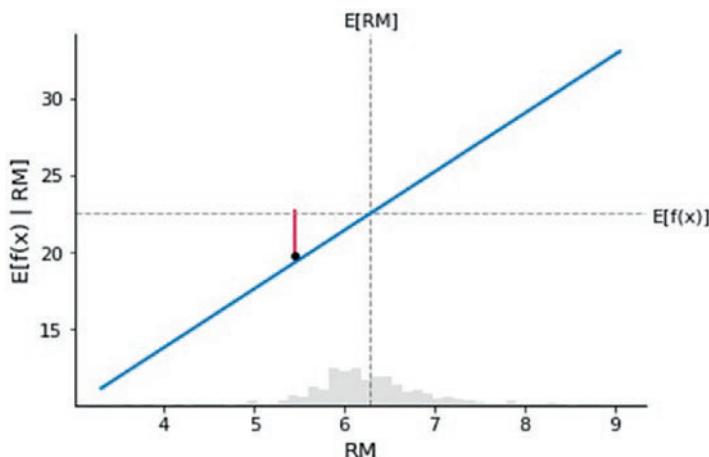


Рис. 5.5. Страна 18 из набора данных наложена на PDP

На рис. 5.5 мы видим, что вклад характеристики RM в прогнозируемое значение целевого столбца является линейной зависимостью, как показано синей прямой линией. Красная линия отображает разницу между прогнозируемым и средним прогнозируемым значением. Значение SHAP равно прогнозируемому.

```
X100 = shap.utils.sample(X,100)
model.predict(X100).mean()
model.predict(X100).min()
model.predict(X100).max()
shap_values[18:19,:]
X[18:19]
model.predict(X[18:19])
shap_values[18:19,:].values.sum() + shap_values[18:19,:].base_values
```

Таким образом, прогнозируемый результат для записи № 18 равен 16.178, и сумма значений SHAP от различных характеристик плюс базовое значение, которое является средним прогнозируемым, эквивалентно прогнозируемому значению, равному 16.178.

```
shap.plots.scatter(shap_values[:, "RM"])
```

На рис. 5.6 зависимость является линейной, поскольку значения SHAP генерируются с использованием линейной модели в качестве объяснителя. При переключении на нелинейную модель можно ожидать, что зависимость будет нелинейной.

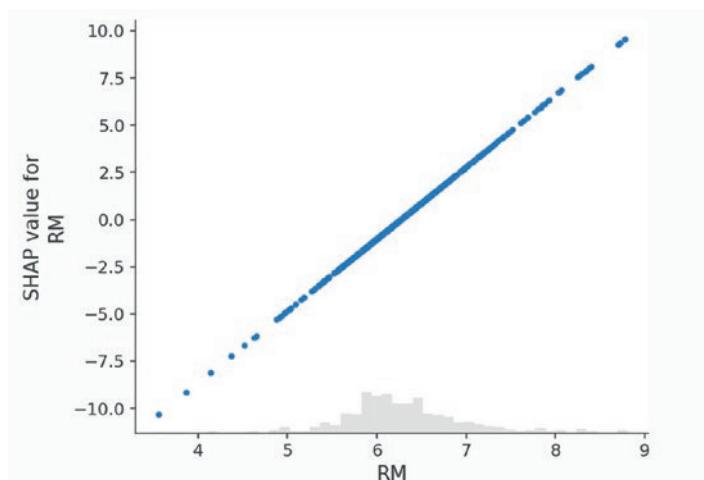


Рис. 5.6. Линейная зависимость между значениями RM и SHAP для RM

```
# диаграмма водопада показывает, как мы переходим от shap_values.base_values к
# model.predict(X)[sample_ind]
shap.plots.waterfall(shap_values[sample_ind], max_display=14)
```

На рис. 5.7 на горизонтальной оси дано среднее значение прогнозируемого результата, которое составляет 22.841, а на поле рисунка отображены значения SHAP для различных характеристик. Предполагаемые значения каждой характеристики из набора данных отображаются серым цветом, отрицательные значения SHAP – синим, а положительные значения SHAP – красным. На вертикальной оси показан также прогнозируемый результат для 18-й записи, равный 16.178.

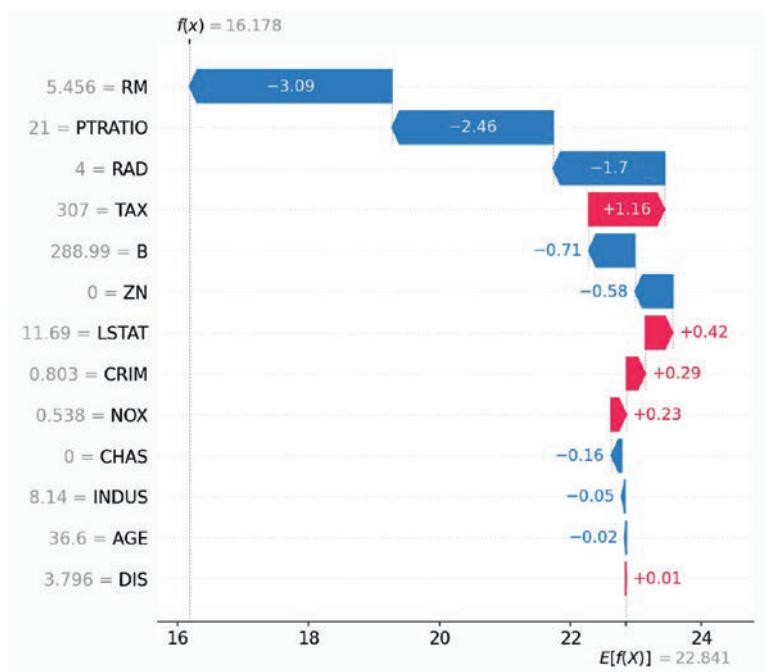


Рис. 5.7. Связь между прогнозируемым результатом и значениями SHAP

ИСПОЛЬЗОВАНИЕ ИНТЕРПРЕТАЦИИ, ОБЪЯСНЯЮЩЕЙ МОДЕЛЬ ПОВЫШЕНИЯ

В этом разделе для прогнозирования цен на жилье будут использоваться обобщенные аддитивные модели (generalized additive models – GAM). Модель можно объяснить с помощью библиотеки SHAP. Обобщенная аддитивная модель может быть обучена с использованием библиотеки интерпретации Python, а затем обученный объект может быть пропущен через модель SHAP с целью генерации объяснений для моделей повышения.

Библиотека интерпретации может быть установлена тремя способами:

```
!pip install interpret-core
```

Это без каких-либо зависимостей с использованием процесса установки pip.

```
conda install -c interpret-core
```

Здесь используется распределение anaconda. Установить с помощью терминала можно из среды conda.

```
git clone https://github.com/interpretml/interpret.git && cd interpret/scripts
&& make install-core
```

Это прямо из источника с помощью GitHub.

Библиотека интерпретации Python поддерживает два вида алгоритмов.

- **Модели Glassbox.** Они предназначены для того, чтобы быть более интерпретируемыми, используя фреймворк scikit-learn, поддерживающий тот же уровень точности, что и современная библиотека sklearn. Они поддерживают четыре различных типа моделей – линейные, деревья решений, правила принятия решений и модели на основе повышения.
- **Объяснители Blackbox.** Они предназначены, чтобы дать приблизительные объяснения того, как ведет себя модель, и прогнозы модели.

Эти варианты алгоритмов будут полезны, когда ни один из компонентов модели машинного обучения не интерпретируется.

Они поддерживают объяснения Shapely, объяснения LIME, графики частичных зависимостей и анализ чувствительности Морриса.

```
# подгонка модели GAM к данным
import interpret.glassbox
model_ebm = interpret.glassbox.ExplainableBoostingRegressor()
model_ebm.fit(X, y)
```

Во-первых, необходимо импортировать модуль `glassbox` из интерпретации, инициализировать объяснимый повышающий регрессор `ExplainableBoostingRegressor` и подогнать модель. Объектом модели является `model_ebm`.

```
#объяснение модели GAM с помощью SHAP
explainer_ebm = shap.Explainer(model_ebm.predict, X100)
shap_values_ebm = explainer_ebm(X)
```

Вы будете пробовать обучающий набор данных и отберете 100 образцов, чтобы создать фон для генерации пояснений с использованием библиотеки SHAP. В объяснителе SHAP `shap.Explainer` вы используете `model_ebm.predict` и берете 100 образцов для создания объяснений.

```
# создание стандартного графика частичной зависимости с наложенным единственным
# значением SHAP
fig,ax = shap.partial_dependence_plot(
    "RM", model_ebm.predict, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False,
    shap_values=shap_values_ebm[sample_ind:sample_ind+1,:]
)
```

На рис. 5.8 показана модель на основе повышения. Можно увидеть поэтапный разрыв нелинейной кривой и нелинейную зависимость между значениями RM и прогнозируемым целевым столбцом, который является средним значением цен на жилье. Здесь снова объяснена та же запись № 18, как показано красной прямой линией.

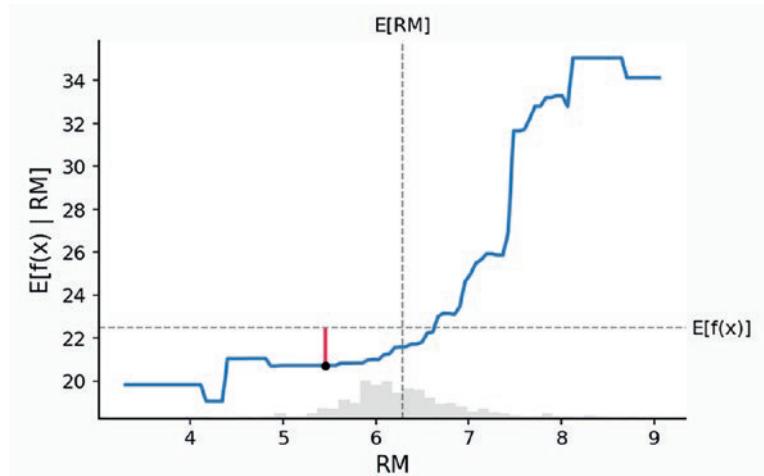


Рис. 5.8. График PDP модели повышения для характеристики RM

```
shap.plots.scatter(shap_values_ebm[:, "RM"])
```

На рис. 5.9 связь показана нелинейной кривой. На начальном участке ожидаемое значение не сильно увеличивается при увеличении RM, но затем значение SHAP увеличивается по экспоненте (см. также рис. 5.10).

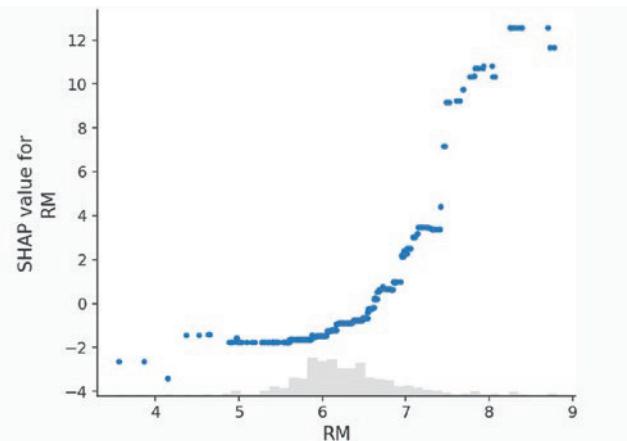


Рис. 5.9. RM и SHAP для RM

```
# диаграмма водопада показывает, как мы переходим от explainer.expected_value к
# model.predict(X)[sample_ind]
shap.plots.waterfall(shap_values_ebm[sample_ind], max_display=14)
```

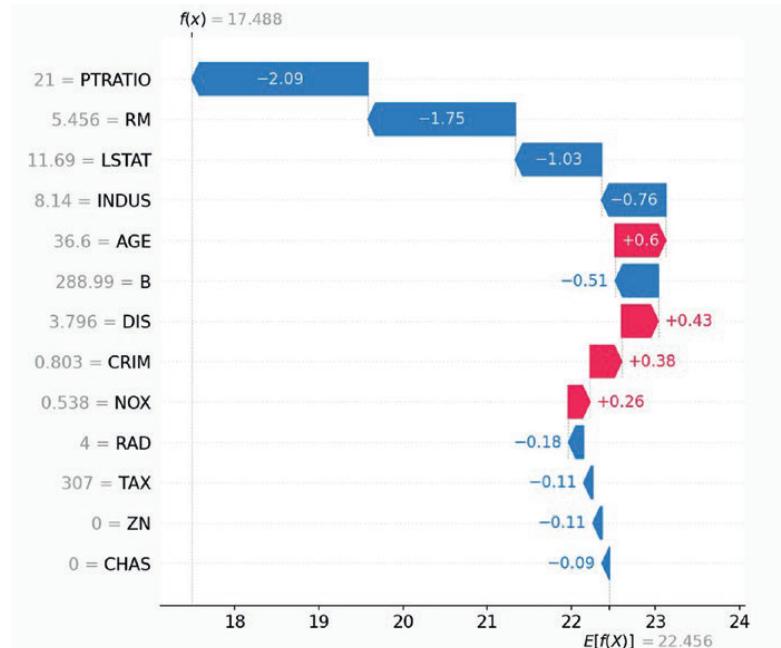


Рис. 5.10. Диаграмма водопада для того же набора переменных

```
shap.plots.beeswarm(shap_values_ebm, max_display=14)
```

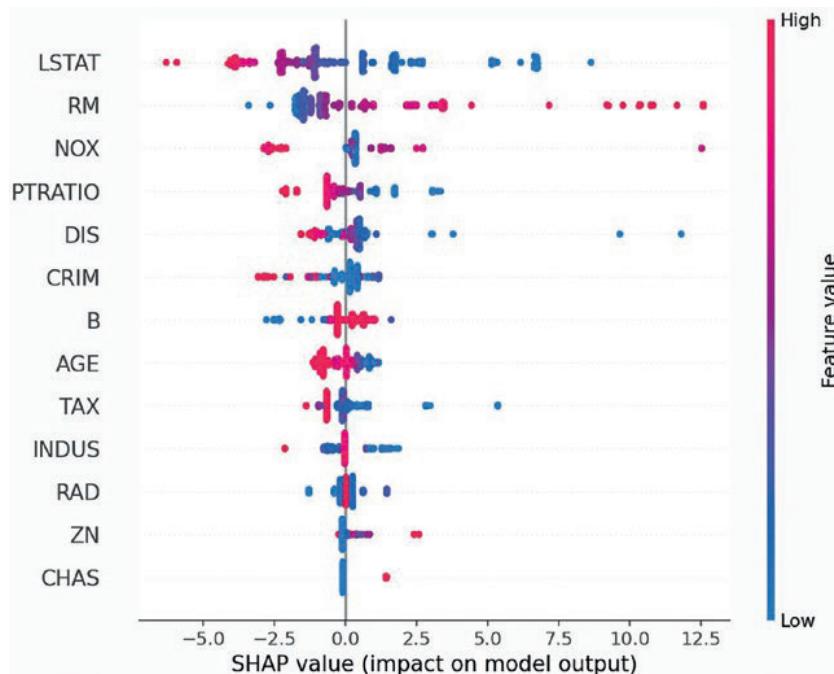


Рис. 5.11. Связь значений характеристик со значениями SHAP

На рис. 5.11 показана другая визуализация связи значений SHAP и характеристик.

```
# обучение модели XGBoost
import xgboost
model_xgb = xgboost.XGBRegressor(n_estimators=100, max_depth=2).fit(X, y)

# объяснение модели GAM с помощью SHAP
explainer_xgb = shap.Explainer(model_xgb, X100)
shap_values_xgb = explainer_xgb(X)

# построение стандартного графика частичной зависимости с наложением
единственного значения SHAP
fig,ax = shap.partial_dependence_plot(
    "RM", model_xgb.predict, X, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False,
    shap_values=shap_values_ebm[sample_ind:sample_ind+1,:]
)
```

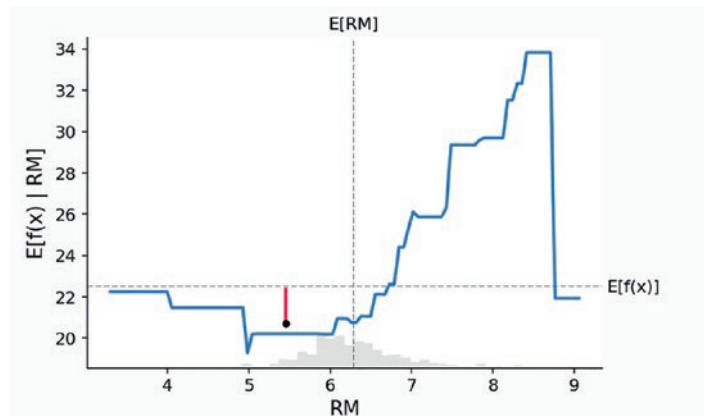


Рис. 5.12. PDP для RM и извлеченных значений SHAP

На рис. 5.12 используется модель регрессии с повышением экстремального градиента, чтобы объяснить ансамблевые модели, и здесь также вы видите, что есть нелинейные отношения.

```
shap.plots.scatter(shap_values_xgb[:, "RM"])
```

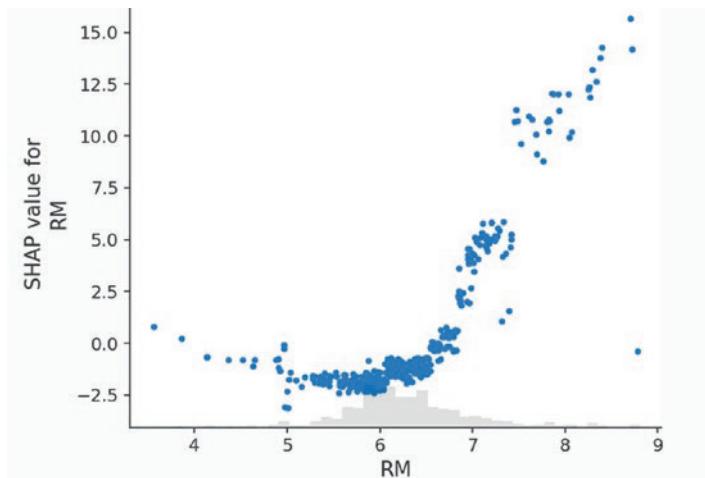


Рис. 5.13. Диаграмма рассеяния RM и значений SHAP

На рис. 5.13 представлена кривая, которая показывает нелинейную связь между значениями RM и SHAP.

```
shap.plots.scatter(shap_values_xgb[ :, "RM"] , color=shap_values)
```

Рисунок 5.14 отображает ту же нелинейную связь с дополнительным наложением характеристики TAX (красным цветом) и показывает – чем выше значение RM, тем выше компонент TAX, и наоборот.

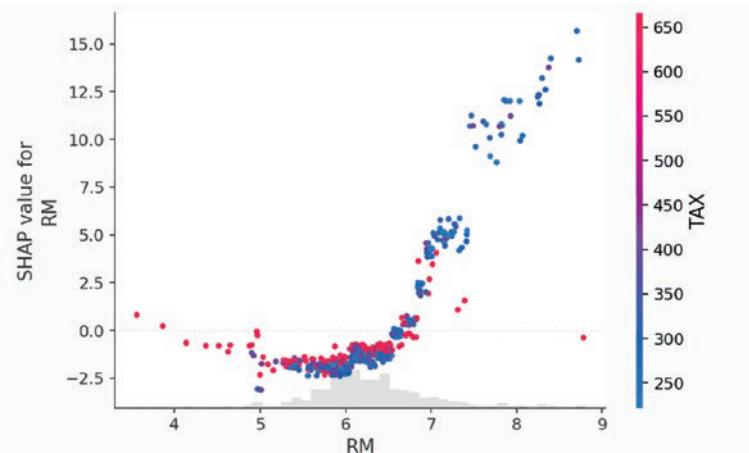


Рис. 5.14. К диаграмме рассеяния значений RM и SHAP RM добавлена характеристика TAX

МОДЕЛЬ КЛАССИФИКАЦИИ АНСАМБЛЕЙ: SHAP

Для регрессионной модели вы видели характеристики SHAP и важность характеристик, включая график частичной зависимости для нескольких из них. Теперь давайте рассмотрим распространенный и популярный набор данных классификации переписи доходов (обычно известный как взрослый набор данных). Он популярен, потому что легко понятен, узнаваем и появляется почти во всех примерах кода машинного обучения.

```
# классический взрослый набор данных переписи
X_adult,y_adult = shap.datasets.adult()

# простая линейная логистическая модель
model_adult = sklearn.linear_model.LogisticRegression(max_iter=10000)
model_adult.fit(X_adult, y_adult)
```

В приведенный выше скрипт поступает набор данных переписи доходов, который уже является частью библиотеки SHAP. Модель логистической регрессии обучена, поскольку это – задача бинарной классификации. Наконец, процесс подгонки модели создает обученную модель.

```
def model_adult_proba(x):
    return model_adult.predict_proba(x)[:,1]
def model_adult_log_odds(x):
    p = model_adult.predict_log_proba(x)
    return p[:,1] - p[:,0]
```

В примере классификации, использующем набор данных переписи доходов, целевой столбец имеет две метки: люди, зарабатывающие более 50 тыс. долл. в год, и люди с меньшим доходом. Это – модель бинарной классификации, и поэтому в качестве базовой вы можете использовать модель логистической регрессии, чтобы продемонстрировать PDP как линейную функцию значений SHAP. Затем можете взять ансамблевую модель, чтобы сравнить PDP и показать, как функция становится нелинейной. Нелинейные и сложные отношения получены ансамблевой моделью. Две функции, называемые `model_adult_proba` и `model_adult_log_odds`, выводят значения вероятности двух классов и логарифмическое отношение шансов соответственно.

```
# построение стандартного графика частичной зависимости
sample_ind = 18
fig,ax = shap.partial_dependence_plot(
    "Capital Gain", model_adult_proba, X_adult, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)
```

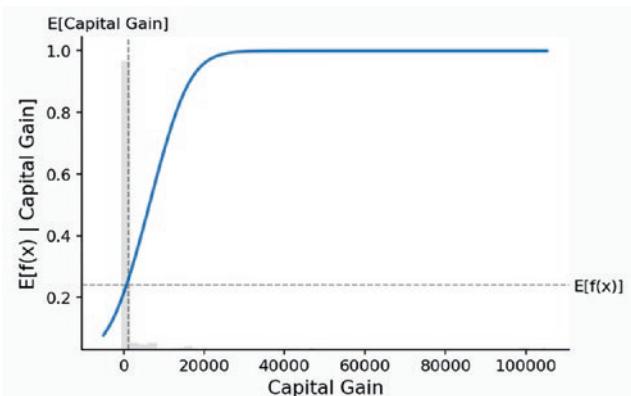


Рис. 5.15. PDP между приростом капитала и вероятностью иметь менее 50 тыс. долл.

Ось x на рис. 5.15 показывает переменную прироста капитала, а ось y – предсказанную вероятность иметь менее чем 50 тыс. долл. Значения вероятности изменяются от 0.05 до почти 1.00 (ближе к 100 %, но не совсем). Объяснение здесь состоит в том, что, когда сумма прироста капитала превышает 20 тыс. долл., она не оказывает влияния на предсказанное значение вероятности. До 20 тыс. долл., когда прирост капитала увеличивается, предсказанная вероятность также увеличивается.

```
# вычисление значений SHAP для линейной модели
background_adult = shap.maskers.Independent(X_adult, max_samples=100)
explainer = shap.Explainer(model_adult_proba, background_adult)
shap_values_adult = explainer(X_adult[:1000])

shap.plots.scatter(shap_values_adult[:, "Age"])
```

Рисунок 5.16 изображает линейное соотношение, но есть вариации в распределении точек данных, которые показывают, что нелинейность, возможно, существует, и что это не может быть отражено правильно при использовании линейной модели. Следовательно, есть потребность в ансамблевой модели.

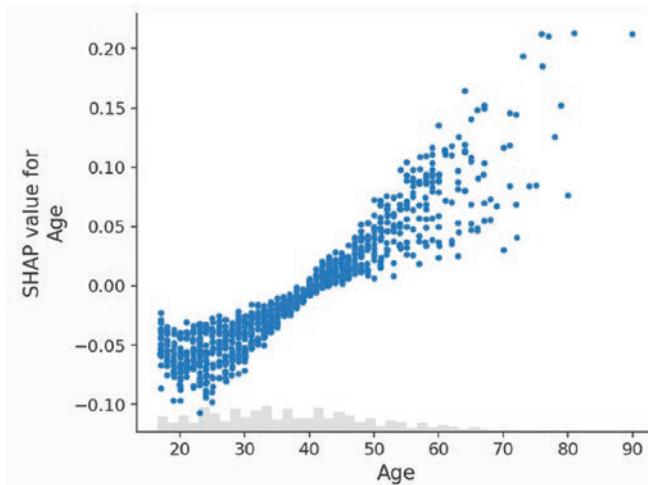


Рис. 5.16. Диаграмма рассеяния между возрастом и значением SHAP для возраста для линейной модели

```
# вычисление значений SHAP для линейной модели
explainer_log_odds = shap.Explainer(model_adult_log_odds, background_adult)
shap_values_adult_log_odds = explainer_log_odds(X_adult[:1000])

shap.plots.scatter(shap_values_adult_log_odds[:, "Age"])
# построение стандартного графика частичной зависимости
sample_ind = 18
fig,ax = shap.partial_dependence_plot(
    "Age", model_adult_log_odds, X_adult, model_expected_value=True,
    feature_expected_value=True, show=False, ice=False
)

# обучение модели XGBoost
model = xgboost.XGBClassifier(n_estimators=100, max_depth=2).fit(X_adult,
y_adult)

# вычисление значений SHAP
explainer = shap.Explainer(model, background_adult)
shap_values = explainer(X_adult)

# установка версии дисплея данных для графического отображения (имеет строковые
# значения)
shap_values.display_data = shap.datasets.adult(display=True)[0].values
```

Теперь давайте рассмотрим модель классификатора с повышением экстремального градиента. Как модель повышения, она улучшает точность классификатора и дает объяснения предсказанных результатов.

```
shap.plots.bar(shap_values)
```

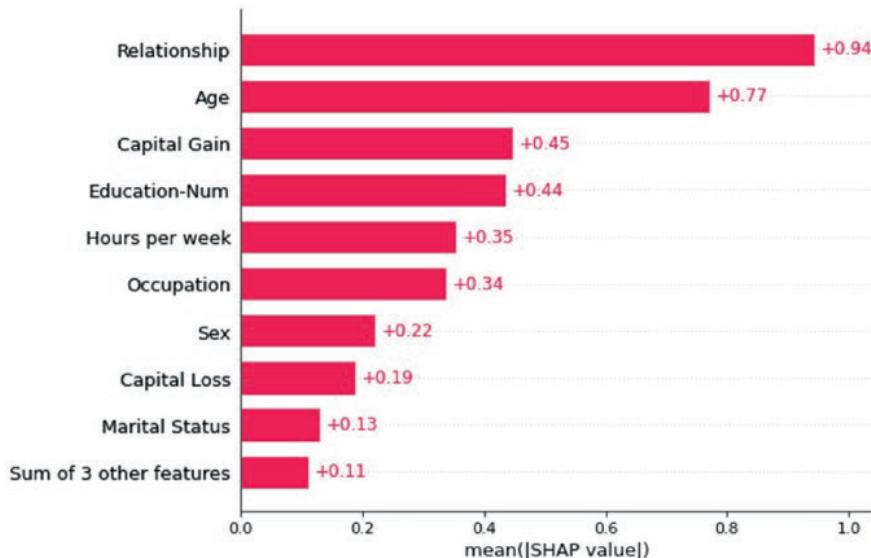


Рис. 5.17. Средние абсолютные значения SHAP для оценки характеристик

На рис. 5.17 переменная «степень родства» показывает самое высокое среднее абсолютное значение SHAP, которое является отражением того, что эта переменная наиболее важна в предсказании класса целевого столбца. Вторая наиболее важная переменная – возраст, затем прирост капитала и т. д.

```
shap.plots.bar(shap_values.abs.max(0))
```

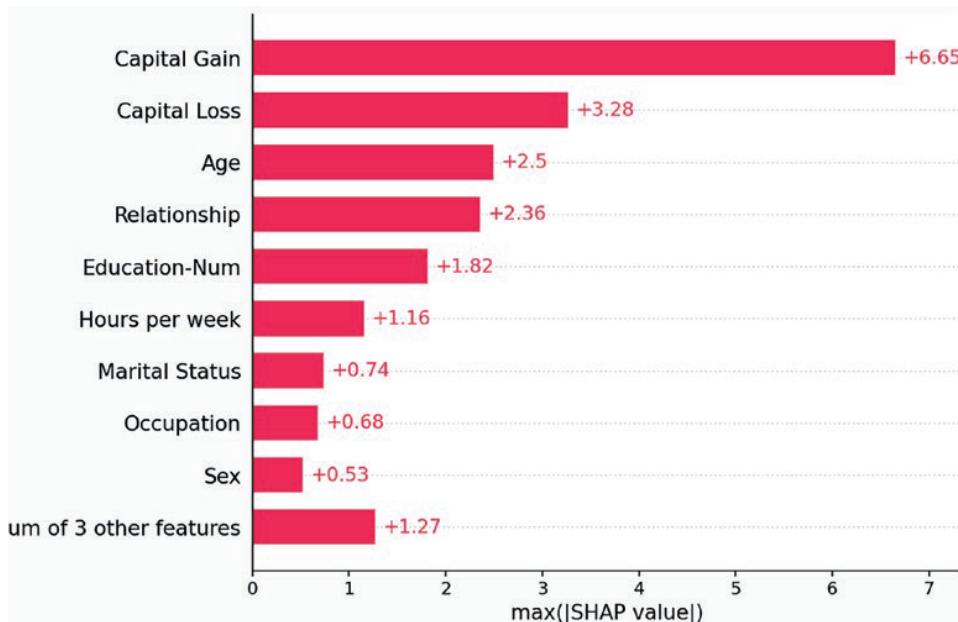


Рис. 5.18. Максимальные абсолютные значения SHAP для различных характеристик

На рис. 5.18 характеристика прироста капитала имеет самое высокое значение SHAP, таким образом, это могло быть одним из выбросов. Точно так же потери капитала – возможно, выброс. Это разъясняет рис. 5.19.

На рис. 5.19 значения SHAP показывают, как значения характеристик влияют на результат моделирования. Синие точки указывают на низкие значения характеристик, а красные – на высокие значения.

```
shap.plots.beeswarm(shap_values)
```

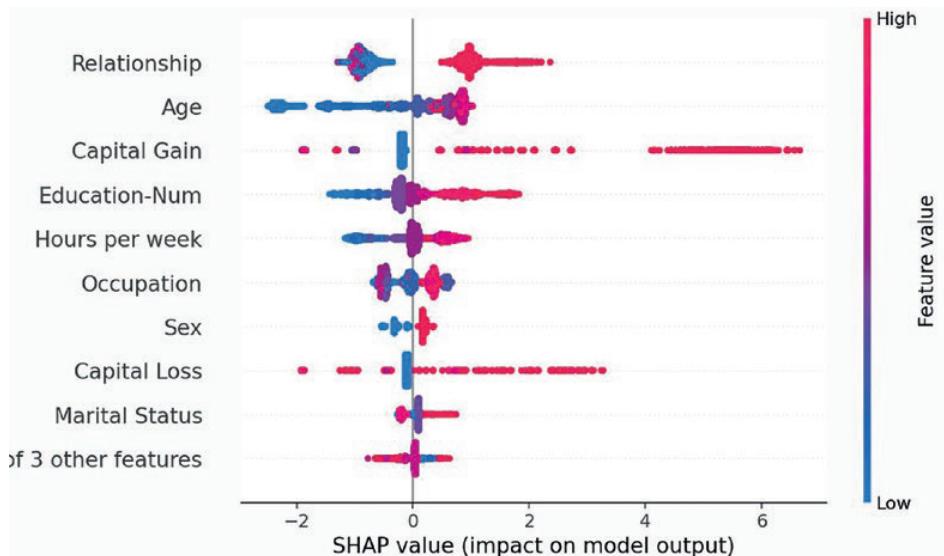


Рис. 5.19. Связь между значениями SHAP и значениями характеристик

```
shap.plots.beeswarm(shap_values.abs, color="shap_red")
```

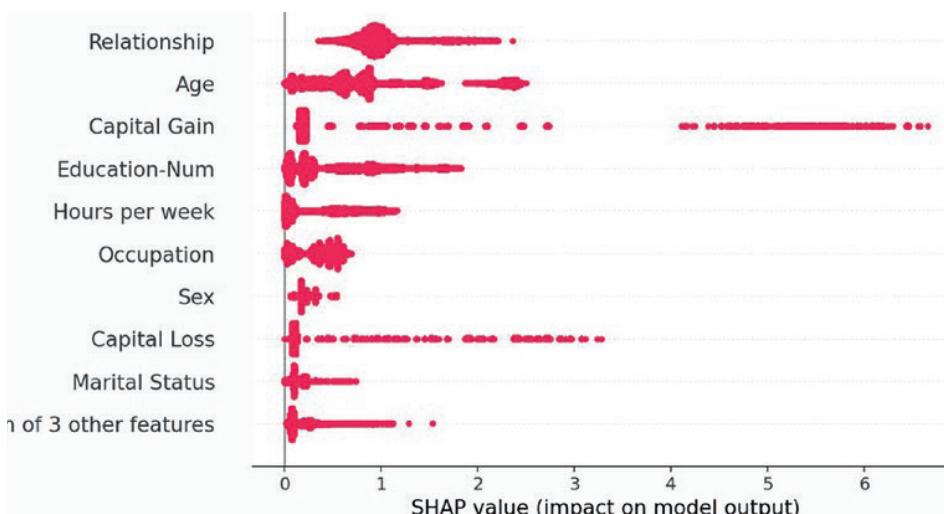


Рис. 5.20. Абсолютные значения SHAP

На рис. 5.20 не видны отрицательные значения SHAP, и две характеристики, прирост и потери капитала, имеют большие изменения в значениях SHAP, что означает, что вариация в фактических значениях этих двух характеристик могла породить противоречивые прогнозы.

Давайте рассмотрим характеристику «степень родства» на рис. 5.21. Она имеет высокие значения SHAP до 430-го экземпляра. После этого значения SHAP становятся отрицательными до 580-го экземпляра и продолжаются до 610-го. Затем все экземпляры этой характеристики производят отрицательные значения SHAP. Этот график показывает вариации в значениях SHAP, связанные с различными экземплярами записей в обучающем наборе данных.

```
shap.plots.heatmap(shap_values[:1000])
```

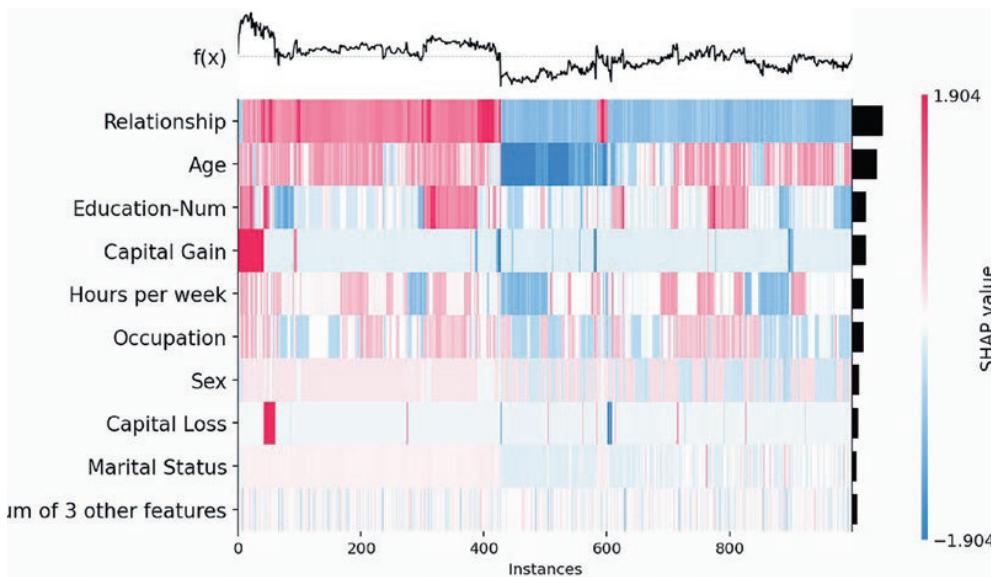


Рис. 5.21. Изменение значений характеристик в зависимости от экземпляра записи и значения SHAP

Рисунок 5.22 объясняет нелинейную зависимость между переменной возраста и значениями SHAP для нее. Эта нелинейность произведена моделью классификатора с повышением экстремального градиента.

```
shap.plots.scatter(shap_values[:, "Age"])
```

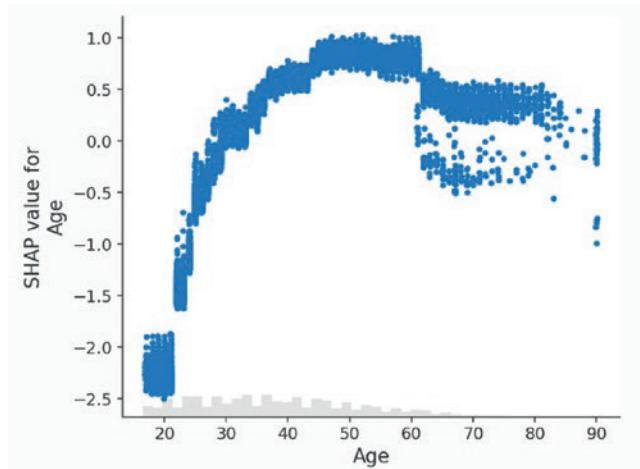


Рис. 5.22. Возраст и значение SHAP для него

```
shap.plots.scatter(shap_values[:, "Age"], color=shap_values)
```

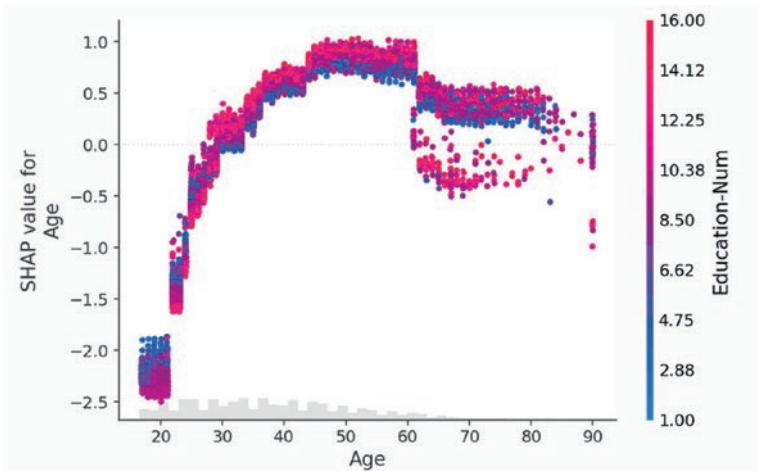


Рис. 5.23. Возраст и значение SHAP возраста с наложенной длительностью обучения

На рис. 5.23 показана связь возраста и длительности обучения со значением SHAP для возраста.

```
shap.plots.scatter(shap_values[:, "Age"], color=shap_values[:, "Education-Num"])
```

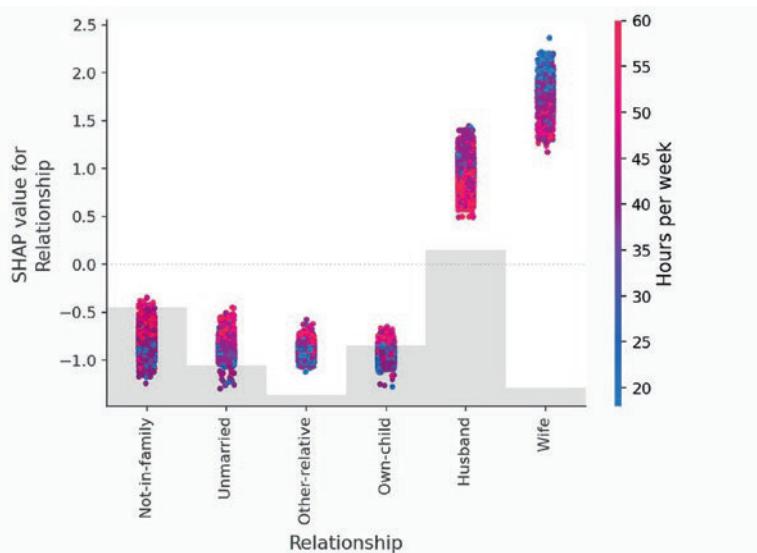


Рис. 5.24. Степень родства и значения SHAP для степени родства с наложением числа рабочих часов в неделю

На рис. 5.24 есть четкое отличие мужа, жены и остальных значений степени родства. Значения SHAP для мужа и жены положительны, а для других четырех значений – отрицательны.

Использование SHAP для объяснения категориальных моделей повышения

Когда характеристики модели классификации содержат категориальные переменные, мы обычно выполняем горячее кодирование для характеристик, чтобы определить влияние каждой из них на целевой столбец. В этом примере мы будем пользоваться библиотекой CatBoost Python наряду с SHAP, чтобы генерировать объяснения.

Для установки библиотеки CatBoost можно использовать команду `pip`.

```
!pip install catboost
shap.initjs()
import catboost
from catboost.datasets import *
import shap
```

Набор данных переписи о доходах содержит категориальные характеристики, такие как степень родства, пол, занятость и семейное положение. Эти характеристики не могут быть непосредственно переданы алгоритмам машинного обучения при помощи `sklearn` или любого другого конвейера, например `Keras`, `TensorFlow` и т. д. Эти категориальные характеристики сначала должны быть преобразованы в кодировщик меток и затем – в горячий кодировщик. Это

становится очень громоздким, если у вас есть более ста характеристик, которые являются категориальными. Следовательно, вам нужна библиотека, которая может обработать этот процесс автоматически. Этой потребности отвечает библиотека CatBoost.

Функция `initjs()` загружает код визуализации JavaScript в Jupyter Notebook. Следовательно, вышеупомянутый синтаксис используется для загрузки функциональности JS.

Регрессионная модель CatBoost или модель классификации работает поверх модели повышения градиента, отсюда название CatBoost.

```
X,y = shap.datasets.boston()

model = CatBoostRegressor(iterations=300, learning_rate=0.1, random_seed=123)
model.fit(X, y, verbose=False, plot=False)

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X)

# визуализация объяснения первого прогноза
shap.force_plot(explainer.expected_value, shap_values[0,:], X.iloc[0,:])
```



Рис. 5.25. График силы для прогноза записи

На рис. 5.25 мы пытаемся объяснить локальный прогноз. На нем показано объяснение прогноза для записи 0 из обучающего набора данных. Для примера мы взяли набор данных о ценах на жилье в Бостоне, обучили модель CatBoost, используя параметры по умолчанию, и сохранили модель. В качестве следующего шага сгенерирован объект объяснителя с использованием древовидного объяснителя, и объект объяснителя снова используется для генерации значений SHAP с применением обучающего набора данных X в качестве входа. Наконец, создана визуализация, использующая ожидаемое значение и значение SHAP для первой записи из обучающего набора данных. Для первой записи предсказанная цена на жилье 25.35, т. е. 25 350 долл. Средняя предсказанная цена на недвижимость равна 22.53, что составляет 22 530 долл. Такие характеристики, как возраст, PTRATIO и LSTAT, продвигают прогноз выше (красная стрела, направленная вправо). RAD, CRIM и RM продвигают прогноз вниз (синяя стрела, направленная влево). Значения, отображенные на горизонтальной оси серым, являются ожидаемыми значениями для всех образцов.

```
# визуализация прогнозов обучающего набора данных
shap.force_plot(explainer.expected_value, shap_values, X)
```

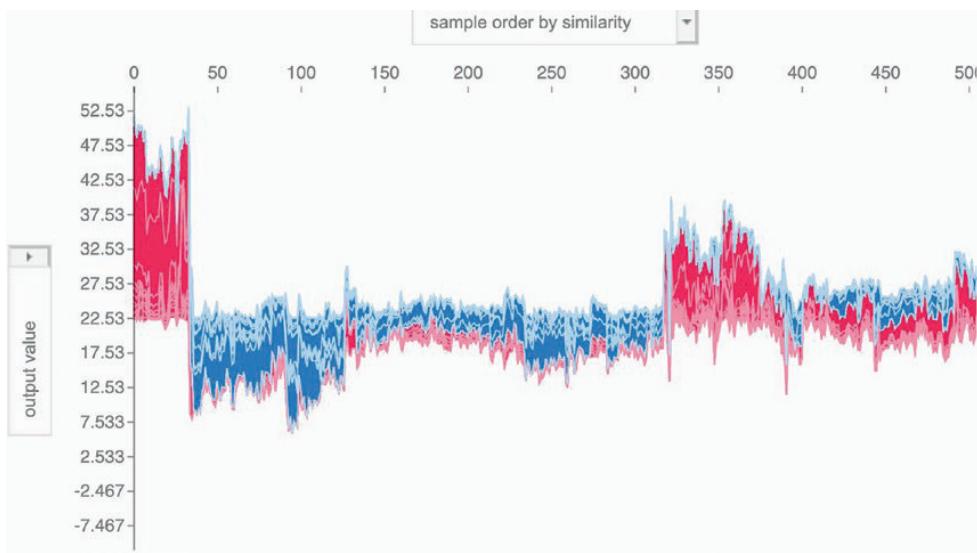


Рис. 5.26. Объяснитель модели, выбранный по сходству образцов

На рис. 5.26 образцы выбраны по сходству образцов. Горизонтальная ось показывает образцы, а вертикальная – предсказанное выходное значение. Это интересная визуализация: просто наведя курсор мыши, можно увидеть значения SHAP-характеристик, предсказанное выходное значение и соответствующий образец, который был выбран как вход объяснителя. Эта диаграмма дает полное представление о понимании предсказуемого результата и вкладов характеристик.

Чтобы объяснить применение классификатора CatBoost, используем набор данных Amazon. Характеристики набора данных перечислены в табл. 5.6.

Таблица 5.6. Описание данных набора данных Amazon

Имя характеристики	Описание
ACTION	ACTION равен 1, если ресурс был разрешен, 0, если нет
RESOURCE	Идентификатор для каждого ресурса
MGR_ID	Идентификатор сотрудника у менеджера текущей записи идентификатора сотрудника; у сотрудника может быть только один менеджер
ROLE_ROLLUP_1	Идентификатор категории групп ролей компании 1 (например, американская разработка)
ROLE_ROLLUP_2	Идентификатор категории групп ролей компании 2 (например, американская розничная торговля)
ROLE_DEPTNAME	Ролевое описание отдела компании (например, розничная торговля)
ROLE_TITLE	Ролевое описание обязанностей в бизнесе компании (например, старший инженер по розничной торговле)
ROLE_FAMILY_DESC	Расширенное описание семейства ролей в компании (например, менеджер розничной торговли, разработка программного обеспечения)

Имя характеристики	Описание
ROLE_FAMILY	Описание семейства ролей в компании (например, менеджер розничной торговли)
ROLE_CODE	Ролевой код, этот код уникален для каждой роли (например, менеджер)

Когда сотрудник приступает к работе в отделе разработки программного обеспечения, ему необходим определенный доступ и полномочия для выполнения своих должностных обязанностей. Есть выполняемый вручную процесс предоставления доступа сотруднику, но во время работы сотрудник может встретиться с ограничениями. Нет никакого автоматизированного механизма, чтобы предоставить доступ к необходимым программным системам. Цель этого варианта использования состоит в моделировании исторических данных для определения потребности в доступе сотрудника. Целевой столбец – ACTION; это 1, если ресурс разрешен, и 0, если доступ к ресурсу запрещен. Остальные характеристики используются, чтобы предсказать целевую характеристику.

```
train_df, test_df = catboost.datasets.amazon()
train_df
set(train_df.ACTION)

y_train = train_df.ACTION
X_train = train_df.drop('ACTION',axis=1)
cat_features = list(range(0,X_train.shape[1]))
from catboost import Pool, CatBoostClassifier, cv
# Инициализация CatBoostClassifier
model = CatBoostClassifier(iterations=300,
                            learning_rate=0.1,
                            random_seed=12)

# Подгонка модели
model.fit(X_train,y_train,cat_features=cat_features,verbose=False,
plot=False)
```

У процесса обучения модели с использованием классификатора CatBoost есть некоторая сложность, что означает сложные отношения, которые не могут быть аппроксимированы одной функцией. Если характеристики имеют линейную зависимость, можно использовать прямую линию, чтобы отобразить функцию. Однако если характеристики имеют круговой или зигзагообразный вид, то это называют сложным шаблоном, поскольку нет никакой математической функции, которая может аппроксимировать их вид. В таких сценариях или вам нужны более чем одна функция, или нормализация данных, чтобы соответствовать любой математической функции.

В данном сценарии значения SHAP вычислены приблизительно из-за сложности набора данных характеристик.

```

eval_dataset = [X_train.iloc[0:1]]
eval_dataset

# Получение ожидаемых классов
preds_class = model.predict(X_train)

# Получение ожидаемых вероятностей для каждого класса
preds_proba = model.predict_proba(X_train)

# Получение ожидаемого RawFormulaVal
preds_raw = model.predict(X_train,
                           prediction_type='RawFormulaVal')

preds_class
preds_proba
preds_raw

```

Обученная модель сохранена в объекте модели. Вы можете взять два обзора в качестве примеров, чтобы найти вероятность для экземпляра и необработанного значения формулы. Образец может быть любой точкой данных из обучающего набора данных. В скрипте возьмем первую и 91-ю запись в качестве примеров.

```

import numpy as np
np.log(0.9964/(1-0.9964))
np.exp(5.62)/(1+np.exp(5.62))
cat_features

```

Первая запись показывает ожидаемую вероятность 99.64 % для класса 1. Значение логарифмической вероятности равно 5.61. Логарифмическая вероятность равна $\log(P/1 - P)$, где P является вероятностью класса 1. Чтобы получить вероятность исходного значения прогноза (raw prediction), можно использовать формулу $\exp(\text{raw prediction}) / (1 + \exp(\text{raw prediction}))$. Например, для первой записи $\exp(5.61) / (1 + \exp(5.61)) = 0.9964$. Подобным способом можно интерпретировать исходное значение прогноза -3.4734 для 91-й записи. Модель CatBoost предоставляет возможности предсказывать различные типы прогноза (см. табл. 5.7).

Таблица 5.7. Типы прогнозов, генерируемых объектом модели CatBoost

Тип прогноза	Описание
Probability	Значение вероятности, соответствующее каждому классу
Class	Будет создана метка класса
RawFormula Value	Значение логарифмической вероятности
Exponent	Экспоненциальное значение вероятности прогноза
LogProbability	Логарифм вероятности

Различные типы прогнозов из модели CatBoost дают больше понимания значения прогноза и того, как модель генерирует прогноз. Связь между всеми

типами прогнозов обеспечивает лучшее их понимание. Вы преобразуете исходное значение прогноза, чтобы получить вероятность класса 1.

```
from catboost.datasets import *
train_df, test_df = catboost.datasets.amazon()
y = train_df.ACTION
X = train_df.drop('ACTION', axis=1)
cat_features = list(range(0, X.shape[1]))

model = CatBoostClassifier(iterations=300, learning_rate=0.1, random_seed=12)
model.fit(X, y, cat_features=cat_features, verbose=False, plot=False)

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(Pool(X, y, cat_features=cat_features))

test_objects = [X.iloc[7:9], X.iloc[56:58]]

for obj in test_objects:
    print('Probability of class 1 = {:.4f}'.format(model.predict_proba(obj)[0][1]))
    print('Formula raw prediction = {:.4f}'.format(model.predict(obj,
        prediction_type='RawFormulaVal')[0]))
    print('\n')
```

```
shap.force_plot(explainer.expected_value, shap_values[0,:], X.iloc[0,:])
```



```
shap.force_plot(explainer.expected_value, shap_values[91,:], X.iloc[91,:])
```



Рис. 5.27. График силы для записи 1 из обучающего набора данных

График силы на рис. 5.27 показывает, как исходное значение прогноза образовано на основе полученных (синий, нижний график) и отправленных (красный, верхний график) значений характеристик. Значения от 1.124 до 6.124 являются логарифмическим отношением шансов вероятности прогноза относительно класса 1. Среднее значение логарифмического отношения шансов – 3.624. Полученные значения характеристик RESOURCE, ROLE_TITLE, ROLE_ROLLUP_2 и т. д. уменьшают ожидаемые значения отношения шансов. Отправленные характеристики (ROLE_DEPTNAME, MGR_ID и т. д.) увеличивают ожидаемое отношение шансов. Окончательное значение 5.61 является ожидаемым исходным значением вероятности, также известным как отношение шансов.

```
shap.summary_plot(shap_values, X)
```

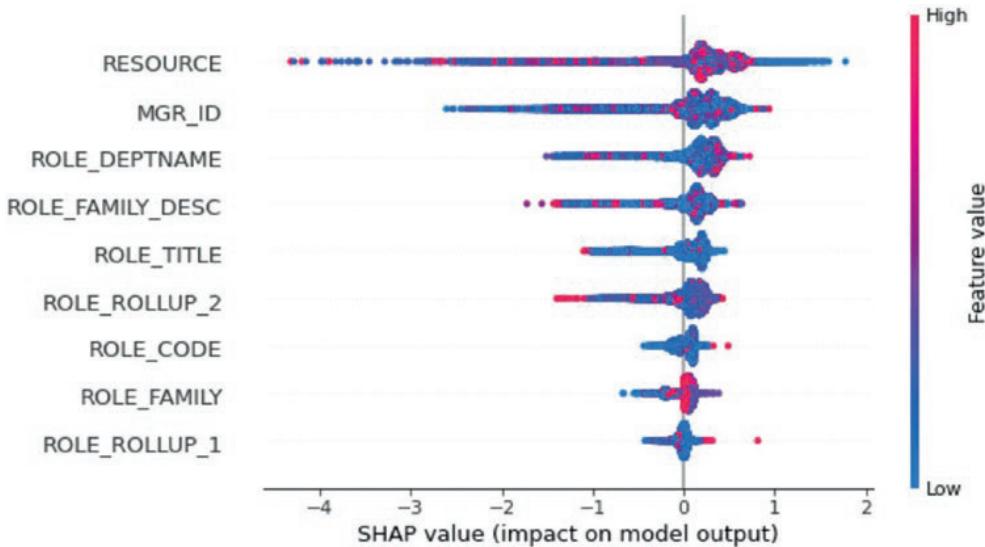


Рис. 5.28. Значения SHAP и характеристики, оказывающие влияние на результат моделирования

Рисунок 5.28 показывает характеристики и их важность во влиянии на результат моделирования. RESOURCE – самая эффективная характеристика, затем – MGR_ID и т. д., как изображено на рисунке.

ИСПОЛЬЗОВАНИЕ МНОГОКЛАССОВОЙ КАТЕГОРИАЛЬНОЙ МОДЕЛИ ПОВЫШЕНИЯ SHAP

Категориальная модель классификатора повышения может также использоваться для обучения модели многоклассовой классификации. Для генерации значений SHAP обученный объект модели может быть передан через древовидный объяснитель SHAP (см. рис. 5.29).

```
model = CatBoostClassifier(loss_function = 'MultiClass', iterations=300,
                            learning_rate=0.1, random_seed=123)
model.fit(X, y, cat_features=cat_features, verbose=False, plot=False)
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(Pool(X, y, cat_features=cat_features))

shap.summary_plot(shap_values[0], X)
```

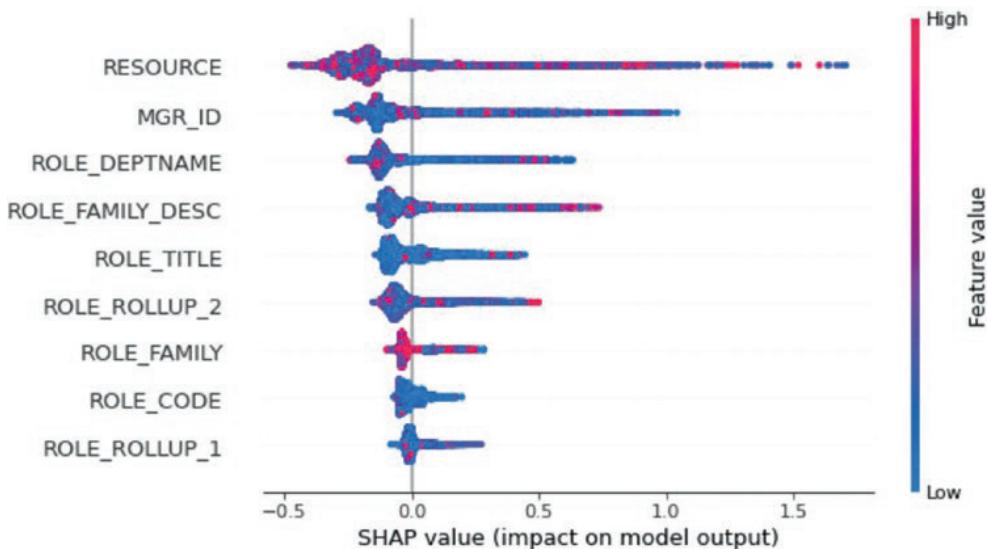


Рис. 5.29. Представление значений характеристик и SHAP

У классификатора CatBoost или регрессии есть много гиперпараметров. В табл. 5.8 показаны некоторые важные параметры и их влияние на результат моделирования.

Таблица 5.8. Гиперпараметры классификатора CatBoost

Параметр	Объяснение
cat_features	Индексы категориальных столбцов, которые присутствуют в обучающем наборе данных
text_features	Индексы текстовых столбцов (определенены как целые числа) или имена (определенны как строки)
learning_rate	Для оптимизации функции потерь, используемой для сокращения шага градиента
Iterations	Максимальное число деревьев, которые могут быть созданы для решения проблем машинного обучения
Max_depth	Глубина дерева
leaf_estimation_method	Метод раньше вычислял значения в листьях Возможные значения: Newton Gradient Exact Зависит от режима и выбранной функции потерь: регрессия с квантилем или функциями потерь MAE – одна итерация регрессия с любой функцией потерь, но с квантилем или MAE – одна итерация градиента режим Classification – десять итераций Newton режим мультиклассификации – одна итерация Newton

Параметр	Объяснение
boosting_type	Схема Boosting Возможные значения: Ordered – обычно обеспечивает лучшее качество на небольших наборах данных, но может быть медленнее, чем схема Plain Plain – классическая схема повышения градиента

Чтобы получить лучшие результаты, указанные выше параметры могут быть изменены и объяснены с использованием библиотеки SHAP.

Использование SHAP для объяснения модели легкой GBM

Как показано в разделе классификатора CatBoost, подобным образом можно также использовать легкую модель повышения градиента (gradient boosting model – GBM). Различие между легкой GBM и моделью CatBoost – то, что легкая GBM быстрее работает в смешанном сценарии данных, где в обучающем наборе данных есть смешанный набор характеристик. Классификатор CatBoost полезен, когда в обучающем наборе данных большое число категориальных характеристик. В легкой GBM вы должны закодировать категориальные характеристики, а в модели CatBoost имеется внутренний механизм их обработки.

Для установки легкой GBM примените следующий код:

```
conda install -c conda-forge lightgbm

from sklearn.model_selection import train_test_split
import lightgbm as lgb
import shap

X,y = shap.datasets.adult()
X_display,y_display = shap.datasets.adult(display=True)

# разделение обучения и тестирования
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=7)
d_train = lgb.Dataset(X_train, label=y_train)
d_test = lgb.Dataset(X_test, label=y_test)

params = {
    "max_bin": 512,
    "learning_rate": 0.05,
    "boosting_type": "gbdt",
    "objective": "binary",
    "metric": "binary_logloss",
    "num_leaves": 10,
    "verbose": -1,
```

```

    "min_data": 100,
    "boost_from_average": True
}

model = lgb.train(params, d_train, 10000, valid_sets=[d_test], early_
stopping_rounds=50, verbose_eval=1000)

```

Можно взять тот же набор данных классификации переписи доходов для обучения классификатора, используя легкую GBM, и генерировать объяснения ожидаемых результатов. Существует много гиперпараметров, которые можно понять из официальной документации легкой GBM-модели. В табл. 5.9 показаны гиперпараметры, которые используются в предыдущих строках кода.

Таблица 5.9. Гиперпараметры легкой модели повышения градиента

Параметры	Объяснение
Max_bin	Максимальное количество ячеек, в которых будут объединены в блок значения характеристики Малое количество ячеек может снизить точность обучения, но увеличить общую производительность (относится к переподгонке)
Learning_rate	Уровень уменьшения
Boosting_type	gbdt, традиционное дерево решений с повышением градиента, алиас: gbrt rf, случайный лес, алиас: random_forest dart, уволенные, соответствует нескольким аддитивным деревьям регрессии goss, основанная на градиенте односторонняя выборка (gradient-based one-side sampling)
Objective	Тип приложения, такой как бинарная классификация, многоклассовая классификация и т. д.
Metric	Метрика (метрики), которая будет оценена на оценочном наборе (наборах)
Num_leaves	Максимальное количество листов в дереве
Min_data	Минимальное количество данных на категориальную группу

После обучения легкой GBM (LGBM) можно перейти к древовидному объяснителю SHAP, чтобы объяснить ожидаемые результаты, а также построить график силы для значений SHAP (рис. 5.30).

```

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X)

shap.force_plot(explainer.expected_value[1], shap_values[1][0,:], X_
display.iloc[0,:])

```



Рис. 5.30. График силы для прогнозирования логарифмических вероятностей со значениями SHAP-характеристик

На рис. 5.30 длительность обучения и возраст увеличивают ожидаемое отношение шансов, а прирост капитала и степень родства уменьшают его. Следовательно, окончательное отношение шансов для записи 1 из набора данных равно -8.43 , что намного ниже, чем среднее отношение шансов, показанное базовым значением -2.43 .

```
shap.force_plot(explainer.expected_value[1], shap_values[1][:1000,:],
X_display.iloc[:1000,:])
```

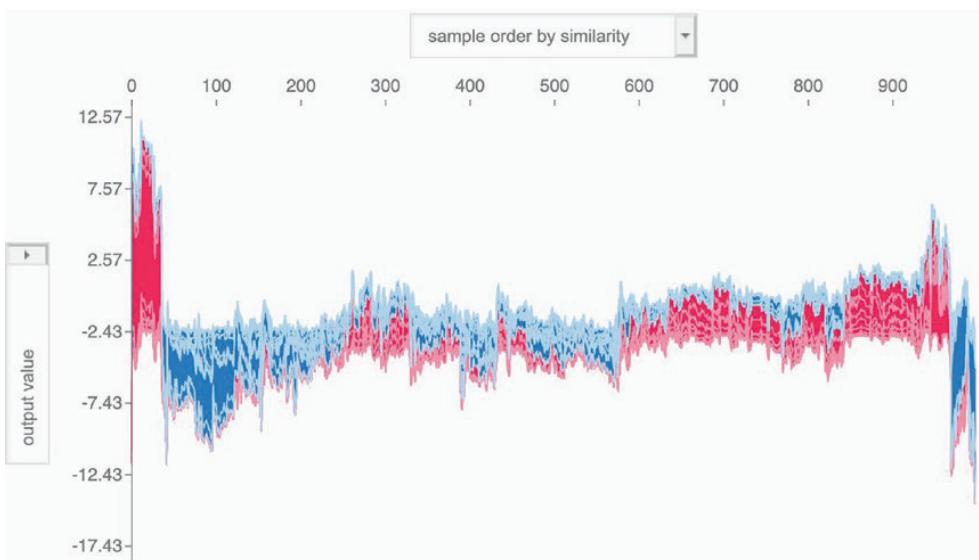


Рис. 5.31. Выбор сходства образцов для получения ожидаемых значений модели LGBM

На рис. 5.31 отношение шансов – результат, и из него можно получить значения вероятности целевых классов, как видно в этой главе. Этот график показывает сходство образцов. Те, которые производят более высокое отношение шансов, находятся между 0 и 50. Остальная часть образцов до записи 900 производит отношения шансов в диапазоне от +3 до -9 . После этого отношение шансов спадает до значительных отрицательных значений. Красная линия показывает значения характеристик, которые увеличивают ожидаемое отношение шансов, а синяя – значения характеристик, которые уменьшают ожидаемое отношение шансов. Средняя линия соответствует значению -2.43 . Если

взять отношения шансов для всего набора данных и затем вычислить среднюю величину, то получим именно -2.43 .

```
shap.summary_plot(shap_values, X)
```

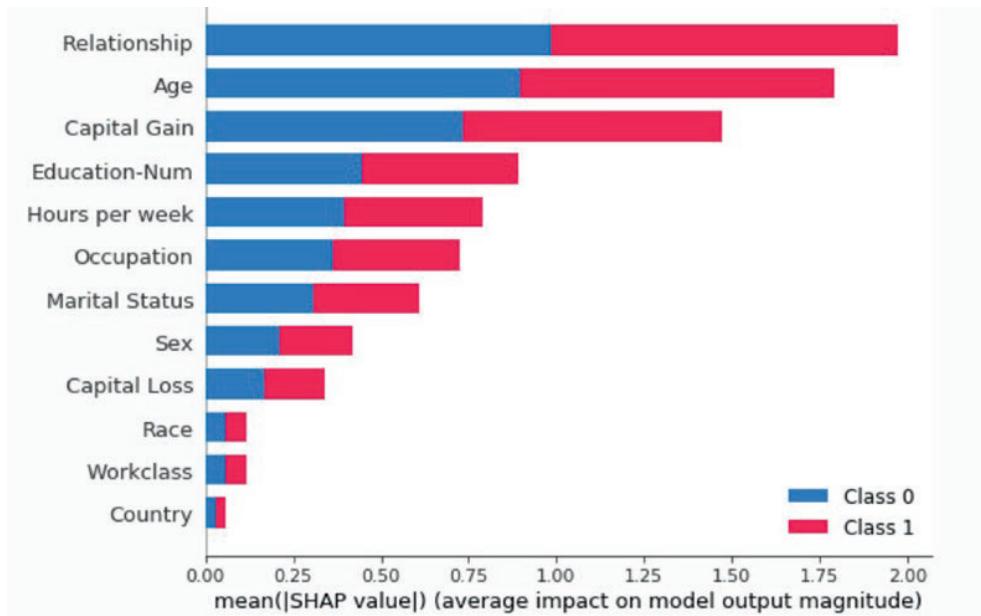


Рис. 5.32. Сводный график

На рис. 5.32 показан сводный график. Он очень важен. Он предоставляет среднее значение SHAP для различных характеристик, соответствующее каждому классу для целевого столбца. На этом графике интересно то, что среднее значение распределено одинаково между двумя классами независимо от значений характеристик.

ЗАКЛЮЧЕНИЕ

Вы рассмотрели несколько моделей на основе повышения для сценариев классификации и регрессионных сценариев. Аналогичным образом можно обучить классификаторы на основе упаковки и получить похожие графики в качестве результата. Графики при этом не изменятся, но интерпретация значений будет меняться при переходе от одной модели к другой. Вот пять важных моментов, на которые вам следует обратить внимание:

- в сценарии регрессии вы прогнозируете целевой столбец, и на основе вклада характеристик можете увидеть, как сдвигается вверх или вниз ожидаемый результат. Таким образом, если вы измените входную характеристику, то сможете легко понять, как это повлияет на целевой результат;
- в сценарии классификации вы прогнозируете логарифмическое отношение шансов в качестве непрерывной переменной и определяете его значение на основе значений входных характеристик. Вы знаете, какие

характеристики важны, поэтому у вас есть точное представление ожидаемого результата, основанное на изменениях входных характеристик;

- будь то упаковка, повышение или укладка, для понимания решения, принятого моделью, могут быть генерированы аналогичная структура PDP, график силы и сводный график. Однако диаграмма, показанная на графике PDP в качестве примера, будет меняться;
- изменения в графиках происходят потому, что модель пытается уловить сложные закономерности, если таковые имеются в данных. По мере увеличения сложности данных модель становится более сложной, и визуально она может выглядеть сложной для интерпретации. Однако выводы из значений остаются неизменными;
- следовательно, исходя из анализа, проведенного в этой главе, не имеет значения, сколько ансамблевых моделей вы охватываете. Важнее создать основу, которой можно следовать для рассмотрения других ансамблевых моделей.

ГЛАВА 6

Объяснимость для моделей временных рядов

Модель временного ряда – это способ создания многоступенчатого прогноза на будущий период времени. Существуют статистические модели и модели на основе машинного обучения, которые можно применить для создания прогнозов на будущее на основе исторических данных. Если прогнозам модели можно доверять или нет, какова степень уверенности в отношении прогнозов? В этой главе мы рассмотрим модели, которые могут быть объяснены, и модели, которые не могут быть объяснены.

Модели временных рядов

Основной целью модели временного ряда является оценка значения целевой переменной с использованием времени в качестве независимой переменной. Целевой переменной может быть значение цены акции, количество единиц продукции, сумма выручки, которая поступит на счет компании, или количество посетителей веб-сайта. Прогнозируемые значения являются многошаговыми, поскольку при использовании модели временного ряда мы обычно прогнозируем несколько временных шагов. Модель временного ряда генерирует прогнозные значения. Ожидаемые значения имеют определенные уровни доверия. Чем выше уровень доверия, тем лучше модель, при более низких уровнях доверия модели не хватает стабильности в генерировании ожидаемых значений. Доверительный интервал может быть рассчитан как ожидаемое значение плюс-минус 1.96 (стандартизированное значение из статистической таблицы, соответствующее 95%-ной доверительной вероятности), умноженное на стандартную ошибку остаточного члена, рассчитанного по модели. Это основано на условии, что ошибки являются нормально распределенными.

Модель временных рядов требует, чтобы данные регистрировались через частые временные интервалы без каких-либо перерывов во временном шаге. Данные временного ряда упорядочены по своей природе, поскольку порядок определяет неявную временную последовательность. Инженеру машинного обучения важно создать полезные характеристики данных, чтобы делать правильные прогнозы. В модели временного ряда время является независимой переменной. В одномерной модели временного ряда у вас есть только одна переменная. В модели причинно-следственного прогнозирования используется модель, подобная регрессионной. В модели одномерного временного

ряда характеристиками являются условия авторегрессии, такие как запаздывающие условия, условия скользящего среднего (например, трех- или пятипериодное скользящее среднее) и разностные. Наиболее популярные модели прогнозирования временных рядов полагаются исключительно на исторические значения целевой переменной. Это модели экспоненциального сглаживания и ARIMA (авторегрессионная интегрированная модель скользящего среднего – autoregressive integrated moving average model). Если вы собираетесь их использовать, вы должны зафиксировать компоненты временного ряда на шаге разработки характеристик. Это:

- **тенденция** – признак, когда значение переменной увеличивается, уменьшается или остается постоянным непрерывно в течение нескольких шагов в последовательности. При использовании моделирования временных рядов тенденция рассматривается как характеристика;
- **сезонность** – признак регулярных закономерностей, проявляющихся в определенном интервале, который является периодическим по своей природе. Она непрерывно повторяется в постоянном временном интервале. Модель временного ряда также учитывает сезонные переменные на этапе разработки характеристик;
- **цикличность.** Бизнес-циклы обычно колеблются во времени. Иногда они больше, а иногда меньше. Цикличность – это характеристика, которая помогает в прогнозировании значения целевой переменной.

Помимо компонентов временного ряда, могут быть введены и другие инженерные характеристики, например запаздывающие переменные, переменные на основе скользящего среднего, фиктивные, основанные на событиях, таких как праздники и специальные события, маркетинговые кампании и существование выбросов в данных. Объяснимость и интерпретируемость модели временного ряда необходимы для создания доверия к модели, объяснения прогнозов, а также понимания поведения моделей. Компоненты и характеристики временного ряда, а также дополнительные характеристики важны для интерпретации поведения модели временного ряда. На рис. 6.1 показано, что мы обычно моделируем в одномерной модели прогнозирования временных рядов.

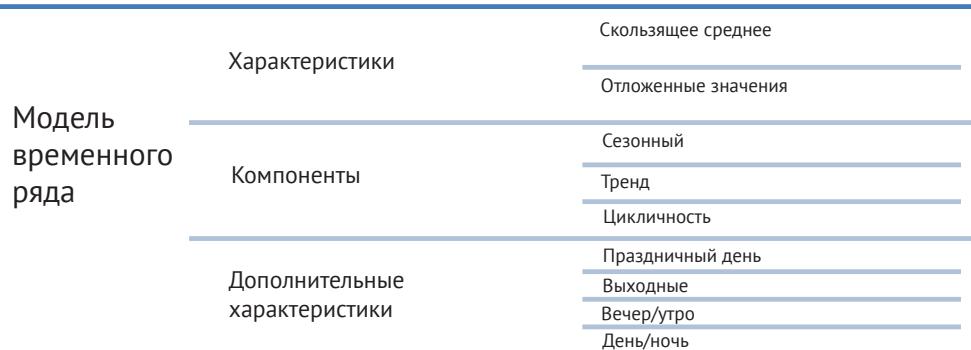


Рис. 6.1. Что обычно моделируют в модели прогнозирования временных рядов

Давайте рассмотрим некоторые случаи использования, где применима модель временного ряда, а также требуется ее объяснимость:

- данные, получаемые с датчиков IoT-устройств, установленных в различных местах;
- финансовые данные, такие как цены на акции;
- экономические показатели, такие как ВВП, FII, ИП, инфляция и т. д.;
- системные журналы и журналы ошибок компьютеров;
- прогнозирование спроса в отрасли цепочки поставок.

Для решения различных сценариев использования у нас есть список алгоритмов, которые могут генерировать прогнозы на будущее. Однако они не просты для понимания. Есть и сложные модели. Не все алгоритмы обладают интерпретируемостью и объяснимостью. Но важно объяснить модели бизнес-лидерам. Эти алгоритмы следующие:

- модель авторегрессионного интегрированного скользящего среднего;
- обобщенная авторегрессионная модель условной гетероскедастичности;
- байесова вероятностная модель;
- векторная авторегрессионная модель;
- нейросетевая авторегрессионная модель;
- модель рекуррентной нейронной сети;
- модель долгой краткосрочной памяти;
- модель управляемых рекуррентных нейронов.

ВЫБОР ПОДХОДЯЩЕЙ МОДЕЛИ

Перед использованием в производственном сценарии модели временных рядов должны быть оценены с помощью метрик и других методов диагностики.

Метрики:

- среднеквадратичная ошибка (root mean square error – RMSE) – это функция масштаба. Чем она меньше, тем лучше модель;
- средняя абсолютная процентная ошибка (mean absolute percentage error – MAPE) – этот показатель не зависит от масштаба. Предпочтительно, чтобы он был меньше 10 %, согласно отраслевому эталону.

После того как модель прогнозирования подобрана, важно знать метрики, чтобы определить, насколько хорошо модель соответствует данным. Понятие, характерное для модели временных рядов, – это белый шум, означающий ошибки, которые не могут быть объяснены моделью. Ошибка может считаться белым шумом, если выполняются следующие условия:

- остаточные члены не коррелированы, т. е. значение автокорреляционной функции (autocorrelated function – ACF) равно 0;
- остатки имеют нормальное распределение.

Если два вышеуказанных свойства не выполняются, то еще существуют возможности для улучшения модели.

СТРАТЕГИЯ ПРОГНОЗИРОВАНИЯ

Если бизнесу необходимо создание прогноза на высшем иерархическом уровне и доступны детализированные данные, то прогнозирование может быть построено двумя способами:

- макроуровнем с использованием высшей иерархии (пример – прогноз объема продаж для категории одежды осуществляется на макроуровне, а для каждой единицы хранения (stock-keeping unit – SKU) – на микроуровне);
- микроуровнем с последующим агрегированием с использованием минимально возможной детализации.

ДОВЕРИТЕЛЬНЫЙ ИНТЕРВАЛ ПРОГНОЗОВ

Не все модели временных рядов предоставляют доверительные интервалы для ожидаемых значений. Доверительный интервал прогнозируемых значений дает осмысленное понимание. Чем он выше, тем лучше модель. Если модели не соответствуют доверительному интервалу, возможны два сценария:

- **событие «черного лебедя»** – доверительный интервал прогнозируемого значения нарушается фактическим значением в большой степени, что приводит к хаосу;
- **событие «серого лебедя»** – доверительный интервал меньше ожидаемого значения и может привести к неоптимальному результату.

На рис. 6.2 показаны ежедневные продажи, и исторические данные используются для составления прогноза на будущее. Пунктирная линия отображает прогнозируемые значения, полученные с помощью модели, две параллельные черные линии показывают доверительный интервал для прогнозируемых значений, где верхний порог соответствует верхней линии, а нижний порог – нижней линии. Если прогнозируемые значения превышают верхний порог, как в точке А, это называется событием «черного лебедя», а если ниже нижнего порога в точке В, то это – событие «серого лебедя».

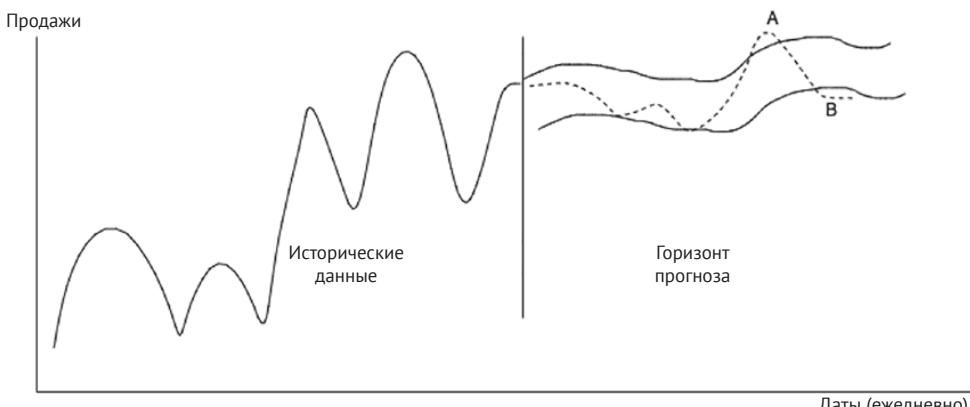


Рис. 6.2. События «черного лебедя» и «серого лебедя»

Что происходит с доверием?

Задача XAI – объяснить конечному пользователю, следует ли ему верить данному прогнозу или выбрать конкурирующие модели, которые могут быть более надежными. Приведенные выше два сценария, «черный лебедь» и «серый лебедь», чрезвычайно трудно спрогнозировать. Если они произойдут один или два раза, то все в порядке. Если модель не сможет часто идентифицировать такие сценарии, то у пользователя не будет доверия к модели. Одним из способов контроля таких событий является создание доверительного интервала при более высоком пороговом значении, например 90 %. Другие сценарии должны быть объяснены с помощью параметров модели.

В наборе данных есть отметка даты и цена некоторого товара. Это считается одномерной моделью временного ряда. Вам необходимо построить прогнозируемые значения на будущее.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
df = pd.read_csv('/Users/pradmishra/Downloads/XAI Book Apress/monthly_csv.csv',
                 index_col=0)
df
# линейный график временного ряда
# линейный график набора данных
df.plot()
```

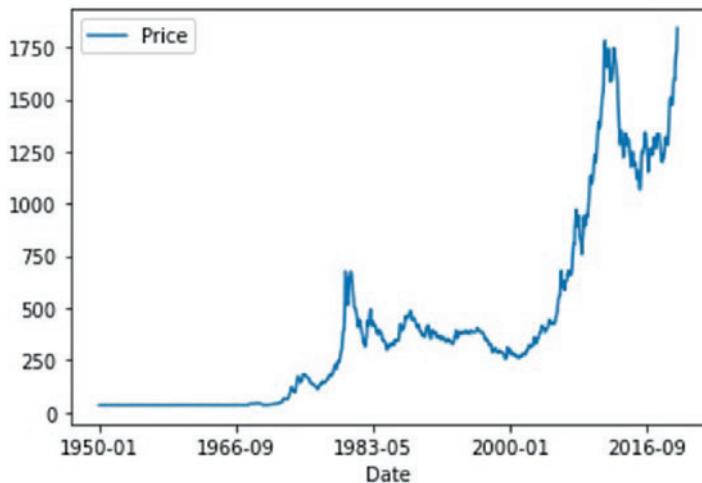


Рис. 6.3. Цена по датам

Данные на рис. 6.3 показывают колебания цен с тенденцией и скачком к концу 2016 года. В модели временного ряда существует некоторая степень сезонности. Сезонность означает, что одни и те же значения цены будут иметь место в определенный период времени и такие события будут повторяться.

Если этот показатель больше единицы, то можно подтвердить, что в данных существует сезонность. Чтобы включить сезонность в прогнозируемые значения, можно расширить модель временного ряда до вариантов модели с поправкой на сезонность. Но если вы хотите устраниТЬ влияние сезонных элементов из прогнозируемого значения, необходимо скорректировать временной ряд с помощью метода разности. Разность представляет собой временной ряд минус его собственные значения в прошлом – раньше на один период. Сезонно скорректированные данные могут храниться отдельно.

```
# сезонные различия
differenced = df.diff(12)
# обрезка первого года пустых данных
differenced = differenced[12:]
# сохранение дифференцированного набора данных в файл
differenced.to_csv('seasonally_adjusted.csv', index=False)
# plot differenced dataset
differenced.plot()
plt.show()
```

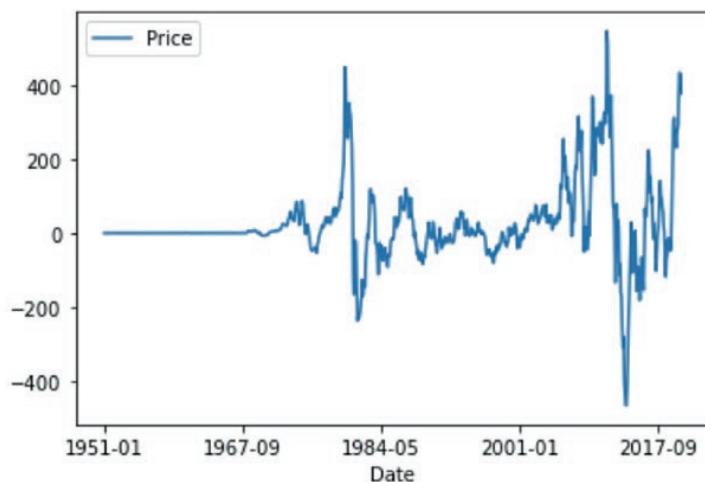


Рис. 6.4. Данные временного ряда с сезонной разницей

На рис. 6.4 представлен тот же временной ряд, но с сезонной корректировкой. Авторегрессионные интегрированные модели скользящего среднего (ARIMA) наиболее часто используются в промышленности. Они объясняют автокорреляции в данных. Стационарный временной ряд можно определить как ряд, значения которого не зависят от того, когда он фактически произошел. Любой временной ряд с тенденциями и сезонностью не является стационарным в отличие от временного ряда с цикличностью. Один из способов сделать его таковым – это применение метода дифференцирования, поскольку стационарность является одним из предположений модели ARIMA. Дифференцирование помогает уменьшить тенденции и сезонность в данных и, следовательно, сделать их стационарными. Автокорреляционная функция (АКФ) является

одним из способов определения стационарности; АКФ быстро падает до нуля. Однако для нестационарного временного ряда значение корреляции остается довольно высоким, а его спад очень незначителен.

```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(df)
plt.show()
```

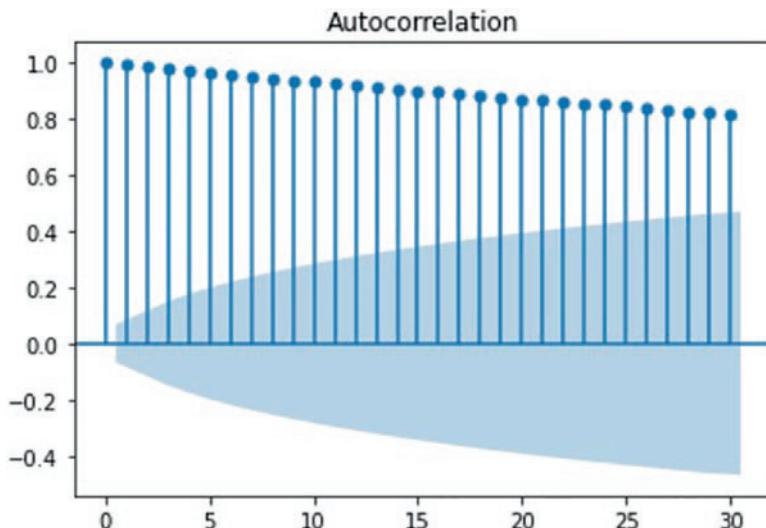


Рис. 6.5. Автокорреляционная функция цены

Вид автокорреляционной функции на рис. 6.5 говорит о том, что ряд является нестационарным, так как коэффициент корреляции до 30 лагов составляет более 0.80. Затененный цвет показывает уровень доверия к значению коэффициента корреляции. Поскольку коэффициент корреляции не равен нулю на уровне до 30 лагов, ряд является нестационарным по своей природе. Так как вы знаете, что ряд не является стационарным, вы не можете применить стандартную модель ARIMA.

Существует альтернативный способ – превратить проблему в задачу машинного обучения с авторегрессионным контролем, чтобы понять, влияет ли какой-либо лаг на прогнозируемое значение. Поэтому мы создаем 12 лаговых значений в качестве независимых характеристик для прогнозирования фактического ряда цен в качестве зависимой переменной.

```
# переосмыслить как обучение с учителем
dataframe = pd.DataFrame()
for i in range(12,0,-1):
    dataframe['t-' + str(i)] = df.shift(i).values[:,0]
dataframe['t'] = df.values[:,0]
print(dataframe.head(13))
dataframe = dataframe[13:]
# сохранение в новом файле
dataframe.to_csv('lags_12months_features.csv', index=False)
```

В приведенном выше скрипте мы рассчитываем лаговые значения. На рис. 6.6 они представлены в виде таблицы.

	t-12	t-11	t-10	t-9	t-8	t-7	t-6	t-5	t-4	t-3
0	NaN									
1	NaN									
2	NaN									
3	NaN	34.73								
4	NaN	34.73	34.73							
5	NaN	NaN	NaN	NaN	NaN	NaN	34.73	34.73	34.73	
6	NaN	NaN	NaN	NaN	NaN	34.73	34.73	34.73	34.73	
7	NaN	NaN	NaN	NaN	34.73	34.73	34.73	34.73	34.73	
8	NaN	NaN	NaN	34.73	34.73	34.73	34.73	34.73	34.73	
9	NaN	NaN	34.73	34.73	34.73	34.73	34.73	34.73	34.73	
10	NaN	34.73	34.73	34.73	34.73	34.73	34.73	34.73	34.73	
11	34.73	34.73	34.73	34.73	34.73	34.73	34.73	34.73	34.73	
12	34.73	34.73	34.73	34.73	34.73	34.73	34.73	34.73	34.73	
	t-2	t-1	t							
0	NaN	NaN	34.73							
1	NaN	34.73	34.73							
2	34.73	34.73	34.73							
3	34.73	34.73	34.73							
4	34.73	34.73	34.73							
5	34.73	34.73	34.73							
6	34.73	34.73	34.73							
7	34.73	34.73	34.73							
8	34.73	34.73	34.73							
9	34.73	34.73	34.73							
10	34.73	34.73	34.73							
11	34.73	34.73	34.72							
12	34.73	34.72	34.72							

Рис. 6.6. Обработанные данные после создания лаговых переменных для разработки модели⁵

12 значений лага являются характеристиками, и можно использовать регрессор случайного леса для составления различных комбинаций характеристик с целью построения дерева решений и ансамбля прогнозов модели, взяв среднее значение прогнозируемых значений из всех деревьев. Таким образом, можно определить, какая характеристика является более важной.

```
# разделение на вход и выход
df = pd.read_csv('lags_12months_features.csv')
data = df.values
X = data[:,0:-1]
```

⁵ NaN – (not a number) не число. В арифметике с плавающей запятой (точкой) NaN используется для представления исключительных случаев – значений выражений, которые невозможно представить в виде действительного (хотя бы и бесконечного) числа. При этом все биты экспоненты устанавливаются в 1. Примеры NaN: значение квадратного корня из отрицательного числа; сумма бесконечностей с разными знаками; результат попытки умножить бесконечность на ноль или поделить ноль на ноль; ситуация, когда переменная просто не была инициализирована. <https://translate.academic.ru/nan/en/ru/>. – Прим. перев.

```

y = data[:, -1]
from sklearn.ensemble import RandomForestRegressor
# подгонка модели случайного леса
model = RandomForestRegressor(n_estimators=500, random_state=1)
model.fit(X, y)
# показать оценки важности
print(model.feature_importances_)
# построить график оценки важности
names = dataframe.columns.values[0:-1]
ticks = [i for i in range(len(names))]
plt.bar(ticks, model.feature_importances_)
plt.xticks(ticks, names)
plt.show()

```

Мы рассмотрели 500 деревьев решений и использовали их характеристики для обучения регрессионной модели случайного леса. На рис. 6.7 оценивается важность характеристик.

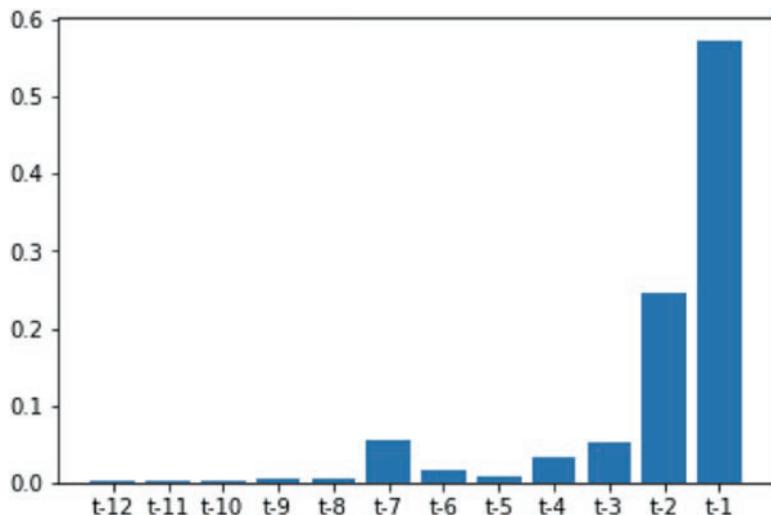


Рис. 6.7. Важность характеристик модели случайного леса

На рис. 6.7 видно, что лаг 1 ($t-1$) является наиболее важной характеристикой, лаг 2 ($t-2$) – второй по важности, лаг 7 ($t-7$) – третьей, за ним следуют лаг 3 и лаг 4, как показано с помощью ($t-3$) и ($t-4$) соответственно. Остальные характеристики не слишком важны для модели.

```

from sklearn.feature_selection import RFE
# отбор характеристик
rfe = RFE(RandomForestRegressor(n_estimators=500, random_state=1),
n_features_to_select=4)
fit = rfe.fit(X, y)

```

```
# отчет о выбранных характеристиках
print('Selected Features:')
names = dataframe.columns.values[0:-1]
for i in range(len(fit.support_)):
    if fit.support_[i]:
        print(names[i])
# построение графика ранжирования характеристик
names = dataframe.columns.values[0:-1]
ticks = [i for i in range(len(names))]
plt.bar(ticks, fit.ranking_)
plt.xticks(ticks, names)
plt.show()
```

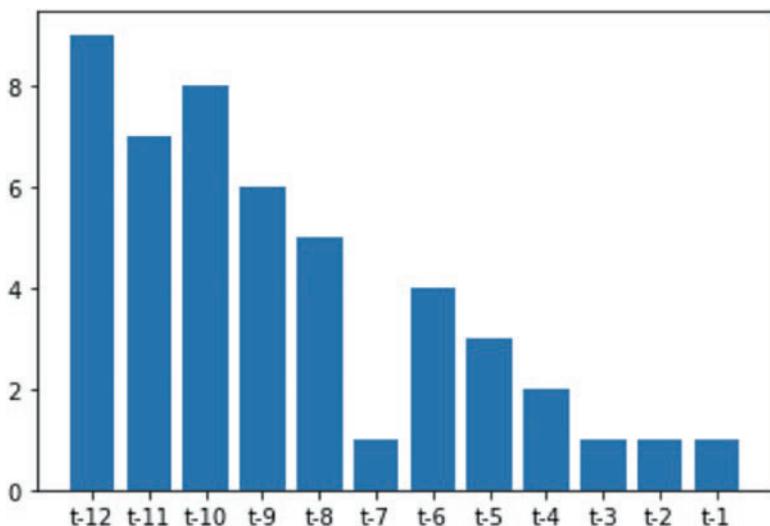


Рис. 6.8. Важность характеристик по алгоритму RFE

Рекурсивное исключение характеристик (recursive feature elimination – RFE) – это популярный алгоритм для удаления избыточных из списка характеристик в среде контролируемого обучения (рис. 6.8). Существует два варианта настройки RFE – либо количество выбираемых характеристик, либо выбор алгоритма, используемого для выбора характеристик. RFE действует как алгоритм «обертки» поверх контролируемой модели на основе регрессии или модели на основе классификации. Это означает, что вам нужна модель, а затем вы можете применить RFE поверх нее. В приведенном выше сценарии используется RFE поверх регрессионной модели случайного леса. Наиболее важными выбранными характеристиками являются лаги 7, 3, 2 и 1. Остальные восемь характеристик не выбираются алгоритмом. Следовательно, непрерывные и последовательные лаги не оказывают влияния на целевую характеристику, однако выбранные характеристики влияют на результат прогнозирования. Характеристиками модели временного ряда, как обсуждалось ранее в этой главе, являются только лаговые переменные и переменные скользящего среднего. Поэтому объект временного

ряда может быть смоделирован как авторегрессионный (AR), если используются только лаговые переменные. Он может быть только скользящим средним (MA), если в качестве характеристик используются только члены скользящего среднего. Авторегрессия означает, что лаговые значения могут быть использованы в качестве характеристик для прогнозирования фактического временного ряда.

```
# пример авторегрессии
from statsmodels.tsa.ar_model import AutoReg
from random import random
# подгонка модели
model = AutoReg(y, lags=1)
model_fit = model.fit()
model_fit.summary()
```

Предыдущий скрипт показывает, что если вы используете только лаг 1 для прогнозирования текущего периода времени, то это называется моделью AR 1. Это похоже на линейную регрессию (см. табл. 6.1).

Таблица 6.1. Результаты авторегрессионной модели

Зависимая переменная:	y	Число наблюдений:	834
Модель:	AutoReg(1)	Логарифмическая вероятность	-3864.852
Метод:	Условный метод максимального правдоподобия	S.D. of innovations	25.047
Дата:	Пн, 1 марта 2021	ИИС	6.449
Время:	23:08:04	BIC	6.466
Экземпляр:	1	HQIC	6.455
	834		
		Коэффициент	Стандартная ошибка
член перехвата	0.5136	1.186	0.433
y.L1	1.0039	0.002	522.886
Корни			
		Действительная часть	Мнимая часть
AR.1	0.9961	+0.0000j	0.9961
			Модуль
			Частота

Вы можете записать уравнение в виде:

$$Y_t = 0.5136 + 1.0039 * Y_{t-1}.$$

Коэффициент 1.0039 имеет значение р 0.000, что меньше 0.05. Это означает, что при доверительной вероятности 95 % и статистическом уровне значимости 5 % первый член лага является достаточно значимым, чтобы повлиять на фактический временной ряд Y_t . Это можно интерпретировать так: если лаговый член изменится на одну единицу, фактическое значение ряда по прогнозу изменится на 0.39 %.

```
# пример авторегрессии
from statsmodels.tsa.ar_model import AutoReg
from random import random
# подгонка модели
model = AutoReg(y, lags=2)
model_fit = model.fit()
model_fit.summary()
```

Аналогичный анализ можно провести, принимая во внимание два лага, и получить уравнение:

$$Y_t = 0.6333 + 1.2182 * Y_{t-1} - 0.2156 * Y_{t-2}.$$

Коэффициенты 1.2182 и 0.2156 имеют значение р 0.000 что меньше 0.05. Это означает 95%-ный доверительный уровень и 5%-ный статистический уровень значимости. Для первого и второго лагов это достаточно значимо, чтобы повлиять на фактический временной ряд Y_t . Это можно интерпретировать так: если член лага изменится на одну единицу, то фактическое значение ряда по прогнозам изменится на 0.22 %. Коэффициент лага 2 можно интерпретировать так: если лаг 2 изменится на одну единицу, то серия Y_t уменьшится в 0.2156 раза (см. табл. 6.2).

Таблица 6.2. Результаты модели авторегрессии с двумя лагами

Зависимая переменная:	y	Число наблюдений:	834
Модель:	AutoReg(2)	Логарифмическая вероятность	-3841.483
Метод:	Условный метод максимального правдоподобия	S.D. of innovations	24.489
Дата:	Пн, 1 марта 2021	ИИС	6.406
Время:	23:08:14	BIC	6.429
Экземпляр:	2	HQIC	6.415
	834		
		Коэффициент	Стандартная ошибка
член перехвата	0.6333	1.161	0.0546
y.L1	1.2182	0.034	35.614
y.L2	-0.2156	0.034	-6.274
Корни		z	P> z
		[0.025	0.975]
		-1.642	2.908
		1.151	1.285
		-0.283	-0.148

	Действительная часть	Мнимая часть	Модуль	Частота
AR.1	0.9967	+0.0000j	0.9967	0.0000
AR.2	4.6536	+0.0000j	4.6536	0.0000

```
# пример авторегрессии
from statsmodels.tsa.ar_model import AutoReg
from random import random
# подгонка модели
model = AutoReg(y, lags=12)
model_fit = model.fit()
model_fit.summary()
```

Подобно описанному выше сценарию с двумя лагами, можно расширить модель для сценариев с 12 лагами и сделать аналогичную интерпретацию (см. табл. 6.3 и 6.4).

Таблица 6.3. Результат модели авторегрессии с 12 лаговыми переменными

Зависимая переменная:	y	Число наблюдений:	834
Модель:	AutoReg(12)	Логарифмическая вероятность	-3776.103
Метод:	Условный метод максимального правдоподобия	S.D. of innovations	23.923
Дата:	Пн, 1 марта 2021	AIC	6.384
Время:	23:08:22	BIC	6.464
Экземпляр:	12	HQIC	6.415
	834		

Таблица 6.4. Таблица коэффициентов для вышеуказанной модели

	Коэффициент	Стандартная ошибка	z	P> z	[0.025	0.975]
Член перехвата	0.7456	1.147	0.650	0.516	-1.503	2.994
y.L1	1.2473	0.035	35.827	0.000	1.179	1.316
y.L2	-0.3653	0.056	-6.569	0.000	-0.474	-0.256
y.L3	0.1999	0.057	3.504	0.000	0.088	0.312
y.L4	-0.1471	0.057	-2.558	0.011	-0.260	-0.034
y.L5	0.2303	0.058	3.981	0.000	0.117	0.344
y.L6	-0.1920	0.058	-3.290	0.001	-0.306	-0.078
y.L7	0.0802	0.058	1.373	0.170	-0.034	0.195
y.L8	-0.1004	0.058	-1.730	0.084	-0.214	0.013
y.L9	0.0705	0.058	1.215	0.224	-0.043	0.184
y.L10	-0.0587	0.058	-1.016	0.309	-0.172	0.054
y.L11	0.1843	0.056	3.270	0.001	0.074	0.295
y.L12	-0.1475	0.035	-4.155	0.000	-0.217	-0.078

При выборе модели ARIMA необходимо определить порядок модели. В следующем скрипте порядок определяется как (0,0,1), что означает (p,d,q), где p – член авторегрессии, d – дифференциация, а q – скользящее среднее. Таким образом, порядок 0, 0 и 1 означает модель скользящего среднего.

```
# пример скользящего среднего
from statsmodels.tsa.arima.model import ARIMA
from random import random
# подгонка модели
model = ARIMA(y, order=(0, 0, 1))
model_fit = model.fit()
# создание прогноза
yhat = model_fit.predict(len(y), len(y))
print(yhat)
model_fit.summary()
```

Таблица 6.5. Результаты сезонной модели ARIMA

Зависимая переменная:	y	Число наблюдений:	834
Модель:	ARIMA(0, 0, 1)	Логарифмическая вероятность	-5736.156
Дата:	Пн, 1 марта 2021	ИИС	11478.312
Время:	23:09:14	BIC	11492.490
Экземпляр:	0	HQIC	11483.748
	-834		
Тип ковариации:	opg		
Коэффициент	Стандартная ошибка	z	P> z
const	422.8527	28.368	14.906 0.000
ma.L1	0.9997	0.007	134.053 0.000
sigma2	5.476e+04	3750.622	14.601 0.000
Ljung-Box (L1) (Q):	668.38	Jerque-Bera (JB):	333.81
Вероятность(Q):	0.00	Вероятность (JB):	0.00
Гетероскедастичность (H):	3.33	Асимметрия:	1.51
Вероятность (H) (двусторонняя):	0.00	Эксцесс:	4.40

Следующий скрипт показывает результаты ARIMA порядка 2, 0, 1 (2 лага авторегрессии, 0 дифференциации и 1 член скользящего среднего)

```
# пример ARMA
from statsmodels.tsa.arima.model import ARIMA
from random import random
# подгонка модели
model = ARIMA(y, order=(2, 0, 1))
model_fit = model.fit()
# создание прогноза
yhat = model_fit.predict(len(y), len(y))
print(yhat)
model_fit.summary()
```

Таблица 6.6. Результаты модели SARIMAX, коэффициенты и уровень их статистической значимости

Зависимая переменная:	y	Число наблюдений:	834			
Модель:	ARIMA(2, 0, 1)	Логарифмическая вероятность	-3849.432			
Дата:	Пн, 1 марта 2021	ИИС	7708.864			
Время:	23:11:45	BIC	7732.495			
Экземпляр:	0	HQIC	7717.924			
	-834					
Тип ковариации:	opg					
Коэффициент	Стандартная ошибка	z	P> z	[0.025	0.975]	
const	422.5370	3593.885	0.118	0.906	-6621.347	7466.421
ar.L1	0.4603	0.043	10.759	0.000	0.376	0.544
ar.L2	0.5390	0.043	12.585	0.000	0.455	0.623
ma.L1	0.7660	0.032	23.996	0.000	0.703	0.829
sigma2	592.5712	11.593	51.113	0.000	569.849	615.294
Ljung-Box (L1) (Q):	0.93	Jerque-Bera (JB):	5757.42			
Вероятность (Q):	0.34	Вероятность (JB):	0.00			
Гетероскедастичность (H):	178.08	Асимметрия:	1.17			
Вероятность (H) (двусторонняя):	0.00	Эксцесс:	15.66			

Сводные результаты всех моделей можно интерпретировать подобно тому, как это было сделано выше. Если вы захотите ввести сезонные компоненты в модель ARIMA, то она превратится в модель SARIMA. Порядок сезонов также быть определен в соответствии с табл. 6.7.

Таблица 6.7. Параметры модели SARIMA и их объяснение

Параметры	Объяснение
Endog	Наблюдаемый процесс временного ряда y
Order	Порядок (p,d,q) модели по количеству параметров AR, дифференциации и параметров MA
seasonal_order	Порядок (p,d,q) сезонной составляющей модели для параметров AR, дифференциации, параметров MA и периодичности
Trend	Параметр, управляющий полиномиальным детерминированным трендом $A(t)$
enforce_stationarity	Нужно ли преобразовывать параметры AR для обеспечения стационарности в авторегрессионном компоненте модели. По умолчанию равно <code>true</code>

```
# пример ARMA
from statsmodels.tsa.arima.model import ARIMA
from random import random
# подгонка модели
model = ARIMA(y, order=(2, 1, 1))
model_fit = model.fit()
# создание прогноза
yhat = model_fit.predict(len(y), len(y))
print(yhat)
model_fit.summary()

# пример SARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from random import random
# подгонка модели
model = SARIMAX(y, order=(1, 1, 1), seasonal_order=(0, 0, 0, 0))
model_fit = model.fit(disp=False)
# создание прогноза
yhat = model_fit.predict(len(y), len(y))
print(yhat)
model_fit.summary()
```

Таблица 6.8. Результаты модели SARIMA

Зависимая переменная:	y	Число наблюдений:	834
Модель:	SARIMAX(1, 1, 1)	Логарифмическая вероятность	-3841.210
Дата:	Пн, 1 марта 2021	ИИС	7688.419
Время:	23:13:53	BIC	7702.594
Экземпляр:	0	HQIC	7693.854
	-834		
Тип ковариации:	opg		
Коэффициент	Стандартная ошибка	z	P> z
ar.L1	-0.5398	0.043	-12.639 0.000
ma.L1	0.7664	0.032	24.047 0.000
sigma2	592.5977	11.206	52.880 0.000
Ljung-Box (L1) (Q):	0.87	Jerque-Bera (JB):	5724.72
Вероятность(Q):	0.35	Вероятность (JB):	0.00
Гетероскедастичность (H):	184.64	Асимметрия:	1.13
Вероятность (H) (двусторонняя):	0.00	Эксцесс:	15.64

Значения коэффициентов ρ из табл. 6.8 являются статистически значимыми на уровне значимости 5 %, так как они меньше 0.05. Результат модели, полученный выше, использует в основном OLS в качестве метода измерения, а метод OLS очень похож на модель множественной линейной регрессии. Следовательно, коэффициенты модели, их значимость и интерпретация могут быть легко получены.

ВРЕМЕННЫЕ РЯДЫ: LIME

Используя объяснимую библиотеку LIME, можно взять 12 лагов в качестве характеристик, обучить регрессионную модель и объяснить прогнозы, как показано ниже. В качестве первого шага вам необходимо установить библиотеку LIME, если ее еще нет:

```

explainer.feature_frequencies
# запрос объяснения модели LIME
i = 60
exp = explainer.explain_instance(np.array(X)[i],
                                  new_model.predict,
                                  num_features=12
)
exp.show_in_notebook(show_table=True)

```

В приведенном выше скрипте мы рассматриваем модель временного ряда как модель контролируемого обучения и используем 12 лагов в качестве характеристик. Из библиотеки LIME мы берем табличный объяснитель (tabular explainer). На рис. 6.9 показано объяснение записи № 60.

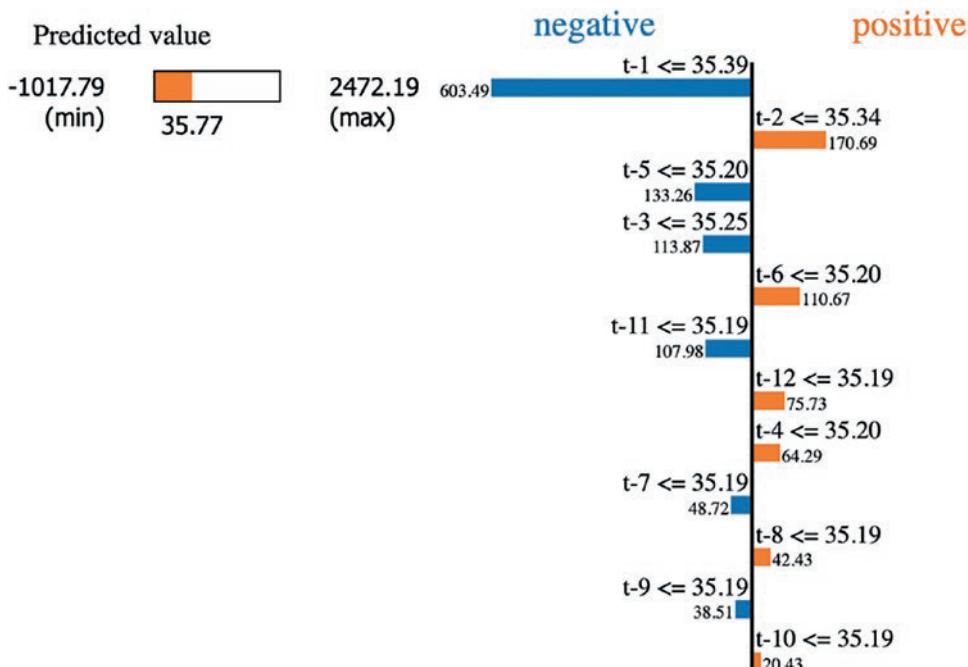


Рис. 6.9. Объяснение LIME для записи 60

Ожидаемое значение равно 35.77, а нижнее и верхнее пороговые значения отражают доверительный интервал результата прогноза. Положительные и отрицательные факторы, влияющие на прогноз, показаны на рис. 6.9. Лаг 1 является наиболее важной характеристикой, второй по важности является лаг 2, затем лаг 5, лаг 3, лаг 6 и т. д. Лаговые значения как характеристики и их вклад в прогнозируемое значение показаны на рис. 6.10.

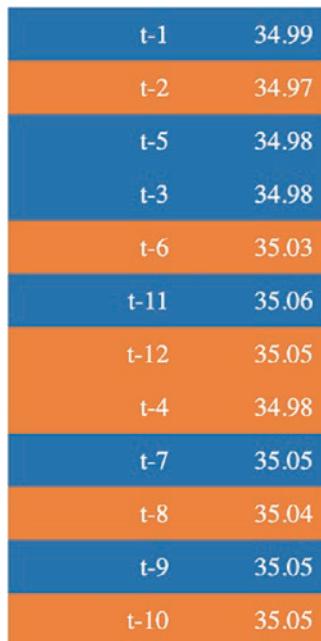


Рис. 6.10. Важность характеристик для модели

```
# Код для SP-LIME
import warnings
from lime import submodular_pick

# Не забудьте преобразовать датафрейм в матричные значения
# SP-LIME возвращает объяснения набора образцов, чтобы обеспечить не избыточную
# глобальную
# границу принятия решений исходной модели
sp_obj = submodular_pick.SubmodularPick(explainer, np.array(X),
                                            new_model.predict,
                                            num_features=12,
                                            num_exps_desired=10)

import matplotlib.pyplot as plt
plt.savefig('[exp.as_pyplot_figure() for exp in sp_obj.sp_explanations
].png', dpi=300)
images = [exp.as_pyplot_figure() for exp in sp_obj.sp_explanations ]
exp.predicted_value
```

Субмодульная выборка для объяснения моделей обеспечивает глобальное понимание модели по набору экземпляров из данных. В приведенном выше скрипте рассматриваются все 12 лаговых характеристик, 10 экземпляров для просмотра пользователем или показа объяснений.

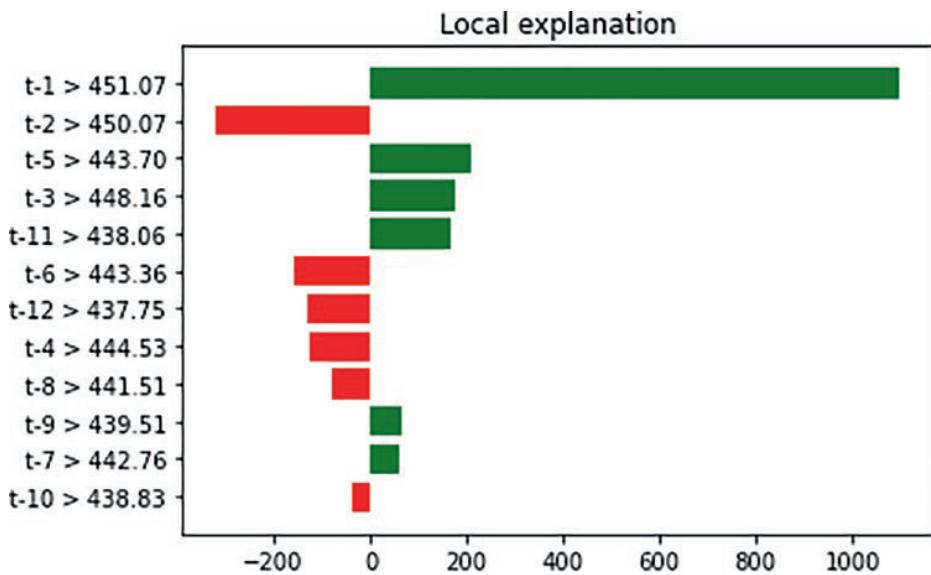


Рис. 6.11. Локальное объяснение записи 0

Лаг 1 на рис. 6.11 очень важен для прогнозирования ожидаемого значения. Зеленые столбики улучшают предсказание, а красные уменьшают предсказанное значение. Это локальное объяснение для первой записи. Аналогичным образом можно рассмотреть другие записи и генерировать подобные объяснения.

ЗАКЛЮЧЕНИЕ

В этой главе вы узнали, как интерпретировать модель временного ряда. Вы рассмотрели модель временного ряда как модель контролируемого обучения и интерпретировали значение признаков важности для всех характеристик. Вы изучили модели авторегрессии, модели скользящего среднего и интегрированные модели скользящего среднего. Используя LIME в качестве библиотеки объяснимости, вы освоили важность лагов в прогнозировании целевого значения. Кроме того, изучили характеристики положительных и отрицательных лагов, которые вносят вклад в ожидаемый результат.

ГЛАВА 7

Объяснимость для NLP

В этой главе объясняется использование библиотек Python ELI5 и SHAP в таких задачах обработки естественного языка (natural language processing – NLP), как модели классификации текстов. Прогнозные решения, принимаемые моделями в задачах контролируемого машинного обучения, относятся к неструктурированным данным. Классификация текста – это задача, в которой необходимо рассматривать текстовые предложения или фразы в качестве входных данных и классифицировать их по дискретным категориям. Примером может служить классификация новостей, где в качестве входных данных используется контент, а выходные классифицируются на политику, бизнес, спорт, технологии и т. д. Аналогичным примером является обнаружение спама в классификации электронной почты, где содержимое почты используется в качестве входных данных и классифицируется на спам или не спам. В этом сценарии важно знать: если письмо классифицируется как спам, то почему? Какие маркеры, присутствующие в содержимом, действительно приводят к такому прогнозу? Это представляет интерес для конечного пользователя. Когда я говорю о задачах NLP, их множество, но я ограничусь классификацией текста и другими подобными задачами, например распознаванием сущностей, пометками частей речи и анализом настроения.

ЗАДАЧИ ОБРАБОТКИ ЕСТЕСТВЕННОГО ЯЗЫКА

Сегодня весь мир связан между собой Всемирной паутиной (World Wide Web). Неструктурированные текстовые данные встречаются повсюду. Они находятся в социальных сетях, в разговорах по электронной почте, в чатах через различные приложения, на HTML-страницах, в текстовых документах, в службах поддержки клиентов, ответах на онлайн-опросы и многом другом. Ниже перечислены некоторые варианты использования неструктурированных данных:

- классификация документов, когда на входе находится текстовый документ, а выходом может быть бинарный класс или многоклассовый ярлык;
- иногда если настроения помечены, то они также соответствуют сценарию классификации документов. В противном случае классификация настроений будет классификацией на основе лексикона;
- модели распознавания именованных сущностей (named entity recognition – NER), где на входе находится текстовый документ, а на выходе – класс именованных сущностей;
- резюме текста, когда большой текст сжат и представлен в компактной форме.

В классификации текстов NLP обычными являются модели NER и предсказание следующего слова, где вход – это предложение или векторы слов, а выход – метка, которую необходимо классифицировать или предсказать. До наступления эры машинного обучения задача классификации текста решалась вручную, когда группа аннотаторов читала, понимала смысл текста, выраженного в документе, и относила его к определенному классу. С масштабным ростом объема неструктурированных данных ручной способ классификации стал очень сложным. Теперь некоторые аннотированные данные могут быть введены в компьютер и применен алгоритм обучения, чтобы обученную модель можно было в будущем использовать для прогнозирования.

Объяснимость для классификации текстов

Неструктурированные документы или входные текстовые векторы очень многомерны. Прогностические модели, используемые для классификации документов, должны быть объяснены, потому что причины, лежащие в основе прогноза, или особенности, лежащие в основе предсказанного класса, должны быть показаны конечному пользователю. В случае классификации текста модель предсказывает класс 1 против класса 2, поэтому важно знать ключевые слова, которые являются положительными и отрицательными для класса 1. В многоклассовой классификации это становится более сложным, поскольку вам необходимо объяснить все ключевые слова, приводящие к предсказанию определенного класса. В этой главе мы рассмотрим оба сценария.

На рис. 7.1 показаны шаги, необходимые для использования XAI в моделях классификации текста.



Рис. 7.1. Шаги в объяснении текста, где используется модель ИИ

Входные тексты представляются в виде вектора, а матрица терминов документа используется для разработки модели классификации, и, наконец, предсказанный класс должен быть объяснен с помощью библиотеки объяснимости, чтобы узнать особенности или слова, которые ответственны за прогнозирование метки конкретного класса.

Набор данных для классификации текста

В этой главе вы будете использовать большой набор данных отзывов о фильмах (Large Movie Review Dataset) с сайта <http://ai.stanford.edu/~amaas/data/sentiment/>.

Этот набор данных содержит 50 000 отзывов, и 25 000 уже помечены бинарным классификационным тегом, например положительный или отрицательный отзыв. Еще 25 000 отзывов предназначены для тестирования. Вы собираетесь использовать этот набор данных для создания модели классификации текста с целью прогнозирования отзывов о фильмах. После прогнозирования проанализируете причины прогнозов и объясните важность особенностей и других элементов объяснения этой модели машинного обучения для классификации текстов. В этой задаче вы предоставляете модели машинного обучения предложения отзыва, и модель предсказывает положительную или отрицательную эмоцию. Следовательно, модель ML рассматривается здесь как модель «черного ящика». В связи с этим возникает вопрос: можно ли доверять результатам, полученным с помощью модели «черного ящика»?

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
from keras.datasets import imdb
imdb = pd.read_csv('IMDB Dataset.csv')
imdb.head()
```

Набор данных фильмов IMDB содержит отзывы и настроения. Настроения помечаются как положительные и отрицательные, что представляет собой бинарную систему меток классов. Отзывы проходят очистку, удаляются стоп-слова, и текст преобразуется в векторы. Следующий скрипт показывает разбиение набора данных на обучающий и тестовый наборы с помощью функции `train_test_split` из библиотеки `sklearn`:

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegressionCV
from sklearn.pipeline import make_pipeline
vec = CountVectorizer()
clf = LogisticRegressionCV()
clf.fit(vec,imdb.sentiment)
```

Чтобы разделить данные на обучающий и тестовый наборы, мы используем функцию `train_test_split` из библиотеки `sklearn`. Векторизатор подсчета преобразует лексемы и представляет их в виде векторов, которые в свою очередь являются матричным представлением терминов документа. Вектор подсчета инициализируется и сохраняется в `vec`, а затем инициализируется модель логистической регрессии с перекрестной валидацией для создания обучающей модели. Инициализированный объект хранится в `clf`. Последним шагом является обучение модели с помощью метода `fit`.

Конвейер – это метод, в котором все шаги предварительной обработки, обучения модели и проверки могут быть последовательны и запущены один раз. В этом конвейере вы преобразуете текстовые данные из слов в вектор подсчета, а затем пропускаете их через классификатор. Текстовые предложения сначала преобразуются в таблицу структурированных данных. Это происходит с ис-

пользованием подсчета количества слов, присутствующих в каждом предложении. В приведенном выше примере обзоры рассматриваются как документы. Каждый документ разбивается на слова. Уникальный список слов собирается из всех отзывов, а затем каждое слово рассматривается как характеристика. Если взять количество каждого слова в различных обзорах, это называется векторизатором подсчета. Поскольку это набор данных для анализа настроения отзывов о фильмах, он классифицируется на положительный и отрицательный класс, что приводит к задаче бинарной классификации. Конвейер сохраняет последовательность преобразований, шагов обучения и выполняется вызовом метода `fit` из библиотеки `sklearn`.

```
from sklearn import metrics

def print_report(pipe):
    y_test = imdb.sentiment
    y_pred = pipe.predict(imdb.review)
    report = metrics.classification_report(y_test, y_pred)
    print(report)
    print("accuracy: {:.3f}".format(metrics.accuracy_score(y_test,
        y_pred)))
print_report(pipe)
```

В приведенном выше фрагменте скрипта вы предсказываете настроения на тестовом наборе данных и вычисляете метрику ошибки для точности классификации. В данном примере точность классификации составляет 95 %, что является очень хорошей точностью.

Объяснение с помощью ELI5

Чтобы объяснить модель классификации текста, вам необходимо установить библиотеку ELI5 для объясняемого ИИ (XAI). Она известна под названием «*объясни, как будто мне 5*» (explain it like I am 5). Эту библиотеку можно установить либо с помощью `pip install`, либо с помощью `conda install` из дистрибутива Anaconda.

```
!pip install eli5
import eli5
eli5.show_weights(clf, top=10) # этот результат не имеет смысла, так как вес
# и имена характеристик отсутствуют
```

После установки библиотеки вы можете написать оператор импорта для вызова библиотеки.

```
eli5.show_weights(clf, feature_names=vec.get_feature_names(),
target_names=set(imdb.sentiment))
#имеет смысл
```

Результаты, полученные с помощью модуля ELI5 для базового линейного классификатора, не имеют смысла. Он предоставляет веса и характеристики.

Имена характеристик не имеют смысла. Чтобы сделать их осмысленными, вы можете задать имена характеристик и целей.

y=positive top features

Weight?	Feature
+0.875	excellent
+0.753	refreshing
+0.733	perfect
+0.716	superb
... 51892 more positive ...	
... 49984 more negative ...	
-0.713	lacks
-0.718	poor
-0.726	forgettable
-0.728	laughable
-0.757	lame
-0.764	horrible
-0.771	dull
-0.775	fails
-0.793	disappointing
-0.813	terrible
-0.858	poorly
-0.889	boring
-0.970	disappointment
-1.071	awful
-1.334	waste
-1.375	worst

Рис. 7.2. Характеристики, полученные из модели, для класса положительных настроений

Характеристики на рис. 7.2, выделенные зеленым цветом, являются положительными для целевого класса «положительный» (positive), а красным выделены отрицательные характеристики, которые поддерживают другой класс. Вес характеристики вместе со значением веса класса указывает на ее относительную важность при отнесении настроений к определенному классу.

ВЫЧИСЛЕНИЕ ВЕСОВ ХАРАКТЕРИСТИК ДЛЯ ЛОКАЛЬНОГО ОБЪЯСНЕНИЯ

Весовые коэффициенты характеристик рассчитываются с помощью пути принятия решений. Путь принятия решений представляет собой последовательность операторов «если/тогда/иначе» (if/else/then), которые связывают прогнозируемый класс от корня дерева с ветвями дерева. В приведенном выше примере логистическая регрессия используется в качестве модели для обучения классификатора настроений, поэтому веса являются коэффициентами модели логистической регрессии. Для сложной модели, такой как случайный лес или градиентная модель повышения, используется путь принятия решений дерева. По сути, веса – это параметры оценщика, которые используются на этапе обучения модели.

ЛОКАЛЬНОЕ ОБЪЯСНЕНИЕ. ПРИМЕР 1

```
eli5.show_prediction(clf, imdb.review[0], vec=vec,
                     target_names=set(imdb.sentiment)) # объяснение локального
                                         # прогноза
```

y=positive (probability 0.845, score 1.698) top features

Contribution?	Feature
+1.715	Highlighted in text (sum)
-0.016	<BIAS>

one of the other reviewers has mentioned that after watching just 1 oz episode you'll be hooked. they are right, as this is exactly what happened with me.

the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint hearted or timid. this show pulls no punches with regards to drugs, sex or violence. its is hardcore, in the classic use of the word.

it is called oz as that is the nickname given to the oswald maximum security state penitentiary. it focuses mainly on emerald city, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda. em city is home to many..aryans, muslims, gangstas, latinos, christians, italians, irish and more....so scuffles, death stares, dodgy dealings and shady agreements are never far away.

i would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. forget pretty pictures painted for mainstream audiences, forget charm, forget romance...oz doesn't mess around. the first episode i ever saw struck me as so nasty it was surreal, i couldn't say i was ready for it, but as i watched more, i developed a taste for oz, and got accustomed to the high levels of graphic violence. not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) watching oz, you may become comfortable with what is uncomfortable viewing....thats if you can get in touch with your darker side.

Рис. 7.3. Выделение важных характеристик предложений для большей ясности

Предвзятость (BIAS) – это член перехвата из классификатора логистической регрессии. Мы можем использовать векторизатор подсчета и линейный классификатор, следовательно, сопоставление вполне очевидно, так как вес каждого слова соответствует коэффициентам линейного классификатора. Чтобы объяснить один отзыв о фильме, что называется локальной интерпретацией, можно пропустить его через функцию показа прогноза. Поскольку это бинарный классификатор, можно рассчитать логарифмическую вероятность (log odds). Она равна 1.698. Если вы используете формулу $\exp(\text{log odds})/1+\exp(\text{log odds})$, то получите значение вероятности 0.845. Существует 84.5 % вероятности того, что отзыв[0] относится к положительному настроению. Выделенный зеленым цветом текст на рис. 7.3 приводит к положительной log odds, а текст красного цвета уменьшает log odds. Общий вклад представляет собой сумму вкладов каждого текста обоих цветов, а результат – это окончательное значение log odds.

ЛОКАЛЬНОЕ ОБЪЯСНЕНИЕ. ПРИМЕР 2

Аналогичным образом можно объяснить отзывы № 123 и 100. Отзыв 123 классифицируется как отрицательный, и вероятность для отрицательного класса составляет 0.980. Отзыв № 100 классифицирован как положительный с вероятностью 0.670 (см. рис. 7.4).

```
eli5.show_prediction(clf, imdb.review[123], vec=vec,
                     target_names=set(imdb.sentiment)) # объяснение локального прогноза
```

y=negative (probability 0.980, score -3.877) top features

Contribution?	Feature
+3.861	Highlighted in text (sum)
+0.016	<BIAS>

ah yes the 1980s , a time of reaganomics and sly , chuck and a host of other action stars hiding in a remote jungle blowing away commies . at the time i couldn't believe how movies like rambo , missing in action and uncommon valor (and who can forget the ridiculous red dawn ?) made money at the box office , they're turgid action crap fests with a rather off putting right wing agenda and they have dated very badly . troma's war is a tongue in cheek take on these type of movies but you've got to ask yourself did they need spoofing in the first place ? of course not . troma's war lacks any sort of sophistication - though it does make the point that there's no real difference between right wing tyrants and left wing ones - and sometimes feels more like a grade z movie than a send up . maybe it is ?

Рис. 7.4. Локальное объяснение для прогноза отрицательного класса

ЛОКАЛЬНОЕ ОБЪЯСНЕНИЕ. ПРИМЕР 3

В примере 1 мы рассмотрели первую запись, где вероятность положительного класса составляет более 80 %. В примере 2 вероятность отрицательного класса составляет 98 %. Слова, выделенные зеленым цветом, вносят вклад в отрицательный класс прогноза (см. рис. 7.5).

```
eli5.show_prediction(clf, imdb.review[100], vec=vec,
                     target_names=set(imdb.sentiment)) # объяснение локального прогноза
```

y=positive (probability 0.670, score 0.709) top features

Contribution?	Feature
+0.725	Highlighted in text (sum)
-0.016	<BIAS>

this short film that inspired the soon-to-be full length feature - spatula madness - is a hilarious piece that contends against similar cartoons yielding multiple writers. the short film stars edward the spatula who after being fired from his job, joins in the fight against the evil spoons. this premise allows for some funny content near the beginning, but is barely present for the remainder of the feature. this film's 15-minute running time is absorbed by some odd-ball comedy and a small musical number. unfortunately not much else lies below it. the plot that is set up doesn't really have time to show. but it's surely follows it plot better than many high-budget hollywood films. this film is worth watching at least a few times. take it for what it is, and don't expect a deep story.

Рис. 7.5. Выделение характеристик для положительных и отрицательных классов

ОБЪЯСНЕНИЕ ПОСЛЕ УДАЛЕНИЯ СТОП-СЛОВ

Приведенный выше анализ выполнен при сохранении стоп-слов, чтобы сохранить контекст вводимого текста. Если удалить отсюда стоп-слова, то из входного вектора будет удалено несколько избыточных характеристик. Вы сможете получить для объяснения более значимые характеристики. Предвзятость – это член перехвата в модели. В приведенном выше скрипте мы рассмотрели пакет слов без удаления стоп-слов.

```
vec = CountVectorizer(stop_words='english')
clf = LogisticRegressionCV()
pipe = make_pipeline(vec, clf)
pipe.fit(imdb.review, imdb.sentiment)
print_report(pipe)
```

Таблица 7.1. Точность классификатора модели после очистки

	Точность	Отзыв	f1-score	Поддержка
Отрицательный	0.95	0.94	0.95	25 000
Положительный	0.94	0.95	0.95	25 000
Аккуратность			0.95	50 000
Макросреднее	0.95	0.95	0.95	50 000
Взвешенное среднее	0.95	0.95	0.95	50 000
Аккуратность: 0.948				

Мы попробовали линейный классификатор. По мере увеличения сложности модели мы получаем более уточненные лексемы и более уточненный контекст (см. табл. 7.1). В качестве примера обновили модель логистической регрессии до модели логистической регрессии с перекрестной проверкой. При этом отзыв № 0 имеет более точную оценку вероятности, которая, составляя 78.7 %, немного ниже, чем в предыдущей модели.

```
eli5.show_prediction(clf, imdb.review[0], vec=vec,
                     target_names=set(imdb.sentiment),
                     targets=['positive'])

y=positive (probability 0.787, score 1.307) top features
```

Contribution? Feature

+1.315	Highlighted in text (sum)
--------	---------------------------

-0.008	<BIAS>
--------	--------

one of the other reviewers has mentioned that after watching just 1 oz episode you'll be hooked. they are right, as this is exactly what happened with me.

the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint hearted or timid. this show pulls no punches with regards to drugs, sex or violence. it is hardcore, in the classic use of the word.

it is called oz as that is the nickname given to the oswald maximum security state penitentiary. it focuses mainly on emerald city, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda. em city is home to many..aryans, muslims, gangstas, latinos, christians, italians, irish and more....so scuffles, death stares, dodgy dealings and shady agreements are never far away.

i would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. forget pretty pictures painted for mainstream audiences, forget charm, forget romance...oz doesn't mess around. the first episode i ever saw struck me as so nasty it was surreal, i couldn't say i was ready for it, but as i watched more, i developed a taste for oz, and got accustomed to the high levels of graphic violence. not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) watching oz, you may become comfortable with what is uncomfortable viewing....thats if you can get in touch with your darker side.

Рис. 7.6. Выделенная важность характеристик для положительного класса

Текст отзыва о фильме (рис. 7.6) показывает повторение некоторых ключевых слов. Если в тексте слишком много повторений, то подсчеты в векторизаторе подсчета будут завышены. Если есть много слов, количество которых завышено, то это отразится в списке важности характеристик. Во избежание влияния повышенного количества слов в векторизаторе подсчета необходимо применить метод нормализации, чтобы все характеристики получили равную важность в процессе классификации. Для этого можно ввести новый векторизатор, который называется векторизатор частоты слов и обратной частоты до-

кументов (term frequency and inverse document frequency), широко известный как (tf-idf)-векторизатор. Для расчета значения tf-idf можно использовать следующую формулу:

*Значение tf-idf для i-го слова в j-м документе = частота i-го слова в j-м документе * log(общее количество документов / количество документов, содержащих i-е слово).*

Функция tf-idf легко доступна в модуле извлечения характеристик из текстового модуля (`feature_extraction`) библиотеки `sklearn`. Вам не обязательно вычислять значения tf-idf; вы можете применить следующий метод:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vec = TfidfVectorizer()
clf = LogisticRegressionCV()
pipe = make_pipeline(vec, clf)
pipe.fit(imdb.review, imdb.sentiment)

print_report(pipe)
```

Таблица 7.2. Отчет о классификации для модели на основе векторизатора tf-idf

	Точность	Отзыв	f1-score	Поддержка
Отрицательный	0.96	0.95	0.95	25 000
Положительный	0.95	0.96	0.95	25 000
Аккуратность			0.95	50 000
Макросреднее	0.95	0.95	0.95	50 000
Взвешенное среднее	0.95	0.95	0.95	50 000
Аккуратность: 0.954				

Согласно табл. 7.2 и рис. 7.7, нет видимой разницы в точности модели после применения преобразования tf-idf. Однако значения вероятности и оценки незначительно изменяются.

```
eli5.show_prediction(clf, imdb.review[0], vec=vec,
                     target_names=set(imdb.sentiment),
                     targets=['positive'])
```

y=positive (probability 0.793, score 1.340) top features

Contribution ⁷	Feature
+1.526	Highlighted in text (sum)
-0.186	<BIAS>

one of the other reviewers has mentioned that after watching just 1 oz episode you'll be hooked. they are right, as this is exactly what happened with me.

the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint hearted or timid. this show pulls no punches with regards to drugs, sex or violence. its is hardcore, in the classic use of the word.

it is called oz as that is the nickname given to the oswald maximum security state penitentiary. it focuses mainly on emerald city, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda. em city is home to many..aryans, muslims, gangstas, latinos, christians, italians, irish and more...so scuffles, death stares, dodgy dealings and shady agreements are never far away.

i would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. forget pretty pictures painted for mainstream audiences, forget charm, forget romance...oz doesn't mess around. the first episode i ever saw struck me as so nasty it was surreal, i couldn't say i was ready for it, but as i watched more, i developed a taste for oz, and got accustomed to the high levels of graphic violence. not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) watching oz, you may become comfortable with what is uncomfortable viewing...thats if you can get in touch with your darker side.

Рис. 7.7. Выделение характеристик для положительного класса после применения tf-idf

```
vec = TfidfVectorizer(stop_words='english')
clf = LogisticRegressionCV()
pipe = make_pipeline(vec, clf)
pipe.fit(imdb.review, imdb.sentiment)

print_report(pipe)
```

Таблица 7.3. Отчет о классификации после удаления стоп-слов

	Точность	Отзыв	f1-score	Поддержка
Отрицательный	0.96	0.95	0.96	25 000
Положительный	0.95	0.96	0.96	25 000
Аккуратность			0.96	50 000
Макросреднее	0.96	0.96	0.96	50 000
Взвешенное среднее	0.96	0.96	0.96	50 000
Аккуратность: 0.956				

Tf-idf – это векторное представление текстовых данных. Это очень полезно, когда характеристики, основанные на подсчете, имеют высокую частоту. Чтобы избежать влияния часто встречающихся слов на алгоритм классификации, предпочтительнее использовать векторизатор на основе tf-idf. При инициализации векторизатора tf-idf если указать опцию stop word, то для стоп-слов векторы создаваться не будут. Кроме того, при вычислении tf-idf количество стоп-слов не будет учитываться. Они определяются как слова, которые не имеют никакого значения, но помогают понять контекст вместе с другими словами. Как характеристики, они не добавляют большой ценности в задачу классификации текста, скорее увеличивают размер данных. Следовательно, удаление стоп-слов поможет получить более высокую точность за счет уменьшения размера данных (см. рис. 7.8).

```
eli5.show_prediction(clf, imdb.review[0], vec=vec,
                     target_names=set(imdb.sentiment),
                     targets=['positive'])
```

y=positive (probability 0.755, score 1.125) top features

Contribution?	Feature
+1.227	Highlighted in text (sum)
-0.102	<BIAS>

one of the other **reviewers** has mentioned that after **watching just 1 oz episode** you'll be **hooked**. they are **right**, as this is exactly what happened with me.

the first thing that **struck** me about **oz** was its **brutality** and **unflinching** scenes of violence, which set in **right** from the word go. **trust** me, this is not a show for the faint hearted or timid. this show **pulls no punches** with regards to drugs, sex or violence. its **hardcore**, in the **classic** use of the word.

it is called **oz** as that is the **nickname** given to the oswald maximum security state penitentiary. it **focuses** mainly on **emerald city**, an **experimental** section of the prison where all the cells have **glass fronts** and face inwards, so **privacy** is not high on the agenda. **em city** is home to many. **aryans**, **muslims**, **gangstas**, **latinos**, **christians**, **italians**, **irish** and more....so **scuffles**, **death stakes**, **dodgy dealings** and **shady agreements** are never **far away**.

i would say the main appeal of the show is due to the fact that it goes where other shows **wouldn't dare**. **forget** pretty pictures painted for **mainstream** audiences, **forget charm**, **forget romance...oz doesn't mess** around. the first **episode** i ever saw **struck** me as so nasty it was **surreal**, **i couldn't say** i was ready for it, but as i watched more, i developed a taste for **oz**, and got accustomed to the high **levels** of graphic violence. not **just** violence, but injustice (**crooked guards** who'll be sold out for a nickel, inmates who'll **kill** on order and get away with it, well mannered, **middle class** inmates being **turned** into prison bitches due to their lack of street **skills** or prison **experience**) **watching oz**, you may become **comfortable** with what is **uncomfortable viewing...thats** if you can get in **touch** with your **darker side**.

Рис. 7.8. Важные характеристики, выделенные из более совершенной модели

КЛАССИФИКАЦИЯ ТЕКСТА НА ОСНОВЕ N-ГРАММ

Рассмотрим два новых подхода к разработке модели классификации текста. Для классификации текста можно использовать либо слова, либо символы. Следующий скрипт анализирует символы, отбирает биграммы до пяти грамм и сохраняет эти n-граммы в качестве характеристик в векторизаторах tf-idf. После создания вектора данные поступают в конвейер, и происходит обучение модели. Точность обучения для классификации настроений составляет 99.8 %, что очень хорошо. Теперь можно объяснить индивидуальный прогноз и посмотреть, какие факторы способствуют позитивному настроению, а какие – негативному (см. табл. 7.4 и рис. 7.9).

```
vec = TfidfVectorizer(stop_words='english', analyzer='char',
                      ngram_range=(2,5))
clf = LogisticRegressionCV()
pipe = make_pipeline(vec, clf)
pipe.fit(imdb.review, imdb.sentiment)
print_report(pipe)
```

Векторизатор на основе N-грамма используется в качестве метода предварительной обработки, и N-граммы от 2 до 5 рассматриваются для создания обучающей модели с использованием векторизатора TF-IDF; в табл. 7.4 показан отчет о классификации.

Таблица 7.4. Отчет о классификации модели с использованием N-грамма

	Точность	Отзыв	f1-score	Поддержка
Отрицательный	1.00	1.00	1.00	25 000
Положительный	1.00	1.00	1.00	25 000
Аккуратность			1.00	50 000
Макросреднее	1.00	1.00	1.00	50 000
Взвешенное среднее	1.00	1.00	1.00	50 000
Аккуратность: 0.998				

```
eli5.show_prediction(clf, imdb.review[0], vec=vec,
                     target_names=set(imdb.sentiment),
                     targets=['positive'])
```

y=positive (probability 0.949, score 2.914) top features

Contribution?	Feature
+3.106	Highlighted in text (sum)
-0.192	<BIAS>

one of the other reviewers has mentioned that after watching just 1 oz episode you'll be hooked. they are right, as this is exactly what happened with me.

the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint hearted or timid. this show pulls no punches with regards to drugs, sex or violence. its is hardcore, in the classic use of the word.

it is called oz as that is the nickname given to the oswald maximum security state penitentiary. it focuses mainly on emerald city, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda. em city is home to many..aryans, muslims, gangstas, latinos, christians, italians, irish and more...so scuffles, death stares, dodgy dealings and shady agreements are never far away.

i would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. forget pretty pictures painted for mainstream audiences, forget charm, forget romance...oz doesn't mess around. the first episode i ever saw struck me as so nasty it was surreal, i couldn't say i was ready for it, but as i watched more, i developed a taste for oz, and got accustomed to the high levels of graphic violence. not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) watching oz, you may become comfortable with what is uncomfortable viewing...thats if you can get in touch with your darker side.

Рис. 7.9. Выделенные характеристики модели – 2 и 5 г

В приведенном выше скрипте мы подобрали вклад в нулевой обзор, который является общим результатом зеленых и красных выделенных элементов текста, а это +3.106. Поскольку общий результат равен +3.106, он классифицируется как позитивное настроение, чему также соответствует вероятность позитивного настроения 0.949, а логарифмическая вероятность для классификации составляет 2.914. -0.192 – это предвзятость, которая также называется членом перехвата для логистической регрессионной модели.

Векторизатор подсчета подходит, когда длина текстовых документов короче. Если длина текста больше (каждое слово имеет высокую частоту или повторяется несколько раз в тексте), то следует использовать векторизатор tf-idf. Векторизатор – это просто слова и их частоты, представленные в виде таблицы или массива. Из следующего скрипта следует, что векторизатор подсчета с удалением стоп-слов уменьшает количество характеристик и делает прогноз очень значимым. После обучения модели можно увидеть, что точность составляет 94.8 %.

Также, когда вы объясняете первый обзор, сумма результатов зеленого и красного цвета выделенного текста равна +1.315, и, поскольку общий результат является положительным, настроение отмечается как позитивное, с вероятностью 78.7 % и логарифмической вероятностью 1.307. Можно продолжать обновлять извлечение характеристик и обновлять тип модели, чтобы сделать прогноз лучше. Чем лучше модель, тем лучше объяснение классификации настроения. Следующий скрипт показывает вектор tf-idf в качестве характеристики без процесса удаления стоп-слов.

Вы можете видеть, что в выделенном тексте на рис. 7.10 есть несколько стоп-слов, bigramms и триграмм, которые неправильно помечены цветом. Чтобы исправить это, можете использовать такую опцию в анализаторе, как `char_wb`. Это анализатор, который вы уже видели раньше. Он разбивает текст на символы в виде n-грамм, но этот процесс занимает время. Кроме того, `char_wb` лучше настраивается, чтобы взять в качестве характеристик те n-граммы, которые не перекрывают друг друга.

```
vec = TfidfVectorizer(stop_words='english', analyzer='char_wb',
                      ngram_range=(2,5))

clf = LogisticRegressionCV()

pipe = make_pipeline(vec, clf)

pipe.fit(imdb.review, imdb.sentiment)

print_report(pipe)

eli5.show_prediction(clf, imdb.review[0], vec=vec,
                     target_names=set(imdb.sentiment),
                     targets=['positive'])
```

y=positive (probability 0.969, score 3.458) top features

Contribution?	Feature
+3.595	Highlighted in text (sum)
-0.137	<BIAS>

one of the other reviewers has mentioned that after watching just 1 oz episode you'll be hooked. they are right, as this is exactly what happened with me.

the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint hearted or timid. this show pulls no punches with regards to drugs, sex or violence. its is hardcore, in the classic use of the word.

it is called oz as that is the nickname given to the oswald maximum security state penitentiary. it focuses mainly on emerald city, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda. em city is home to many.aryans, muslims, gangstas, latinos, christians, italians, irish and more....so scuffles, death stares, dodgy dealings and shady agreements are never far away.

i would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. forget pretty pictures painted for mainstream audiences, forget charm, forget romance...oz doesn't mess around. the first episode i ever saw struck me as so nasty it was surreal, i couldn't say i was ready for it, but as i watched more, i developed a taste for oz, and got accustomed to the high levels of graphic violence. not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) watching oz, you may become comfortable with what is uncomfortable viewing....thats if you can get in touch with your darker side.

Рис. 7.10. Неперекрывающиеся символьные n-граммы как важность характеристик

На этот раз результаты выглядят лучше. Например, «смотреть только 1» – это отрицательная n-грамма, которая не пересекается ни с одной другой n-граммой, что является значимым. Аналогично другие n-граммы являются

более точными. Если отзыв о фильме ограничивается несколькими строками, можно использовать векторизатор подсчета или векторизатор tf-idf, но если отзывы становятся длинными, то на помощь приходит векторизатор хеширования. При увеличении длины отзыва увеличивается объем словарного запаса, поэтому векторизатор хеширования будет весьма полезен. В следующем скрипте представлен векторизатор хеширования вместе с классификатором одной из продвинутых ML-моделей стохастического градиентного спуска. В результате получается модель лучше, чем предыдущая (рис. 7.11). Это объясняется тем, что предыдущие модели также были немного переподогнаны. Чтобы исследовать объясимость, можно взять тот же самый текст нулевого отзыва и объяснить классификационную метку, а также рассмотреть слова, которые способствовали положительному и отрицательному настроению.

```
from sklearn.feature_extraction.text import HashingVectorizer
from sklearn.linear_model import SGDClassifier
vec = HashingVectorizer(stop_words='english', ngram_range=(1,2))
clf = SGDClassifier(random_state=42)
pipe = make_pipeline(vec, clf)
pipe.fit(imdb.review, imdb.sentiment)

print_report(pipe)

eli5.show_prediction(clf, imdb.review[0], vec=vec,
                     target_names=set(imdb.sentiment),
                     targets=['positive'])
```

y=positive (score 0.269) top features

Contribution ²	Feature
+0.243	Highlighted in text (sum)
+0.025	<BIAS>

one of the other reviewers has mentioned that after watching just 1 oz episode you'll be hooked. they are right, as this is exactly what happened with me.

the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint hearted or timid. this show pulls no punches with regards to drugs, sex or violence. its is hardcore, in the classic use of the word.

it is called oz as that is the nickname given to the oswald maximum security state penitentiary. it focuses mainly on emerald city, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda. em city is home to many..aryans, muslims, gangstas, latinos, christians, italians, irish and more...so scuffles, death stares, dodgy dealings and shady agreements are never far away.

i would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. forget pretty pictures painted for mainstream audiences, forget charm, forget romance...oz doesn't mess around. the first episode i ever saw struck me as so nasty it was surreal, i couldn't say i was ready for it, but as i watched more, i developed a taste for oz, and got accustomed to the high levels of graphic violence. not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) watching oz, you may become comfortable with what is uncomfortable viewing....thats if you can get in touch with your darker side.

Рис. 7.11. Уточненные выделенные характеристики лучше объясняют прогноз

Первый комментарий к обзору вызывает положительные чувства, общая положительная оценка составляет +0.243, а оценка равна 0.269. В нем учитывались только соответствующие характеристики, выделенные зеленым и красным цветом.

```
from eli5.sklearn import InvertableHashingVectorizer
import numpy as np
ivec = InvertableHashingVectorizer(vec)
sample_size = len(imdb.review) // 10
X_sample = np.random.choice(imdb.review, size=sample_size)
ivec.fit(X_sample)
eli5.show_weights(clf, vec=ivec, top=20,
                  target_names=set(imdb.sentiment))
```

Инвертируемый векторизатор хеширования (invertible hashing vectorizer) помогает в представлении весов вместе с именами характеристик из векторизатора хеширования без использования огромного словаря. На рис. 7.12 показаны положительные и отрицательные характеристики.

y=positive top features	
Weight?	Feature
+4.547	excellent
+3.812	great
+3.293	wonderful ...
+3.270	perfect ...
+3.195	amazing ...
+3.066	brilliant ...
+3.038	best
... 456783	<i>more positive ...</i>
... 454840	<i>more negative ...</i>
-3.070	supposed ...
-3.109	unfortunately
-3.110	minutes
-3.269	stupid
-3.408	worse ...
-3.502	dull ...
-3.819	terrible
-4.213	poor
-4.268	waste
-4.479	bad
-4.562	boring ...
-5.084	awful ...
-6.464	worst

Рис. 7.12. Лучшие характеристики для прогноза положительного класса

ОБЪЯСНИМОСТЬ МНОГОКЛАССОВОЙ КЛАССИФИКАЦИИ ТЕКСТА ПО МЕТКАМ

Давайте рассмотрим модель многоклассовой классификации, когда есть более двух категорий в целевом столбце. Многоклассовая классификация – это еще один случай, когда зависимая переменная или целевой столбец может иметь более двух категорий или меток. Объяснимость здесь требуется для каждой метки соответствующего класса, поэтому должны быть показаны важные по-

ложительные и отрицательные характеристики. Давайте воспользуемся набором данных о жалобах клиентов, где целью является прогнозирование категории или типа жалоб клиентов.

```
import pandas as pd
df = pd.read_csv('complaints.csv')
df.shape
df.info()

# Создание нового датафрейма с двумя столбцами
df1 = df[['Product', 'Consumer complaint narrative']].copy()

# Удаление отсутствующих значений (NaN)
df1 = df1[pd.notnull(df1['Consumer complaint narrative'])]

# Присвоение второму столбцу более простого имени
df1.columns = ['Product', 'Consumer_complaint']
df1.shape

# Поскольку вычисления занимают много времени (с точки зрения CPU),
# данные были отобраны
df2 = df1.sample(20000, random_state=1).copy()

# Переименование категорий
df2.replace({'Product':
              {'Credit reporting, credit repair services, or other personal
               consumer reports':
                  'Credit reporting, repair, or other',
               'Credit reporting': 'Credit reporting, repair, or other',
               'Credit card': 'Credit card or prepaid card',
               'Prepaid card': 'Credit card or prepaid card',
               'Payday loan': 'Payday loan, title loan, or personal loan',
               'Money transfer': 'Money transfer, virtual currency, or money
               service',
               'Virtual currency': 'Money transfer, virtual currency, or
               money service'}},
              inplace=True)

df2.head()

pd.DataFrame(df2.Product.unique())
```

Источником набора данных является сайт <https://catalog.data.gov/dataset/consumercomplaint-database>. Набор данных очищается и подготавливается для прогнозирования категории жалобы. После чтения данных создается объект Python df. Следующий результат показывает первые несколько записей из набора данных.

Из df вы получаете копию, содержащую только описание жалобы клиента и столбец продукта для разработки модели многоклассовой классификации, которая присваивается объекту df1. Затем данные очищаются путем удаления значений NaN (*not a number* – не число) и введения более простых имен для переменных.

Длинные описания с большим количеством текста очищаются с помощью более коротких и более подходящих имен, чтобы можно было лучше анализировать данные. Существует 13 уникальных категорий продуктов, в отношении которых были упомянуты жалобы клиентов.

```
# Создание нового столбца 'category_id' с закодированными категориями
df2['category_id'] = df2['Product'].factorize()[0]
category_id_df = df2[['Product', 'category_id']].drop_duplicates()

# Словари для будущего использования
category_to_id = dict(category_id_df.values)
id_to_category = dict(category_id_df[['category_id', 'Product']].values)

# Новый датафрейм
df2.head()
df2.Product.value_counts()
```

Для обучения модели необходимо преобразовать текст целевого столбца в числа. Поэтому вы создаете столбец `category_id` путем факторизации столбца `Product`. Следующие две строки кода в приведенном выше скрипте создают соответствие между кодированным числом и описанием класса. Это полезно для обратного преобразования после генерации прогнозов. В следующих строках кода вы создаете вектор `tf-idf` с n-граммами, взятыми от 1 до 2, и удаляете стоп-слова, сохраняя сублинейную частоту слов (term frequency – TF) и минимальную частоту документа равной 5. Если вы хотите уменьшить предвзятость, вызванную длиной документа, то следует использовать сублинейную частоту слов `TRUE`. Как известно из закона Ципфа, частота любого слова обратно пропорциональна его рангу. Минимальная частота документа подразумевает, что при составлении словаря следует игнорировать слова, частота которых ниже заданного порога. В следующем скрипте это 5, т. е. что любое слово с частотой меньше 5 не будет включено в словарь.

Выходные данные следующего скрипта показывают количество каждого класса в целевом столбце многоклассовой классификации. Следующий скрипт подгоняет логистическую регрессию для многоклассовой модели, используя в качестве отправной точки векторизатор подсчета.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegressionCV
from sklearn.pipeline import make_pipeline

vec = CountVectorizer()
clf = LogisticRegressionCV()
pipe = make_pipeline(vec, clf)
pipe.fit(df2.Consumer_complaint, df2.Product)

from sklearn import metrics

def print_report(pipe):
    y_test = df2.Product
    y_pred = pipe.predict(df2.Consumer_complaint)
```

```

report = metrics.classification_report(y_test, y_pred)
print(report)
print("accuracy: {:.3f}".format(metrics.accuracy_score(y_test,
y_pred)))
print_report(pipe)

```

	точность	отзыв	f1-score	поддержка
Банковский счет или услуга	0.97	0.74	0.84	441
Расчетный или сберегательный счет	0.91	0.87	0.89	849
Потребительский кредит	0.98	0.59	0.74	296
Кредитная карта или карта предоплаты	0.91	0.85	0.88	2000
Кредитная отчетность, ремонт или другое	0.87	0.95	0.91	8231
Взыскание долгов	0.87	0.85	0.86	4075
Денежные переводы, виртуальная валюта или денежные услуги	0.92	0.80	0.86	358
Денежные переводы	1.00	0.70	0.82	53
Ипотека	0.96	0.95	0.95	2216
Другие финансовые услуги	1.00	0.57	0.73	7
Платежный кредит, титульный кредит или персональный кредит	0.93	0.67	0.78	312
Студенческий кредит	0.94	0.85	0.89	788
Лизинг или кредит на транспортное средство	0.93	0.68	0.79	374
аккуратность			0.89	20000
макро среднее	0.94	0.78	0.84	20000
взвешенное среднее	0.89	0.89	0.89	20000

Рис. 7.13. Точность классификации для многоклассовой модели классификации текста

Согласно рис. 7.13, точность модели кажется нормальной, поскольку она составляет 89 %. В многоклассовой классификации обычно рассматривается f1-score, который является средним гармоническим значением точности и отзыва. Приемлемый предел f1-score составляет 75 % или более. Видно, что для потребительского кредита и других финансовых услуг он составляет менее 75 %. В качестве отправной точки рассмотрим термины в роли весов для каждого класса в задаче классификации. Результат не имеет смысла, поскольку названия терминов недоступны (табл. 7.5).

```

import eli5
eli5.show_weights(clf, top=10)
# Результат не имеет смысла, поскольку вес и имена характеристик здесь отсутствуют

```

Таблица 7.5. Образцы характеристик, важных для каждого класса

y=Mortgage top features		y=Other financial service top features		y=Payday loan, title loan, or personal loan top features		y=Student loan top features		y=Vehicle loan or lease top features	
Weight ⁷	Feature	Weight ⁷	Feature	Weight ⁷	Feature	Weight ⁷	Feature	Weight ⁷	Feature
+1.119	x15196	+0.229	x14201	+0.578	x14062	+1.081	x15458	+0.607	x4596
+0.777	<BIAS>	+0.201	x6495	+0.535	x16848	+0.776	x22119	+0.597	x24811
+0.769	x9037	+0.191	x11457	+0.362	x10873	+0.735	x14077	+0.524	x3171
+0.621	x15082	+0.167	x2853	+0.358	x4665	+0.473	x19395	+0.395	x20363
+0.533	x20813	+0.167	x15115	+0.311	X13958	+0.436	x1888	+0.352	x1467
+0.506	x18974	+0.153	x13149	+0.296	x1987	+0.427	x8458	+0.318	x13715
+0.486	x7980	+0.150	x22755	...2746 более позитивный...		+0.419	x14062	+0.310	x15678
+0.463	x5262	...424 более позитивный...		...23451 более негативный...		+0.410	x20478	+0.301	x23501
...7456 более позитивный...		...25773 более негативный...		-0.295	x15196	+0.380	x10211	...3355 более позитивный...	
...18741 более негативный...		-0.157	x1528	-0.315	x22119	...4286 более позитивный...		...22842 более негативный...	
-0.475	x4600	-0.164	x23119	-0.315	x4600	...21911 более негативный...		-0.295	x25367
-0.481	x24811	-2.364	<BIAS>	-0.537	<BIAS>	-0.381	x3427	-0.610	<BIAS>

Поскольку имена характеристик в виде терминов отсутствуют на табл. 7.5, вы не сможете найти в этом смысла. Нужно немного изменить скрипт, как показано ниже, чтобы включить имена характеристик из объекта `vec`, а также целевые имена как уникальные. Таким образом, вы можете видеть, что термины, положительно влияющие на метку класса, выделены зеленым цветом, а термины, которые отрицательно влияют на нее, – красным. Для каждой категории, такой как банковский счет, услуга или потребительский кредит, вы можете увидеть основные характеристики. Поскольку у вас более 10 категорий, нельзя уместить все положительные и отрицательные термины на одном экране, поэтому вы делаете различные снимки как минимум для пяти меток одновременно, чтобы обеспечить читаемость терминов.

```
eli5.show_weights(clf, feature_names=vec.get_feature_names(),
                  target_names=set(df2.Product))
#СМЫСЛ
```

Таблица 7.6. Главные характеристики для каждого класса с названиями характеристик

y=Bank account or service top features		y=Checking or savings account top features		y=Consumer Loan top features		y=Credit card or prepaid card top features		y=Credit reporting, repair, or other top features	
Weight ⁷	Feature	Weight ⁷	Feature	Weight ⁷	Feature	Weight ⁷	Feature	Weight ⁷	Feature
+0.613	scottrade	+0.423	schwab	+0.604	vehicle	+0.899	<BIAS>	+2.290	<BIAS>
+0.373	2016	+0.387	citibank	+0.556	car	+0.763	card	+1.208	equifax
+0.372	overdraft	+0.365	bank	+0.400	loan	+0.445	capital	+1.078	experian
+0.369	greentree	+0.341	<BIAS>	+0.352	2016	+0.427	purchase	+0.999	transunion
+0.368	debit	+0.340	deposit	+0.351	financial	+0.420	limit	+0.607	inquiries
+0.362	bank	+0.314	overdraft	+0.300	finance	+0.417	minimum	+0.565	freeze
+0.327	requirements	+0.311	bonus	+0.285	dealer	+0.388	rewards	+0.540	mortgage
+0.323	2015	+0.307	debit	+0.265	contract	+0.386	visa	+0.483	inaccurate
+0.310	branch	+0.289	charged	+0.253	corporation	+0.381	unemployment	+0.464	report
+0.306	deposited	+0.288	branch	+0.252	paying	+0.378	issued	+0.447	remove
+0.295	club	+0.287	won	+0.251	credit	+0.376	netspend	+0.444	background
+0.289	hold	+0.287	access	+0.229	tree	+0.366	capitol	+0.432	trans
+0.279	atm	+0.279	link	...2767 более положительный...		+0.359	found	+0.421	mine
+0.277	fee	+0.275	funds	...25420 более отрицательный...		+0.344	citibank	+0.398	inquiry
...3662 более положительный...		...4762 более положительный...		-0.231	2018	+0.340	city	+0.387	removed
...22525 более отрицательный...		...21425 более отрицательный...		-0.245	problem	+0.337	discover	+0.379	delinquent
-0.287	<BIAS>	-0.365	2016	-0.267	2020	...6420 более положительный...		+0.372	victim
-0.287	company	-0.371	payments	-0.272	2019	...19767 более отрицательный...		...8710 более положительный...	
-0.297	2019	-0.397	2015	-0.273	don	-0.387	loans	...17477 более отрицательный...	
-0.341	payment	-0.417	credit	-0.283	didnt	-0.416	mortgage	-0.381	calls
-0.351	tried	-0.444	payment	-0.284	accounts	-0.437	debit	-0.384	servicer
-0.378	2018	-0.484	debt	-0.585	<BIAS>	-0.727	loan	-0.416	american

Результат в табл. 7.6 показывает первые пять классов из целевого столбца и их главные характеристики, положительные и отрицательные.

Какие основные выводы можно сделать из этих таблиц? Возьмем класс ипотеки. Такие слова, как ипотека, контракт, рефинансирование, модификация, обслуживающая организация и ditech, – это основные положительные характеристики, которые помогли в прогнозировании класса ипотеки. Аналогичным образом можно интерпретировать и объяснить все другие классы и слова из жалоб потребителей, действительно помогающие текстовому документу предсказать конкретный класс как категорию продукта, на который жалуется потребитель. Таким образом, это не только выявление основных характеристик, но и весь контекст, приводящий к прогнозированию метки класса.

ЛОКАЛЬНОЕ ОБЪЯСНЕНИЕ. ПРИМЕР 1

Следующий скрипт и рис. 7.14 рассматривают первую жалобу потребителя. Прогноз для банковского счета или услуги имеет очень низкую вероятность 0.008, что означает, что первая жалоба не относится к этому классу.

```
eli5.show_prediction(clf, np.array(df2.Consumer_complaint)[0], vec=vec,
target_names=set(df2.Product)) # объяснение локального прогноза
np.array(df2.Consumer_complaint)[0]
```



Рис. 7.14. Объяснение прогноза класса для первой жалобы

Поскольку вероятность равна 0.083, первая жалоба не относится к расчетному или сберегательному счету (рис. 7.15).

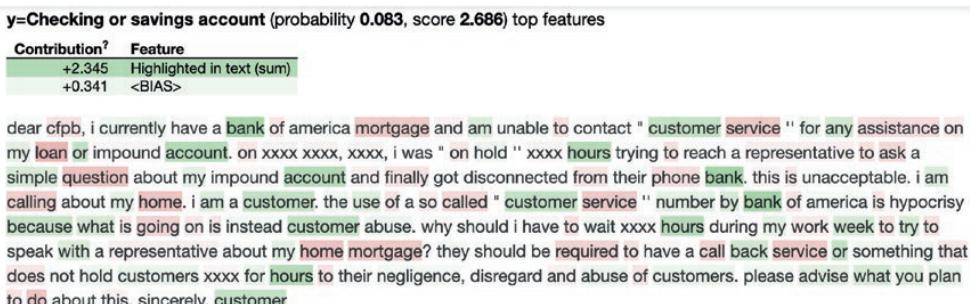


Рис. 7.15. Важность характеристик для расчетного или сберегательного счета

Рис. 7.16 показывает, что первая жалоба потребителя относится к ипотеке, так как вероятность этого составляет 84.3 %, или 0.843. Слова, которые действительно внесли вклад в прогноз, выделены зеленым цветом.

y=Mortgage (probability 0.843, score 5.006) top features

Contribution?	Feature
+4.229	Highlighted in text (sum)
+0.777	<BIAS>

dear cfpb, i currently have a bank of america mortgage and am unable to contact " customer service " for any assistance on my loan or impound account. on xxxx xxxx, xxxx, i was " on hold " xxxx hours trying to reach a representative to ask a simple question about my impound account and finally got disconnected from their phone bank. this is unacceptable. i am calling about my home. i am a customer. the use of a so called " customer service " number by bank of america is hypocrisy because what is going on is instead customer abuse. why should i have to wait xxxx hours during my work week to try to speak with a representative about my home mortgage? they should be required to have a call back service or something that does not hold customers xxxx for hours to their negligence, disregard and abuse of customers. please advise what you plan to do about this. sincerely, customer

Рис. 7.16. Важность характеристик для прогнозирования класса ипотеки

Ниже представлен фактический текст первой жалобы потребителя:

Уважаемое CFPB, в настоящее время у меня ипотечный кредит от Bank of America, и я не могу связаться со ''службой поддержки клиентов \\'' для получения какой-либо помощи по моему кредиту или конфискованному счету. В XXXX XXXX, XXXX, я просидел "в режиме ожидания \\'' XXXX часов, пытаясь дозвониться до представителя, чтобы задать простой вопрос о моем конфискованном счете, и в конце концов был отключен от их телефонного банка. \nЭТО НЕПРИЕМЛЕМО. Я звоню по поводу своего дома. Я клиент. Использование Bank of America так называемого "номера обслуживания клиентов\\'' - это лицемерие, потому что то, что происходит, - это насилие над клиентом. \nПочему я должен ждать XXXX часов в течение своей рабочей недели, чтобы поговорить с представителем по поводу моего ипотечного кредита? Они должны быть обязаны иметь службу обратного звонка или что-то подобное, что не заставляет клиентов ждать XXXX часов, что свидетельствует об их халатности, пренебрежении и жестоком обращении с клиентами. \nПожалуйста, сообщите, что вы планируете сделать в этой связи. \nИскренне, клиент'

Аналогичным образом вы можете проверить жалобу № 123 и убедиться, что она принадлежит к классу взыскания долгов. Вероятность того, что текст относится к взысканию долга, составляет 0.740 (74 %). Если вы объясняете прогнозы в терминах точного текста, который внес свой вклад, это называется локальной интерпретацией.

ЛОКАЛЬНОЕ ОБЪЯСНЕНИЕ. ПРИМЕР 2

Рассмотрим другой пример, запись № 123 из набора данных по жалобам потребителей. Важность характеристик для каждого класса показана ниже. Следующий скрипт берет жалобу № 123, и сгенерированный локальный прогноз показывает, что она принадлежит к классу «кредитная отчетность, ремонт или другое». Вероятность составляет 0.515 (51.5 %), см. рис. 7.17.

```
eli5.show_prediction(clf, np.array(df2.Consumer_complaint)[100], vec=vec,
                     target_names=set(df2.Product)) # объяснение локального
                                         # прогноза
```

y=Credit reporting, repair, or other (probability **0.515**, score **3.151**) top features

Contribution?	Feature
+2.290	<BIAS>
+0.861	Highlighted in text (sum)

id theft victim xxxx and xxxx (not my account) xxxx { not my account } xxxx bank xxxx xxxx xxxx (not my account)

y=Debt collection (probability **0.154**, score **1.944**) top features

Contribution?	Feature
+1.828	<BIAS>
+0.116	Highlighted in text (sum)

id theft victim xxxx and xxxx (not my account) xxxx { not my account } xxxx bank xxxx xxxx xxxx (not my account)

Рис. 7.17. Важность характеристик с положительными и отрицательными словами

Приведенный выше результат получен без предварительной обработки или тонкой настройки. Более точный результат можно ожидать при дальнейшей тонкой настройке и предварительной обработке текста. Стандартной практикой является удаление стоп-слов. Следующий скрипт удаляет стоп-слова и снова запускает конвейер обучения модели:

```
vec = CountVectorizer(stop_words='english')
clf = LogisticRegressionCV()
pipe = make_pipeline(vec, clf)
pipe.fit(df2.Consumer_complaint, df2.Product)

print_report(pipe)

eli5.show_prediction(clf, np.array(df2.Consumer_complaint)[0], vec=vec,
                     target_names=set(df2.Product))
```

Поскольку многие жалобы не содержат достаточного количества текста, после удаления стоп-слов было потеряно много характеристик, поэтому точность немного снизилась, до 88.1 %, как показано в табл. 7.7.

Таблица 7.7. Отчет о классификации для нескольких классов на основе уточненной модели

	Точность	Отзыв	f1-score	Поддержка
Банковский счет или услуга	0.95	0.71	0.81	441
Расчетный или сберегательный счет	0.90	0.85	0.87	849
Потребительский кредит	0.96	0.52	0.68	296
Кредитная карта или карта предоплаты	0.89	0.84	0.86	2000
Кредитная отчетность, ремонт или другое	0.86	0.95	0.90	8231
Взыскание долгов	0.86	0.84	0.85	4075
Денежные переводы, виртуальная валюта или денежные услуги	0.92	0.80	0.85	358
Денежные переводы	0.97	0.62	0.76	53

Окончание табл. 7.7

	Точность	Отзыв	f1-score	Поддержка
Ипотека	0.95	0.94	0.95	2216
Другие финансовые услуги	1.00	0.43	0.60	7
Платежный кредит. титульный кредит или персональный кредит	0.93	0.62	0.75	312
Студенческий кредит	0.94	0.84	0.89	788
Лизинг или кредит на транспортное средство	0.93	0.67	0.78	374
Аккуратность			0.88	20 000
Средний макро	0.93	0.74	0.81	20 000
Взвешенный макро	0.88	0.88	0.88	20 000
Аккуратность: 0.881				

ЛОКАЛЬНОЕ ОБЪЯСНЕНИЕ. ПРИМЕР 3

Возможно взять уточненную модель и создать локальные интерпретации для жалобы потребителя № 0. Оценка вероятности теперь немного уточнена, она составляет 76.4 %, как показано на рис. 7.18.

```
eli5.show_prediction(clf, np.array(df2.Consumer_complaint)[0], vec=vec,
                     target_names=set(df2.Product))
```

y=Mortgage (probability 0.764, score 4.301) top features

Contribution?	Feature
+3.469	Highlighted in text (sum)
+0.831	<BIAS>

dear cfpb, i currently have a bank of america mortgage and am unable to contact " customer service " for any assistance on my loan or impound account. on xxxx xxxx, xxxx, i was " on hold " xxxx hours trying to reach a representative to ask a simple question about my impound account and finally got disconnected from their phone bank. this is unacceptable. i am calling about my home. i am a customer. the use of a so called " customer service " number by bank of america is hypocrisy because what is going on is instead customer abuse. why should i have to wait xxxx hours during my work week to try to speak with a representative about my home mortgage? they should be required to have a call back service or something that does not hold customers xxxx for hours to their negligence, disregard and abuse of customers. please advise what you plan to do about this. sincerely, customer

Рис. 7.18. Локальное объяснение прогноза для одной жалобы

Аналогичным образом изменим векторизатор со счетного на tf-idf, без удаления стоп-слов, и посмотрим, как это повлияет на локальную интерпретацию. Вы можете увидеть это при помощи следующего скрипта:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vec = TfidfVectorizer()
clf = LogisticRegressionCV()
pipe = make_pipeline(vec, clf)
pipe.fit(df2.Consumer_complaint, df2.Product)
print_report(pipe)
```

Таблица 7.8. Отчет о классификации на основе модели векторизатора tf-idf

	Точность	Отзыв	f1-score	Поддержка
Банковский счет или услуга	0.95	0.73	0.82	441
Расчетный или сберегательный счет	0.86	0.90	0.88	849
Потребительский кредит	0.92	0.53	0.68	296
Кредитная карта или карта предоплаты	0.88	0.89	0.89	2000
Кредитная отчетность, ремонт или другое	0.91	0.94	0.92	8231
Взыскание долгов	0.88	0.88	0.88	4075
Денежные переводы, виртуальная валюта или денежные услуги	0.88	0.83	0.86	358
Денежные переводы	0.96	0.49	0.65	53
Ипотека	0.93	0.97	0.95	2216
Другие финансовые услуги	0.00	0.00	0.00	7
Платежный кредит, титульный кредит или персональный кредит	0.91	0.66	0.76	312
Студенческий кредит	0.93	0.90	0.92	788
Лизинг или кредит на транспортное средство	0.85	0.74	0.79	374
Аккуратность			0.90	20 000
Средний макро	0.84	0.73	0.77	20 000
Взвешенный макро	0.90	0.90	0.90	20 000
Аккуратность: 0.901				

Согласно табл. 7.8, подход tf-idf увеличивает общую точность до 90.1 %, а локальная интерпретация для первой жалобы имеет лучшую оценку вероятности 72.5 %.

```
eli5.show_prediction(clf, np.array(df2.Consumer_complaint)[0], vec=vec,
                     target_names=set(df2.Product))
```

В качестве следующего шага экспериментируем с удалением стоп-слов в tf-idf, чтобы увидеть какие-либо различия в локальной объяснимости прогноза для первой жалобы потребителя (см. табл. 7.9).

```
vec = TfidfVectorizer(stop_words='english')
clf = LogisticRegressionCV()
pipe = make_pipeline(vec, clf)
pipe.fit(df2.Consumer_complaint, df2.Product)
print_report(pipe)
```

Таблица 7.9. Отчет о классификации для модели многоклассовой классификации

	Точность	Отзыв	f1-score	Поддержка
Банковский счет или услуга	0.93	0.76	0.83	441
Расчетный или сберегательный счет	0.87	0.90	0.88	849
Потребительский кредит	0.94	0.60	0.74	296
Кредитная карта или карта предоплаты	0.88	0.90	0.89	2000
Кредитная отчетность, ремонт или другое	0.91	0.94	0.93	8231
Взыскание долгов	0.89	0.88	0.89	4075
Денежные переводы, виртуальная валюта или денежные услуги	0.88	0.85	0.87	358
Денежные переводы	0.96	0.49	0.65	53
Ипотека	0.94	0.97	0.96	2216
Другие финансовые услуги	0.00	0.00	0.00	7
Платежный кредит, титульный кредит или персональный кредит	0.92	0.70	0.80	312
Студенческий кредит	0.93	0.90	0.92	788
Лизинг или кредит на транспортное средство	0.86	0.73	0.79	374
Аккуратность			0.91	20 000
Средний макро	0.84	0.74	0.78	20 000
Взвешенный макро	0.91	0.91	0.90	20 000
Аккуратность: 0.906				

Удаление стоп-слов не оказало существенного влияния на общую точность модели. Она увеличилась незначительно – с 90.1 до 90.6 %. Кроме того, оценка вероятности для класса ипотеки изменилась до 73.5 %.

```
eli5.show_prediction(clf, np.array(df2.Consumer_complaint)[0], vec=vec,
                     target_names=set(df2.Product))

y=Mortgage (probability 0.735, score 4.000) top features

Contribution? Feature
+3.153 Highlighted in text (sum)
+0.847 <BIAS>
```

dear cfpb, i currently have a bank of america mortgage and am unable to contact " customer service " for any assistance on my loan or impound account. on xxxx xxxx, xxxx, i was " on hold " xxxx hours trying to reach a representative to ask a simple question about my impound account and finally got disconnected from their phone bank. this is unacceptable. i am calling about my home. i am a customer. the use of a so called " customer service " number by bank of america is hypocrisy because what is going on is instead customer abuse. why should i have to wait xxxx hours during my work week to try to speak with a representative about my home mortgage? they should be required to have a call back service or something that does not hold customers xxxx for hours to their negligence, disregard and abuse of customers. please advise what you plan to do about this. sincerely, customer

Рис. 7.20. Важность характеристик для класса ипотеки после удаления стоп-слов

Важность характеристик необходима для понимания положительного и отрицательного вклада факторов в классификацию. На рис. 7.20 зеленым цветом выделена положительная, а красным – отрицательная важность характеристик.

В качестве дальнейшего шага совершенствования можно взять символы в качестве анализатора и применить для создания характеристик n-грамм до пяти грамм. Точность при этом увеличивается до 91.4 %, незначительно превышая предыдущие результаты.

```
vec = TfidfVectorizer(stop_words='english', analyzer='char',
                      ngram_range=(2,5))
clf = LogisticRegressionCV()
pipe = make_pipeline(vec, clf)
pipe.fit(df2.Consumer_complaint, df2.Product)

print_report(pipe)
```

Отчет по многоклассовой классификации после итерации показан в табл. 7.10. Вероятность для класса ипотеки снова увеличилась до 74.2 % (см. рис. 7.20).

Таблица 7.10. Отчет о многоклассовой классификации после итерации

	Точность	Отзыв	f1-score	Поддержка
Банковский счет или услуга	0.95	0.75	0.84	441
Расчетный или сберегательный счет	0.88	0.91	0.90	849
Потребительский кредит	0.98	0.56	0.71	296
Кредитная карта или карта предоплаты	0.89	0.91	0.90	2000
Кредитная отчетность, ремонт или другое	0.92	0.95	0.94	8231
Взыскание долгов	0.89	0.90	0.90	4075
Денежные переводы, виртуальная валюта или денежные услуги	0.90	0.85	0.87	358
Денежные переводы	1.00	0.38	0.55	53
Ипотека	0.94	0.97	0.95	2216
Другие финансовые услуги	0.00	0.00	0.00	7
Платежный кредит, титульный кредит или персональный кредит	0.91	0.63	0.75	312
Студенческий кредит	0.94	0.90	0.92	788
Лизинг или кредит на транспортное средство	0.91	0.74	0.81	374
Аккуратность			0.91	20 000
Средний макро	0.86	0.73	0.77	20 000
Взвешенный макро	0.91	0.91	0.91	20 000
Аккуратность: 0.914				

```
eli5.show_prediction(clf, np.array(df2.Consumer_complaint)[0], vec=vec,
                     target_names=set(df2.Product))
```

y=Mortgage (probability 0.742, score 3.989) top features

Contribution?	Feature
+3.370	Highlighted in text (sum)
+0.619	<BIAS>

dear cfpb, i currently have a bank of america mortgage and am unable to contact " customer service " for any assistance on my loan or impound account. on xxxx xxxx, xxxx, i was " on hold " xxxx hours trying to reach a representative to ask a simple question about my impound account and finally got disconnected from their phone bank. this is unacceptable. i am calling about my home. i am a customer. the use of a so called " customer service " number by bank of america is hypocrisy because what is going on is instead customer abuse. why should i have to wait xxxx hours during my work week to try to speak with a representative about my home mortgage? they should be required to have a call back service or something that does not hold customers xxxx for hours to their negligence, disregard and abuse of customers. please advise what you plan to do about this. sincerely, customer

Рис. 7.20. Локальный прогноз для класса ипотеки

На рис. 7.20 показаны для класса ипотеки положительные слова как характеристики, выделенные зеленым цветом фона, и отрицательные слова – как характеристики, выделенные красным.

Аналогичным образом для создания характеристик можно взять символы и неперекрывающиеся n-граммы до пяти грамм. Это снижает общую точность до самого низкого уровня 88.4 %, но вы можете проверить локальное объяснение, чтобы увидеть, улучшило ли оно объяснимость, или нет.

```
vec = TfidfVectorizer(stop_words='english', analyzer='char_wb',
                      ngram_range=(2,5))
clf = LogisticRegressionCV()
pipe = make_pipeline(vec, clf)
pipe.fit(df2.Consumer_complaint, df2.Product)
print_report(pipe)
```

На самом деле это не улучшило объяснение. Как видно на рис. 7.21, только слово «ипотека» является важным. Остальные слова выделены очень светло-зеленым цветом. Даже красный – очень слабый.

```
eli5.show_prediction(clf, np.array(df2.Consumer_complaint)[0], vec=vec,
                     target_names=set(df2.Product))
```

Mortgage (probability 0.720, score 3.996) top features

Contribution?	Feature
+3.422	Highlighted in text (sum)
+0.575	<BIAS>

cfpb, i currently have a bank of america mortgage and am unable to contact " customer service " for any assistance on my loan or impound account. on xxxx xxxx, xxxx, i was " on hold " xxxx hours trying to reach a representative to ask a simple question about my impound account and finally got disconnected from their phone bank. this is unacceptable. i am calling about my home. i am a customer. the use of a so called " customer service " number by bank of america is hypocrisy because what is going on is instead customer abuse. why should i have to wait xxxx hours during my work week to try to speak with a representative about my home mortgage? they should be required to have a call back service or something that does not hold customers xxxx for hours to their negligence, disregard and abuse of customers. please advise what you plan to do about this. sincerely, customer

Рис. 7.21. Выделен текст характеристик

Чтобы позаботиться о длине словарного запаса, можно использовать векторизатор хеширования со стохастической моделью градиентного повышения в качестве классификатора. Точность еще больше снизилась – до 85.7 %. Тем не менее вы можете проверить интерпретацию.

```
from sklearn.feature_extraction.text import HashingVectorizer
from sklearn.linear_model import SGDClassifier

vec = HashingVectorizer(stop_words='english', ngram_range=(1,2))
clf = SGDClassifier(random_state=42)
pipe = make_pipeline(vec, clf)
pipe.fit(df2.Consumer_complaint, df2.Product)
print_report(pipe)
```

Таблица 7.11. Отчет о многоклассовой классификации после применения векторизатора хеширования

	Точность	Отзыв	f1-score	Поддержка
Банковский счет или услуга	0.97	0.47	0.63	441
Расчетный или сберегательный счет	0.77	0.79	0.78	849
Потребительский кредит	0.98	0.49	0.66	296
Кредитная карта или карта предоплаты	0.82	0.83	0.83	2000
Кредитная отчетность, ремонт или другое	0.86	0.93	0.89	8231
Взыскание долгов	0.86	0.82	0.84	4075
Денежные переводы, виртуальная валюта или денежные услуги	0.91	0.73	0.81	358
Денежные переводы	1.00	0.68	0.81	53
Ипотека	0.86	0.96	0.91	2216
Другие финансовые услуги	0.50	0.14	0.22	7
Платежный кредит, титульный кредит или персональный кредит	0.95	0.54	0.69	312
Студенческий кредит	0.91	0.84	0.87	788
Лизинг или кредит на транспортное средство	0.93	0.52	0.66	374
Аккуратность			0.86	20 000
Средний макро	0.87	0.67	0.74	20 000
Взвешенный макро	0.86	0.86	0.85	20 000
Аккуратность: 0.857				

В табл. 7.11 показана точность отчета о многоклассовой классификации.

Хотя общая точность модели снизилась, оценка вероятности увеличилась до 78 %. Однако только доминирующая характеристика «ипотека» лидирует в классификации. Остальные не имеют значения. Это предвзятая модель. Другие ключевые слова имеют очень низкие веса. Интенсивность зеленого и красного цвета определяет значение веса. Например, «ипотека» имеет высокий вес, потому что она темно-зеленая. У всех остальных слов светло-зеленый оттенок (рис. 7.22).

```
eli5.show_prediction(clf, np.array(df2.Consumer_complaint)[0], vec=vec,
                     target_names=set(df2.Product))

y=Mortgage (score 0.780) top features
Contribution? Feature
+2.050 Highlighted in text (sum)
-1.270 <BIAS>
```

dear cfpb, i currently have a bank of america mortgage and am unable to contact " customer service " for any assistance on my loan or impound account. on xxxx xxxx, xxxx, i was " on hold " xxxx hours trying to reach a representative to ask a simple question about my impound account and finally got disconnected from their phone bank. this is unacceptable. i am calling about my home. i am a customer. the use of a so called " customer service " number by bank of america is hypocrisy because what is going on is instead customer abuse. why should i have to wait xxxx hours during my work week to try to speak with a representative about my home mortgage? they should be required to have a call back service or something that does not hold customers xxxx for hours to their negligence, disregard and abuse of customers. please advise what you plan to do about this. sincerely, customer

Рис. 7.22. Более уточненные и лучше выделенные характеристики для прогнозирования класса ипотеки

```
from eli5.sklearn import InvertableHashingVectorizer
import numpy as np

ivec = InvertableHashingVectorizer(vec)
sample_size = len(df2.Consumer_complaint) // 10
X_sample = np.random.choice(df2.Consumer_complaint, size=sample_size)
ivec.fit(X_sample)
eli5.show_weights(clf, vec=ivec, top=20,
                  target_names=set(df2.Product))
```

Используя инвертируемый векторизатор хеширования, можно получить названия характеристик и их относительные веса в прогнозе класса (см. табл. 7.12–7.14).

Таблица 7.12. Положительные и отрицательные характеристики (с их весами) для соответствующих классов в задаче многоклассовой классификации

y=Bank account or service top features		y=Checking or savings account top features		y=Consumer Loan top features		y=Credit card or prepaid card top features	
Weight ⁷	Feature	Weight ⁷	Feature	Weight ⁷	Feature	Weight ⁷	Feature
+1.096	requirements	+2.169	branch	+0.432	vehicle	+4.756	card
+0.725	promotion	+2.145	deposit	+0.409	finance ...	+2.565	purchase
+0.661	citigold	+1.696	bonus	+0.404	car	+2.422	capital
+0.658	overdraft	+1.641	bank	+0.403	auto loan	+2.404	minimum
+0.649	miles	+1.591	deposited	+0.394	dealer	+2.228	discover
+0.633	met	+1.447	funds	+0.380	vehicle xxxx...	+2.187	synchrony
+0.629	fee	+1.433	checking...	+0.375	FEATURE[503180)	+2.154	express
+0.608	citi	+1.414	atm	+0.350	bmw	+2.144	cards
+0.573	FEATURE[428730]	+1.351	savings	+0.346	corporation	+2.015	citi
+0.570	2016		account	+0.336	xxxx vehicle	+1.987	amex
+0.520	debit	+1.341	savings	+0.325	contract	+1.934	charges
+0.519	bank	+1.294	debit	+0.291	door...	+1.921	visa
+0.511	xx 2016	+1.283	chase	+0.273	come	+1.825	purchases
+0.505	checks	+1.229	debit card	+0.271	car repossessed	+1.816	american
+0.504	negative	... 65301 более позитивный...		+0.263	westlake financial	+1.816	express...
+0.499	charge	... 75832 более негативный 45800 более позитивный...		+1.779	citibank
+0.497	branch	-1.147	mortgage	... 60839 более негативный...		... 108311 более позитивный...	
... 51135 более позитивный...		-1.156	transfer...	-0.294	don 117322 более негативный...	
... 64901 более негативный...		-1.311	paypal	-0.319	didn	-2.063	experian...
-0.516	didn	-1.469	2016	-0.340	im	-2.299	mortgage
-1.254	<BIAS>	-1.506	<BIAS>	-0.341	mortgage	-2.373	debit card
-2.153	FEATURE[301946]	-1.676	credit	-1.134	<BIAS>	-2.492	loan
		-1.929	payment...			-2.579	debit

Таблица 7.13. Положительные и отрицательные характеристики (с их весами) для соответствующих классов в задаче многоклассовой классификации

y=Credit reporting, repair, or other top features		y=Debt collection top features		y=Money transfer, virtual currency, or money service top features		y=Money transfers top features	
Weight ⁷	Feature	Weight ⁷	Feature	Weight ⁷	Feature	Weight ⁷	Feature
+6.313	experian ...	+5.378	debt...	+4.269	coinbase	+0.411	wire
+6.148	equifax	+3.592	collection	+1.652	app	+0.398	western
+5.390	transunion	+3.478	collect	+1.593	paypal	+0.394	western union
+3.496	inquiries	+3.426	owe	+1.314	transfer...	+0.369	sent money
+2.969	report	+2.883	calling ...	+1.228	cash app	+0.306	money...
+2.916	reporting	+2.685	collections	+1.150	tickets	+0.302	union
+2.775	inquiry	+2.553	hospital	+0.898	transaction ...	+0.267	service
+2.500	00 xxxx...	+2.477	owed ...	+0.881	wallet...	+0.265	paypal
+2.330	xxxx account	+2.324	recovery	+0.846	buyer	+0.264	ebay
+2.128	removed	+2.192	medical	+0.823	moneygram	+0.257	refused ...
+1.998	freeze	+2.078	calls	+0.798	support	+0.254	send
+1.965	late	+1.972	midland ...	+0.766	withdraw	+0.238	bank xxxx
+1.939	inaccurate	+1.676	lie	+0.755	cash	+0.236	tracking
... 135798 более позитивный 123540 более позитивный...		+0.706	money...	+0.234	sent...
... 146309 более негативный...		... 136030 более негативный...		+0.692	usd	+0.215	seller
-2.006	capital	-1.656	car	+0.670	use xxxx	+0.210	transfer...
-2.017	notice	-1.665	customer	... 40678 более позитивный 10299 более позитивный ...	
-2.039	calls	-2.380	mortgage	... 51535 более негативный...		... 16171 более негативный...	
-2.057	bank	-2.525	loan	-0.756	card	-0.224	FEATURE[628860]
-2.110	owe	-2.910	transunion	-0.758	credit	-0.252	case...
-2.185	collection.	-3.192	equifax	-0.792	FEATURE[537929]	-0.256	FEATURE[14521]
-3.499	debt..	-3.751	experian ...	-1.228	<BIAS>	-1.182	<BIAS>

Таблица 7.14. Положительные и отрицательные характеристики (с их весами) для соответствующих классов в задаче многоклассовой классификации

y=Mortgage top features		y=Other financial service top features		y=Payday loan, title loan, or personal loan top features		y=Student loan top features		y=Vehicle loan or lease top features	
Weight ²	Feature	Weight ²	Feature	Weight ²	Feature	Weight ²	Feature	Weight ²	Feature
+7.850	mortgage	+0.180	lower	+0.836	00 loan	+6.329	navient	+0.971	car
+5.020	escrow	+0.120	help	+0.810	payday	+3.108	loans	+0.688	santander
+4.525	modification	+0.114	money...	+0.510	cash	+3.027	student	+0.611	toyota
+3.341	home	+0.104	irs	+0.484	loan	+2.066	repayment	+0.603	dmv
+2.990	foreclosure	+0.087	karma	+0.465	lending	+1.920	aes	+0.557	auto
+2.959	closing	+0.087	credit karma	+0.459	borrowed	+1.877	school	+0.548	vehicle
+2.955	appraisal	+0.086	credit score	+0.459	taking	+1.758	student loan	+0.526	nissan
+2.848	refinance...	+0.081	return	+0.429	apr	+1.575	private	+0.492	ally
+2.627	ocwen	+0.081	process	+0.415	payday loan	+1.436	forbearance	+0.485	motor
+2.621	ditech	+0.079	forced	+0.377	store	+1.397	forgiveness	+0.471	lease ...
+2.548	servicing	+0.079	creditors	+0.376	payoff	+1.375	education	+0.452	title
+2.403	properly	+0.078	promised	+0.357	state	+1.190	nelnet	+0.431	acceptance
+2.231	lender	+0.076	hard ...	+0.336	800	+1.177	paying	+0.390	FEATURE(800554]
+2.132	house	+0.076	said	+0.355	800 00	+1.176	income	+0.385	ford
+2.055	sale	+0.073	credit	+0.335	personal loan	+1.144	plan	+0.379	dealer
+1.914	nationstar	+0.071	td	+0.334	working	... 61062 более позитивный...		+0.374	extra
... 110314 более позитивный...		+0.071	attorney	+0.334	lady	... 70420 более негативный...		+0.373	financial
... 118461 более негативный...		+0.071	power	... 44935 более позитивный...		-1.090	modification	... 51906 более позитивный...	
-1.921	student	+0.071	attorney	... 59073 более негативный...		-1.194	report	... 64554 более негативный...	
-1.981	car	+0.071	power	-0.457	student	-1.221	<BIAS>	-0.414	home
-2.054	debt...	... 1925 более позитивный...		-0.585	mortgage	-1.309	credit	-0.506	mortgage
-2.092	reporting	... 3705 более негативный...		-1.184	<BIAS>	-2.083	mortgage	-1.203	<BIAS>
		-1.098	<BIAS>						

ЗАКЛЮЧЕНИЕ

Классификация текстов является наиболее важным примером использования в области обработки естественного языка. Она полезна для классификации жалоб потребителей, классификации необработанного текста на различные сущности, а также классификации настроений на позитивные и негативные. Речь идет как о бинарной, так и о многоклассовой классификации. Другими вариантами ее использования являются моделирование тем и резюмирование, которые были включены в эту главу, поскольку эти задачи не относятся к моделям контролируемого машинного обучения. Мы рассматривали только модели контролируемого машинного обучения, потому что интерпретация обычно теряется, когда генерируется результат прогнозирования класса. Пользователь, очевидно, будет думать о том, какие лексемы или характеристики рассматриваются и почему предсказанный класс такой-то и такой-то, и т. д. Однако для задач, основанных на неконтролируемом обучении в области NLP, существуют более простые способы для понимания отношений, но в неконтролируемом обучении нет прогнозов, которые необходимо объяснить конечному пользователю. В этой главе мы больше сосредоточились на различных методах предварительной обработки, моделях и их объяснимости.

ГЛАВА 8

Справедливость модели ИИ, использующей сценарий «что, если»

В этой главе объясняется использование инструмента «что, если» (What-If Tool – WIT) для объяснения предвзятости в моделях ИИ, таких как модели регрессии на основе машинного обучения, модели классификации и многоклассовые модели классификации. Как специалист по исследованию данных, вы отвечаете не только за разработку модели машинного обучения, но и за обеспечение того, чтобы модель не была предвзятой и чтобы новые наблюдения обрабатывались беспристрастно. Крайне важно проверять решения и проверять справедливость алгоритмов. Google разработал инструмент «что, если» для решения проблемы справедливости модели в моделях машинного обучения. Мы рассмотрим реализацию WIT в трех возможных моделях ML: ML для задач, основанных на регрессии, ML для бинарной классификации и ML для полиноминальных моделей.

Что такое WIT?

Инструмент What-If Tool (WIT) – это инструмент с открытым исходным кодом, выпущенный компанией Google в 2019 году для тестирования моделей машинного обучения. Он был разработан для понимания причинно-следственной связи между драйверами и итоговыми переменными, а также изменениями в данных и переменными результата. Под драйверами здесь подразумеваются независимые предикторы для переменных результата. Благодаря простоте использования, привлекательной визуализации и объяснимости инструмент WIT получил широкое признание и распространение. При минимальном кодировании пользователь может визуально исследовать поведение любой модели машинного обучения. Моделирование различных входных данных для обученной модели машинного обучения необходимо для разработки объяснимого и ответственного ИИ. WIT доступен в трех форматах – обычном Jupyter Notebook из окна Anaconda Navigator, Google Colab Notebook, а также визуализации TensorBoard для любой модели на основе глубокого обучения. В настоящее время доступна версия инструмента 1.8.0, и она преодолевает некоторые ограничения предыдущих версий (см. рис. 8.1).



Рис. 8.1. Блок-схема выявления предвзятости

Обучающие данные могут быть разделены на основе определенных защищенных атрибутов исходя из вашего вопроса. В качестве примера, допустим, у вас есть раса и пол как два атрибута и вам нужен ответ с точки зрения объяснимости, например:

- является ли модель предвзятой по отношению к определенной расе;
- является ли модель предвзятой по отношению к определенному полу, предполагая наличие мужчин и женщин.

Возьмем второй пункт – пол – как защищенную переменную. Первый пол является привилегированным, а второй – непривилегированным. Вы можете подготовить две обучающие модели и, используя их, генерировать прогнозы для обучающих данных. Если для какой-либо конкретной записи прогнозы обеих моделей совпадают, то эта запись или точка данных является непредвзятой; если они отличаются, значит – предвзятой. На основании этого можно сделать вывод о том, является ли модель предвзятой по отношению к полу или расе.

УСТАНОВКА WIT

Здесь приведен код для установки инструмента What-If Tool:

```

import pandas as pd
import warnings
warnings.filterwarnings("ignore")

!pip install witwidget

from witwidget.notebook.visualization import WitConfigBuilder
from witwidget.notebook.visualization import WitWidget

```

Прежде чем запускать функцию `WitWidget()` из среды блокнота (Notebook), пожалуйста, выполните следующие две строки из командной строки:

```
jupyter nbextension install --py --symlink --sys-prefix witwidget
jupyter nbextension enable --py --sys-prefix witwidget
```

После завершения установки необходимо создать следующие функции, чтобы подготовить входные данные для создания модели и визуализации. Следующий скрипт создает спецификации характеристик на основе спецификации TensorFlow, поскольку инструмент WIT зависит от бэкенда TensorFlow для форматирования входных данных:

```

!pip install xgboost

# если вы получили ошибку в xgboost
# запустите

conda install -c conda-forge xgboost

import pandas as pd
import xgboost as xgb
import numpy as np
import collections
import witwidget

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.utils import shuffle
from witwidget.notebook.visualization import WitWidget, WitConfigBuilder

```

Мы будем использовать набор данных по оттоку, так как использовали его в других главах.

```

df = pd.read_csv('ChurnData.csv')
df.head()

del df['Unnamed: 0']
df.head()

import pandas as pd
import numpy as np
import tensorflow as tf
import functools

```

```
# Создает спецификацию характеристики tf из указанного датафрейма и столбцов
def create_feature_spec(df, columns=None):
    feature_spec = {}
    if columns == None:
        columns = df.columns.values.tolist()
    for f in columns:
        if df[f].dtype is np.dtype(np.int64):
            feature_spec[f] = tf.io.FixedLenFeature(shape=(), dtype=tf.int64)
        elif df[f].dtype is np.dtype(np.float64):
            feature_spec[f] = tf.io.FixedLenFeature(shape=(), dtype=tf.float32)
        else:
            feature_spec[f] = tf.io.FixedLenFeature(shape=(), dtype=tf.string)
    return feature_spec

# Создает простые числовые и категориальные столбцы характеристик из
# спецификации характеристики
# и список столбцов из этой спецификации для использования.
#
# ПРИМЕЧАНИЕ: Модели могут работать лучше с некоторыми техническими
# характеристиками, такими как числовые столбцы с привязкой
# и хеш-привязка/встраивание столбцов для категориальных характеристик.

def create_feature_columns(columns, feature_spec):
    ret = []
    for col in columns:
        if feature_spec[col].dtype is tf.int64 or feature_spec[col].dtype is tf.float32:
            ret.append(tf.feature_column.numeric_column(col))
        else:
            ret.append(tf.feature_column.indicator_column(
                tf.feature_column.categorical_column_with_vocabulary_
                list(col, list(df[col].unique()))))
    return ret
```

В приведенной выше функции, имея набор данных, вы проверяете входные данные и формируете целочисленные, плавающие и строковые столбцы. Следующая служебная функция генерирует входные данные из примеров на основе TensorFlow для использования на этапе обучения модели:

```
# Входная функция для предоставления входных данных для модели из tf.Examples
def tfexamples_input_fn(examples, feature_spec, label, mode=tf.estimator.ModeKeys.EVAL,
                       num_epochs=None,
                       batch_size=64):
```

```

def ex_generator():
    for i in range(len(examples)):
        yield examples[i].SerializeToString()
dataset = tf.data.Dataset.from_generator(
    ex_generator, tf.dtypes.string, tf.TensorShape([]))
if mode == tf.estimator.ModeKeys.TRAIN:
    dataset = dataset.shuffle(buffer_size=2 * batch_size + 1)
dataset = dataset.batch(batch_size)
dataset = dataset.map(lambda tf_example: parse_tf_example(tf_example,
label, feature_spec))
dataset = dataset.repeat(num_epochs)
return dataset

```

Следующая функция анализирует пример функции TensorFlow, используя спецификации характеристик, определенные функцией спецификации характеристик:

```

# Преобразование прототипов Tf.Example в характеристики для входной функции
def parse_tf_example(example_proto, label, feature_spec):
    parsed_features = tf.io.parse_example(serialized=example_proto,
features=feature_spec)
    target = parsed_features.pop(label)
    return parsed_features, target

```

Модель на основе TensorFlow требует пакетной загрузки входных данных для шага обучения.

```

# Преобразование датафрейма в список прототипов tf.Example
def df_to_examples(df, columns=None):
    examples = []
    if columns == None:
        columns = df.columns.values.tolist()
    for index, row in df.iterrows():
        example = tf.train.Example()
        for col in columns:
            if df[col].dtype is np.dtype(np.int64):
                example.features.feature[col].int64_list.value.
append(int(row[col]))
            elif df[col].dtype is np.dtype(np.float64):
                example.features.feature[col].float_list.value.
append(row[col])
            elif row[col] == row[col]:
                example.features.feature[col].bytes_list.value.
append(row[col].encode('utf-8'))
        examples.append(example)
    return examples

```

Следующая функция преобразует датафрейм, предоставленный пользователем, в столбец двоичных значений (0 и 1). Функция преобразует столбец меток в числовой для бинарной модели классификации. Аналогично для модели многоклассовой классификации столбец меток или целевой столбец также преобразуется в столбец кодировщика меток путем применения функции кодировщика меток.

```
# Преобразует столбец датафрейма в столбец из 0 и 1 на основе предоставленного
# теста.
# Используется для преобразования столбцов меток в числовые столбцы для
# бинарной классификации с
# использованием оценщика TF.
def make_label_column_numeric(df, label_column, test):
    df[label_column] = np.where(test(df[label_column]), 1, 0)
```

Случай использования, который мы рассматриваем в этой главе, – это сценарий классификации оттока клиентов в телекоммуникационной отрасли. Учитывается история телекоммуникационных клиентов, в частности абонентский стаж, код региона, в котором находится клиент, код штата, включенные или не включенные международные вызовы, план голосовой почты, число сообщений голосовой почты, общее время разговора за день в минутах, общее количество звонков в дневное время и общая сумма оплаты в дневное время. Аналогично вечерние иочные данные об использовании также считаются характеристиками. Они помогают спрогнозировать возможность оттока в будущем. Если вы можете предсказать правильный исход заранее, то можно устраниить факторы, приводящие к возможному оттоку, и удержать клиента. Удержание клиента всегда предпочтительнее, чем приобретение нового, так как это обходится дорого, а удержание существующих клиентов значительно дешевле. Таким образом, в процессе прогнозирования оттока вы можете разработать соответствующие стратегии, чтобы удержать клиента от оттока путем выявления причин оттока и проверки предубеждений и сценариев «что, если» заранее.

```
import numpy as np

# Установите столбец в наборе данных, который вы хотите, чтобы модель
# предсказывала
label_column = 'churn'

# Сделайте столбец меток числовым (0 и 1) для использования в нашей модели.
# В этом случае примеры с целевым значением "да" считаются относящимися
# к классу
# '1' (положительный), а все остальные примеры - к классу # '0'
# (отрицательный).
make_label_column_numeric(df, label_column, lambda val: val == 'yes')

# Установите список всех столбцов из набора данных, которые мы будем
# использовать для входа в модель.
```

```

input_features = [
    'account_length', 'area_code', 'international_plan',
    'voice_mail_plan', 'number_vmail_messages', 'total_day_minutes',
    'total_day_calls', 'total_day_charge', 'total_eve_minutes',
    'total_eve_calls', 'total_eve_charge', 'total_night_minutes',
    'total_night_calls', 'total_night_charge', 'total_intl_minutes',
    'total_intl_calls', 'total_intl_charge',
    'number_customer_service_calls']

# Создайте список, содержащий все входные характеристики и столбец меток
features_and_labels = input_features + [label_column]

features_and_labels

examples = df_to_examples(df)

num_steps = 5000 #@param {type: "number"}

# Создайте спецификацию характеристик для классификатора
feature_spec = create_feature_spec(df, features_and_labels)

# Определите и обучите классификатор
train_inpf = functools.partial(tfexamples_input_fn, examples, feature_spec,
label_column)

classifier = tf.estimator.LinearClassifier(
    feature_columns=create_feature_columns(input_features, feature_spec))

classifier.train(train_inpf, steps=num_steps)

test_df = pd.read_csv('churnTest.csv')

num_datapoints = 2000

tool_height_in_px = 700

from witwidget.notebook.visualization import WitConfigBuilder
from witwidget.notebook.visualization import WitWidget

make_label_column_numeric(test_df, label_column, lambda val: val == 'yes')

test_examples = df_to_examples(test_df[0:num_datapoints])

# Настройка инструмента с тестовыми примерами и обученным классификатором
config_builder = WitConfigBuilder(test_examples).set_estimator_and_feature_
spec(
    classifier, feature_spec).set_label_vocab(['Churn', 'No Churn'])

WitWidget(config_builder, height=tool_height_in_px)

```

Тестовый набор данных содержит 1667 образцов, которые вы визуализируете с помощью инструмента Google What-If Tool. Приведенный выше скрипт генерирует визуализацию с двумя панелями. Левая панель показывает различные оп-

ции, такие как редактор точек данных, производительность и справедливость модели, характеристики. Правая панель показывает визуализацию и другие опции.

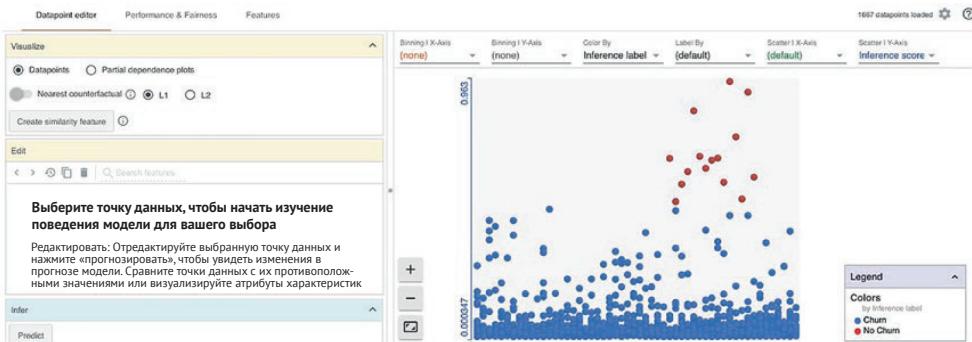


Рис. 8.2. Изучение поведения модели с помощью WIT

На рис. 8.2 в левом блоке находятся точки данных и графики частичных зависимостей, которые находятся в редакторе точек данных. Вторая вкладка содержит параметры производительности и справедливости, а третья – характеристики. На правой стороне ось у показывает оценку вывода и упорядочена по метке вывода. Красные точки – это клиенты без оттока, а синие – клиенты с оттоком. Здесь используется опция «метка по умолчанию».

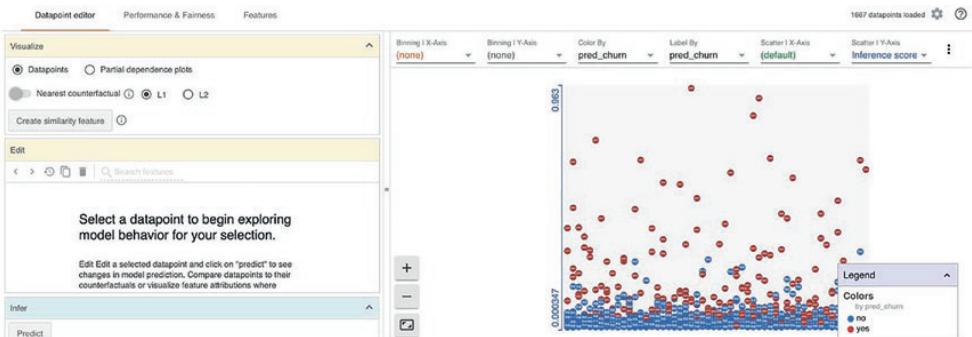


Рис. 8.3. Представление оттока и отсутствия оттока по типу прогнозирования

На рис. 8.3 синие точки – это случаи без оттока, а красные – с оттоком. Для записи № 1073 в тестовом наборе данных вероятность оттока составляет 0.016 %, а отсутствия оттока – 98.4 %, что является хорошим прогнозом. Однако, если посмотреть на пример № 799, вероятность оттока составляет 49.2 %, а отсутствия оттока – 50.4 %. Такие случаи весьма неоднозначны для модели. Модель не знает, как отличить случаи оттока и отсутствия оттока. По умолчанию инструмент WIT использует порог классификации 0.5. Если порог вероятности превышает 0.5, он предсказывает положительный класс, в противном случае – отрицательный. Здесь положительный класс – это отсутствие оттока, а отрицательный – отток. В правой части панели на диаграмме рассеяния показана оценка заключения, которая является оценкой вероятности. Если оценка за-

ключения больше 0.5, то это положительный класс, в противном случае – отрицательный. Есть несколько пограничных случаев, поэтому требуется дополнительная настройка модели. Таким образом, для получения лучшего результата очень важно точно настроить модель с помощью различных гиперпараметров (см. рис. 8.4).

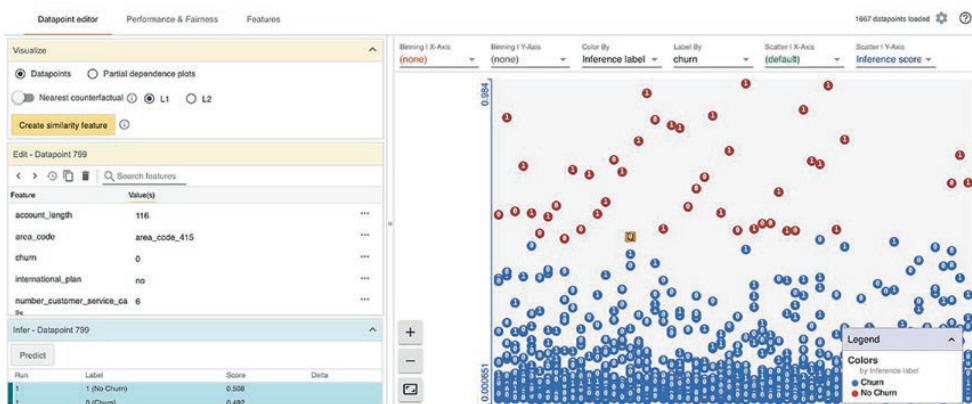


Рис. 8.4. Анализ точки данных 799

Редактор точек данных на левой панели открывает возможность просмотра и редактирования отдельных значений записей и показывает, что происходит с выводом прогноза. Это демонстрирует мощь вывода модели и дает подробное объяснение результатов прогноза. Для того же номера примера 799 если изменить абонентский стаж до 210, а общее количество звонков за день до 227 и некоторые другие параметры, а затем нажать кнопку **Predict** в левом боковом меню, то можно увидеть, что оценка оттока уменьшается с 49.2 до 25.8 %, а вероятность отсутствия оттока увеличивается с 50.8 до 74.2 %, что соответствует второму варианту моделирования (см. рис.8.5).



Рис. 8.5. Изменение оценки заключения из-за небольшого изменения значений характеристик

Вы также можете увидеть график частичной зависимости (PDP) отдельных характеристик. PDP вкратце показывает предельный вклад одной или двух характеристик в прогнозируемый результат модели машинного обучения.

На рис. 8.6 дано распределение случаев оттока и отсутствия оттока, предсказанное моделью. Если принять во внимание оценку заключения по оси X, то оценка вывода составляет менее 10 %. Все случаи предсказаны как отток. Здесь следует отметить, что при 49–59 % вы можете увидеть как случаи оттока, так и отсутствия оттока, что указывает на недостаточную прогностическую способность модели, потому что тогда ваша модель не способна различать классы churn и no-churn.

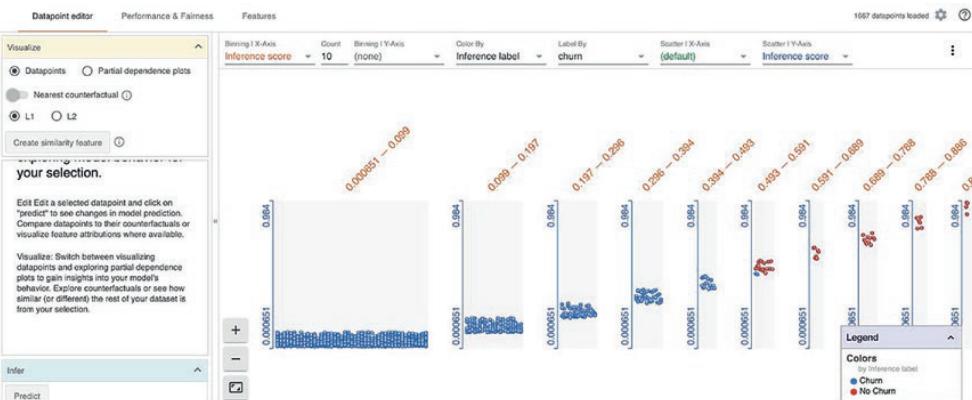


Рис. 8.6. Представление пакета оценок для прогнозирования оттока клиентов

На рис. 8.6 оценка вывода разбита на 10 групп, и представление показывает существование двух классов в каждой группе. Это поможет вам выбрать порог, который следует использовать для классификации целевого класса. Порог применяется к значению вероятности.

График частичной зависимости для всех независимых переменных, используемых в модели в качестве входных переменных, приведен на рис. 8.7. PDP отсортированы по вариациям.

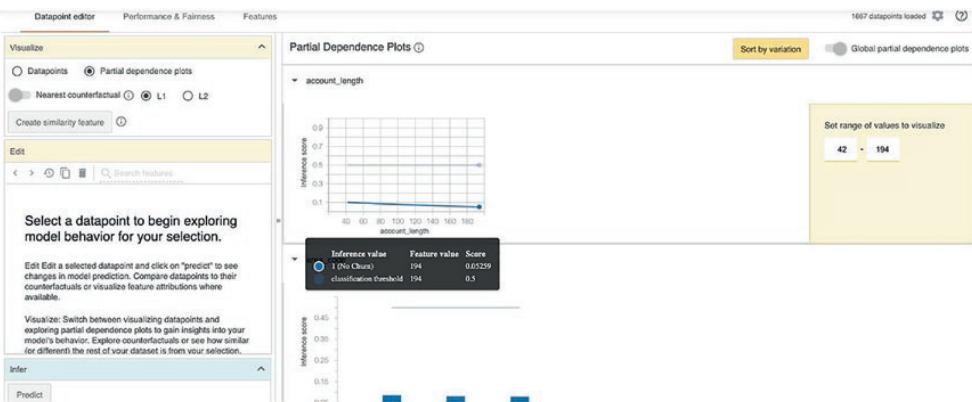


Рис. 8.7. График частичной зависимости для абонентского стажа

По мере увеличения абонентского стажа оценка вывода постепенно снижается (см. рис. 8.8).

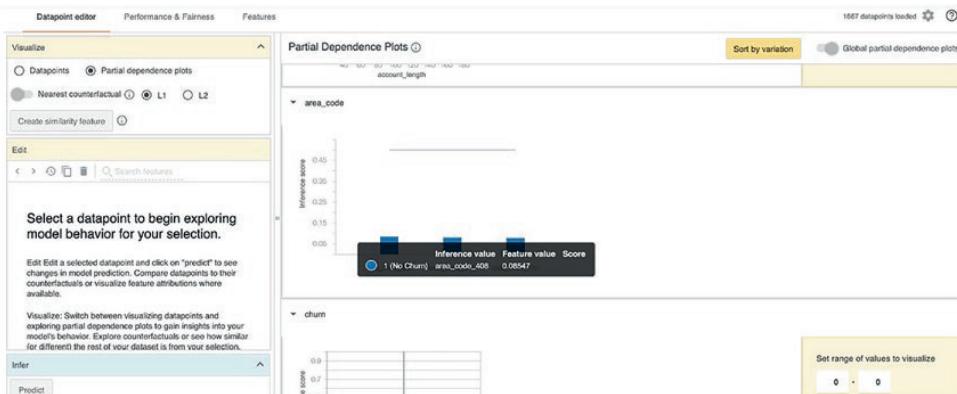


Рис. 8.8. PDP для переменной код региона

Оценивая общую эффективность модели, вы можете рассмотреть матрицу путаницы, которая иллюстрирует соответствие и несоответствие между случаями оттока и отсутствия оттока в вашей модели.

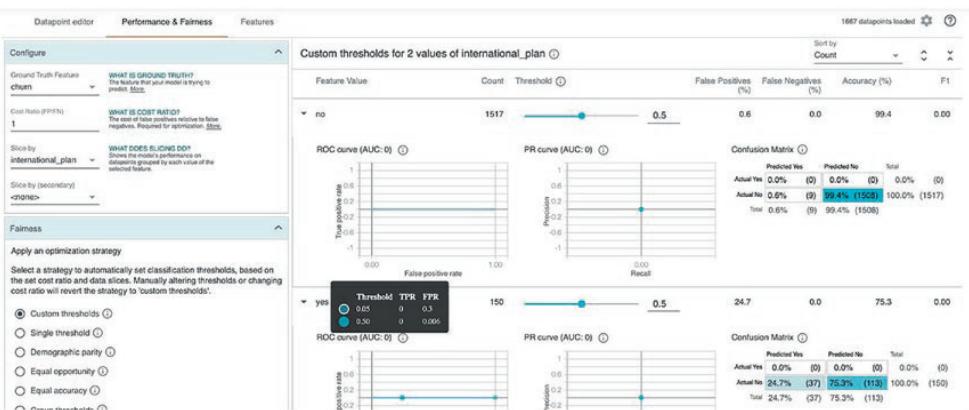


Рис. 8.9. Экран производительности и справедливости

На экране производительности и справедливости, показанном на рис. 8.9, вы можете увидеть ROC-кривую, кривую AUC, матрицу путаницы и соотношение затрат. Вы можете исправить пороговое значение вероятности, чтобы изменить матрицу классификации.

МЕТРИКА ОЦЕНКИ

Точность: Как часто ваша модель правильно предсказывает положительную метку.

Отзыв: Сколько положительных классов в наборе данных правильно предсказывает модель?

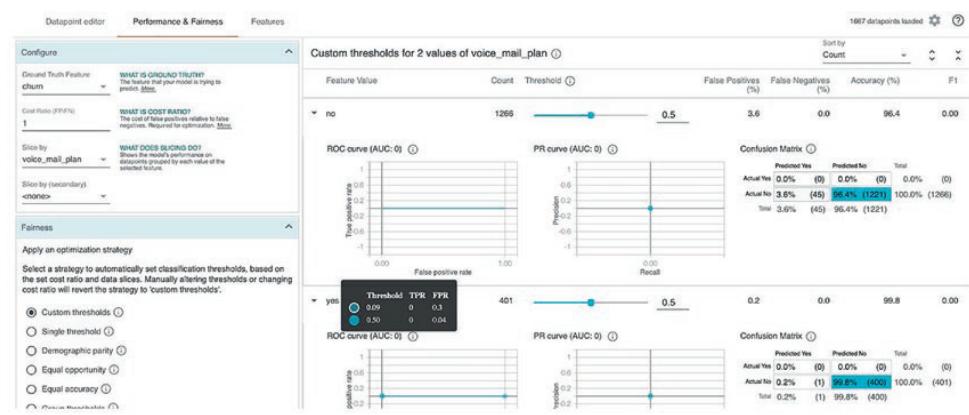


Рис. 8.10. Экран оценки, показывающий точность модели

ЗАКЛЮЧЕНИЕ

В этой главе обсуждалось, как интерпретировать результаты модели классификации с использованием инструмента What-If Tool и как определить характеристики, которые играют решающую роль в классификации случаев оттока и отсутствия оттока. Кроме того, вы использовали методы локальной интерпретации отдельных точек данных.

ГЛАВА 9

Объяснимость для моделей глубокого обучения

Модели на основе глубоких нейронных сетей постепенно становятся основой искусственного интеллекта и машинного обучения. Будущее добычи данных будет определяться использованием передовых методов моделирования на основе искусственных нейронных сетей. Итак, почему нейронные сети приобретают такое большое значение, хотя они были изобретены в пятидесятых годах прошлого века? Заимствованные из области компьютерных наук, нейронные сети можно определить, как параллельную систему обработки информации, в которой входы связаны друг с другом для передачи информации, подобно нейронам в человеческом мозге, что позволяет выполнять такие действия, как распознавание лиц и изображений. В теории нейронные сети существуют уже более 50 лет, но реализация их проектов на практике стала возможной после определенных достижений в области вычислений, а именно эволюции графических процессоров (GPU) и тензорных процессоров (TPU) в выполнении высокопроизводительных вычислений, умножении крупномасштабных матриц и т. д. В этой главе вы узнаете о применении методов на основе нейронных сетей для решения различных задач интеллектуального анализа данных, таких как классификация, регрессия, прогнозирование и уменьшение числа характеристик. Искусственная нейронная сеть (artificial neural network – ANN) функционирует аналогично работе человеческого мозга, где миллиарды нейронов связаны друг с другом для обработки информации и генерации идей.

Объяснение моделей DL

Биологическая сеть мозга обеспечивает основу для соединения элементов в реальном сценарии для обработки информации и генерации идей. Иерархия нейронов соединена через слои, где выход одного слоя становится входом для другого. Информация передается от одного слоя к другому в виде весов. Веса, связанные с каждым нейроном, содержат информацию, благодаря которой распознавание и рассуждения для следующего уровня становятся проще. Искусственная нейронная сеть – это очень популярный и эффективный метод, состоящий из слоев, связанных с весами. Связь между различными слоями регулируется математическим уравнением, которое передает информацию от одного слоя к другому. На самом деле внутри одной модели искусственной нейронной сети работает множество уравнений. На рис. 9.1 показана общая

архитектура модели на основе нейронной сети, где отображены входной, выходной и скрытый слои.

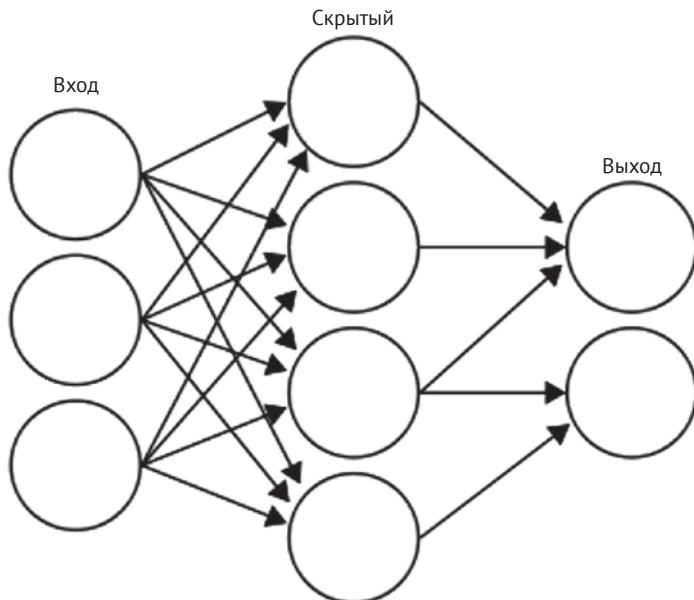


Рис. 9.1. Образец структуры нейронной сети

Существуют три слоя (входной, скрытый и выходной), и они являются ядром любой архитектуры нейронных сетей. ANN являются мощным инструментом для решения многих реальных задач, таких как классификация, регрессия и выбор характеристик. Они обладают способностью обучаться на новых входных данных, чтобы улучшить выполнение задач классификации или регрессии, а также адаптироваться к изменениям во входной среде. Каждый кружок на рис. 9.1 представляет собой нейрон. Одно из основных преимуществ глубокого обучения заключается в том, что нам не нужно сосредоточиваться на создании характеристик для обучения модели. Взаимодействие характеристик не обязательно создаваться человеком, оно может появляться автоматически в процессе обучения.

Существуют различные варианты нейронных сетей с алгоритмической или вычислительной точки зрения, которые используются во множестве различных сценариев. Ниже я собираюсь концептуально объяснить несколько из них, чтобы вы могли понять их использование в практических приложениях:

- **нейронная сеть с одним скрытым слоем** – это самая простая форма нейронной сети, как показано на рис. 9.1. В ней есть только один скрытый слой;
- **нейронные сети с несколькими скрытыми слоями** – в этой форме более чем один скрытый слой соединяет входные данные с выходными. Сложность вычислений в этой форме возрастает, и она требует больше вычислительной мощности системы для обработки информации;

- **нейронные сети с прямой передачей** – в этой форме архитектуры нейронной сети информация передается в одном направлении от одного слоя к другому, итерации с первого уровня обучения не происходит;
- **нейронные сети с обратным распространением** – в этой форме нейронной сети есть два важных этапа. Прямая передача работает путем передачи информации от входного слоя к скрытому и от скрытого – к выходному. Во-вторых, выходной слой вычисляет ошибку и передает ее предыдущим слоям.

Существует еще один вид классификации, основанный на использовании и структуре, которую можно объяснить следующим образом:

- **рекуррентная нейронная сеть** (recurrent neural network – RNN) – в основном используется для последовательной обработки информации, такой как обработка аудио, классификация текста и т. д.;
- **глубокая нейронная сеть** (deep neural network – DNN) – используется для решения задач с высокой размерностью структурированных данных, где взаимодействие характеристик настолько сложно, что очень утомительно строить каждую характеристику вручную;
- **конволюционная нейронная сеть** (convolutional neural network – CNN) – в основном используется для данных изображений, где размеры пикселей создают матрицу высокой размерности. Возникает необходимость синтезировать информацию в свернутом виде для классификации изображений.

Архитектура модели нейронной сети с прямой передачей показана на рис. 9.2, а метод обратного распространения объясняется в следующем разделе.

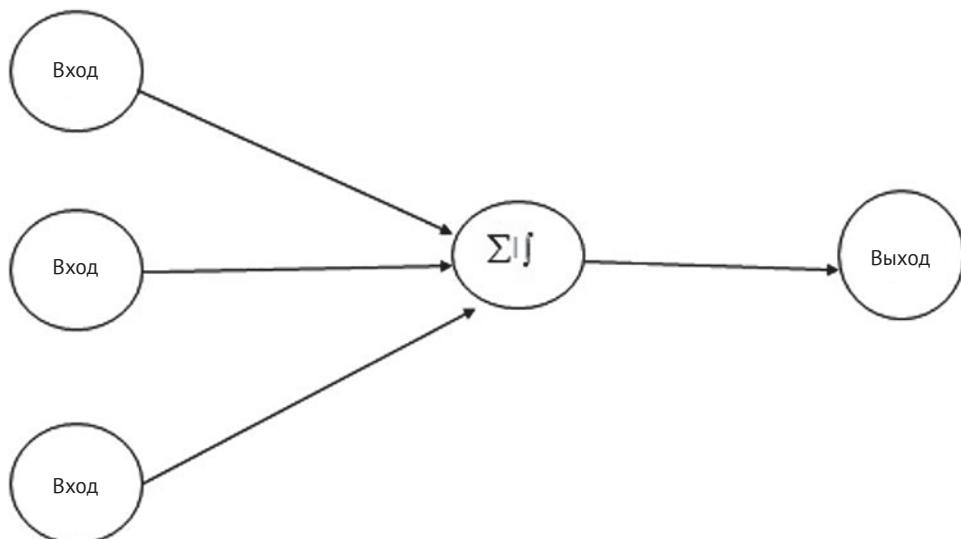


Рис. 9.2. Базовая структура NN с прямой передачей

Использование SHAP с DL

Как уже упоминалось в предыдущих главах, SHAP – это отличная библиотека для всех видов объяснений моделей, локальной и глобальной интерпретации моделей, причем интерпретация происходит из значения Шепли (*Shapely*). Важно знать, что значение Шепли полностью зависит от теоретико-игрового подхода. Следующая формула объясняет, как происходит вычисление значения Шепли внутри библиотеки:

$$\phi_i(v) = \frac{1}{|N|!} \sum_R [v(P_i^R \cup \{i\}) - v(P_i^R)],$$

где ϕ – значение Шепли;

N – количество игроков (характеристик);

P_i^R – набор игроков с заказом;

$v(P_i^R)$ – вклад набора игроков с заказом;

$v(P_i^R \cup \{i\})$ – вклад набора игроков с заказом и i -го игрока.

Использование Deep SHAP

Deep SHAP – это фреймворк для получения значений SHAP из модели, основанной на глубоком обучении, разработанной либо с использованием моделей на базе Keras, либо моделей на базе TensorFlow. Вы можете взять пример с набором данных MNIST, где в точке выборки данных отображаются значения пикселей в виде числа, а модель глубокого обучения обучается классифицировать изображение. Значения SHAP показаны ниже.

Если мы сравним модели машинного обучения с моделями глубокого обучения, то первые все еще поддаются интерпретации, но модели нейронных сетей, особенно глубоких, являются «черным ящиком» по своей природе, что представляется препятствием для принятия моделей ИИ в промышленности, потому что никто не может интерпретировать применение прогноза модели глубокого обучения. Deep Learning Important Features (DeepLIFT) – это фреймворк, который появился на свет в октябре 2019 года. Он предназначен для применения метода декомпозиции результата прогнозирования модели глубокой нейронной сети. Это делается путем обратного распространения вклада всех нейронов глубокой нейронной сети в каждую характеристику входного сигнала. Фреймворк DeepLIFT делает это возможным, сравнивая активацию каждого нейрона с их соответствующей активацией, и присваивает оценку, которая известна как оценка вклада.

Использование Alibi

Помимо DeepLIFT и Deep SHAP, существует еще одна библиотека с открытым исходным кодом под названием Alibi. Это библиотека на базе Python, предназначенная для объяснения моделей машинного обучения. Следующий код объясняет, как установить Alibi и как ускорить поиск Alibi с помощью другой библиотеки под названием Ray.

```
!pip install alibi
```

Приведенный выше код устанавливает библиотеку Alibi для объяснения моделей.

```
!pip install alibi[ray]
```

Библиотека `alibi[ray]` является зависимой библиотекой и также должна быть установлена.

```
!pip install tensorflow_datasets
```

Вы можете использовать `tensorflow_datasets`, чтобы собрать несколько наборов данных из библиотеки для использования в этой главе.

```
!pip install keras
```

Библиотека Keras предназначена для обучения моделей глубокого обучения.

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
import matplotlib.pyplot as plt
%matplotlib inline

batch_size = 128
num_classes = 10
epochs = 12

# размеры входного изображения
img_rows, img_cols = 28, 28
```

MNIST – это набор данных для классификации цифр, содержащий изображения цифр, написанные от руки и помеченные экспертами, что позволяет модели изучать образцы. `Conv2D` – это сверточный двумерный слой, преобразующий веса и выполняющий пространственную свертку над пикселями изображения. Значения, которые вы задаете в `Conv2D`, являются числом фильтров, на основе которых будут обучаться сверточные слои. После свертки требуется максимальное объединение для сжатого представления пикселей в более низкоразмерном пространстве. Следовательно, каждый сверточный слой нуждается в слое `max pool`. Слой `max pool` требует размера выходной матрицы. В следующем скрипте используется 32 фильтра в слое `Conv2D`, и размер ядра или размер фильтра составляет (3,3). Максимальное объединение обычно используется для уменьшения пространственных размеров выходного объема. Размер ядра должен быть комбинацией нечетных чисел, например (1,1), (3,3) или (5,5), для того чтобы фильтр ядра проходил через объемное пространство плавно. `Stride` – это

значение, которое заставляет фильтры перемещаться в пространстве пикселей. `stride=1` означает, что фильтр перемещается на одно место вправо в матрице пикселей. Достигнув края матрицы пикселей, он опускается вниз и продолжает двигаться до самой нижней возможной строки, а затем – влево. Когда вы перемещаете ядро с нечетным номером в пиксельном пространстве, чтобы собрать все свернутые характеристики, иногда фильтр не доходит до краев матрицы пикселей. Чтобы это произошло, вы добавляете строку и столбец вокруг исходной матрицы пикселей. Это называется добавлением. Добавление может быть нулевым и одинаковым.

```
# данные, разделенные на обучающий и тестовый наборы
(x_train, y_train), (x_test, y_test) = mnist.load_data()
if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

Приведенный выше скрипт показывает общие шаги для получения данных из библиотеки Keras и использования загруженных данных для нормализации обучающих и тестовых образцов. Это выполняется путем деления каждого отдельного значения пикселя на его наибольшее значение. После этого 60 000 образцов откладываются для обучения и 10 000 – для тестирования. Размер изображения здесь составляет 28×28 пикселей, размер пакета – 128, а количество классов – 10 для каждой цифры изображения одного класса. Кроме того, для обучения модели используется 10 итераций или эпох.

```
# преобразование векторов классов в бинарные матрицы классов
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                activation='relu',
                input_shape=input_shape))
```

```
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Модель классификации изображений, которая используется здесь в качестве примера, состоит из трех частей, плюс применение свертки и максимального объединения для уменьшения размерности. Использование Dropout и Flatten применяется для удаления весов с низкой вероятностью, чтобы оградить модель от переподгонки. Flatten добавляет слои для изменения формы данных для дальнейшего применения матричного перемножения. Также используется плотный слой для создания полностью подключенной модели нейронной сети, которая работает в низкоразмерном пространстве, чтобы классифицировать целевой класс с большей точностью. Функции активации – это передаточные функции, которые используются после точечного перемножения весов и признаков, полученных от предыдущего слоя. Чтобы получить вероятность для каждого класса, необходимо использовать функцию активации softmax на последнем слое модели нейронной сети.

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

После того как структура модели разработана, следующим шагом является компиляция модели. Шаг компиляции в библиотеке Keras имеет определенные параметры, важными из них являются функция потерь, оптимизатор и метрика. Кроме них, существуют и другие параметры, которые используются для тонкой настройки модели и ее переобучения, чтобы оградить модель от переподгонки. Выбор функции ошибки или функции потерь должен основываться на задаче. Если целевой столбец имеет более двух классов, то можно использовать категориальную перекрестную энтропию. Роль оптимизатора заключается в том, чтобы оптимизировать функцию потерь и определить шаг, на котором функция потерь минимальна. Функция метрики заключается в вычислении точности модели, соответствующей каждой эпохе.

```
model.fit(x_train, y_train,
           batch_size=batch_size,
           epochs=epochs,
           verbose=1,
           validation_data=(x_test, y_test))
```

Результаты по всем эпохам не могут быть представлены в журналах, поэтому они усечены, но следите за журналами, чтобы определить итерации, после

которых модель перестает обучаться или точность становится низкой и т. д. Из следующего результата следует, что тестовая точность модели классификации составляет 85.28 %.

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

import tensorflow as tf
tf.compat.v1.disable_v2_behavior() # запустите это, если получите ошибку,
                                 связанныю с тензором
```

Объяснитель SHAP для глубокого обучения

Библиотека SHAP имеет модуль объяснителя глубокого обучения, который содержит представление, позволяющее показать положительные и отрицательные атрибуты или вклад в классификацию в целом и для каждого класса в отдельности.

```
background = x_train[np.random.choice(x_train.shape[0], 100, replace=False)]
explainer = shap.DeepExplainer(model, background)

shap_values = explainer.shap_values(x_test[1:5])
# отображение атрибутов характеристик
shap.image_plot(shap_values, -x_test[1:5])
```

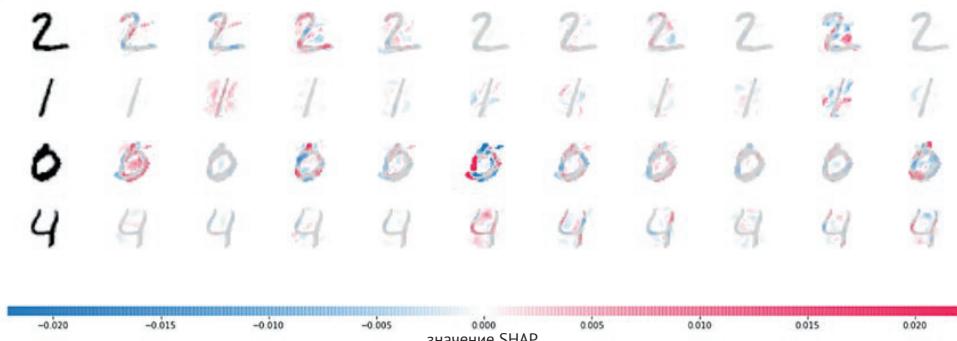


Рис. 9.3. DeepExplainer для классификации изображений

На рис. 9.3 значения, выделенные красным цветом, являются положительными классами для прогнозируемого класса, а выделенные синим – отрицательными. На рисунке показаны значения SHAP для 2, 1, 0 и 4. Модель генерирует 10 прогнозов, соответствующих этим четырем значениям изображений. Красные значения делают прогнозы близкими к входному изображению, показанному слева, а отрицательные уменьшают прогнозы по сравнению с входным изображением. Deep SHAP был разработан на основе статьи, опубликованной в NIPS.

DeepLIFT производит оценку значений SHAP, используя подход, аналогичный обратному распространению, после того как генерируется прогноз для модели классификации и регрессии.

```
# отображение атрибутов характеристик
shap.image_plot(shap_values, -x_test[5:9])
```

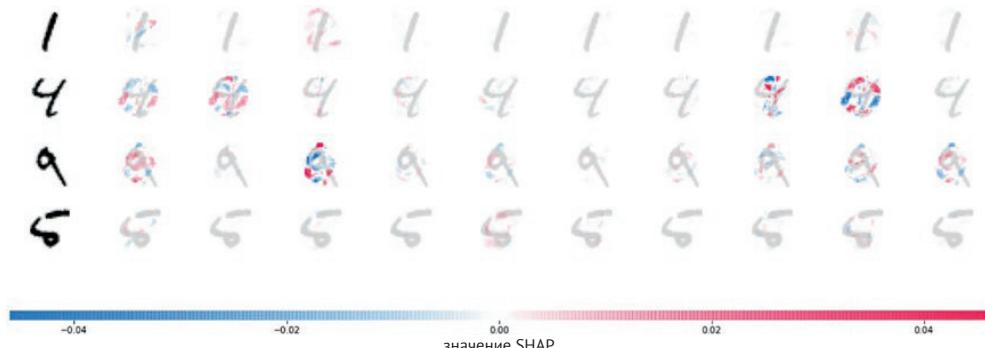


Рис. 9.4. DeepExplainer еще для нескольких записей

На рис. 9.4 напечатаны еще четыре цифры и изображены соответствующие им оценки SHAP. Если вы посмотрите на цифры 5 и 1, то увидите единообразие. Однако если посмотреть на цифры 4 и 9, то здесь есть неоднозначность, так как есть места, где оценка SHAP не помогает предсказать класс.

```
# отображение атрибутов характеристик
shap.image_plot(shap_values, -x_test[9:13])
```

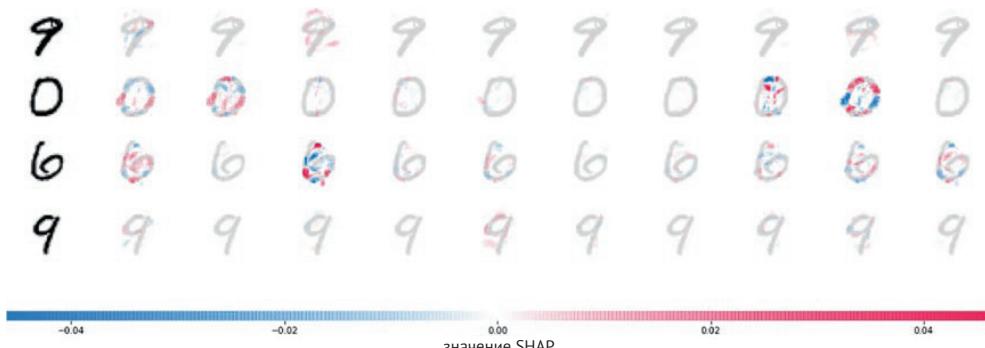


Рис. 9.5. DeepExplainer для еще четырех цифр

На рис. 9.5 то же самое можно увидеть на цифрах 0 и 6, так как обе, кажется, написаны очень похожим образом. Однако цифра 9 является достаточно четкой.

ЕЩЕ ОДИН ПРИМЕР КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

Набор данных CIFAR-10 взят с сайта www.cs.toronto.edu/~kriz/cifar.html. У нас есть выборка из 60 000 изображений, состоящая из цветных изображений 32×32 с 10 классами, которые необходимо прогнозировать или классифицировать. Эти классы: самолет, автомобиль, птица и т. д. Модели глубокого обучения можно объяснить с помощью двух объяснителей, DeepExplainer и GradientExplainer.

```
from keras.datasets import cifar10
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.optimizers import SGD, Adam, RMSprop

import matplotlib.pyplot as plt
# из quiver_engine import server
# CIFAR_10 - это набор из 60K изображений по 32x32 пикселя в 3 каналах
IMG_CHANNELS = 3
IMG_ROWS = 32
IMG_COLS = 32
#константа
BATCH_SIZE = 128
NB_EPOCH = 20
NB_CLASSES = 10
VERBOSE = 1
VALIDATION_SPLIT = 0.2
OPTIM = RMSprop()
```

Вы не можете взять весь набор образцов и пропустить их через конволюционный и плотный слой. Этот процесс очень медленный и трудоемкий, а также требующий больших вычислительных затрат. Следовательно, необходимо использовать небольшие порции образцов из обучающего набора и инкрементально обновлять веса после каждой эпохи. Поэтому размер порции – 128, количество эпох – 20, оптимизатор – RMSPROP, а размер проверочной выборки составляет 20 %. Это гиперпараметры, которые были использованы в качестве примера. Является ли это конфигурацией с наилучшим набором гиперпараметров? Возможно, нет, так как гиперпараметры могут быть установлены после многих итераций. Выбор наилучшего из них может быть сделан с помощью сеточного поиска или многократных итераций.

```
#загрузка набора данных
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
print('X_train shape:', X_train.shape)
print(X_train.shape[0], 'train samples')
```

```
print(X_test.shape[0], 'test samples')
# преобразование в категориальный
Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
Y_test = np_utils.to_categorical(y_test, NB_CLASSES)

# конвертация в числа с плавающей точкой и нормализация
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255

# сеть

model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=(IMG_ROWS, IMG_COLS, IMG_CHANNELS)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(NB_CLASSES))
model.add(Activation('softmax'))

model.summary()
```

Суммарное количество нейронов показано в столбце param. Существует 4 200 842 обучаемых параметра, и нет ни одного необучаемого.

```
# обучение
#optim = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=OPTIM,
               metrics=['accuracy'])
history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE,
                     epochs=NB_EPOCH, validation_split=VALIDATION_SPLIT,
                     verbose=VERBOSE)
print('Testing...')
score = model.evaluate(X_test, Y_test,
                       batch_size=BATCH_SIZE, verbose=VERBOSE)
print("\nTest score:", score[0])
print('Test accuracy:', score[1])
```

После обучения лучшей модели классификации вы можете сохранить объект модели для использования позднее при генерации выводов.

```
#сохранение модели
model_json = model.to_json()
open('cifar10_architecture.json', 'w').write(model_json)
model.save_weights('cifar10_weights.h5', overwrite=True)
# список всех данных в истории
print(history.history.keys())
# обобщение истории для точности
# plt.plot(mo)
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

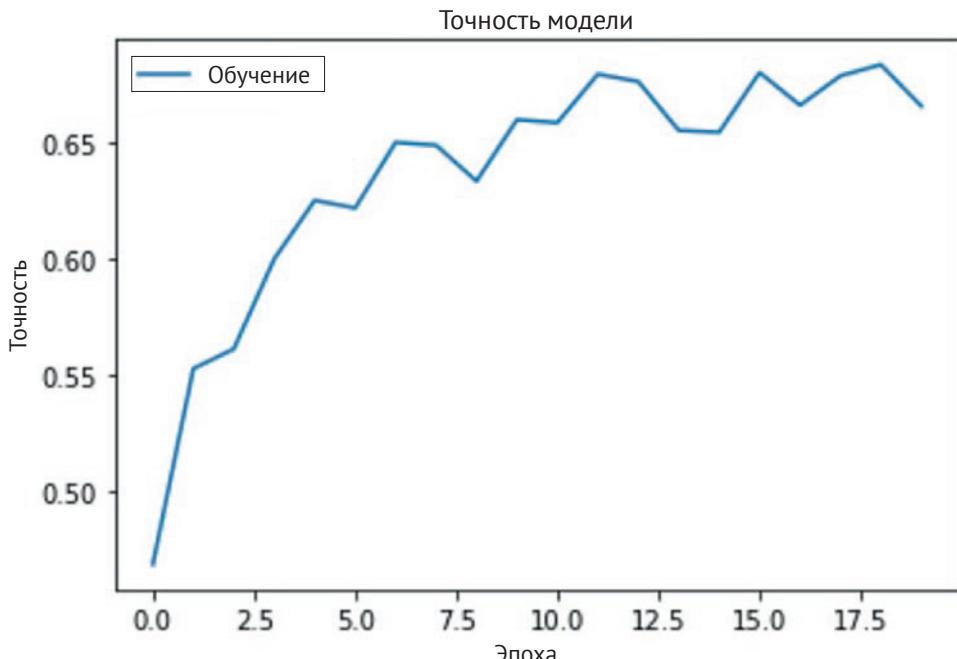


Рис. 9.6. Точность обучаемой модели для каждой эпохи

Из рис. 9.6 видно, что самая высокая точность обучения достигается в районе 10-й эпохи. После этого точность довольно нестабильна. Наблюдается зигзагообразная картина.

```
# обобщение истории по потерям
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
```

```
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

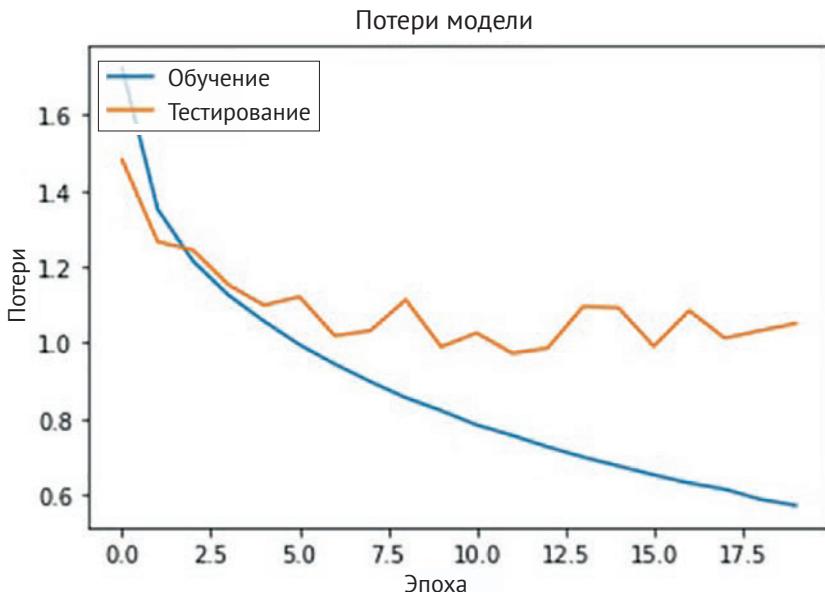


Рис. 9.7. Потери при обучении и тестировании модели

Из рис. 9.7 видно, что потери при обучении и тестировании модели идут рука об руку до 3-й эпохи. После этого значения расходятся, и потери при тестировании показывают зигзагообразную картину, что означает нестабильность потерь, а потери при обучении продолжают падать, что является явным признаком чрезмерной подгонки.

Прежде чем генерировать объяснения модели, важно обеспечить стабильную, точную и надежную модель. В противном случае очень трудно обосновать вывод, так как модель будет производить случайные выводы и объяснения.

Использование SHAP

После получения приемлемой модели глубокого обучения с хорошей точностью важно объяснить прогнозы модели. Кроме того, было бы интересно узнать, как оценки SHAP приводят к предсказанию класса (см. рис. 9.8 и табл. 9.1).

```
background = X_train[np.random.choice(X_train.shape[0],100, replace=False)]
explainer = shap.DeepExplainer(model,background)

shap_values = explainer.shap_values(X_test[10:15])

# отображение атрибутов характеристик
shap.image_plot(shap_values, -X_test[10:15])
```

```
import pandas as pd
pd.DataFrame(model.predict_classes(X_test)).head(10)
```

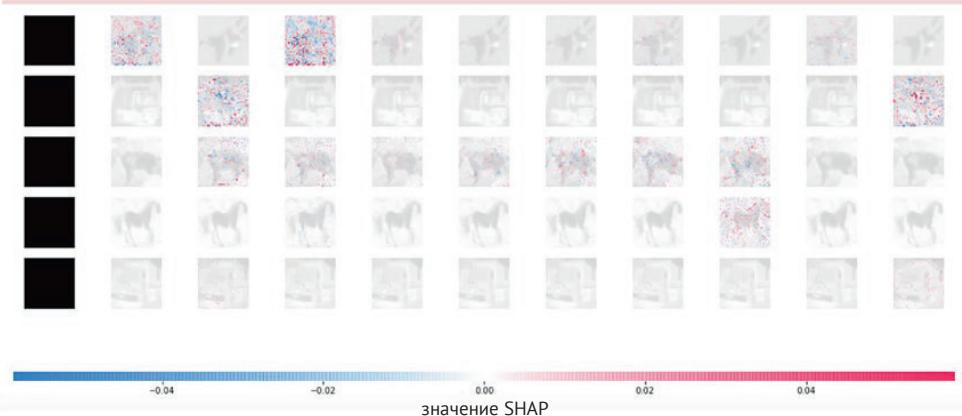


Рис. 9.8. Атрибуты SHAP характеристик

Таблица 9.1. Прогнозируемый класс для первых 10 записей

Номер строки	Прогнозируемый класс
0	3
1	8
2	8
3	0
4	6
5	6
6	1
7	6
8	3
9	1

В табл. 9.2 имена объектов, образцы изображений и числа в скобках показывают, как они закодированы в целевом классе приведенной выше модели.

```
import pandas as pd
pd.DataFrame(np.round(model.predict_proba(X_test),3)).head(10)
```

Таблица 9.2. Кодирование объектов в целевом классе приведенной выше модели

Самолет (0)										
Автомобиль (1)										
Птица (2)										
Кошка (3)										
Олень (4)										
Собака (5)										
Лягушка (6)										
Лошадь (7)										
Корабль (8)										
Грузовик (9)										

Таблица 9.3. Значения вероятности для каждого из классов, представленных в табл. 9.2

	0	1	2	3	4	5	6	7	8	9
0	0.000	0.001	0.002	0.928	0.007	0.053	0.000	0.000	0.008	0.000
1	0.000	0.005	0.000	0.000	0.000	0.000	0.000	0.000	0.995	0.000
2	0.164	0.013	0.005	0.001	0.001	0.000	0.000	0.001	0.797	0.019
3	0.661	0.002	0.105	0.001	0.003	0.000	0.001	0.000	0.226	0.000
4	0.000	0.000	0.037	0.073	0.289	0.002	0.598	0.000	0.000	0.000
5	0.001	0.000	0.035	0.033	0.023	0.024	0.877	0.003	0.000	0.003

Окончание табл. 9.3

	0	1	2	3	4	5	6	7	8	9
6	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	0.009	0.000	0.036	0.001	0.000	0.000	0.954	0.000	0.000	0.000
8	0.003	0.000	0.029	0.665	0.121	0.147	0.008	0.025	0.001	0.000
9	0.001	0.936	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.061

DEEP EXPLAINER ДЛЯ ТАБЛИЧНЫХ ДАННЫХ

Давайте применим модель глубокого обучения к сложному набору данных. Назовем это прогнозированием качества вина. Наилучшая точность, которую можно получить для этого набора данных, составляет 62 % из-за сложности и неоднозначности характеристик. Ни один алгоритм не может увеличить точность выше 70 или 80 %. Следовательно, это сложный набор данных. Существуют различные классы вин, которые необходимо классифицировать. Набор данных взят из репозитория машинного обучения UCI по адресу <https://archive.ics.uci.edu/ml/datasets/wine+quality>. В него включены два набора данных, относящихся к образцам красного и белого вина с севера Португалии. Цель состоит в том, чтобы смоделировать качество вина на основе физико-химических тестов. Два набора данных относятся к красному и белому вариантам португальского вина Винью Верде (Portuguese Vinho Verde). В связи с вопросами конфиденциальности и логистики доступны только физико-химические (входные) и сенсорные (выходные) переменные (например, нет данных о сортах винограда, марке вина, цене и т. д.). Эти наборы данных можно рассматривать как задачи классификации или регрессии. Классы упорядочены и несбалансированы (например, нормальных вин гораздо больше, чем отличных или плохих).

```
from tensorflow import keras
from sklearn.model_selection import cross_val_score, KFold
from keras.wrappers.scikit_learn import KerasRegressor
from keras.layers import Dense, Dropout
from keras.models import Sequential
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Приведенный выше скрипт импортирует библиотеки, необходимые для создания модели глубокого обучения для табличного набора данных. Следующий скрипт называет характеристики, которые могут быть использованы для прогнозирования качества вина:

```
feature_names = [  
    "fixed acidity",  
    "volatile acidity",  
    "citric acid",  
    "residual sugar",  
    "chlorides",  
    "free sulfur dioxide",  
    "total sulfur dioxide",  
    "density",  
    "pH",  
    "sulphates",  
    "alcohol",  
    "quality",  
]
```

Следующий скрипт считывает два набора данных из репозитория машинного обучения UCI:

```
red_wine_data = pd.read_csv(  
    'https://archive.ics.uci.edu/ml/machine-learning-databases/winequality/  
    winequality-red.csv', names=feature_names, sep=';', header=1)  
white_wine_data = pd.read_csv(  
    'https://archive.ics.uci.edu/ml/machine-learning-databases/winequality/  
    winequality-white.csv', names=feature_names, sep=';', header=1)  
wine_data = red_wine_data.append(white_wine_data)  
wine_features = wine_data[feature_names].drop('quality', axis=1).values  
wine_quality = wine_data['quality'].values  
  
scaler = StandardScaler().fit(wine_features)  
wine_features = scaler.transform(wine_features)
```

Модель глубокого обучения имеет входные данные из 11 характеристик. Это трехслойная глубокая нейронная сетевая модель.

```
model = Sequential()  
model.add(Dense(1024, input_dim=11, activation='tanh'))  
model.add(Dense(512, activation='tanh'))  
model.add(Dense(64, activation='tanh'))  
model.add(Dense(1))  
model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])  
  
model.summary()  
history = model.fit(wine_features, wine_quality,  
    batch_size=BATCH_SIZE,  
    epochs=N_EPOCH,  
    validation_split=VALIDATION_SPLIT,  
    verbose=VERBOSE)
```

```
background = wine_features[np.random.choice(wine_features.shape[0],100,  
replace=False)]  
explainer = shap.DeepExplainer(model,background)  
shap_values = explainer.shap_values(wine_features)  
  
pd.DataFrame(np.array(shap_values).reshape(6495,11))  
np.round(model.predict(wine_features),0)
```

Положительные значения SHAP помогают сделать прогноз, а отрицательные уменьшают силу прогноза.

ЗАКЛЮЧЕНИЕ

В этой главе вы узнали, как интерпретировать вывод для модели классификации на основе трех наборов данных – MNIST, CIFAR и набора данных о качестве вина. При использовании модели глубокого обучения было проведено обучение, и мы объяснили прогнозы модели с помощью DeepExplainer из библиотеки SHAP. Роль DeepExplainer заключается в том, чтобы показать положительные и отрицательные значения SHAP-изображений, чтобы объяснить любую неоднозначность или перекрытие в задаче классификации. В задаче структурированных данных, которой является сценарий классификации качества вина, он также объясняет, какие характеристики положительно влияют на прогноз, а какие – отрицательно. Разница между моделью машинного обучения и моделью глубокого обучения заключается в том, что в глубоком обучении отбор характеристик происходит автоматически, и при обучении глубокой нейронной сети происходит итерационный процесс. Из-за этого сложно объяснить кому-либо прогнозы модели. Сейчас эта задача значительно упростилась благодаря использованию DeepExplainer для объяснения прогнозов модели.

ГЛАВА 10

Контрфактуальные объяснения для моделей XAI

В этой главе объясняется использование What-If Tool (WIT) для объяснения контрфактуальных определений в моделях ИИ, таких как модели регрессии на основе машинного обучения, модели классификации и модели многоклассовой классификации. Как специалист по исследованию данных, вы не просто разрабатываете модель машинного обучения. Вы следите за тем, чтобы ваша модель не была предвзятой и была справедливой в отношении решений, которые она примет для новых наблюдений, предсказываемых в будущем. Очень важно проверить решения и убедиться в справедливости алгоритма. Google разработал инструмент What-If Tool для решения проблемы справедливости в моделях машинного обучения. Мы рассмотрим реализацию WIT в трех моделях ML: ML для задач на основе регрессии, ML для биномиальных моделей классификации и ML для полиноминальных моделей.

Что такое CFE?

Контрфактуальные объяснения (counterfactual explanations – CFE) – это причинно-следственные связи в процессе прогнозирования моделей ML. Я проиллюстрирую прогноз или классификацию на одном примере. Контрфактуальные объяснения – это способ создания гипотетических сценариев и генерирования прогнозов в соответствии с этими гипотетическими сценариями. Это понятно бизнес-пользователям, неспециалистам по анализу данных и обычным людям, не имеющим отношения к прогнозирующему моделированию. Иногда мы обнаруживаем, что связь между входом и выходом не является причинно-следственной, но все же хотим образовать такую связь для создания прогнозов. Контрфактуальные объяснения являются локальными объяснениями для прогнозов, что означает, что для отдельных прогнозов они генерируют соответствующие объяснения.

ПРИМЕНЕНИЕ CFE

Контрфактуальные объяснения могут быть созданы как для задач регрессии, так и для задач классификации. Мы будем использовать библиотеку Alibi на базе Python. Alibi работает на Python версии 3.6+ и выше. Для ее установки можно использовать следующие команды:

```
!pip install alibi
!pip install alibi[ray]
import alibi
alibi.__version__
```

После установки, пожалуйста, проверьте версию с помощью следующего кода, чтобы убедиться, что установка завершена: `alibi.__version__`

CFE с помощью Alibi

Контрфактуальные объяснения прогнозов модели могут быть созданы с помощью Alibi. После установки модели вы можете следовать процессу инициализации, подгонки и прогнозирования с помощью функций Alibi. Для работы библиотеки Alibi требуется обученная модель и новый набор данных, для которых необходимо создать прогнозы. Например, вы можете использовать модель экстремального повышения градиента для получения объекта обученной модели, который в дальнейшем может быть использован с помощью библиотеки Alibi для генерации контрфактуальных объяснений. Следующий скрипт устанавливает модель xgboost и библиотеку witwidget. Если модель xgboost показывает ошибку, можно использовать приведенную ниже команду conda для установки библиотеки xgboost на базе Python.

```
import pandas as pd
!pip install xgboost

import pandas as pd
import xgboost as xgb
import numpy as np
import collections
import witwidget

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.utils import shuffle
from witwidget.notebook.visualization import WitWidget, WitConfigBuilder

# если вы получили ошибку в xgboost
# запустите
# conda install -c conda-forge xgboost

data = pd.read_csv('diabetes.csv')
data.head()
data.columns

from sklearn.model_selection import train_test_split
```

Вы используете набор данных `diabetes.csv`. Используя характеристики, показанные в табл. 10.1, вы должны предсказать, будет ли кто-то диабетиком, или нет.

Таблица 10.1. Словарь данных для характеристик

Имя характеристики	Описание
Беременности	Количество беременностей
Глюкоза	Уровень глюкозы в крови
Артериальное давление	Уровень диастолического артериального давления
Толщина кожи	Толщина кожи измеряется в мм
Инсулин	Введенные дозы инсулина
ИМТ	Индекс массы тела субъекта
Родословная диабета	Функция родословной диабета субъекта
Возраст	Возраст субъекта
Результат	0 – отсутствие диабетиков, 1 – диабетики

```
# Разделение данных на обучающий и тестовый наборы
y = data['Outcome']
x = data[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=1234)
```

Все данные разбиваются на обучающий и тестовый объекты, `x_train` и `y_train`, для разработки обученного объекта модели. В следующем скрипте выполняется обучение с помощью модели логистической регрессии с использованием классификатора экстремального повышения градиента. Классификатор экстремального повышения градиента имеет множество гиперпараметров, но мы будем использовать выбор гиперпараметров по умолчанию. Гиперпараметры – это параметры, которые помогают точно настроить модель, чтобы получить наилучшую ее версию.

```
# Обучение модели, на это уйдет несколько минут
bst = xgb.XGBClassifier(
    objective='reg:logistic'
)
bst.fit(x_train, y_train)
# Получение прогнозов на тестовом наборе и вывод оценки точности
y_pred = bst.predict(x_test)
acc = accuracy_score(np.array(y_test), y_pred)
print(acc, '\n')

# Вывод матрицы путаницы
print('Confusion matrix:')
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

Объект обученной модели хранится под именем `bst`. Используя этот объект, вы можете определить точность модели. Базовая точность составляет 72.9 %.

Точность лучшей модели равна 72.9 %. Вы можете сохранить лучшую модель и загрузить ее в Alibi для генерации прогнозов и создания контрафактуальных объяснений.

```
# Сохраните модель, чтобы мы могли ее развернуть
bst.save_model('model.bst')
bst
x_train.head()
bst.predict(x_train.head())
y_train.head()
x_train.iloc[3] = x_train.iloc[1]
x_train.head()
bst.predict(x_train.head())
```

Для человека в записи № 8 модель предсказывает диабет, и фактическим результатом также является диабет. Стока № 651 (третья запись из обучающего набора данных) не имеет диабета, как видно из `y_train` записи № 651.

Если немного изменить значения в строке № 651, прогноз для этой записи изменится (табл. 10.2). Изменения вносятся в следующем скрипте.

Таблица 10.2. Изменения в значениях характеристик влияют на прогнозирование

Имя характеристики	Старое значение	Новое значение
Беременности	1	5
Глюкоза	117	97
Артериальное давление	60	76
Толщина кожи	23	27
Инсулин	106	0
ИМТ	33.8	35.6
Родословная диабета	0.466	0.378
Возраст	27	52

Следующий скрипт показывает важность характеристик в прогнозировании диабета у человека. Уровень глюкозы считается наиболее важной характеристикой, за ней следуют ИМТ, возраст и уровень инсулина.

```
bst.predict(x_train.head())
x_train.iloc[3] = [1, 117, 60, 23, 106, 33.8, 0.466, 27]
x_train.head()

result = pd.DataFrame()
```

```
result['features_importance'] = bst.feature_importances_
result['feature_names'] = x_train.columns
result.sort_values(by=['features_importance'], ascending=False)
```

Можно вносить небольшие изменения в значения характеристик и наблюдать за изменениями переменной результата. При небольших изменениях прогноз не изменился.

```
# внесем небольшие изменения в значения
x_train.iloc[2] = [2, 146, 70, 38, 360, 28.0, 0.337, 29]
x_train.iloc[2]
x_train.columns

bst.predict(x_train.head())
y_train.head()

# если изменим такие важные характеристики, как уровень глюкозы и возраст

x_train.iloc[2] = [3, 117, 76, 36, 245, 31.6, 0.851, 27]
x_train.iloc[2]
x_train.columns
bst.predict(x_train.head())
y_train.head()
```

При очередном изменении значений входных характеристик предсказание класса диабета не изменилось. Продолжая изменять значения, вы найдете место, где небольшое отклонение значения может привести к изменению прогноза класса. Вручную повторять все подобные сценарии громоздко и сложно. Даже изменения значений наиболее важных характеристик не влияют на прогноз целевого класса. Следовательно, существует необходимость в определении таких пограничных случаев с помощью некоторой системы. Такую систему предлагает Alibi.

```
# если изменим такие важные характеристики, как глюкоза, ИМТ и возраст

x_train.iloc[2] = [3, 117, 76, 36, 245, 30.6, 0.851, 29]
x_train.iloc[2]
x_train.columns

bst.predict(x_train.head())
y_train.head()

import tensorflow as tf
tf.get_logger().setLevel(40) # подавлять сообщения об устаревании
tf.compat.v1.disable_v2_behavior() # отключить поведение TF2, поскольку код alibi
# по-прежнему опирается на конструкции TF1
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.models import Model, load_model
```

```

from tensorflow.keras.utils import to_categorical
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import os
from alibi.explainers import CounterFactualProto

print('TF version: ', tf.__version__)
print('Eager execution enabled: ', tf.executing_eagerly()) # False

```

Вы можете использовать слой Keras с TensorFlow в качестве бэкенда для обучения объекта модели нейронной сети. Alibi предоставляет класс `explainers`, в котором есть модуль для генерации контрфактуальных проторезультатов. Следующий скрипт создает класс модели нейронной сети как определяемую пользователем функцию. Эта функция `nn_model()` принимает входные данные и применяет функцию активации выпрямленного линейного блока. Затем она создает еще один скрытый слой с 40 нейронами, применяет ту же функцию активации и использует стохастический градиентный спуск в качестве оптимизатора и функцию потерь в качестве категориальной перекрестной энтропии.

```

x_train.shape, y_train.shape, x_test.shape, y_test.shape
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

np.random.seed(42)
tf.random.set_seed(42)

def nn_model():
    x_in = Input(shape=(8,))
    x = Dense(40, activation='relu')(x_in)
    x = Dense(40, activation='relu')(x)
    x_out = Dense(2, activation='softmax')(x)
    nn = Model(inputs=x_in, outputs=x_out)
    nn.compile(loss='categorical_crossentropy', optimizer='sgd',
               metrics=['accuracy'])
    return nn

```

Следующий скрипт показывает сводку модели нейронной сети. При обучении модели используется размер порции 64 и 500 эпох. После обучения модели она сохраняется в формате h5. Формат h5 – это переносимый объект модели для загрузки и переноса модели с одной платформы на другую.

```

nn = nn_model()
nn.summary()
nn.fit(x_train, y_train, batch_size=64, epochs=500, verbose=0)
nn.save('nn_diabetes.h5', save_format='h5')

```

Следующий скрипт загружает объект обученной модели, использует тестовые наборы данных и обеспечивает точность теста 71.8 %:

```
nn = load_model('nn_diabetes.h5')
score = nn.evaluate(x_test, y_test, verbose=0)
print('Test accuracy: ', score[1])
```

Генерация контрфактуалов осуществляется на основе ближайшего прототипа класса. Мы берем одну строку из тестового набора с восемью характеристиками и изменяем ее форму так, чтобы она стала доступна для функции генерации контрфактуального прототипа.

```
# Создание контрфактуала на основе ближайшего прототипа класса
X = np.array(x_test)[1].reshape((1,) + np.array(x_test)[1].shape)
shape = X.shape
shape
```

Следующий скрипт принимает объект модели нейронной сети. Используя лямбда-функцию, создается функция прогнозирования. Затем используется контрфактуальная функция Proto (табл. 10.3).

Таблица 10.3. Гиперпараметры контрфактуальной функции Proto

Параметры	Объяснение
predict_fn	Модель Keras или TensorFlow, или функция прогнозирования любой другой модели, возвращающая вероятности классов
Shape	Форма входных данных, начиная с размера порции
use_kdtree	Использовать ли k-мерные деревья для члена потерь прототипа, если нет доступного кодировщика
Theta	Константа для члена потерь при поиске прототипа
max_iterations	Максимальное количество итераций для поиска контрфактуала
feature_range	Кортеж с допустимыми минимальным и максимальным диапазонами для возмущенных экземпляров
c_init, c_steps	Начальное значение для масштабирования термина потерь атаки, количество итераций для настройки постоянного масштабирования члена потерь атаки

```
# определение модели
nn = load_model('nn_diabetes.h5')

predict_fn = lambda x: nn.predict(x)

# инициализация объяснителя, подгонка и генерация контрфактуала
cf = CounterFactualProto(predict_fn, shape, use_kdtree=False, theta=10.,
                           max_iterations=1000,
                           feature_range=(np.array(x_train).min(axis=0),
```

```

np.array(x_train).max(axis=0)),
c_init=1., c_steps=10)
cf.fit(x_train)

```

Вышеуказанный результат генерируется в процессе контрафактуальной подгонки или в процессе обучения. Следующий скрипт показывает наименьшие возможные значения по всем характеристикам:

```
x_train.min(axis=0)
```

Метод `explain` принимает входные данные и генерирует контрафактические объяснения, показанные в табл. 10.4. Метод `explain` также создает следующий вывод локального объяснения:

```

explanation = cf.explain(X)
print(explanation)

```

Таблица 10.4. Интерпретация контрафактуалов

Вывод	Интерпретация
Cf.X	Экземпляр контрафактуала
Cf.class	Предполагаемый класс для контрафактуала
Cf.proba	Прогнозируемые вероятности класса для контрафактуала
Cf.grads_graph	Значения градиента, вычисленные из TF-графа относительно входных характеристик контрафактуала
Cf.grads_num	Численные значения градиента относительно входных характеристик контрафактуала
orig_class	Предполагаемый класс для исходного экземпляра
orig_proba	Прогнозируемые вероятности класса для исходного экземпляра
All	Словарь с итерациями в качестве ключей и для каждой итерации список контрафактуалов, найденных в этой итерации, в качестве значений

```

print(f'Original prediction: {explanation.orig_class}')
print('Counterfactual prediction: {}'.format(explanation.cf['class']))
Original prediction: 0
Counterfactual prediction: 1

```

В приведенном выше скрипте все ключи обеспечивают результаты итераций при нахождении значений контрафактуала. На итерации 3 вы получаете значения контрафактуала. Результаты остальных из девяти итераций не имеют контрафактуальных значений. Первоначальный прогноз – отсутствие диабета, а прогноз, основанный на контрафактуальной информации, – диабет.

```
explanation = cf.explain(X)
explanation.all
explanation.cf

{'X': array([[2.5628092e+00, 1.7726180e+02, 7.6516624e+01, 2.4388657e+01,
              7.2130630e+01, 2.6932917e+01, 7.8000002e-02, 2.2944651e+01]], 
            dtype=float32),
 'class': 1,
 'proba': array([[0.4956944, 0.5043056]], dtype=float32),
 'grads_graph': array([[ -0.7179985 , -5.005951 , 23.374054 , -0.96334076,
                        3.8287811 , -13.371899 , -0.38599998, -5.9150696 ]], 
                  dtype=float32),
 'grads_num': array([[ 18.74566078, 38.92183304, -117.42114276,
                      24.19948392, -35.82239151, 62.04843149, 93.54948252,
                      25.92801861]])}

explanation.orig_class
explanation.orig_proba
X
```

КОНТРФАКУАЛ ДЛЯ ЗАДАЧ РЕГРЕССИИ

В регрессионном сценарии контрфактуальные объяснения можно увидеть, рассмотрев нейросетевую модель с использованием набора данных о ценах на жилье в Бостоне. Регрессия – это задача, в которой целевой столбец является непрерывной переменной и можно использовать смесь переменных – непрерывных и категориальных. Набор данных о ценах на жилье в Бостоне является обычным набором данных для изучения регрессии. Я выбрал его для демонстрации контрфактуалов, так как пользователи лучше знакомы с этим набором данных.

```
import tensorflow as tf
tf.get_logger().setLevel(40) # подавлять сообщения об устаревании
tf.compat.v1.disable_v2_behavior()# отключить поведение TF2, так как код alibi
# по-прежнему опирается на конструкции TF1
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.utils import to_categorical
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import os
from sklearn.datasets import load_boston
from alibi.explainers import CounterFactualProto

print('TF version: ', tf.__version__)
print('Eager execution enabled: ', tf.executing_eagerly()) # False
```

```

boston = load_boston()
data = boston.data
target = boston.target
feature_names = boston.feature_names

y = np.zeros((target.shape[0],))
y[np.where(target > np.median(target))[0]] = 1

data = np.delete(data, 3, 1)
feature_names = np.delete(feature_names, 3)

mu = data.mean(axis=0)
sigma = data.std(axis=0)
data = (data - mu) / sigma

idx = 475
x_train,y_train = data[:idx,:], y[:idx]
x_test, y_test = data[idx:,:], y[idx:]
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

np.random.seed(42)
tf.random.set_seed(42)

def nn_model():
    x_in = Input(shape=(12,))
    x = Dense(40, activation='relu')(x_in)
    x = Dense(40, activation='relu')(x)
    x_out = Dense(2, activation='softmax')(x)
    nn = Model(inputs=x_in, outputs=x_out)
    nn.compile(loss='categorical_crossentropy', optimizer='sgd',
               metrics=['accuracy'])
    return nn

```

Приведенный выше скрипт представляет собой нейросетевую модель для регрессии. Скрипт ниже представляет собой краткое описание архитектуры нейронной сети:

```

nn = nn_model()
nn.summary()
nn.fit(x_train, y_train, batch_size=64, epochs=500, verbose=0)
nn.save('nn_boston.h5', save_format='h5')

```

Объект обученной модели `nn_boston.h5` сохраняется для создания контрфактуалов.

```
nn = load_model('nn_boston.h5')
score = nn.evaluate(x_test, y_test, verbose=0)
print('Test accuracy: ', score[1])

X = x_test[1].reshape((1,) + x_test[1].shape)
shape = X.shape

# определение модели
nn = load_model('nn_boston.h5')

# инициализация объяснителя, подгонка и генерация контрфактуала
cf = CounterFactualProto(nn, shape, use_kdtree=True, theta=10.,
max_iterations=1000,
            feature_range=(x_train.min(axis=0), x_train.
                           max(axis=0)),
            c_init=1., c_steps=10)

cf.fit(x_train)
explanation = cf.explain(X)
```

Результат, полученный из контрфактуалов, можно интерпретировать так, как показано в табл. 10.4. Доля единиц жилья, занимаемых владельцами и построенных до 1940 года, составляет 93.6 %, а более низкий статус населения составляет 18.68 %. Для повышения цен на жилье доля занимаемых владельцами единиц, построенных до 1940 года, должна снизиться на 5.95 %, а более низкий статус населения – на 4.85%.

```
print(f'Original prediction: {explanation.orig_class}')
print('Counterfactual prediction: {}'.format(explanation.cf['class']))
Original prediction: 0
Counterfactual prediction: 1

orig = X * sigma + mu
counterfactual = explanation.cf['X'] * sigma + mu
delta = counterfactual - orig
for i, f in enumerate(feature_names):
    if np.abs(delta[0][i]) > 1e-4:
        print('{}: {}'.format(f, delta[0][i]))

print('% owner-occupied units built prior to 1940: {}'.format(orig[0][5]))
print('% lower status of the population: {}'.format(orig[0][11]))
% owner-occupied units built prior to 1940: 93.6
% lower status of the population: 18.68
```

ЗАКЛЮЧЕНИЕ

В этой главе мы рассмотрели контрафактуальные объяснения задач, связанных с регрессией и классификацией. Контрафактуальная задача состоит в том, чтобы найти аналогичную точку данных в обучающих данных, которая дает прогноз, отличный от прогноза помеченного класса. Контрафактуальное объяснение заключается в том, что если две точки данных очень похожи, то обе должны иметь схожий прогноз или схожий результат. Не должно быть разных результатов в целевом классе. В примере прогнозирования диабета если два человека имеют схожие характеристики, то либо у обоих нет диабета, либо у обоих он есть. Не должно быть так, что у одного есть, а у другого нет. Существование различной контрафактуальной информации фактически создает путаницу с точки зрения конечного пользователя, что порождает недоверие к моделям ИИ. Чтобы укрепить это доверие, важно генерировать контрафактуальные объяснения.

ГЛАВА 11

Контрастные объяснения для машинного обучения

Контрастное обучение – это новый подход к поиску схожих и несхожих объектов-кандидатов в конвейере машинного обучения. Контрастное объяснение направлено на поиск сходства между двумя характеристиками, чтобы помочь в прогнозировании класса. Типичные модели «черные ящики», обученные на различных типах гиперпараметров и куче параметров, оптимизированных на различных эпохах и скоростях обучения, очень трудно интерпретировать и еще труднее объяснить, почему модель предсказала класс А, а не класс В. Необходимость генерировать больше объяснений, чтобы бизнес-пользователи могли понять прогнозы, привлекла многих разработчиков к созданию инновационных фреймворков, способных генерировать значение. Контрастное объяснение фокусируется на объяснении того, почему модель предсказала класс А, а не класс В. Поиск причин помогает бизнес-пользователю понять поведение модели. В этой главе мы будем использовать библиотеку Alibi для задачи классификации изображений с помощью фреймворка на основе TensorFlow.

Что такое CE для ML?

Чтобы понять, что такое контрастное объяснение (contrastive explanation – CE) для машинного обучения, давайте возьмем пример процесса одобрения кредита или оценки кредитного риска банка. Ни один банк не предоставит кредит рискованному клиенту. Аналогично ни один банк не откажет в кредите нерискованному клиенту. Когда заявка на получение ссуды или кредитная заявка отклоняется, это не просто произвольное решение, принятое кем-то в банке. Скорее, это решение, принятое моделью ИИ, которая учитывает множество характеристик человека, включая финансовую историю и другие факторы. Человек, которому было отказано в кредите, может задаться вопросом, какой аспект определяет его право на получение кредита, что отличает одного человека от другого с точки зрения права на получение кредита и какого элемента не хватает в его профиле среди прочего. Аналогичным образом при идентификации интересующих объектов из множества других доступных изображений что отличает интересующий объект от остальных изображений? Контрастные объяснения – это характеристики, которые отличают класс от других классов.

Контрастные объяснения подобны человеческим беседам, помогающим человеку вести более заинтересованный разговор. Существует два понятия, связанных с контрастными объяснениями:

- позитивные утверждения (pertinent positives – PP);
- негативные утверждения (pertinent negatives – PN).

Объяснение PP обнаруживает присутствие тех характеристик, которые необходимы для того, чтобы модель машинного обучения идентифицировала тот же класс, что и предсказанный. Например, доход и возраст человека определяют его чистую стоимость. Вы хотите, чтобы ваша модель машинного обучения определила класс людей с высоким уровнем чистой стоимости по наличию высокого дохода и значительного возраста. Это можно сравнить с объяснением якорей в модели машинного обучения. PN является обратной стороной PP, оно объясняет характеристики, которые должны отсутствовать в записи для сохранения исходного выходного класса. Это некоторые исследователи также называют *контрафактуальным объяснением*.

CEM, использующие Alibi

Контрастные объяснения для моделей помогают прояснить конечному пользователю, почему то или иное событие или предсказание произошло, в отличие от другого. Чтобы объяснить концепцию контрастных объяснений для модели (contrastive explanations for a model – CEM) и то, как это может быть реализовано через библиотеки на базе Python, давайте воспользуемся Alibi. В качестве первого шага мы разработаем модель глубокого обучения на базе Keras с TensorFlow в качестве бэкенда. В следующем скрипте есть различные операторы импорта модулей и методов, которые могут быть использованы при разработке модели глубокого обучения. CEM имеется в модуле explainer библиотеки Alibi.

```
import tensorflow as tf
tf.get_logger().setLevel(40) # подавлять сообщения об устаревании
tf.compat.v1.disable_v2_behavior() # отключить поведение TF2, так как код alibi
# по-прежнему опирается на конструкции TF1
import tensorflow.keras as keras
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Conv2D, Dense, Dropout, Flatten,
MaxPooling2D, Input, UpSampling2D
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.utils import to_categorical

import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import os
from alibi.explainers import CEM
```

```
print('TF version: ', tf.__version__)
print('Eager execution enabled: ', tf.executing_eagerly()) # False
```

При объяснении наличия РР и отсутствия РН важно определить и упорядочить соответствующие характеристики, а также классифицировать важные и неважные характеристики. Если на этапе обучения модели у вас больше последних, то действительно не имеет значения, относятся ли они к РР, или РН. Неважные характеристики из модели вообще не имеют значения. Для объяснения СЕМ выбрана классификация цифр в наборе данных MNIST, потому что многие разработчики и инженеры ML знакомы с этим набором данных. Поэтому они могут хорошо соотнести его с концепцией СЕМ. Следующий скрипт разделяет набор данных на обучающий и тестовый наборы, а образец изображения цифры 4 показан на рис. 11.1:

```
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
print('x_train shape:', x_train.shape, 'y_train shape:', y_train.shape)
plt.gray()
plt.imshow(x_test[4]);
```

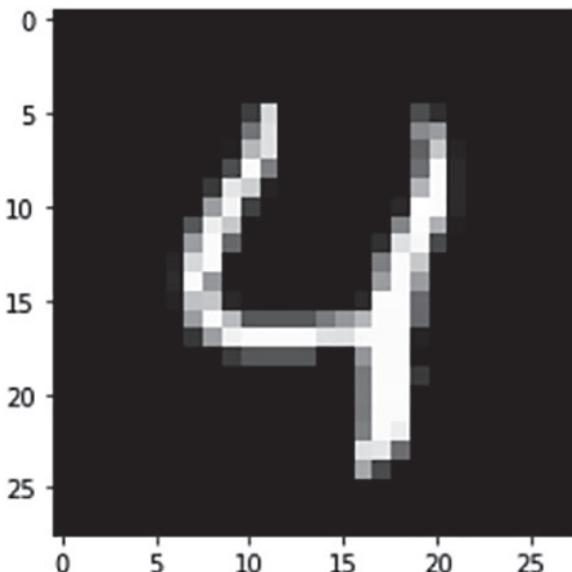


Рис. 11.1. Образец изображения для цифры 4

Мы собираемся разработать модель классификации. В качестве следующего шага необходимо нормализовать характеристики, чтобы ускорить процесс обучения модели глубокого обучения. Поэтому делим значения пикселей на наибольшее значение (255).

```
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
x_train = np.reshape(x_train, x_train.shape + (1,))
```

```

x_test = np.reshape(x_test, x_test.shape + (1,))
print('x_train shape:', x_train.shape, 'x_test shape:', x_test.shape)

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
print('y_train shape:', y_train.shape, 'y_test shape:', y_test.shape)

xmin, xmax = -.5, .5
x_train = ((x_train - x_train.min()) / (x_train.max() - x_train.min())) *
(xmax - xmin) + xmin
x_test = ((x_test - x_test.min()) / (x_test.max() - x_test.min())) *
(xmax - xmin) + xmin

```

Следующий скрипт показывает шаги по созданию модели конволовационной нейронной сети:

- входной набор данных представляет собой фигуру размером 28×28 пиксей;
- в сверточном 2D-слое применяется 64 фильтра и размер ядра 2, с такой же прокладкой и выпрямленной линейной единицей в качестве функции активации;
- после применения сверточного слоя необходимо применить max pooling для создания абстрактных характеристик, которые могут быть использованы последующими слоями;
- отсев обычно применяется для того, чтобы оградить модель от переподгонки;
- существует три слоя сверточных 2D-фильтров, за которыми следуют слой max pooling и слой отсева, применяемые для уменьшения размерности данных;
- цель применения свертки и max pooling заключается в том, чтобы получить слой с меньшим количеством нейронов, который может быть использован для обучения полностью подключенной модели нейронной сети;
- если данные должны быть слажены, чтобы изменить форму, то используется плотный слой, который представляет собой модель полностью подключенной нейронной сети;
- наконец, применяется слой softmax для того, чтобы генерировать вероятность класса относительно каждого класса цифр;
- на шаге компиляции необходимо указать категориальную перекрестную энтропию в качестве функции потерь, adam в качестве оптимизатора и точность в качестве метрики.

```

def cnn_model():
    x_in = Input(shape=(28, 28, 1)) # входной слой
    x = Conv2D(filters=64, kernel_size=2, padding='same',
               activation='relu')(x_in) # сверточный слой

```

```
x = MaxPooling2D(pool_size=2)(x) # слой max pooling
x = Dropout(0.3)(x) #отсев для предотвращения переподгонки

x = Conv2D(filters=32, kernel_size=2, padding='same',
           activation='relu')(x) # второй сверточный слой
x = MaxPooling2D(pool_size=2)(x) # слой max pooling
x = Dropout(0.3)(x) #отсев для предотвращения переподгонки

x = Conv2D(filters=32, kernel_size=2, padding='same',
           activation='relu')(x) # третий сверточный слой
x = MaxPooling2D(pool_size=2)(x) #слой max pooling
x = Dropout(0.3)(x) #отсев для предотвращения переподгонки

x = Flatten()(x) # сглаживание для изменения формы матрицы
x = Dense(256, activation='relu')(x) # это для обучения
# слоя полностью подключенной нейронной сети
x = Dropout(0.5)(x) # снова отсев для предотвращения переподгонки
x_out = Dense(10, activation='softmax')(x) # последний выходной слой

cnn = Model(inputs=x_in, outputs=x_out)
cnn.compile(loss='categorical_crossentropy', optimizer='adam',
             metrics=['accuracy'])

return cnn
```

Сводка по модели конволюционной нейронной сети представлена в табличном виде ниже. На этапе обучения модели необходимо указать размер порции и количество эпох. Затем вы можете сохранить обученный объект модели в формате h5.

```
cnn = cnn_model()
cnn.summary()
cnn.fit(x_train, y_train, batch_size=64, epochs=5, verbose=1)
cnn.save('mnist_cnn.h5', save_format='h5')
```

Для создания значимых СЕМ необходимо создать модель с высокой точностью, иначе СЕМ будут лишены согласованности. Следовательно, рекомендуется сначала обучить, отладить или найти наилучшую модель, обеспечивающую точность не менее 85 %. В данном примере точность модели на тестовом наборе данных составляет 98.73 %, поэтому можно ожидать значимых объяснений СЕМ.

```
# Оценка модели на тестовом наборе
cnn = load_model('mnist_cnn.h5')
score = cnn.evaluate(x_test, y_test, verbose=0)
print('Test accuracy: ', score[1])
```

Для генерации пояснений СЕМ необходим объект модели, который классифицирует входные данные в определенный класс. СЕМ пытается генерировать два возможных объяснения:

- найти минимальное количество информации, которое должно присутствовать во входных данных и которого достаточно для того, чтобы сгенерировать одну и ту же классификацию класса. Это называется РР;
- обнаружить отсутствие минимального количества информации во входных данных, достаточного для того, чтобы прогноз класса не изменился. Это называется РН.

Стремление найти минимум информации, которая, возможно, может изменить прогноз или помочь в сохранении прогноза, как правило, является наиболее важным значением характеристики из набора входных данных. Сопоставление входных данных с абстрактным слоем, сформированным из данных, используется для определения наличия или отсутствия минимальной информации. Абстрактный слой называется автокодировщиком (autoencoder). Для задачи классификации изображений это может быть конволюционный автокодировщик. Автокодировщик обучается с помощью входных данных во входном слое, а также в выходном слое. Модель обучается точно предсказывать тот же выход, что и вход. Как только входные и выходные данные совпадают, тогда веса внутреннего скрытого слоя в модели нейронной сети могут быть выведены как значения автокодировщика. Эти значения помогают в определении минимальной доступности информации в любом входном наборе данных. Следующий скрипт показывает, как обучить модель автокодировщика. Это модель нейронной сети. Она принимает 28×28 пикселей на вход и выдает 28×28 пикселей в качестве выхода.

```
# Определение и обучение автокодировщика, он работает как модель анализа
# основных компонентов
def ae_model():
    x_in = Input(shape=(28, 28, 1))
    x = Conv2D(16, (3, 3), activation='relu', padding='same')(x_in)
    x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)
    x = MaxPooling2D((2, 2), padding='same')(x)
    encoded = Conv2D(1, (3, 3), activation=None, padding='same')(x)

    x = Conv2D(16, (3, 3), activation='relu', padding='same')(encoded)
    x = UpSampling2D((2, 2))(x)
    x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)
    decoded = Conv2D(1, (3, 3), activation=None, padding='same')(x)
    autoencoder = Model(x_in, decoded)
    autoencoder.compile(optimizer='adam', loss='mse')

    return autoencoder

ae = ae_model()
ae.summary()
```

```
ae.fit(x_train, x_train, batch_size=128, epochs=4, validation_data=(x_test,  
x_test), verbose=0)  
ae.save('mnist_ae.h5', save_format='h5')
```

Сводка модели показывает архитектуру и общее количество обучаемых параметров. Обучаемые параметры имеют обновленные веса в каждой итерации модели. Необучаемых параметров нет. Когда модель автокодировщика готова, можно загрузить объект модели в сессию. Используя функцию `predict` на тестовом наборе, вы можете генерировать декодированные изображения MNIST. Результаты следующего скрипта показывают, что тестовые изображения точно соответствуют изображениям, сгенерированным с помощью функции автокодировщика. Это означает, что модель автокодировщика достаточно надежна в генерации точного результата. Модель автокодировщика состоит из двух частей – кодера и декодера. Роль кодера заключается в преобразовании любого входного сигнала в абстрактный слой, а роль декодера – в восстановлении того же входа из абстрактного слоя.

СРАВНЕНИЕ ОРИГИНАЛЬНОГО ИЗОБРАЖЕНИЯ И ИЗОБРАЖЕНИЯ, СГЕНЕРИРОВАННОГО АВТОКОДИРОВЩИКОМ

На рис. 11.2 представлены исходные изображения из обучающего набора и набор изображений, сгенерированных автокодировщиком. Из этих изображений видно, что модель, сгенерированная автокодировщиком, точно соответствует оригинальному изображению. Поскольку это пример набора данных, совпадение очень близкое, однако в других случаях для создания таких близких изображений требуется большое количество тренировок. Хорошо обученная модель автокодировщика будет очень полезна для создания контрастных объяснений.

```
ae = load_model('mnist_ae.h5')  
decoded_imgs = ae.predict(x_test)  
n = 5  
plt.figure(figsize=(20, 4))  
for i in range(1, n+1):  
    # отображение оригинала  
    ax = plt.subplot(2, n, i)  
    plt.imshow(x_test[i].reshape(28, 28))  
    ax.get_xaxis().set_visible(False)  
    ax.get_yaxis().set_visible(False)  
    # отображение восстановленного  
    ax = plt.subplot(2, n, i + n)  
    plt.imshow(decoded_imgs[i].reshape(28, 28))  
    ax.get_xaxis().set_visible(False)  
    ax.get_yaxis().set_visible(False)  
plt.show()
```

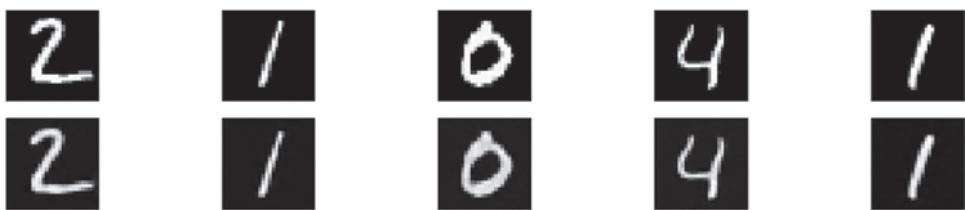


Рис. 11.2. Сравнение изображений, созданных моделью автокодировщика, с оригиналами изображения

В первой строке показаны реальные изображения из первой записи тестового набора данных, а во второй – предсказанные изображения, сгенерированные автокодировщиком.

```
ae = load_model('mnist_ae.h5')
decoded_imgs = ae.predict(x_test)
n = 5
plt.figure(figsize=(20, 4))
for i in range(1, n+1):
    # отображение оригинала
    ax = plt.subplot(2, n, i)
    plt.imshow(x_test[i].reshape(28, 28))
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    # отображение восстановленного
    ax = plt.subplot(2, n, i + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```

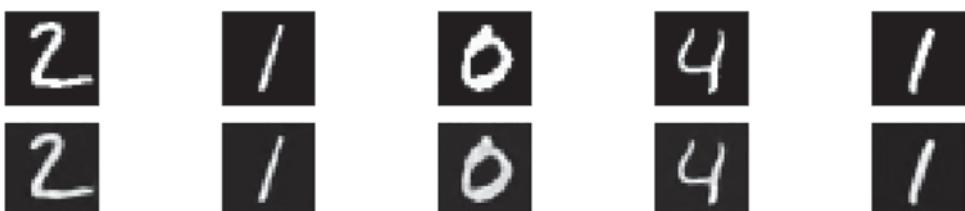


Рис. 11.3. Объяснение негативных утверждений

Приведенный выше скрипт показывает отображение и реконструкцию одного и того же изображения (рис. 11.3), которое приведено ниже, как 5 (рис. 11.4).

```
idx = 15
X = x_test[idx].reshape((1,) + x_test[idx].shape)
plt.imshow(X.reshape(28, 28));
```

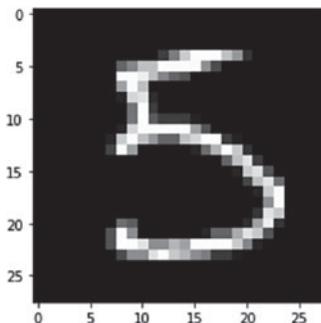


Рис. 11.4. Отображение оригинального изображения для цифры 5

```
# прогноз модели  
cnn.predict(X).argmax(), cnn.predict(X).max()
```

Модель CNN предсказывает входное значение 5 с вероятностью 99.95 % (табл. 11.1).

Таблица 11.1. Объяснения параметров СЕМ

Параметр	Объяснения
Mode	PN или PP
Shape	Форма входного экземпляра
Kappa	Минимальная необходимая разница между вероятностью прогнозирования для возмущенного экземпляра в прогнозируемом классе в качестве исходного экземпляра, и максимальной вероятностью в других классах для того, чтобы первый член потерь был минимизирован
Beta	Вес члена потерь L1
Gamma	Вес члена потерь автокодировщика
C_steps	Количество обновлений
Max iterations	Количество итераций
Feature_range	Диапазон характеристик для возмущенного экземпляра
Lr	Начальная скорость обучения

Следующий скрипт показывает негативное предсказание из объяснения объекта, созданного на основе образца X. Негативный анализ говорит о том, что некоторые важные характеристики отсутствуют в 5, в противном случае он был бы классифицирован как 8. Эта недостающая информация является минимальной информацией, которая отличает прогноз класса 5 от прогноза класса 8. На рис.11.5 цифра 8 наложена на цифру 5.

```

mode = 'PN' # 'PN' (негативное утверждение) или 'PP' (позитивное утверждение)
shape = (1,) + x_train.shape[1:] # instance shape
kappa = 0. # минимально необходимая разница между вероятностью предсказания
# для возмущенного экземпляра на классе, предсказанным исходным экземпляром,
# и максимальной вероятностью для других классов для того, чтобы первый член
# потерь был минимизирован
beta = .1# вес члена потерь L1
gamma = 100 # вес необязательного члена потерь автокодировщика
c_init = 1. # начальный вес с члена потерь, побуждающий предсказывать
# другой класс (PN) или
# тот же класс (PP) для возмущенного экземпляра по сравнению с
# исходным экземпляром, который необходимо объяснить
c_steps = 10 # число обновлений для с
max_iterations = 1000 # число итераций для каждого значения с
feature_range = (x_train.min(),x_train.max())# диапазон характеристик для
# возмущённого экземпляра
clip = (-1000.,1000.) # обрезание градиента
lr = 1e-2 # начальная скорость обучения
no_info_val = -1. # значение, плавающее или вида характеристики, которое может
# рассматриваться как не содержащее информации для прогнозирования
# Возмущения в сторону этого значения означают удаление
# характеристик, а отсутствие означает добавление характеристик
# для наших изображений MNIST, фон (-0.5) является наименее
# информативным, поэтому положительные/отрицательные возмущения
# означают # добавление/удаление характеристик инициализировать
# CEM объяснятель и экземпляр объяснения
cem = CEM(cnn, mode, shape, kappa=kappa, beta=beta, feature_range=feature_
range, gamma=gamma, ae_model=ae, max_iterations=max_iterations,
c_init=c_init, c_steps=c_steps, learning_rate_init=lr, clip=clip,
no_info_val=no_info_val)
explanation = cem.explain(X)
print(f'Pertinent negative prediction: {explanation.PN_pred}')
plt.imshow(explanation.PN.reshape(28, 28));

```

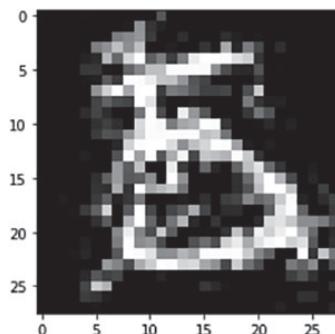


Рис. 11.5. Негативный прогноз (цифра 8 по сравнению с исходным изображением 5)

Позитивное объяснение может быть получено для той же цифры 5, что означает, какие признаки вы абсолютно точно ищете в изображении, чтобы классифицировать цифру как 5. Это называется позитивным объяснением.

```
# Теперь генерируем позитив
mode = 'pp'

# инициализируем CEM-объяснитель и экземпляр объяснения
cem = CEM(cnn, mode, shape, kappa=kappa, beta=beta, feature_range=feature_
           range, gamma=gamma, ae_model=ae, max_iterations=max_iterations,
           c_init=c_init, c_steps=c_steps, learning_rate_init=lr, clip=clip,
           no_info_val=no_info_val)
explanation = cem.explain(X)

print(f'Pertinent positive prediction: {explanation.PP_pred}')
plt.imshow(explanation.PP.reshape(28, 28));
```

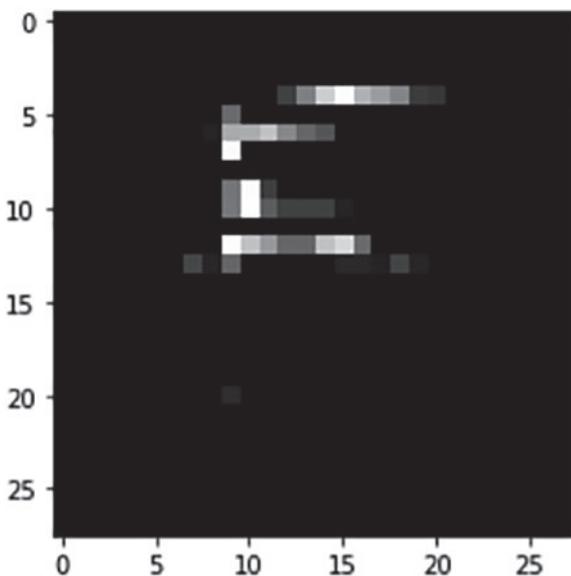


Рис. 11.6. Позитивное объяснение для цифры 5

В приведенном выше скрипте мы генерируем позитив. Позитивное объяснение гласит, что значения пикселей, показанные на рис. 11.6, минимально необходимы для прогнозирования изображения, как цифры 5.

Объяснения СЕМ для табличных данных

Для любых табличных данных (также известных как структурированные данные) строки – это примеры, а столбцы – это характеристики. Вы можете использовать тот же процесс, что и в приведенной выше модели конволюционной нейронной сети, для задачи простой многоклассовой классификации с использованием известного набора данных IRIS.

```
# СЕМ для структурированного набора данных
import tensorflow as tf
tf.get_logger().setLevel(40) # подавлять сообщения об устаревании
tf.compat.v1.disable_v2_behavior() # отключить поведение TF2, так как код alibi
# по-прежнему опирается на конструкции TF1

from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.utils import to_categorical
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import os

import pandas as pd
import seaborn as sns
from sklearn.datasets import load_iris
from alibi.explainers import CEM

print('TF version: ', tf.__version__)
print('Eager execution enabled: ', tf.executing_eagerly()) # False
```

Приведенный выше скрипт показывает операторы импорта, необходимые для того, чтобы модель Alibi могла генерировать объяснения РР и РН.

```
dataset = load_iris()
feature_names = dataset.feature_names
class_names = list(dataset.target_names)
# масштабирование данных
dataset.data = (dataset.data - dataset.data.mean(axis=0)) / dataset.data.
    std(axis=0)
idx = 145
x_train,y_train = dataset.data[:idx,:], dataset.target[:idx]
x_test, y_test = dataset.data[idx+1:,:], dataset.target[idx+1:]
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

Модель IRIS имеет четыре характеристики и три класса в целевом столбце – это setosa, versicolor и virginica. Первые 145 записей являются обучающим набором данных, а 5 записей используются для тестирования модели. В целевом столбце требуется категориальное кодирование или наборы данных `y_train` и `y_test`, поскольку столбец содержит строковые значения.

Следующая функция модели нейронной сети принимает четыре характеристики в качестве входных данных, и модель обучается с помощью полностью подключенной сети, использующей функцию `dense` из Keras. В качестве функции потерь используется категориальная перекрестная энтропия, а в качестве

оптимизатора – стохастический градиентный спуск. Размер порции для обучения модели составляет 16 с 500 эпохами. Обучается очень небольшое количество параметров.

```
def lr_model():
    x_in = Input(shape=(4,))
    x_out = Dense(3, activation='softmax')(x_in)
    lr = Model(inputs=x_in, outputs=x_out)
    lr.compile(loss='categorical_crossentropy', optimizer='sgd',
    metrics=['accuracy'])
    return lr

lr = lr_model()
lr.summary()
lr.fit(x_train, y_train, batch_size=16, epochs=500, verbose=0)
lr.save('iris_lr.h5', save_format='h5')
```

После обучения модели она сохраняется под именем `iris_lr.h5`. В следующем скрипте загружается обученный объект модели и инициализируется функция CEM со всеми параметрами, описанными ранее в табл. 11.1.

```
idx = 0
X = x_test[idx].reshape((1,) + x_test[idx].shape)
print('Prediction on instance to be explained: {}'.format(class_names[np.
argmax(lr.predict(X))]))
print('Prediction probabilities for each class on the instance: {}'.
format(lr.predict(X)))

mode = 'PN' # 'PN' (негативное утверждение) или 'PP' (позитивное утверждение)
shape = (1,) + x_train.shape[1:] # форма экземпляра
кappa = .2 # минимально необходимая разница между вероятностью предсказания
# для возмущенного экземпляра на классе, предсказанным исходным
# экземпляром, и максимальной вероятностью для других классов
# для того, чтобы первый член потерь был минимизирован
beta = .1 # вес члена потерь L1
c_init = 10. # начальный вес с членом потерь, побуждающим предсказывать
# другой класс (PN) или
# тот же класс (PP) для возмущенного экземпляра по сравнению с
# исходным экземпляром, который необходимо объяснить
c_steps = 10 # число обновлений для с
max_iterations = 1000 # число итераций для каждого значения с
feature_range = (x_train.min(axis=0).reshape(shape)-.1, # диапазон
характеристик для возмущенного экземпляра
x_train.max(axis=0).reshape(shape)+.1) # может быть либо
# float, либо массив формы (1xfeatures)
clip = (-1000.,1000.) # обрезание градиента
```

```

lr_init = 1e-2 # начальная скорость обучения
# определение модели
lr = load_model('iris_lr.h5')
# инициализируем СЕМ объяснитель и экземпляр объяснения
cem = CEM(lr, mode, shape, kappa=kappa, beta=beta, feature_range=feature_
range,
           max_iterations=max_iterations, c_init=c_init, c_steps=c_steps,
           learning_rate_init=lr_init, clip=clip)
cem.fit(x_train, no_info_type='median') # нам нужно определить, какие значения
# характеристик
# содержат наименьшее количество
# информации относительно прогнозов
# здесь мы будем наивно полагать, что
# медиана значений характеристик
# не содержит никакой информации, знание
# предмета
# поможет!
explanation = cem.explain(X, verbose=False)

```

В приведенном выше скрипте исходным экземпляром является virginica, но объяснение для соответствующего негатива прогнозирует, что это versicolor. Только разница в третьей характеристике дает разницу в прогнозе. Тот же пример можно использовать и для прогнозирования класса соответствующих позитивов.

```

print(f'Original instance: {explanation.X}')
print('Predicted class: {}'.format(class_names[explanation.X_pred]))

print(f'Pertinent negative: {explanation.PN}')
print('Predicted class: {}'.format(class_names[explanation.PN_pred]))
expl = {}
expl['PN'] = explanation.PN
expl['PN_pred'] = explanation.PN_pred

mode = 'PP'

# определение модели
lr = load_model('iris_lr.h5')
# инициализируем СЕМ объяснитель и экземпляр объяснения
cem = CEM(lr, mode, shape, kappa=kappa, beta=beta, feature_range=feature_
range,
           max_iterations=max_iterations, c_init=c_init, c_steps=c_steps,
           learning_rate_init=lr_init, clip=clip)
cem.fit(x_train, no_info_type='median')
explanation = cem.explain(X, verbose=False)

print(f'Pertinent positive: {explanation.PP}')
print('Predicted class: {}'.format(class_names[explanation.PP_pred]))

```

В приведенном выше скрипте мы объясняем соответствующий позитивный предсказанный класс. Это класс virginica, а фактический также является классом virginica. Мы создаем визуальный способ отображения PP и PN с использованием характеристик и результатов модели СЕМ, объект expl и столбцы с именами PN, PN_pred, PP и PP_pred. Мы создаем датафрейм, содержащий исходные данные и имена характеристик, включая целевой класс. Это требуется для визуализации данных. Библиотека Seaborn Python используется для демонстрации PN и PP в графическом виде на рис. 11.7.

```
expl['PP'] = explanation.PP
expl['PP_pred'] = explanation.PP_pred

df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df['species'] = np.array([dataset.target_names[i] for i in dataset.target])

pn = pd.DataFrame(expl['PN'], columns=dataset.feature_names)
pn['species'] = 'PN_' + class_names[expl['PN_pred']]

pp = pd.DataFrame(expl['PP'], columns=dataset.feature_names)
pp['species'] = 'PP_' + class_names[expl['PP_pred']]

orig_inst = pd.DataFrame(explanation.X, columns=dataset.feature_names)
orig_inst['species'] = 'orig_' + class_names[explanation.X_pred]
df = df.append([pn, pp, orig_inst], ignore_index=True)

fig = sns.pairplot(df, hue='species', diag_kind='hist');
```

Контрастные объяснения обычно создаются путем проецирования характеристик в скрытое пространство как абстрактных, а затем – путем рассмотрения только полезных характеристик для того, чтобы модель различала целевой класс.

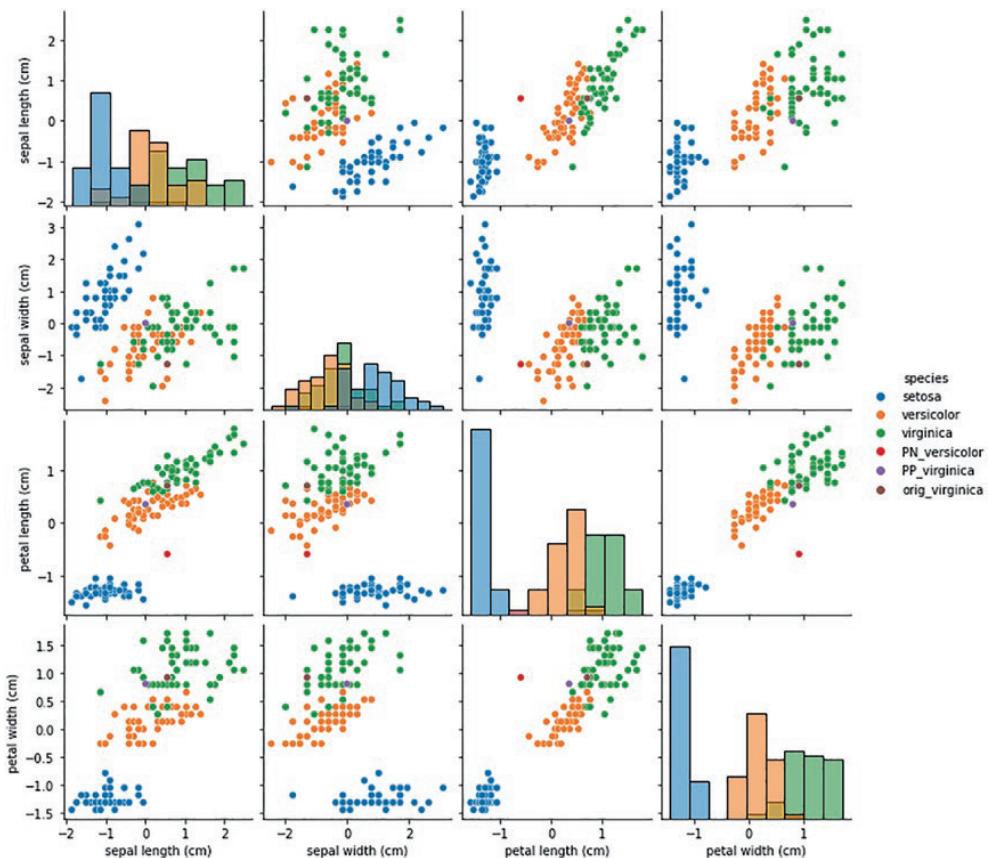


Рис. 11.7. Визуализация позитивных и негативных утверждений

ЗАКЛЮЧЕНИЕ

В этой главе вы изучили методы и библиотеки, которые могут создать контрастное объяснение как для задачи классификации изображений, где у вас есть набор данных MNIST для распознавания почерка, так и для задачи классификации структурированных данных с использованием простого набора данных IRIS. Как позитивные, так и негативные результаты извлекаются из модуля контрастного объяснения в библиотеке Alibi. Этот СЕМ обеспечивает большую ясность предсказания классов и делает вывод о том, почему определенный класс прогнозируется и почему не прогнозируется.

ГЛАВА 12

Модельно независимые объяснения путем определения инвариантности прогноза

Терминология инвариантности, взятая из области математики, объясняет, что прогнозы, сгенерированные моделью машинного обучения, остаются неизменными, если мы вмешиваемся в объясняющие переменные, учитывая тот факт, что модель генерируется посредством формальной причинно-следственной связи. Существует разница между каузальной (причинной) и некаузальной моделью машинного обучения. Каузальная модель показывает реальную причинно-следственную связь между переменной отклика и объясняющими переменными. Непричинная модель отображает случайную связь, прогнозы меняются много раз. Единственный способ понять разницу между этими моделями – только объяснения. В этой главе мы будем использовать библиотеку Python под названием Alibi для создания объяснений, не зависящих от модели.

Что такое независимость от модели?

Термин «модельно независимый» подразумевает независимость от модели. Объяснения моделей машинного обучения, которые действительно независимо от типов моделей, называются модельно независимыми. Объясняя поведение, присущее модели машинного обучения, мы обычно не предполагаем никакой глубинной структуры поведения, отображаемой самой моделью. В этой главе мы будем использовать модельно независимые объяснения для описания инвариантности прогноза. С человеческой точки зрения всегда было сложно понять, как работает модель машинного обучения. Люди всегда удивляются и задают вопросы о ее поведении. Для того чтобы понять модельно независимое поведение, многие думают, что они могут делать предсказания относительно поведения модели.

Что такое якорь?

Сложные модели обычно рассматриваются как модели «черного ящика», так как очень трудно понять, почему был сделан прогноз. Концепция якорей – это не что иное, как выяснение правил. Здесь под этим словом подразумеваются точные правила, которые объясняют поведение модели, принимая во внимание

ние локальные и глобальные факторы. Другими словами, якоря – это условия «если/тогда/иначе» (if/then/else), которые фиксируют прогноз независимо от значений других характеристик.

Алгоритм якоря построен на основе объяснений, не зависящих от модели. Он имеет три основные функции:

- **покрытие** подразумевает, как часто можно изменять структуру, чтобы предсказать поведение модели машинного обучения;
- **точность** – насколько точно человек предсказывает поведение модели;
- **усилия**, необходимые для интерпретации поведения модели или для понимания прогнозов, генерируемых поведением модели. Оба сценария трудно объяснить.

Объяснения якорей с помощью Alibi

Чтобы сгенерировать объяснения якорей для модели «черного ящика», мы будем использовать библиотеку Python Alibi. В следующем скрипте показаны операторы импорта, необходимые для объяснения концепции. Мы будем использовать классификатор случайного леса, который также является моделью «черного ящика», поскольку существует ряд деревьев, генерирующих прогнозы, а прогнозы собираются для получения конечного результата.

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from alibi.explainers import AnchorTabular
from alibi.datasets import fetch_adult
```

В библиотеке Alibi есть несколько наборов данных для генерирования примеров, которые можно использовать для объяснения концепции. Здесь мы будем использовать набор данных о взрослых. Он известен, как набор данных о доходах по переписи населения, доступный в репозитории машинного обучения UCI. Здесь речь идет о модели бинарной классификации. Есть несколько характеристик для прогнозирования категории дохода и того, больше ли он 50 тыс. долл. или меньше.

```
adult = fetch_adult()
adult.keys()
```

Приведенный выше скрипт переносит данные в среду Jupyter. Ключами являются dict_keys(['data', 'target', 'feature_names', 'target_names', 'category_map']).

```
data = adult.data
target = adult.target
feature_names = adult.feature_names
category_map = adult.category_map
```

Функция отображения категорий ролей предназначена для отображения всех категориальных или строковых столбцов, присутствующих в данных.

```
from alibi.utils.data import gen_category_map
```

Категориальная карта – это просто отображение фиктивной переменной, потому что вы не можете использовать категориальные или строковые переменные в их нынешнем виде для вычислений.

```
np.random.seed(0)
data_perm = np.random.permutation(np.c_[data, target])
data = data_perm[:, :-1]
target = data_perm[:, -1]
```

Перед обучением модели необходимо произвольно распределить данные, имеющиеся в наборе данных, и использовать 33 000 записей для обучения модели и 2560 записей для тестирования или проверки.

```
idx = 30000
X_train, Y_train = data[:idx, :], target[:idx]
X_test, Y_test = data[idx+1:, :], target[idx+1:]

X_train.shape, Y_train.shape
X_test.shape, Y_test.shape

ordinal_features = [x for x in range(len(feature_names)) if x not in
list(category_map.keys())]
ordinal_features
```

После выбора порядковых характеристик (ordinal_features) можно применить их преобразователь, запущенный из конвейера, следующим образом:

```
ordinal_transformer = Pipeline(steps=[('imputer', SimpleImputer(strategy='median')),
('scaler', StandardScaler())])
categorical_features = list(category_map.keys())
categorical_features
```

Порядковые характеристики – это номера характеристик 0, 8, 9 и 10, а категориальные – 1, 2, 3, 4, 5, 6, 7 и 11. Для порядковых характеристик применяется стандартная шкала. Если есть отсутствующие значения, вы вводите медианные.

```

categorical_transformer = Pipeline(steps=[('imputer', SimpleImputer(strategy='median')),
                                         ('onehot', OneHotEncoder(handle_
                                         unknown='ignore'))])

categorical_transformer
preprocessor = ColumnTransformer(transformers=[('num', ordinal_transformer,
                                                 ordinal_features),
                                                 ('cat', categorical_transformer,
                                                 categorical_features)])

```

Для категориальных характеристик в случае отсутствующих значений вводится значение медианы и выполняется горячее кодирование для преобразования категориальной переменной. Если значение категориальной переменной неизвестно, оно игнорируется.

```

Preprocessor
ColumnTransformer(transformers=[('num',
                                  Pipeline(steps=[('imputer',
                                                   SimpleImputer(strategy='median')),
                                                 ('scaler', StandardScaler())])),
                                  [0, 8, 9, 10]),
                   ('cat',
                    Pipeline(steps=[('imputer',
                                   SimpleImputer(strategy='median')),
                                   ('onehot',
                                    OneHotEncoder(handle_
                                         unknown='ignore'))])),
                   [1, 2, 3, 4, 5, 6, 7, 11])])

```

Теперь модуль препроцессора готов к обучению модели. Функция `preprocessor.fit` преобразует набор данных для обучения модели. Модель случайного леса для классификации инициализируется с 50 деревьями в качестве количества оценок. Инициализированный объект хранится в `clf`. Функция `clf.fit` обучает модель.

```

preprocessor.fit(X_train)
np.random.seed(0)
clf = RandomForestClassifier(n_estimators=50)
clf.fit(preprocessor.transform(X_train), Y_train)

predict_fn = lambda x: clf.predict(preprocessor.transform(x))
print('Train accuracy: ', accuracy_score(Y_train, predict_fn(X_train)))
Train accuracy: 0.9655333333333334

print('Test accuracy: ', accuracy_score(Y_test, predict_fn(X_test)))
Test accuracy: 0.855859375

```

В качестве следующего шага мы собираемся использовать или подогнать якорь пояснителя к табличным данным. Якорь таблицы имеет требования к параметрам, показанные в табл. 12.1.

Таблица 12.1. Объяснения параметров для якоря таблицы

Параметры	Объяснение
Predictor	Объект модели, имеющий функцию предсказания, которая принимает входные данные и генерирует выходные данные
Feature_names	Список характеристик
Categorical_names	Словарь, в котором ключами являются столбцы характеристик, а значениями – категории для характеристики
OHE	Являются ли категориальные переменные кодированными горячим кодированием (one hot encoded – OHE), или нет. Если нет, предполагается, что они имеют порядковые кодировки
Seed	Для воспроизводимости

```

explainer = AnchorTabular(predict_fn, feature_names, categorical_
names=category_map, seed=12345)
explainer
explainer.explanations
explainer.fit(X_train, disc_perc=[25, 50, 75])
AnchorTabular(meta={

    'name': 'AnchorTabular',
    'type': ['blackbox'],
    'explanations': ['local'],
    'params': {'seed': 1, 'disc_perc': [25, 50, 75]}})

)
idx = 0
class_names = adult.target_names
print('Prediction: ', class_names[explainer.predictor(X_test[idx].
reshape(1, -1))[0]])
Prediction: <=50K

explanation = explainer.explain(X_test[idx], threshold=0.95)
explanation.name
print('Anchor: %s' % (' AND '.join(explanation.anchor)))
print('Precision: %.2f' % explanation.precision)
print('Coverage: %.2f' % explanation.coverage)
idx = 6
class_names = adult.target_names
print('Prediction: ', class_names[explainer.predictor(X_test[idx].
reshape(1, -1))[0]])
Prediction: >50K

```

Объект `explainer.explain` объясняет прогноз, сделанный классификатором для наблюдения. Функция `explain` имеет следующие параметры, приведенные в табл. 12.2.

Таблица 12.2. Значения параметров функции `explain`

Параметры	Объяснение
Xtest	Новые данные, которые необходимо объяснить
Threshold	Минимальный порог точности
batch_size	Размер партии, используемой для выборки
coverage_samples	Количество образцов, используемых для оценки покрытия из поиска результатов
beam_size	Количество якорей, расширяемых на каждом шаге построения нового якоря
stop_on_first	Если true, алгоритм поиска вернет первый якорь, который удовлетворяет ограничению вероятности
max_anchor_size	Максимальное количество характеристик в результате

```
explanation = explainer.explain(X_test[idx], threshold=0.95)
print('Anchor: %s' % (' AND '.join(explanation.anchor)))

Output : [Anchor: Capital Loss > 0.00 AND Relationship = Husband AND
Marital Status = Married AND Age > 37.00 AND Race = White AND Sex = Male
AND Country = United-States
]

print('Precision: %.2f' % explanation.precision)
print('Coverage: %.2f' % explanation.coverage)
```

В приведенном выше примере точность объяснения модели составляет только 72 %, а покрытие – всего 2 %. Можно сделать вывод, что покрытие и точность низкие из-за несбалансированности набора данных. В нем больше тех, кто зарабатывает меньше 50 тыс. долл., чем тех, кто зарабатывает больше указанной суммы. Аналогичную концепцию якорей можно применить к набору данных для классификации текста. В этом случае мы будем использовать набор данных для анализа настроений.

ЯКОРЬ ТЕКСТА ДЛЯ КЛАССИФИКАЦИИ ТЕКСТА

Функция якоря текста поможет вам получить объяснение якоря набора данных для классификации текста. Это может быть идентификация спама в электронной почте или анализ настроений. Параметры, которые можно настроить для генерации якоря текста, показаны в табл. 12.3.

Таблица 12.3. Объяснения параметров якоря текста

Параметры	Объяснение
Predictor	Обычно это объект обученной модели, который может генерировать прогнозы
sampling_strategy	Метод распределения возмущений, где неизвестные слова заменяются на UNK, образцы сходства – в соответствии с оценкой сходства с вложениями в текст и образцы языковой модели – в соответствии с выходным распределением языковой модели
NLP	Объект spaCy, когда методом выборки является сходство или неизвестность
language_model	Преобразователи маскированной языковой модели

Стратегии выборки, которые мы будем использовать, – это неизвестность и сходство, потому что метод трансформации выходит за рамки этой книги.

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
# import spacy
from alibi.explainers import AnchorText
from alibi.datasets import fetch_movie_sentiment
#from alibi.utils.download import spacy_model
```

Приведенный выше скрипт импортирует необходимые библиотеки и функции из библиотек.

```
movies = fetch_movie_sentiment()
movies.keys()
```

Приведенный выше скрипт загружает помеченные данные для классификации настроений.

```
data = movies.data
labels = movies.target
target_names = movies.target_names
```

Сюда входят данные фильмов, целевые колонки (маркируются как позитивные и негативные) и имена характеристик (представляют собой уникальный набор лексем из текста).

```
train, test, train_labels, test_labels = train_test_split(data, labels,
test_size=.2, random_state=42)
train, test, train_labels, test_labels
```

Приведенный выше скрипт разделяет данные на обучающий и тестовый наборы, а следующий ниже представляет обучающий и проверочный наборы данных.

```
train, val, train_labels, val_labels = train_test_split(train, train_labels,
    test_size=.1, random_state=42)
train, val, train_labels, val_labels

train_labels = np.array(train_labels)
train_labels

test_labels = np.array(test_labels)
test_labels

val_labels = np.array(val_labels)
val_labels

vectorizer = CountVectorizer(min_df=1)
vectorizer.fit(train)
```

Функция векторизатора подсчета создает матрицу терминов документа, где документ представлен в виде строк, а лексемы – в виде характеристик.

```
np.random.seed(0)
clf = LogisticRegression(solver='liblinear')
clf.fit(vectorizer.transform(train), train_labels)
```

Используется логистическая регрессия, поскольку это задача бинарной классификации с решателем lib linear. Следующая функция predict берет объект обученной модели и, учитывая набор тестовых данных, делает прогноз:

```
predict_fn = lambda x: clf.predict(vectorizer.transform(x))
preds_train = predict_fn(train)
preds_train

preds_val = predict_fn(val)
preds_val

preds_test = predict_fn(test)
preds_test

print('Train accuracy', accuracy_score(train_labels, preds_train))
print('Validation accuracy', accuracy_score(val_labels, preds_val))
print('Test accuracy', accuracy_score(test_labels, preds_test))
```

Точности обучения, тестирования и проверки находятся на одном уровне.

Чтобы сгенерировать объяснение якоря, необходимо установить spacy. Так же необходимо установить модуль en_core_web_md, что можно сделать с помощью следующего скрипта из командной строки:

```
python -m spacy download en_core_web_md
import spacy
model = 'en_core_web_md'
#spacy_model(model=model)
nlp = spacy.load(model)
```

Функции AnchorText требуются объект модели `spacy`, поэтому мы инициализируем объект `nlp` из `spacy`.

```
class_names = movies.target_names

# выберите экземпляр для объяснения
text = data[40]
print("* Text: %s" % text)

# вычислите прогноз класса
pred = class_names[predict_fn([text])[0]]
alternative = class_names[1 - predict_fn([text])[0]]
print("* Prediction: %s" % pred)

explainer = AnchorText(
    predictor=predict_fn,
    nlp=nlp
)

explanation = explainer.explain(text, threshold=0.95)
explainer
explanation

print('Anchor: %s' % (' AND '.join(explanation.anchor)))
print('Precision: %.2f' % explanation.precision)

Anchor: watchable AND bleak AND honest
Precision: 0.99

print('\nExamples where anchor applies and model predicts %s:' % pred)
print('\n'.join([x for x in explanation.raw['examples'][-1]['covered_true']]))

print('\nExamples where anchor applies and model predicts %s:' %
      alternative)
print('\n'.join([x for x in explanation.raw['examples'][-1]['covered_false']])))
```

Примеры, когда применяется якорь, а модель предсказывает негатив: усиление UNK UNK результаты UNK честные, но UNK UNK настолько мрачный UNK UNK UNK UNK едва ли можно наблюдать.

ЯКОРЬ ИЗОБРАЖЕНИЯ ДЛЯ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

Аналогично якорю текста у нас есть пояснения к модели якоря изображения для набора данных, предназначенного для классификации изображений. Здесь будет использоваться модный набор данных MNIST. В табл. 12.4 показаны параметры якоря изображения, которые необходимо объяснить.

Таблица 12.4. Объяснения параметров якоря изображения

Параметры	Объяснение
Predictor	Объект модели классификации изображений
image_shape	Форма изображения, подлежащего объяснению
segmentation_fn	Любая из встроенных строк функций сегментации: 'felzenszwalb', 'slic', 'quickshift' или пользовательская функция сегментации (вызываемая), которая возвращает маску изображения с метками для каждого суперпикселя
segmentation_kwargs	Аргументы ключевых слов для встроенных функций сегментации
images_background	Изображения для наложения на суперпиксели
Seed	Для воспроизведения результата

Рассмотрим следующие скрипты по созданию якорей изображений.

```
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, Dense, Dropout, Flatten,
MaxPooling2D, Input
from tensorflow.keras.models import Model
from tensorflow.keras.utils import to_categorical
from alibi.explainers import AnchorImage

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_
data()
print('x_train shape:', x_train.shape, 'y_train shape:', y_train.shape)
```

Набор данных для обучения содержит 60 000 образцов изображений, а набор данных для тестирования – 10 000. Мы собираемся разработать модель сверточной нейронной сети (CNN) для предсказания класса изображения и объяснения соответствующих якорей. Целевой класс имеет 10 меток (см. табл. 12.5).

Таблица 12.5. Описание целевого класса

Класс	Описание
0	Футболка/топ
1	Брюки
2	Пуловер
3	Платье
4	Пальто
5	Сандалии
6	Рубашка
7	Кроссовки
8	Сумка
9	Ботильоны

```
idx = 57
plt.imshow(x_train[idx]);

x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
```

Нормализация характеристик выполняется путем деления значения каждого пикселя на наибольшее его значение (255).

```
x_train = np.reshape(x_train, x_train.shape + (1,))
x_test = np.reshape(x_test, x_test.shape + (1,))
print('x_train shape:', x_train.shape, 'x_test shape:', x_test.shape)

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
print('y_train shape:', y_train.shape, 'y_test shape:', y_test.shape)
```

Модель CNN обычно используется для предсказания классов изображений. Она имеет сверточный двумерный фильтр для лучшего распознавания значений пикселей, за которым следует слой max pooling (иногда это усреднение (average pooling), а иногда максимальное объединение – max pooling) для подготовки абстрактного слоя. Слой отсева предназначен для борьбы с любой проблемой переподгонки, которая может возникнуть при прогнозировании модели.

```
def model():
    x_in = Input(shape=(28, 28, 1))
    x = Conv2D(filters=64, kernel_size=2, padding='same',
```

```

activation='relu'))(x_in)
x = MaxPooling2D(pool_size=2)(x)
x = Dropout(0.3)(x)

x = Conv2D(filters=32, kernel_size=2, padding='same',
activation='relu'))(x)
x = MaxPooling2D(pool_size=2)(x)
x = Dropout(0.3)(x)

x = Flatten()(x)
x = Dense(256, activation='relu'))(x)
x = Dropout(0.5)(x)
x_out = Dense(10, activation='softmax'))(x)

cnn = Model(inputs=x_in, outputs=x_out)
cnn.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

return cnn

cnn = model()
cnn.summary()

cnn.fit(x_train, y_train, batch_size=64, epochs=3)

# Оценить модель на тестовом наборе
score = cnn.evaluate(x_test, y_test, verbose=0)
print('Test accuracy: ', score[1])

```

Суперпиксели обеспечивают объяснение якорей в примере классификации изображений. Следовательно, чтобы определить основную характеристику, можно использовать якорь изображения.

```

def superpixel(image, size=(4, 7)):
    segments = np.zeros([image.shape[0], image.shape[1]])
    row_idx, col_idx = np.where(segments == 0)
    for i, j in zip(row_idx, col_idx):
        segments[i, j] = int((image.shape[1]/size[1]) * (i//size[0]) + j
                           //size[1])
    return segments

segments = superpixel(x_train[idx])
plt.imshow(segments);
predict_fn = lambda x: cnn.predict(x)
image_shape = x_train[idx].shape
explainer = AnchorImage(predict_fn, image_shape, segmentation_
fn=superpixel)

```

```
i = 11
image = x_test[i]
plt.imshow(image[:, :, 0]);

cnn.predict(image.reshape(1, 28, 28, 1)).argmax()

explanation = explainer.explain(image, threshold=.95, p_sample=.8, seed=0)

plt.imshow(explanation.anchor[:, :, 0]);

explanation.meta

explanation.params

{'custom_segmentation': True,
 'segmentation_kwargs': None,
 'p_sample': 0.8,
 'seed': None,
 'image_shape': (28, 28, 1),
 'segmentation_fn': 'custom',
 'threshold': 0.95,
 'delta': 0.1,
 'tau': 0.15,
 'batch_size': 100,
 'coverage_samples': 10000,
 'beam_size': 1,
 'stop_on_first': False,
 'max_anchor_size': None,
 'min_samples_start': 100,
 'n_covered_ex': 10,
 'binary_cache_size': 10000,
 'cache_margin': 1000,
 'verbose': False,
 'verbose_every': 1,
 'kwargs': {'seed': 0}}
```

ЗАКЛЮЧЕНИЕ

В этой главе мы рассмотрели пример табличных данных с использованием набора данных о взрослых, набора данных IMDB отзывов о фильмах для анализа настроений, а также набора данных о моде MNIST для примера классификации изображений. Функция якорей совершенно различна для табличных, текстовых данных и данных изображений. Здесь мы обсудили примеры объяснения классификации и прогнозирования изображений, текста и классификации изображений, используя объяснитель из Alibi в качестве новой библиотеки.

ГЛАВА 13

Объяснимость модели для экспертных систем, основанных на правилах

Системы, основанные на правилах, относительно легче объяснить конечным пользователям, потому что правила могут быть выражены в виде условий «если/иначе» (*if/else*). Действительно, иногда существует несколько условий *if/else*. Экспертные системы детерминированы по своей природе и очень точны, потому что они управляются хорошо установленными правилами и условиями. Необходимость в объяснении возникает, когда несколько правил либо противоречат друг другу, либо сочетаются таким образом, что их трудно интерпретировать. В области искусственного интеллекта экспертная система – это программа, разработанная для моделирования реального процесса принятия решений. Экспертные системы предназначены для решения сложных проблем путем простых рассуждений и представлены в виде условий «если/тогда/иначе» (*if/then/else*). Экспертные системы спроектированы так, чтобы действовать как механизм умозаключений. Роль механизма умозаключений заключается в принятии решения. Иногда экспертные системы просты, например как дерево решений, а иногда очень сложны и требуют базы знаний. Иногда эти базы знаний представляют собой онтологию, являющуюся формой совокупности знаний, с помощью которой можно сформулировать конкретное правило и сделать вывод.

ЧТО ТАКОЕ ЭКСПЕРТНАЯ СИСТЕМА?

База знаний – это хранилище рассуждений, с помощью которого работают условия «если/иначе» в экспертной системе. В любой традиционной компьютерной программе рассуждения и логика встроены в скрипт, что делает невозможным для кого-либо, кроме специалистов в области технологий, понять эти рассуждения. Цель системы, основанной на знаниях, состоит в том, чтобы сделать процесс принятия решений явным. Целью экспертной системы является явное представление правил в формате, который является интуитивным, не требующим объяснений и легко понимаемым для лиц, принимающих решения. Онтология – это формальное представление классов, подклассов и суперклассов. Сущности связаны или ассоциированы через некоторые общие

свойства. Отношения могут быть концепциями, ассоциациями и другими метасвойствами. Классификация онтологии помогает поддерживать формальную структуру базы знаний, на которой работает экспертная система.

Любая экспертная система состоит из двух компонентов:

- базы знаний – формальной структуры, т. е. онтологии;
- механизма умозаключений – либо системы, основанной на правилах, либо системы, основанной на машинном обучении.

Механизм умозаключений использует две различные формы рассуждений для выработки правила принятия решения:

- прямую цепочку;
- обратную цепочку.

ПРЯМАЯ И ОБРАТНАЯ ЦЕПОЧКИ

Рисунок 13.1 объясняет процесс построения прямой цепочки. Абстрактные характеристики могут быть извлечены из базовых, а окончательное решение может быть принято на основе абстрактных характеристик.



Рис. 13.1. Прямая цепочка

Метод прямой цепочки объясняет, что происходит с решением. Предположим, что в проблеме структурированных данных у нас есть характеристики. Они имеют свои собственные значения. Сочетание условий if/else приводит к формированию абстрактной характеристики, которая приводит к окончательному решению. Если кто-то захочет проверить, как было принято окончательное решение, его можно проследить. Механизм умозаключений прослеживает цепочку условий if/else и выводов, чтобы прийти к выводу о принятом решении. Аналогичным образом можно объяснить обратную цепочку на рис. 13.2.

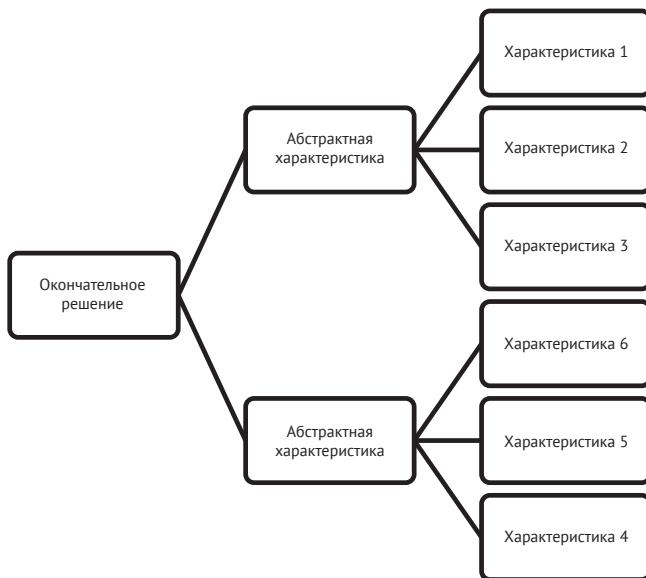


Рис. 13.2. Обратная цепочка

Логика, которой следует обратная цепочка, заключается в том, чтобы понять, почему принимается определенное решение. Механизм принятия решений экспертной системы пытается выяснить условия, которые привели к принятию решения, например спам или не спам. Здесь вероятность того, что наличие определенного символа либо привело к спаму, либо нет, представляет интерес для механизма принятия решений.

ИЗВЛЕЧЕНИЕ ПРАВИЛ С ПОМОЩЬЮ SCIKIT-LEARN

Объяснение правил с помощью модели на основе дерева решений, где предсказание оттока может быть объяснено на основе истории использования клиентов в базе данных по оттоку абонентов, показано с помощью библиотеки Python scikit-learn. Чтобы объяснить процесс создания и извлечения правил, рассмотрим набор данных прогнозирования оттока клиентов в телекоммуникациях. Предварительное прогнозирование вероятного оттока клиентов дает преимущество бизнесу при проведении бесед с ними, понимании их проблем и, следовательно, попытке удержать их. Удержание клиентов обходится дешевле, чем их привлечение. Обучающий набор данных был предварительно обработан, где категориальные состояния столбцов были преобразованы в столбцы при помощи горячего кодирования (one hot encoded columns). Аналогично есть и другие категориальные характеристики (код региона, статус международного тарифного плана, тарифный план голосовой почты и количество звонков в службу поддержки), которые преобразованы в горячо кодированные характеристики.

```
import pandas as pd
df_total = pd.read_csv('Processed_Training_data.csv')
```

После считывания данных в блокнот Jupyter можно просмотреть первые пять записей с помощью функции `head`. Первый столбец – это серийный номер; он не имеет значения, поэтому его можно удалить.

```
df_total.columns  
del df_total['Unnamed: 0']  
df_total.head()df_total.columns  
del df_total['Unnamed: 0']  
df_total.head()
```

Следующий скрипт разделяет целевой столбец как переменную Y и остальные характеристики – как переменную X:

```
import numpy as np  
Y = np.array(df_total.pop('churn_yes'))  
X = np.array(df_total)  
  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import confusion_matrix, classification_report  
  
xtrain,xtest,ytrain,ytest = train_test_split(X,Y,test_size=0.20,random_  
state=1234)  
xtrain.shape,xtest.shape,ytrain.shape,ytest.shape  
  
import numpy as np, pandas as pd, matplotlib.pyplot as plt, pydotplus  
from sklearn import tree, metrics, model_selection, preprocessing  
from IPython.display import Image, display
```

Для визуализации дерева решений в среде Jupyter необходимо установить библиотеку `pydotplus`.

```
!pip install pydotplus
```

Для начала можно обучить простую модель дерева решений, извлечь правила и визуализировать их. Вы можете перейти к сложным моделям, таким как классификаторы случайного леса или классификаторы на основе повышения градиента, которые обучают модель, используя большое количество деревьев, определенное пользователем (от 100 до 10000, но не ограничиваясь этими двумя числами). Идентификация правил в ансамблевых моделях также может быть получена с помощью другой библиотеки под названием RuleFit, которая выходит за рамки данной книги с точки зрения доступности библиотеки. Я включил извлечение правил на основе условного порога и уровня доверия, используя объяснение якорей из библиотеки Alibi в главе 12.

```
dt1 = tree.DecisionTreeClassifier()  
dt1  
dt1.fit(xtrain,ytrain)
```

```
dt1.score(xtrain,ytrain)
dt1.score(xtest,ytest)
```

Инициализируется дерево решений с параметрами по умолчанию, и объект сохраняется как dt1. Затем происходит обучение модели с помощью метода fit, и функция score используется для получения точности модели на основе обучающего и тестового наборов данных. Точность обучения составляет 100 %, а точность тестирования – 89.8 %. Это явный признак чрезмерной подгонки модели. Это происходит из-за параметра «максимальная глубина» (`max_depth`) в классификаторе дерева решений. По умолчанию он равен `None`, что означает, что ветви дерева будут расширяться до тех пор, пока все образцы не будут в него помещены. Если модель переподогнана к данным, то правила, извлеченные из нее, также будут вводить в заблуждение, иногда генерируя ложноположительные, а иногда – ложноотрицательные результаты. Поэтому первым шагом здесь является получение правильной модели. Нам нужно точно настроить гиперпараметры, чтобы создать лучшую и стабильную модель. Тогда правила будут действительны.

```
dt2 = tree.DecisionTreeClassifier(class_weight=None, criterion='entropy',
max_depth=4,
          max_features=None, max_leaf_nodes=None,
          min_impurity_decrease=0.0, min_impurity_split=None,
          min_samples_leaf=10, min_samples_split=30,
          min_weight_fraction_leaf=0.0, presort=False,
          random_state=None, splitter='best')
```

В этой версии модель дерева решений dt2, алгоритм создания ветвей выбран как энтропия, максимальная глубина дерева установлена 4, а также приняты во внимание некоторые другие параметры, способные повлиять на точность модели.

```
dt2.fit(xtrain,ytrain)
dt2.score(xtrain,ytrain)
dt2.score(xtest,ytest)

pred_y = dt2.predict(xtest)
print(classification_report(pred_y,ytest))
```

На второй итерации модели мы сможем сблизить точность обучения и точность тестирования, тем самым устранив возможность переподгонки модели. Точность обучения во второй версии составляет 92 %, а точность тестирования – 89.9 %. Точность классификации в соответствии с прогнозом модели составляет 90 %.

Как только получим лучшую модель, мы сможем извлечь из нее характеристики, чтобы узнать веса.

```
import numpy as np, pandas as pd, matplotlib.pyplot as plt, pydotplus
from sklearn import tree, metrics, model_selection, preprocessing
from IPython.display import Image, display

dt2.feature_importances_
```

Функция `feature_importances_` показывает оценки влияния каждой характеристики на прогноз модели. Функция `export_graphviz` генерирует точечный объект данных, который может быть использован библиотекой `pydotplus` для создания графического представления дерева решений (см. рис. 13.3).

```
dot_data = tree.export_graphviz(dt2,
                                out_file=None,
                                filled=True,
                                rounded=True,
                                feature_names=['state_AL', 'state_AR',
                                'state_AZ', 'state_CA', 'state_CO',
                                'state_CT',
                                'state_DC', 'state_DE', 'state_FL', 'state_HI', 'state_IA',
                                'state_ID', 'state_IL', 'state_IN', 'state_KS', 'state_KY', 'state_LA',
                                'state_MA', 'state_MD', 'state_ME', 'state_MI', 'state_MN', 'state_MO',
                                'state_MS', 'state_MT', 'state_NC', 'state_ND', 'state_NE', 'state_NH',
                                'state_NJ', 'state_NM', 'state_NV', 'state_NY', 'state_OH', 'state_OK',
                                'state_OR', 'state_PA', 'state_RI', 'state_SC', 'state_SD', 'state_TN',
                                'state_TX', 'state_UT', 'state_VA', 'state_VT', 'state_WA', 'state_WI',
                                'state_WV', 'state_WY', 'area_code_area_code_415',
                                'area_code_area_code_510', 'international_plan_yes',
                                'voice_mail_plan_yes', 'num_cust_serv_calls_1',
                                'num_cust_serv_calls_2', 'num_cust_serv_calls_3',
                                'num_cust_serv_calls_4', 'num_cust_serv_calls_5',
                                'num_cust_serv_calls_6', 'num_cust_serv_calls_7',
                                'num_cust_serv_calls_8', 'num_cust_serv_calls_9',
                                'total_day_minutes',
                                'total_day_calls', 'total_day_charge', 'total_eve_minutes',
                                'total_eve_calls', 'total_eve_charge', 'total_night_minutes',
                                'total_night_calls', 'total_night_charge', 'total_intl_minutes',
                                'total_intl_charge', 'total_intl_calls_4.0',
                                'number_vmail_messages_4.0'],
                                class_names=['0', '1'])

graph = pydotplus.graph_from_dot_data(dot_data)
display(Image(graph.create_png()))
```

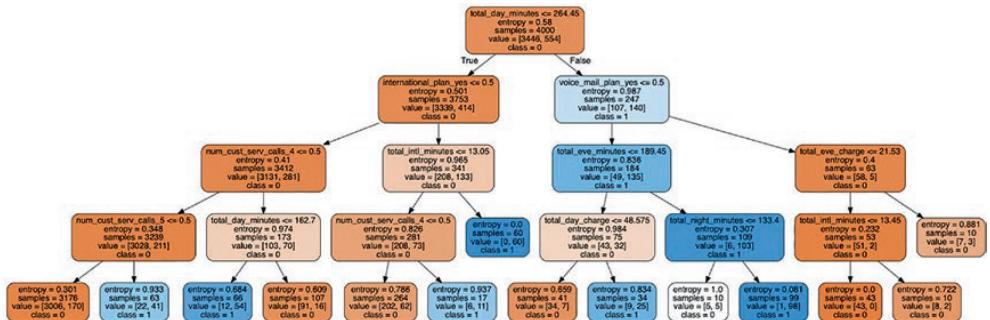


Рис. 13.3. Объект дерева решений, показывающий правила от корневого узла к листовым узлам

Дерево решений начинается с корневого узла, следует логике создания ветвей в соответствии с изменением энтропии или формулой Джини для создания последующих узлов и, наконец, достигает конечного узла, который также считается листовым. Весь путь от корневого до листового узла рассматривается как правило.

```
from sklearn.tree import export_text
tree_rules = export_text(dt1, feature_names=list(df_total.
columns), decimals=0, show_weights=True)
print(tree_rules)
```

Правила на основе дерева можно также экспорттировать в виде текста, чтобы можно было разобрать текст и интегрировать его в другие приложения. Количества правил, генерируемых этой моделью дерева решений, велико, так как модель является переподогнанной. Поэтому доверять ей ошибочно.

Вот почему мы ищем модель с лучшей подгонкой. Мы хотим генерировать заслуживающие доверия правила, которые являются конечными по своей природе, чтобы их было возможно реализовать в производственном сценарии.

```
tree_rules = export_text(dt2, feature_names=list(df_total.
columns), decimals=0, show_weights=True)
print(tree_rules)
--- total_day_minutes <= 264
|--- international_plan_yes <= 0
|   |--- num_cust_serv_calls_4 <= 0
|   |   |--- num_cust_serv_calls_5 <= 0
|   |   |   |--- weights: [3006, 170] class: 0
|   |   |--- num_cust_serv_calls_5 > 0
|   |   |   |--- weights: [22, 41] class: 1
|   |--- num_cust_serv_calls_4 > 0
|   |   |--- total_day_minutes <= 163
|   |   |   |--- weights: [12, 54] class: 1
|   |   |--- total_day_minutes > 163
|   |   |   |--- weights: [91, 16] class: 0
|--- international_plan_yes > 0
```

```

    |   |--- total_intl_minutes <= 13
    |   |   |--- num_cust_serv_calls_4 <= 0
    |   |   |   |--- weights: [202, 62] class: 0
    |   |   |--- num_cust_serv_calls_4 > 0
    |   |   |   |--- weights: [6, 11] class: 1
    |   |--- total_intl_minutes > 13
    |   |   |--- weights: [0, 60] class: 1
--- total_day_minutes > 264
    |--- voice_mail_plan_yes <= 0
        |--- total_eve_minutes <= 189
            |--- total_day_charge <= 49
                |--- weights: [34, 7] class: 0
            |--- total_day_charge > 49
                |--- weights: [9, 25] class: 1
        |--- total_eve_minutes > 189
            |--- total_night_minutes <= 133
                |--- weights: [5, 5] class: 0
            |--- total_night_minutes > 133
                |--- weights: [1, 98] class: 1
--- voice_mail_plan_yes > 0
    |--- total_eve_charge <= 22
        |--- total_intl_minutes <= 13
            |--- weights: [43, 0] class: 0
        |--- total_intl_minutes > 13
            |--- weights: [8, 2] class: 0
    |--- total_eve_charge > 22
        |--- weights: [7, 3] class: 0
tree_rules = export_text(dt2, feature_names=list(df_total.columns))
print(tree_rules)
--- total_day_minutes <= 264.45
    |--- international_plan_yes <= 0.50
        |--- num_cust_serv_calls_4 <= 0.50
            |--- num_cust_serv_calls_5 <= 0.50
                |--- class: 0
            |--- num_cust_serv_calls_5 > 0.50
                |--- class: 1
        |--- num_cust_serv_calls_4 > 0.50
            |--- total_day_minutes <= 162.70
                |--- class: 1
            |--- total_day_minutes > 162.70
                |--- class: 0
    |--- international_plan_yes > 0.50
        |--- total_intl_minutes <= 13.05
            |--- num_cust_serv_calls_4 <= 0.50
                |--- class: 0
            |--- num_cust_serv_calls_4 > 0.50
                |--- class: 1
        |--- total_intl_minutes > 13.05
                |--- class: 1
--- total_day_minutes > 264.45
    |--- voice_mail_plan_yes <= 0.50
        |--- total_eve_minutes <= 189.45
            |--- total_day_charge <= 48.58
                |--- class: 0
            |--- total_day_charge > 48.58
                |--- class: 1

```

```

|   |   |--- total_eve_minutes > 189.45
|   |   |   |--- total_night_minutes <= 133.40
|   |   |   |   |--- class: 0
|   |   |   |   |--- total_night_minutes > 133.40
|   |   |   |   |--- class: 1
|--- voice_mail_plan_yes > 0.50
|   |--- total_eve_charge <= 21.53
|   |   |--- total_intl_minutes <= 13.45
|   |   |   |--- class: 0
|   |   |   |--- total_intl_minutes > 13.45
|   |   |   |--- class: 0
|   |   |--- total_eve_charge > 21.53
|   |   |--- class: 0

```

Объясним правило: если всего минут вызовов за день меньше или равно 264.45 – при наличии подписки на международные тарифные планы – и клиент делает более четырех звонков в службу поддержки, то, скорее всего, он откажется от услуг. И наоборот, если клиент совершает менее четырех звонков в службу поддержки – при неизменности всех остальных факторов, – он, скорее всего, продолжит сотрудничество. Аналогичным образом можно объяснить все остальные правила.

Приведенные выше примеры вытекают из поведения модели, поскольку модель задает пороговые значения при принятии решения об определении сценария «отток или не отток». Однако пороговые значения правила определяются бизнес-пользователем или экспертом в данной области, который обладает достаточными знаниями о постановке проблемы, полученными от экспертной системы.

Взаимодействие с экспертной системой может быть реализовано с помощью методов обработки естественного языка, таких как разработка системы «вопрос/ответ». Однако проектирование и разработка экспертной системы требует выполнения следующих шагов:

- постановки задачи;
- нахождения эксперта в этой области;
- определения экономической эффективности решения.

ПОТРЕБНОСТЬ В СИСТЕМЕ, ОСНОВАННОЙ НА ПРАВИЛАХ

Потребность в экспертной системе, основанной на правилах, может быть обусловлена следующими факторами:

- существует недостаточно данных для обучения алгоритма машинного или глубокого обучения;
- эксперты в данной области формируют правила из своего опыта, который основан на имеющейся таблице истинности. Поиск таких экспертов всегда является сложной задачей;
- существует проблема воспроизведения моделей машинного или глубокого обучения, поскольку есть небольшая неопределенность в прогнозировании или классификации;

- иногда трудно объяснить решения по прогнозированию, создаваемые моделями машинного обучения;
- иногда модели машинного обучения выдают неверные результаты, даже если вы обучили их с помощью правильной процедуры, следовательно, возникает необходимость переписать прогноз машины с помощью основанного на правилах результата.

ПРОБЛЕМЫ ЭКСПЕРТНОЙ СИСТЕМЫ

Основные проблемы экспертной системы, основанной на правилах для объяснения решения, заключаются в следующем:

- слишком большое количество правил может привести к путанице в объяснении результата;
- слишком мало правил может привести к упусканию важного сигнала;
- вы должны управлять конфликтующими правилами и добавить механизм оповещения конечного пользователя о конфликте;
- вы должны найти ансамбли правил для выявления общности между различными правилами;
- если два правила приводят к одному и тому же решению, обратная цепочка может привести к путанице.

ЗАКЛЮЧЕНИЕ

Создание экспертной системы – это трудоемкий процесс. В этой главе было представлено смоделированное представление о том, как можно генерировать и разрабатывать правила для целей объяснимости. Вы узнали о подходах, проблемах и потребностях для системы, основанной на правилах. В реальном мире основные процессы робототехники осуществляются с помощью экспертных систем, основанных на правилах. Модели ИИ могут объяснить решения, но управление действиями ограничено, следовательно, существует потребность в экспертной системе, которая часто отменяет решения модели ИИ, когда они ведут в неблагоприятном направлении. Более того, экспертная система, основанная на правилах, имеет решающее значение для задач, связанных с компьютерным зрением. Модель ИИ можно обмануть, используя некую абстрактную форму изображения, но система, основанная на правилах, может определить ее и распознать аномалии.

ГЛАВА 14

Объяснимость моделей для компьютерного зрения

Задачи компьютерного зрения, такие как классификация изображений и обнаружение объектов, постепенно с каждым днем становятся все более качественными благодаря постоянным исследованиям в области повышения точности моделей, эволюции новых фреймворков и растущему сообществу, использующему открытый исходный код. Результаты последних современных моделей машинного обучения более многообещающи, чем полдесятилетия назад, что вселяет уверенность в том, что мы сможем расширить горизонт проблем за счет решения более сложных задач. Среди примеров использования компьютерного зрения наиболее известными являются розничная торговля, сельское хозяйство, здравоохранение, автомобильная промышленность и игровая индустрия. Некоторые отрасли из этого списка имеют юридический мандат на ответственное использование ИИ. Европейский союз также пытается регулировать применение и использование ИИ. Если прогнозы или классификации, генерируемые моделью классификации изображений, совершают ошибки, пользователи не будут доверять модели и ее прогнозам. Необходимо понять, почему модель ИИ выдала данный прогноз или классификацию, а не другой. Объяснимость модели показывает конечному пользователю, какие именно части изображения заставили модель предсказать класс А по сравнению с другими классами.

Почему объяснимость для данных изображений?

Изображения преобразуются в значения пикселей. Значения пикселей впоследствии используются в качестве характеристики для обучения метки, которую изображение представляет наилучшим образом. Если модель ИИ генерирует неверный прогноз/классификацию, в организации ответственность несут две группы людей – специалист по анализу данных, построивший модель, и заинтересованное лицо, которое одобрило запуск модели в производство. Неправильная классификация данных изображения обусловлена двумя причинами:

- модель не в состоянии учесть корректный набор характеристик для прогнозирования. Вместо этого она выбирает из данных неправильные сигналы и признаки. Правильный набор характеристик не учитывается в процессе обучения;
- не проводится достаточного обучения, чтобы алгоритм мог очень хорошо обобщать неизвестные данные.

ЯКОРЬ ИЗОБРАЖЕНИЯ С ПОМОЩЬЮ ALIBI

Для задач компьютерного зрения, таких как обнаружение объектов и классификация изображений, можно объяснить ключевые характеристики, которые отличают изображение сумки от изображения футболки, например. Этого можно достичь с помощью объяснения якорей, рассмотренного в главе 11 этой книги. Однако этот же метод можно распространить и на другие задачи компьютерного зрения, такие как идентификация поддельного изображения и возможная компрометация изображения. Шаги в этом случае следующие:

- модель обнаружения объектов или классификации изображений может быть обучена для каждой области, например для известных личностей и популярных или распространенных объектов;
- если вы хотите определить подлинность изображения другого человека или обрезанного изображения, можете получить прогноз, но порог вероятности будет немного ниже;
- в этом случае можно задействовать объяснения якорей, чтобы узнать ключевые характеристики или выделяющиеся пиксели, которые приводят к прогнозу;
- теоретически это нормально. На практике вы можете убедиться в этом, используя следующую формулу суперпикселя.

Объяснение якорей – это цвета или контрастные цвета, которые отличают изображение или объект от других изображений. Вам необходимо написать функцию для генерирования суперпикселей для любого заданного изображения.

МЕТОД ИНТЕГРИРОВАННЫХ ГРАДИЕНТОВ

Цель метода интегрированных градиентов заключается в получении оценки важности каждой из характеристик, используемых для обучения модели машинного или глубокого обучения. Мы используем градиентный спуск для обновления весов модели глубокой нейронной сети. При обратном процессе можно использовать интеграцию весов, приписываемых оценке важности характеристики. Градиенты могут быть определены как наклон выходных данных по отношению к входным характеристикам в модели глубокого обучения. Подход с интегрированными градиентами может быть использован для понимания важности пикселя при распознавании правильного изображения в задаче классификации изображений. Градиенты обычно вычисляются для выхода того класса, для которого вероятность по отношению к пиксели входного изображения наиболее высока.

```
import numpy as np
import os

import tensorflow as tf
from tensorflow.keras.layers import Activation, Conv2D, Dense, Dropout
from tensorflow.keras.layers import Flatten, Input, Reshape, MaxPooling2D
```

```

from tensorflow.keras.models import Model
from tensorflow.keras.utils import to_categorical
from alibi.explainers import IntegratedGradients

import matplotlib.pyplot as plt
print('TF version: ', tf.__version__)
print('Eager execution enabled: ', tf.executing_eagerly()) # True

train, test = tf.keras.datasets.mnist.load_data()
X_train, y_train = train
X_test, y_test = test
test_labels = y_test.copy()
train_labels = y_train.copy()

```

Чтобы объяснить работу подхода интегрированных градиентов, воспользуемся набором данных MNIST, который легко понять, и многие люди знакомы с ним. В табл. 14.1 объяснены параметры модели.

Таблица 14.1. Параметры интегрированных градиентов

Параметры	Объяснения
Model	Модель TensorFlow или Keras
Layer	Слой, относительно которого рассчитываются градиенты. Если не указано, градиенты рассчитываются относительно входа
Method	Метод для интегральной аппроксимации. Доступные методы – riemann_left, riemann_right, riemann_middle, riemann_trapezoid и gausslegendre
N_steps	Количество шагов в аппроксимации интеграла пути от базовой линии до входного экземпляра

```

X_train = X_train.reshape(-1, 28, 28, 1).astype('float64') / 255
X_test = X_test.reshape(-1, 28, 28, 1).astype('float64') / 255
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)

load_mnist_model = False
save_model = True

filepath = './model_mnist/' # перейти в каталог, где сохранена модель
if load_mnist_model:
    model = tf.keras.models.load_model(os.path.join(filepath, 'model.h5'))
else:
    # определение модели
    inputs = Input(shape=(X_train.shape[1:]), dtype=tf.float64)
    x = Conv2D(64, 2, padding='same', activation='relu')(inputs)

```

```
x = MaxPooling2D(pool_size=2)(x)
x = Dropout(.3)(x)
x = Conv2D(32, 2, padding='same', activation='relu')(x)
x = MaxPooling2D(pool_size=2)(x)
x = Dropout(.3)(x)
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(.5)(x)
logits = Dense(10, name='logits')(x)
outputs = Activation('softmax', name='softmax')(logits)
model = Model(inputs=inputs, outputs=outputs)
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# обучение модели
model.fit(X_train,
          y_train,
          epochs=6,
          batch_size=256,
          verbose=1,
          validation_data=(X_test, y_test)
         )
if save_model:
    if not os.path.exists(filepath):
        os.makedirs(filepath)
    model.save(os.path.join(filepath, 'model.h5'))
```

Чтобы генерировать интегрированные градиенты, целевая переменная определяет, какой класс выходных данных должен быть рассмотрен для расчета атрибутов с использованием интеграционного подхода.

```
import tensorflow as tf
from alibi.explainers import IntegratedGradients

model = tf.keras.models.load_model(os.path.join(filepath, 'model.h5'))
ig = IntegratedGradients(model,
                          layer=None,
                          method="gausslegendre",
                          n_steps=50,
                          internal_batch_size=100)
```

Импортируется модуль интегрированного градиента из `alibi.explainers`, загружается предварительно обученная модель TensorFlow или Keras, а затем генерируются градиенты. Есть пять различных методов аппроксимации интеграла, основанных на сложности классификации изображения. Не существует

прямого правила или способа узнать, какие методы работают, поэтому необходимо итеративно опробовать все методы.

```
# Инициализация экземпляра IntegratedGradients
n_steps = 50
method = "gausslegendre"
ig = IntegratedGradients(model,
                          n_steps=n_steps,
                          method=method)
```

Количество шагов, которое вы можете увеличить, зависит от вычислительной мощности компьютера и количества образцов в папке для обучения модели.

```
# Вычисление атрибутов для первых 10 изображений в тестовом наборе
nb_samples = 10
X_test_sample = X_test[:nb_samples]
predictions = model(X_test_sample).numpy().argmax(axis=1)
explanation = ig.explain(X_test_sample,
                          baselines=None,
                          target=predictions)
```

Чтобы вычислить атрибуты для первых 10 изображений из тестового набора, можно взять прогнозы (`predictions`) в качестве цели и отсутствие (`None`) в качестве базовой линии. Если выбрать базовую линию `None`, то в качестве базовой линии будет использоваться черный цвет, который является фоновым цветом изображения.

```
# Метаданные объекта объяснения
explanation.meta

# Поля данных из объекта объяснения
explanation.data.keys()

# Получить значения атрибутов из объекта объяснения
attrs = explanation.attributions[0]
```

Получив атрибуты, мы можем показать позитивные и негативные атрибуты для целевого класса в графическом формате.

```
fig, ax = plt.subplots(nrows=3, ncols=4, figsize=(10, 7))
image_ids = [0, 1, 9]
cmap_bound = np.abs(attrs[[0, 1, 9]]).max()

for row, image_id in enumerate(image_ids):
    # оригинальные изображения
    ax[row, 0].imshow(X_test[image_id].squeeze(), cmap='gray')
    ax[row, 0].set_title(f'Prediction: {predictions[image_id]}')
```

```

# атрибуты
attr = attrs[image_id]
im = ax[row, 1].imshow(attr.squeeze(), vmin=-cmap_bound, vmax=cmap_
bound, cmap='PiYG')

# позитивные атрибуты
attr_pos = attr.clip(0, 1)
im_pos = ax[row, 2].imshow(attr_pos.squeeze(), vmin=-cmap_bound,
vmax=cmap_bound, cmap='PiYG')

# негативные атрибуты
attr_neg = attr.clip(-1, 0)
im_neg = ax[row, 3].imshow(attr_neg.squeeze(), vmin=-cmap_bound,
vmax=cmap_bound, cmap='PiYG')

ax[0, 1].set_title('Attributions');
ax[0, 2].set_title('Positive attributions');
ax[0, 3].set_title('Negative attributions');

for ax in fig.axes:
    ax.axis('off')

fig.colorbar(im, cax=fig.add_axes([0.95, 0.25, 0.03, 0.5]));

```

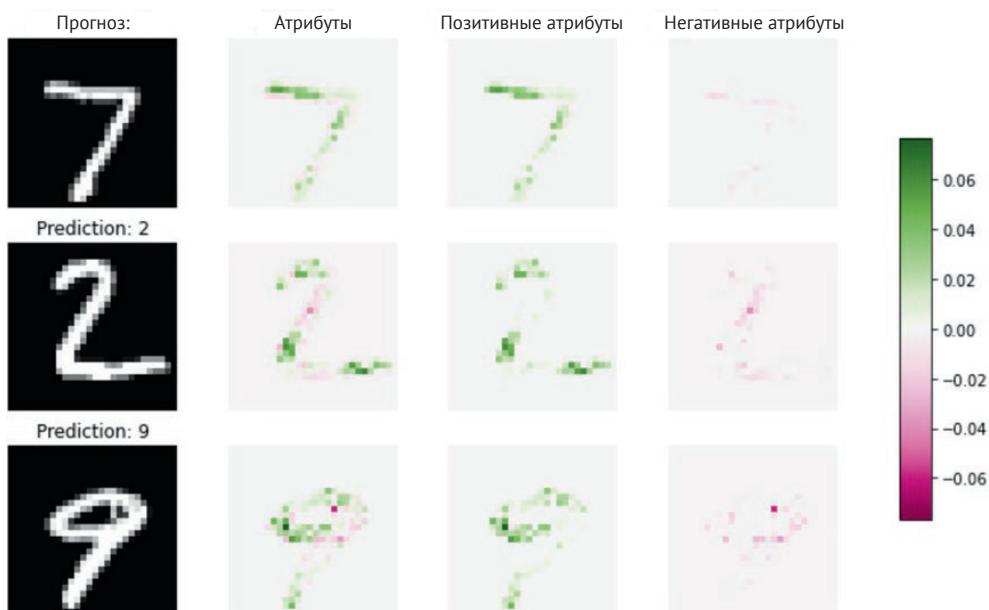


Рис. 14.1. Атрибуты, созданные интегрированными градиентами

На рис. 14.1 предсказанными цифрами являются 7, 2 и 9. Позитивные атрибуты в основном соответствуют окончательному прогнозу, а негативных атрибутов довольно мало.

ЗАКЛЮЧЕНИЕ

Атрибуты изображений важны для объяснения причины определенной классификации изображений. Они могут быть созданы с использованием подхода доминирующего пикселя или идентификации влияющих пикселей изображения. Это может быть полезно в некоторых отраслях. В производстве это может быть использовано для идентификации дефектных продуктов по их изображениям. В здравоохранении это позволяет классифицировать и выявлять аномалии на снимках. Во всех областях важно объяснить, почему модель классифицировала тот или иной класс. Если вы сумеете объяснить прогнозы, то сможете завоевать доверие пользователей к модели и тем самым повысить уровень внедрения моделей ИИ в промышленности для решения сложных бизнес-проблем.

Книги издательства «ДМК Пресс» можно заказать
в торгово-издательском холдинге «КТК Галактика» наложенным платежом,
выслав открытку или письмо по почтовому адресу:

115487, г. Москва, пр. Андропова д. 38 оф. 10.

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.galaktika-dmk.com.

Оптовые закупки: тел. (499) 782-38-89.

Электронный адрес: books@aliants-kniga.ru.

Прадипта Мишра

Объяснимые модели искусственного интеллекта на Python

Модель искусственного интеллекта.

Объяснения с использованием библиотек, расширений
и фреймворков на основе языка Python

Главный редактор *Мовчан Д. А.*

dmkpress@gmail.com

Зам. главного редактора *Сенченкова Е. А.*

Корректор *Абросимова Л. А.*

Верстка *Луценко С. В.*

Дизайн обложки *Мовчан А. Г.*

Формат 70×100 1/16.

Гарнитура «PT Serif». Печать цифровая.

Усл. печ. л. 24,21. Тираж 200 экз.

Веб-сайт издательства: www.dmkpress.com

В этой книге рассматриваются так называемые модели «черного ящика» для повышения адаптивности, интерпретируемости и объяснимости решений, принимаемых алгоритмами искусственного интеллекта (ИИ), с использованием таких фреймворков, как библиотеки Python XAI, TensorFlow 2.0+, Keras, а также пользовательских фреймворков, использующих оболочки Python. Излагаются основы объясимости и интерпретируемости моделей, обсуждаются методы и системы для интерпретации линейных, нелинейных моделей и моделей временных рядов, используемых в ИИ. Вы узнаете, как алгоритм ИИ принимает решение и как сделать модель ИИ интерпретируемой и объяснимой, ознакомитесь с моделями глубокого обучения.

Среди рассматриваемых тем:

- предвзятость и добросовестная этическая практика моделей ИИ;
- оценка надежности и справедливости моделей ИИ;
- разработка механизмов для раскрытия моделей «черного ящика»;
- основные элементы доверия к моделям ИИ;
- повышение уровня внедрения ИИ.

Прадипта Мишра – руководитель отдела искусственного интеллекта в компании L&T Infotech (LTI), возглавляет группу специалистов по анализу данных, вычислительной лингвистике, машинному и глубокому обучению, участвующих в создании продукта следующего поколения Leni – первого в мире виртуального специалиста по обработке данных. Вошел в число 40 лучших специалистов по обработке данных Индии в возрасте до 40 лет, по версии журнала Analytics India. Автор четырех книг. Выступал в университетах, технических учреждениях и на форумах; на конференции TED представил доклад «Могут ли машины думать?».

Интернет-магазин:
www.dmkpress.com

Оптовая продажа:
КТК «Галактика»
books@aliens-kniga.ru

Apress



ISBN 978-5-93700-124-5



9 785937 001245 >