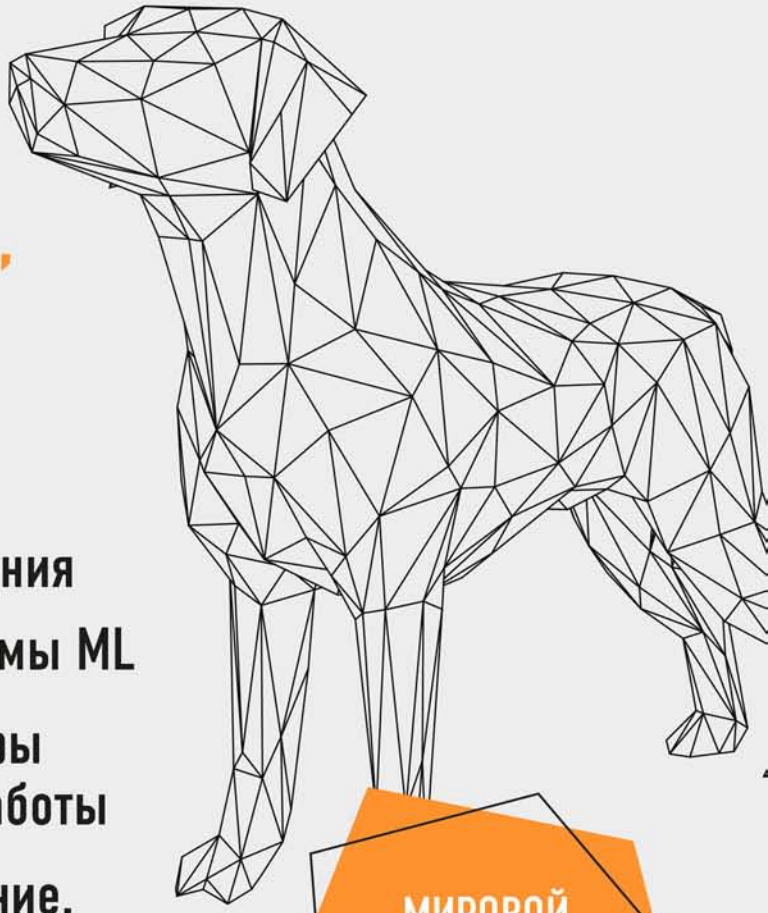
Оливер Теобальд

МАШИННОЕ ОБУЧЕНИЕ ДЛЯ АБСОЛЮТНЫХ НОВИЧКОВ



ВВОДНЫЙ КУРС, ИЗЛОЖЕННЫЙ ПРОСТЫМ ЯЗЫКОМ

- Понятные объяснения
- Основные алгоритмы ML
- Наглядные примеры и практические работы
- Горячее кодирование, классическая статистика и правильный код

МИРОВОЙ КОМПЬЮТЕРНЫЙ БЕСТСЕЛЛЕР

БОМБОРА
ИЗДАТЕЛЬСТВО


МИРОВОЙ
КОМПЬЮТЕРНЫЙ
БЕСТСЕЛЛЕР

Oliver Theobald

MACHINE LEARNING
FOR ABSOLUTE BEGINNERS

Оливер Теобальд

МАШИННОЕ ОБУЧЕНИЕ
ДЛЯ АБСОЛЮТНЫХ НОВИЧКОВ

ВВОДНЫЙ КУРС,
ИЗЛОЖЕННЫЙ ПРОСТЫМ ЯЗЫКОМ

БОМБОРА
ИЗДАТЕЛЬСТВО
Москва 2024

УДК 004.8
ББК 32.813
Т33

Oliver Theobald
MACHINE LEARNING FOR ABSOLUTE BEGINNERS

Теобальд, Оливер.

Т33 Машинное обучение для абсолютных новичков. Вводный курс, изложенный простым языком / Оливер Теобальд; [перевод с английского М. А. Райтмана]. — Москва : Эксмо, 2024. — 208 с. — (Мировой компьютерный бестселлер).

ISBN 978-5-04-190305-3

«Машинное обучение для абсолютных новичков» Оливера Теобальда — это идеальная книга для тех, кто хочет изучить основы машинного обучения (ML) без опыта программирования. Книга содержит основные алгоритмы ML, наглядные примеры, практические работы и обучение классической статистике. Руководство включает в себя материалы по загрузке бесплатных наборов данных, методы очистки и подготовки данных для анализа, основы работы нейронных сетей и многое другое.

УДК 004.8
ББК 32.813

ISBN 978-5-04-190305-3

ИЩИТЕ НАС НА СЛЕДУЮЩИХ РЕСУРСАХ:

Ежемесячный информационный бюллетень
http://eepurl.com/gKjQij
Получайте рекомендации книг, выигрывайте бесплатные экземпляры новых изданий от авторов и знакомьтесь со статьями и новостями, посвященными машинному обучению и науке о данных.

Teachable
http://scatterplotpress.teachable.com/
Здесь вы найдете вводные видеокурсы по машинному обучению, а также бонусные видеоуроки, сопровождающие эту книгу.

Skillshare
www.skillshare.com/user/machinelearning_beginners
На этом сайте представлены вводные видеокурсы по машинному обучению и видеоуроки от других инструкторов.

Instagram[1]
machinelearning_beginners
Здесь вы найдете краткие уроки, цитаты из книг и многое другое!

[1] Соцсеть признана экстремистской и запрещена на территории РФ.

ОГЛАВЛЕНИЕ

1

ПРЕДИСЛОВИЕ

Машины претерпели большие изменения с начала промышленной революции. Они по-прежнему заполняют цеха заводов и фабрик, однако их возможности уже выходят за рамки ручного труда и позволяют решать когнитивные задачи, которые до недавнего времени были под силу только человеку. Судейство песенных конкурсов, вождение автомобилей и выявление мошеннических операций — вот лишь три примера сложных задач, которые способны решать современные машины.

Однако эти замечательные достижения вселяют страх в некоторых людей. Отчасти он связан с опасениями выживальщиков и вечным вопросом: «А что, если?» Что, если разумные машины восстанут против нас в борьбе за существование? Что, если разумные машины произведут потомство, обладающее возможностями, которыми люди никогда не собирались их наделять? Что, если легенда о сингулярности окажется правдой?

Еще один существенный страх связан с угрозой потери рабочего места, и если вы работаете таксистом или бухгалтером, то у вас есть вполне веские основания для беспокойства. Согласно совместному исследованию Национальной статистической службы и компании Deloitte UK, опубликованному BBC в 2015 году, такие рабочие профессии, как работник бара (77%), официант (90%), дипломированный бухгалтер (95%), администратор (96%) и таксист (57%), имеют высокие шансы быть автоматизированными к 2035 году[2]. Однако к результатам исследований, касающихся планируемой автоматизации рабочих мест, и предсказаниям относительно развития машин и искусственного интеллекта (ИИ) следует относиться с некоторой долей скептицизма. В своей книге «Искусственный интеллект. Этапы. Угрозы. Стратегии» Ник Бостром рассуждает о постоянном пересмотре прогнозов относительно развития ИИ и о том, что «два десятилетия — это тот оптимальный срок, который является достаточно близким, чтобы привлекать

[2] "Will A Robot Take My Job?", *The BBC*, дата обращения: 30 декабря 2017 года, **http://www.bbc.com/news/technology-34066941**

внимание и оставаться актуальным, но достаточно далеким для того, чтобы допустить возможность ряда прорывов, которые могли бы к тому времени произойти»[3, 4].

Несмотря на то что ИИ развивается довольно быстро, его широкое внедрение по-прежнему остается неизведанным путем, на котором нам предстоит столкнуться с непредвиденными проблемами, задержками и другими препятствиями. Машинное обучение не сводится к простому щелчку выключателем, позволяющим приказать машине спрогнозировать результаты «Суперкубка» и подать вам вкусный мартини.

Машинное обучение не имеет ничего общего с готовыми аналитическими решениями. Оно опирается на статистические алгоритмы, которые контролируются такими квалифицированными специалистами, как дата-сайентисты и инженеры машинного обучения. Это один из быстро растущих рынков труда, на котором предложение пока не в состоянии удовлетворить спрос.

На самом деле нехватка профессионалов, обладающих необходимым опытом и уровнем подготовки, — одно из основных препятствий, задерживающих развитие сферы ИИ. По словам Чарльза Грина, директора по идейному лидерству компании Belatrix Software:

> «Во-первых, это огромная проблема — найти дата-сайентистов, *людей, обладающих* опытом работы в сфере машинного обучения или навыками анализа и использования данных, а также тех, кто способен создавать необходимые алгоритмы. Во-вторых, несмотря на то что технология все еще находится на стадии становления, существует множество текущих разработок. Очевидно, что сфера ИИ еще очень далека от того, как мы ее себе представляем»[5].

Возможно, с чтения этой книги начнется ваш путь к получению работы в области машинного обучения, а может быть, она просто даст вам о ней

---

[3] Ник Бостром, «Искусственный интеллект. Этапы. Угрозы. Стратегии». Издательство: Манн, Иванов и Фербер, 2016.

[4] Бостром также шутливо замечает, что два десятилетия примерно соответствуют оставшейся продолжительности карьеры типичного прогнозиста.

[5] Matt Kendall, "Machine Learning Adoption Thwarted by Lack of Skills and Understanding," Nearshore Americas, дата обращения: 14 мая 2017 года, **http://www.nearshoreamericas.com/machine-learning-adoption-understanding**

базовое представление, которого будет достаточно для удовлетворения вашего любопытства.

Эта книга представляет собой высокоуровневый обзор, включающий ключевые термины, общее описание рабочего процесса и статистические основы базовых алгоритмов, которые позволят вам начать свой путь. Для разработки и программирования интеллектуальных машин вам в первую очередь необходимо хорошо усвоить классическую статистику, так как алгоритмы, реализованные на ее основе, — это сердце машинного обучения. Они представляют собой метафорические нейроны, отвечающие за искусственные когнитивные способности. Написание кода — это еще одна неотъемлемая часть машинного обучения, которая предусматривает управление и манипулирование большими объемами данных. В отличие от создания целевых веб-страниц с помощью таких конструкторов, как Wix и WordPress, машинное обучение требует использования Python, C++, R или другого языка программирования. Если вы еще не изучили соответствующий язык, вам придется это сделать, чтобы развиваться в этой области. Однако материал, который вы здесь найдете, можно освоить даже без опыта программирования.

Хотя эта книга служит вводным курсом по машинному обучению, ознакомление читателей с основами математики, компьютерного программирования и статистики — не ее цель. Для облегчения понимания материала следующих глав вам могут потребоваться базовые знания в этих областях или доступ к Интернету.

Для тех, кто хочет погрузиться в аспект машинного обучения, связанный с программированием, в главах 17 и 18 представлен весь процесс создания модели машинного обучения с помощью языка Python. Небольшое введение в программирование на языке Python вы найдете в приложении, а информацию о дополнительных обучающих ресурсах — в заключительном разделе книги.

Наконец, видеоуроки и другие онлайн-материалы, сопровождающие эту книгу, вы можете найти на сайте:

https://scatterplotpress.teachable.com/p/ml-code-exercises.

2

ЧТО ТАКОЕ МАШИННОЕ ОБУЧЕНИЕ?

В 1959 году компания IBM опубликовала в журнале *IBM Journal of Research and Development* статью с интригующим и туманным названием. Ее автор, сотрудник IBM Артур Самуэль, исследовал вопрос о применении машинного обучения в контексте игры в шашки «для проверки того факта, что компьютер можно запрограммировать таким образом, чтобы он научился играть в шашки лучше, чем человек, написавший программу»[6].

Рис. 1. История упоминания термина «машинное обучение» в опубликованных книгах. Источник: *Google Ngram Viewer*, 2017.

Несмотря на то что это была не первая опубликованная работа, в которой использовался термин «машинное обучение», Артур Самуэль считается тем человеком, который ввел понятие и дал определение машинному обучению как концепции и специализированной области, известной нам сегодня (рис. 1). В знаковой статье Самуэля под названием *Some Studies in Machine Learning Using the Game of Checkers* («Некоторые исследования в области машинного обучения на примере игры в шашки») машинное обучение

[6] Arthur Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, Vol. 3, Issue. 3, 1959.

было представлено в качестве области информатики, которая изучает возможность компьютеров обучаться без их явного программирования.

Хотя в первоначальном определении Артура Самуэля это понятие не было рассмотрено, ключевая характеристика машинного обучения — это концепция самообучения. Она подразумевает применение статистического моделирования для выявления закономерностей и повышения производительности на основе данных и эмпирической информации без использования прямых команд. Артур Самуэль назвал это способностью обучаться без явного программирования. Он не имел в виду, что машины могут формулировать решения без какого-либо предварительного программирования. Напротив, машинное обучение в значительной степени зависит от вводимого кода. Но он заметил, что машины способны выполнять поставленную задачу, используя входные данные, вместо того чтобы полагаться на входную команду (рис. 2).

Рис. 2. Сравнение входной команды и входных данных

В качестве примера использования входной команды можно рассмотреть ввод выражения «2+2» на языке программирования Python и нажатие кнопки запуска или клавиши ввода для просмотра результата.

```
>>> 2+2
4
>>>
```

Это прямая команда с заранее запрограммированным ответом, характерная для большинства компьютерных приложений. Однако в отличие

от традиционного компьютерного программирования, при котором результаты или решения заранее определяются программистом, машинное обучение предполагает использование входных данных для построения модели принятия решений. Эти решения генерируются путем выявления существующих в данных взаимосвязей и закономерностей с помощью вероятностных рассуждений, проб и ошибок и применения других вычислительно-интенсивных методов. Это означает, что выход модели принятия решений определяется содержанием входных данных, а не правилами, заранее определенными программистом. При этом программист по-прежнему отвечает за подачу данных на вход модели, выбор подходящего алгоритма и настройку его параметров (называемых гиперпараметрами) для уменьшения ошибки прогнозирования. Однако в отличие от традиционного программирования в данном случае машина и разработчик работают на разных уровнях.

В качестве примера предположим, что в результате анализа привычек пользователей YouTube[7] модель принятия решений выявляет значимую взаимосвязь, говорящую о том, что специалисты в области науки о данных любят смотреть видео с котиками. В то же время другая модель выявляет закономерности между физическими показателями бейсболистов и вероятностью получения ими награды «Самый ценный игрок» (MVP) в текущем сезоне.

В первом сценарии машина анализирует предпочтения пользователей YouTube*, опираясь на такие показатели их вовлеченности, как лайки, подписки и повторные просмотры. Во втором сценарии машина оценивает физические показатели бейсболистов, ранее получивших награду MVP, наряду с другими их характеристиками, такими как возраст и уровень образования. Однако ни на одном из этапов модель принятия решений не получает каких-либо указаний, заставляющих ее выдать именно эти два результата. Модель использует методы машинного обучения для выявления сложных закономерностей, существующих во входных данных, без помощи со стороны человека. Это также означает, что при использовании аналогичного набора данных, собранного за другой период времени и включающего другое количество точек данных, модель может выдать несколько иной результат.

Еще одна отличительная особенность машинного обучения — способность модели улучшать качество прогнозов на основе опыта. Подобно тому

[7] Соцсеть признана экстремистской и запрещена на территории РФ.

как люди принимают решения, учитывая успех или неудачу предыдущих попыток, машинное обучение предполагает улучшение процесса принятия решений за счет контактов модели с данными. Взаимодействие с точками данных позволяет модели получить опыт и ознакомиться с существующими в данных закономерностями. И наоборот, недостаточное количество входных данных ограничивает способность модели деконструировать эти закономерности и реагировать на потенциальную дисперсию и случайность в реальных данных. Таким образом, контакт с входными данными позволяет модели лучше понять существующие в них закономерности, в том числе значимость происходящих в них изменений, что ведет к повышению ее эффективности.

Распространенный пример самообучающейся модели — система выявления спама в электронных сообщениях. После получения входных данных эта модель учится отмечать письма с подозрительными темами и текстом, содержащим ключевые слова, которые сильно коррелируют со спамом, отмеченным пользователями в прошлом. К признакам спама могут относиться такие слова, как: дорогой друг, бесплатно, счет, *PayPal*, виагра, казино, платеж, банкротство и победитель. Однако по мере анализа все большего количества данных модель может сталкиваться с исключениями и делать неверные предположения, ведущие к плохим прогнозам. Например, при ограниченном количестве данных для обоснования своего решения система может ошибочно классифицировать следующую тему письма как спам: «Сервис PayPal получил ваш платеж за фильм „Казино Рояль", купленный на eBay».

Несмотря на то что это настоящее письмо, отправленное автоответчиком PayPal, система обнаружения спама выдает ложноположительный результат на основе ранее изученных входных данных. Традиционное программирование особенно сильно подвержено этой проблеме, поскольку работа модели жестко определена заранее установленными правилами. С другой стороны, машинное обучение подчеркивает важность контакта с данными для уточнения прогнозов модели, корректировки слабых предположений и обеспечения ее адекватной реакции на уникальные точки данных, вроде тех, что были описаны в приведенном выше сценарии.

Хотя процесс самообучения модели зависит от данных, большее их количество не всегда приводит к лучшим решениям; входные данные должны быть релевантными. В своей книге *Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World* Брюс Шнайер пишет: «Когда вы

ищете иголку, последнее, что вам следует делать, это наваливать на нее как можно больше сена»[8]. Это означает, что увеличение количества нерелевантных данных может помешать в достижении желаемого результата. Кроме того, количество входных данных должно быть совместимо с имеющимися вычислительными и временными ресурсами.

Тренировочные и тестовые данные

В контексте машинного обучения входные данные обычно делятся на *тренировочные* и тестовые. *Тренировочные* данные используются для разработки модели. В примере с системой обнаружения спама такие данные могут использоваться для выявления ложноположительных результатов, вроде ошибочной классификации сообщения от автоответчика PayPal. После этого в модель необходимо внести соответствующие изменения, например настроить спам-фильтр, исключив из него электронные сообщения, отправленные с адреса payments@paypal.com. Методы машинного обучения позволяют научить модель автоматически обнаруживать эти ошибки (путем анализа исторических примеров спам-сообщений и выявления закономерностей) без непосредственного вмешательства человека.

Когда вы разработаете модель на основе закономерностей, существующих в тренировочных данных, и будете удовлетворены точностью ее прогнозов, вы сможете протестировать модель на оставшихся тестовых данных. Если вас устроит производительность модели на тестовых данных, вы сможете использовать ее для фильтрации и классификации входящих электронных сообщений в реальных условиях. Подробнее о тренировочных и тестовых данных мы поговорим в главе 6.

Анатомия машинного обучения

В заключительном разделе этой главы мы рассмотрим место машинного обучения в более широком ландшафте науки о данных и информатики, включая его связь с родительскими областями и смежными дисциплинами. Это важно, поскольку в специализированной литературе и на курсах по машинному обучению вам будут встречаться родственные термины. Кроме того,

[8] Bruce Schneir, “Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World,” W. W. Norton & Company, First Edition, 2016.

смежные дисциплины бывает трудно различить, особенно если речь идет о машинном обучении и извлечении данных.

Начнем с высокоуровневого представления. Машинное обучение, добыча данных, искусственный интеллект и компьютерное программирование относятся к области информатики, которая охватывает все, что связано с разработкой и использованием компьютеров. Внутри этого всеобъемлющего пространства информатики находится более узкая область — наука о данных, которая охватывает методы и системы извлечения знаний из данных с помощью компьютеров.

Рис. 3. Родословная машинного обучения, представленная в виде ряда матрешек

Третья матрешка слева на рис. 3 обозначает область искусственного интеллекта или ИИ, которая изучает способность машин решать интеллектуальные и когнитивные задачи. Подобно тому, как промышленная революция породила машины, выполняющие физические действия, область ИИ способствует развитию машин, моделирующих когнитивные функции.

Будучи более специализированной областью по сравнению с информатикой и наукой о данных, ИИ охватывает множество подобластей, которые сегодня пользуются популярностью и широко освещаются в новостях. К ним относятся поиск и планирование, рассуждения и представление знаний, восприятие, обработка естественного языка (NLP) и, разумеется, машинное обучение.

Для студентов, интересующихся искусственным интеллектом, машинное обучение — это отличная отправная точка, поскольку обеспечивает более узкую и практическую направленность обучения (по сравнению

с ИИ). Алгоритмы, применяемые в сфере машинного обучения, могут использоваться и в других дисциплинах, включая восприятие и обработку естественного языка. Кроме того, для достижения определенного уровня мастерства в области машинного обучения вполне достаточно получения степени магистра, тогда как для достижения реального прогресса в области ИИ вам может понадобиться докторская степень (рис. 4).

Рис. 4. Визуальное представление взаимосвязи между областями, имеющими отношение к работе с данными

Как уже было сказано, машинное обучение пересекается с такой родственной дисциплиной, как добыча данных, которая занимается обнаружением и поиском закономерностей в больших массивах данных. Обе сферы опираются на инференциальные методы, то есть прогнозирование результатов на основе других результатов и вероятностных рассуждений, и используют схожий набор алгоритмов, включающий анализ главных компонент, регрессионный анализ, деревья решений и методы кластеризации. Однако эти области часто путают между собой, неверно описывают и даже неправильно используют. Например, говорят, что учебник *Data mining:*

Practical machine learning tools and techniques with Java («Добыча данных: Практические инструменты и методы машинного обучения на Java») изначально назывался *Practical machine learning* («Машинное обучение на практике»), но затем, по маркетинговым соображениям, к названию была добавлена фраза *Data mining* (добыча данных)[9].

Наконец, в силу междисциплинарной природы этих областей различные эксперты зачастую по-разному определяют понятия «добыча данных» и «машинное обучение», что усугубляет путаницу, вызванную реальным пересечением дисциплин. Но если машинное обучение предполагает постепенное самообучение и автоматическое обнаружение закономерностей на основе опыта, полученного в результате изучения входящей информации, то добыча данных представляет собой менее автономную технику извлечения скрытых знаний.

Подобно бурению скважины в случайно выбранном месте земной коры, процесс добычи данных не начинается с четкой гипотезы о том, что должно быть обнаружено в итоге, а предполагает поиск еще не известных взаимосвязей и поэтому хорошо подходит для изучения больших наборов данных, содержащих сложные закономерности. Как отмечают авторы книги *Data Mining: Concepts and Techniques*, развитию сферы добычи данных способствовали результаты, достигнутые в области сбора данных и управления их базами с начала 1980-х годов[10], а также острая необходимость в осмыслении все более объемных и сложных наборов данных[11].

В то время как область добычи данных сосредоточена на анализе входных переменных с целью прогнозирования нового результата, машинное обучение предполагает анализ как входных, так и выходных переменных. Сюда входят методы контролируемого обучения, которые позволяют сравнивать известные комбинации входных и выходных переменных для выявления закономерностей и прогнозирования, и методы обучения

---

[9] Remco Bouckaert, Eibe Frank, Mark Hall, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann & Ian Witten, "WEKA — Experiences with a Java Open-Source Project," *Journal of Machine Learning Research*, Edition 11, **https://www.cs.waikato.ac.nz/ml/publications/2010/bouckaert10a.pdf**

[10] Изначально добыча данных называлась «анализом баз данных» и «информационным поиском». В 1990-х годах эта дисциплина стала известна как «обнаружение знаний в базах данных» и «добыча данных».

[11] Jiawei Han, Micheline Kamber & Jian Pei, "Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)," *Morgan Kauffmann*, 3rd Edition, 2011.

с подкреплением, которые предполагают случайное тестирование огромного количества входных переменных для получения желаемого результата. Другая методика машинного обучения, называемая неконтролируемым обучением, позволяет генерировать прогнозы на основе анализа входных переменных при отсутствии известного целевого результата. Эта техника часто используется в сочетании с методами контролируемого обучения или в качестве подготовки к нему и называется полуконтролируемым обучением, и несмотря на некоторые пересечения с областью добычи данных, неконтролируемое обучение отклоняется от таких ее стандартных методов, как анализ ассоциаций и последовательностей (табл. 1).

Табл. 1. Сравнение техник с учетом методов использования входных и выходных данных/переменных

Техника	Входы известны	Выходы известны	Методология
Добыча данных	*V*		Анализ входов для получения неизвестного выхода.
Контролируемое обучение	*V*	*V*	Анализ комбинаций известных входов и выходов для прогнозирования будущих выходов на основе новых входных данных.
Неконтролируемое обучение	*V*		Анализ входных данных для получения выходных — используемые алгоритмы могут отличаться от алгоритмов добычи данных.
Обучение с подкреплением		*V*	Случайное тестирование большого количества входных переменных для получения желаемого результата.

Чтобы прояснить разницу между добычей данных и машинным обучением, рассмотрим пример с двумя командами археологов. Одна команда практически не имеет представления о месте раскопок и использует знания предметной области, чтобы оптимизировать свои инструменты, найти закономерности, расчистить площадку и обнаружить скрытые артефакты. Цель команды заключается в том, чтобы вручную раскопать участок, найти

ценные предметы, а затем упаковать свое оборудование и отправиться дальше. На следующий день археологи вылетают в другое экзотическое место, чтобы начать новый проект, никак не связанный с тем местом, которое они раскопали накануне.

Вторая команда тоже занимается раскопками в исторических местах, но придерживается другой методологии. Эти археологи воздерживаются от раскопок основного котлована в течение нескольких недель. За это время они исследуют другие близлежащие археологические объекты и изучают закономерности в устройстве каждого из них. Посещая эти места, они набираются опыта, тем самым улучшая свою способность интерпретировать закономерности и уменьшая ошибку прогнозирования. Когда приходит время начинать раскопки основного котлована, они используют свое понимание и опыт, полученный при изучении местности, для интерпретации целевого объекта и составления прогнозов.

Как вы могли догадаться, первая команда делает ставку на добычу данных, в то время как вторая полагается на машинное обучение. Хотя обе команды занимаются раскопками ради обнаружения ценных сведений, их цели и методология весьма различны. Команда, использующая машинное обучение, создает самообучающуюся систему, которая использует контакт с данными для улучшения своей прогностической способности. Тем временем команда по добыче данных концентрируется на раскопках целевого участка, используя более прямой и приближенный метод, который полагается скорее на человеческую интуицию, чем на самообучение.

В следующей главе мы более подробно поговорим о самообучении в контексте машинного обучения, а также о том, как входные и выходные переменные используются для составления прогнозов.

3

КАТЕГОРИИ МАШИННОГО ОБУЧЕНИЯ

Область машинного обучения предусматривает несколько сотен статистических алгоритмов, и выбор подходящего алгоритма(ов) для решения поставленной задачи всегда представляет проблему для специалистов в этой области. Однако, прежде чем рассматривать конкретные алгоритмы, важно разобраться с тремя основными категориями машинного обучения, которые различаются по способу обращения с входными и выходными переменными.

Контролируемое обучение

Контролируемое обучение (или обучение с учителем) имитирует человеческую способность выявлять закономерности в известных примерах и использовать эти знания для получения воспроизводимых результатов. Именно так компания Toyota создала свой первый прототип автомобиля. Вместо того чтобы строить догадки или разрабатывать уникальный процесс производства машин, владельцы Toyota создали свой первый прототип, разобрав автомобиль Chevrolet на своем семейном предприятии по производству ткацких станков. Изучив готовый автомобиль (выход) и разобрав его на отдельные компоненты (входы), инженеры Toyota раскрыли особенности процесса проектирования, которые американская компания Chevrolet держала в секрете.

В контексте машинного обучения этот процесс понимания известной комбинации входов и выходов реализуется с помощью так называемого контролируемого обучения. Модель анализирует и расшифровывает взаимосвязи между входными и выходными данными, чтобы выявить основополагающие закономерности. Входные данные называются независимой переменной (и обозначаются прописной буквой «X»), а выходные данные — зависимой переменной (и обозначаются строчной буквой «y»). Примерами зависимой переменной (y) могут быть координаты прямоугольной рамки,

окружающей человека на цифровой фотографии (в системе распознавания лиц), цена дома или класс товара (например: спортивный автомобиль, семейный автомобиль, седан). Соответствующими независимыми переменными, которые предположительно влияют на зависимые, могут быть цвета пикселей, размер и расположение дома и технические характеристики автомобиля. В результате анализа достаточного количества примеров машина создает модель — алгоритмическое уравнение для получения выходного сигнала на основе закономерностей, выявленных при изучении примеров комбинаций входов и выходов.

Используя эту модель, машина может предсказать выходной результат, основываясь исключительно на входных данных. Например, рыночная цена вашего подержанного автомобиля Lexus может быть определена на основе размеченных примеров других машин, недавно проданных через сайт подержанных автомобилей (табл. 2).

Табл. 2. Фрагмент набора данных о подержанных автомобилях

	Вход	**Вход**	**Вход**	**Выход**
	Марка	**Пробег (км)**	**Год выпуска**	**Цена (в долларах)**
Автомобиль 1	Lexus	51715	2012	15985
Автомобиль 2	Lexus	7980	2013	19600
Автомобиль 3	Lexus	82497	2012	14095
Автомобиль 4	Lexus	85199	2011	12490
Автомобиль 5	Audi	62948	2008	13985

Имея доступ к данным о цене продажи других аналогичных автомобилей, модель контролируемого обучения может работать в обратном направлении, чтобы установить взаимосвязь между стоимостью автомобиля (выход) и его характеристиками (вход). После этого на вход данной модели можно будет подать характеристики вашего автомобиля для получения прогнозируемой цены.

В то время как входные данные с неизвестным выходом могут быть поданы в модель для получения прогноза, немаркированные данные не могут использоваться для построения самой модели. При построении модели контролируемого обучения каждый элемент (например: автомобиль,

продукт, клиент) должен предусматривать входные и выходные значения, снабженные метками. В области науки о данных это называется «размеченным набором данных» (рис. 5).

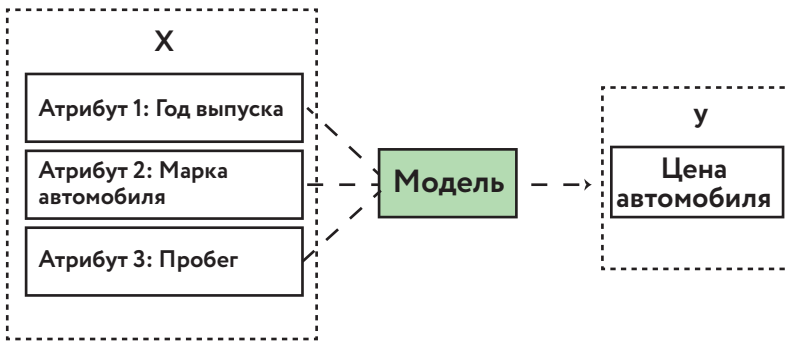


Рис. 5. Входные данные (X) подаются в модель для получения нового прогноза (y)

Примеры распространенных алгоритмов контролируемого обучения — регрессионный анализ (линейная регрессия, логистическая регрессия, нелинейная регрессия), деревья решений, метод *k*-ближайших соседей, нейронные сети и машины опорных векторов, каждый из которых мы рассмотрим в следующих главах.

Неконтролируемое обучение

В случае неконтролируемого обучения (или обучения без учителя) выходные переменные являются немаркированными, а комбинации входных и выходных переменных — неизвестными. Неконтролируемое обучение предполагает анализ взаимосвязей между входными переменными и выявление скрытых закономерностей, которые можно извлечь для создания новых меток, имеющих отношение к возможным выходам.

Например, если вы сгруппируете точки данных исходя из покупательского поведения МСП (малых и средних предприятий) и крупных предприятий, то, скорее всего, получите два кластера таких точек. Это связано с тем, что предприятия разного размера, как правило, имеют различные потребности. Например, когда речь идет о приобретении инфраструктуры облачных вычислений, большинству малых и средних предприятий будет достаточно обычного облачного хостинга и сети доставки контента (CDN).

Однако крупные корпоративные клиенты, скорее всего, приобретут более широкий набор облачных продуктов и комплексные решения, включающие современные средства обеспечения безопасности и такие сетевые продукты, как WAF (Web Application Firewall, межсетевой экран уровня веб-приложений), частное выделенное соединение и VPC (Virtual Private Cloud, виртуальное частное облако). Анализируя покупательские привычки, модель неконтролируемого обучения способна определить эти две группы клиентов без использования специальных меток, классифицирующих конкретную компанию как малую/среднюю или крупную.

Преимущество неконтролируемого обучения заключается в том, что оно позволяет обнаружить в данных еще не известные закономерности (например, существование двух доминирующих типов клиентов) и обеспечивает плацдарм для проведения анализа после выявления новых групп. Неконтролируемое обучение оказывается особенно полезным в сфере обнаружения мошенничества, где наиболее опасные атаки — те, которые еще только предстоит классифицировать. Один из интересных примеров — компания DataVisor, которая построила свою бизнес-модель на базе неконтролируемого обучения. Основанная в 2013 году в Калифорнии, компания защищает клиентов от мошеннических действий в Интернете, к которым относится рассылка спама, публикация поддельных отзывов, установка фальшивых приложений и мошеннические транзакции. В то время как традиционные сервисы для защиты от мошенничества используют модели контролируемого обучения и движки исполнения правил, DataVisor опирается на неконтролируемое обучение для обнаружения еще не классифицированных категорий атак.

На сайте DataVisor говорится о том, что «для обнаружения атак существующие решения полагаются на человеческий опыт при создании правил или на размеченные тренировочные данные, используемые для настройки моделей. Это означает, что они не способны обнаруживать новые атаки, которые еще не были идентифицированы людьми или снабжены специальными метками в наборе тренировочных данных»[12]. Иначе говоря, традиционные решения анализируют цепочки действий, характерные для определенного типа атак, а затем создают правила для прогнозирования и обнаружения соответствующих инцидентов. В данном случае зависимой

[12] "Unsupervised Machine Learning Engine," DataVisor, дата обращения: 19 мая 2017 г., **https://www.datavisor.com/unsupervised-machine-learning-engine**

переменной (выходом) становится сама атака, а независимыми переменными (входами) — распространенные предикторы атаки. Примерами независимых переменных могут быть:

1. **Внезапный крупный заказ от неизвестного пользователя.** Например, постоянные клиенты обычно тратят на один заказ менее 100 долларов, а новый пользователь тратит 8000 долларов сразу после регистрации аккаунта.
2. **Внезапный рост количества пользовательских рецензий.** Как и в случае с большинством книг, посвященных технологиям, продаваемых на сайте Amazon.com, на первое издание этой книги редко публикуется больше одного отзыва в день. В целом примерно 1 из 200 читателей оставляет отзыв на сайте, а большинство книг остаются без отзывов на протяжении нескольких недель или месяцев. Однако я заметил, что на некоторые книги в категории «наука о данных» публикуется по 50–100 рецензий за один день! (Неудивительно, что спустя некоторое время Amazon удаляет эти подозрительные отзывы.)
3. **Одинаковые или похожие отзывы от разных пользователей.** Также на сайте Amazon я иногда замечаю, что положительные отзывы читателей о моей книге появляются под другими книгами (даже с упоминанием моего имени как автора!). Опять же, Amazon в конечном итоге удаляет эти поддельные отзывы и блокирует соответствующие аккаунты за нарушение условий пользования сайтом.
4. **Подозрительный адрес доставки.** Например, для малых предприятий, которые регулярно отправляют продукцию местным клиентам, заказ из отдаленного места (в котором их продукция не рекламируется) в редких случаях может быть признаком мошеннической или злонамеренной деятельности.

В отдельности такие действия, как внезапный крупный заказ или удаленный адрес доставки, не всегда предоставляют достаточно информации для обнаружения сложных киберпреступлений и, скорее всего, приведут к получению серии ложноположительных результатов. Однако модель, отслеживающая комбинации независимых переменных, таких как крупный заказ, сделанный с другого конца света, или резкий рост количества книжных обзоров, в которых используется один и тот же пользовательский контент, обычно выдает более качественные прогнозы.

При контролируемом обучении модель деконструирует и классифицирует эти общие переменные и разрабатывает систему обнаружения для выявления и предотвращения повторных преступлений. Однако изощренные киберпреступники учатся обходить эти простые, основанные на классификации движки исполнения правил, изменяя свою тактику. Например, за некоторое время до осуществления атаки злоумышленники могут зарегистрировать одну или несколько учетных записей и использовать их, имитируя действия настоящих пользователей. История активности позволяет им обходить системы обнаружения, которые уделяют наибольшее внимание именно новым пользователям. В результате решения, опирающиеся на контролируемое обучение, зачастую оказываются неспособными обнаружить «спящие ячейки» до нанесения ими ущерба, особенно когда речь идет о новых типах атак.

Для устранения этих ограничений компания DataVisor и другие поставщики решений для борьбы с мошенничеством используют методы контролируемого обучения. Они анализируют закономерности в активности сотен миллионов учетных записей и выявляют подозрительные связи между пользователями (входные данные), ничего не зная при этом о фактической категории будущих атак (выходные данные). Группируя и выявляя злоумышленников, чьи действия отличаются от стандартного поведения пользователей, компании могут принимать меры по предотвращению новых типов атак (результаты которых еще не известны и не снабжены соответствующими метками).

Примерами подозрительных действий могут быть четыре описанных выше сценария или новые случаи аномального поведения, например большое количество вновь зарегистрированных пользователей с одинаковой фотографией в профиле. Выявляя тонкие взаимосвязи между пользователями, компании, занимающиеся обнаружением мошенничества, такие как DataVisor, способны обнаруживать «спящие ячейки» на стадии их инкубации. Например, множество фальшивых пользователей Facebook могут добавляться друг к другу в друзья и лайкать одни и те же страницы, но не иметь связей с реальными пользователями. Поскольку данный тип мошеннического поведения основан на поддельных связях между аккаунтами, неконтролируемое обучение помогает обнаружить сообщников и разоблачить преступные группировки.

Однако недостаток использования неконтролируемого обучения в том, что, поскольку набор данных не размечен, отсутствие известных выходных

наблюдений не позволяет проверить работу модели, из-за чего ее прогнозы оказываются более субъективными по сравнению с прогнозами модели контролируемого обучения.

Далее в этой книге мы рассмотрим процесс неконтролируемого обучения на примере кластеризации методом *k-средних*. К другим формам неконтролируемого обучения относятся анализ социальных сетей и алгоритмы снижения размерности.

Полуконтролируемое обучение

Полуконтролируемое обучение (или обучение с частичным привлечением учителя) представляет собой гибрид контролируемого и неконтролируемого обучения, которое предполагает использование наборов, содержащих как размеченные, так и неразмеченные данные. Учитывая то, что основная мотивация в данном случае заключается в принципе «чем больше данных, тем лучше», цель полуконтролируемого обучения — это использование неразмеченных данных для повышения надежности предсказательной модели. Один из методов заключается в том, чтобы построить исходную модель на основе размеченных данных (контролируемое обучение), а затем использовать ее для маркировки оставшихся (неразмеченных) данных в наборе. Затем модель можно переобучить на более крупном наборе (с меньшим количеством немаркированных данных или вообще без них). Альтернативный подход заключается в том, чтобы итеративно переобучать модель, используя вновь размеченные данные, которые соответствуют заданному порогу достоверности, и добавляя их в тренировочный набор после достижения ими соответствия установленному пороговому значению. Однако нет никакой гарантии, что модель полуконтролируемого обучения превзойдет модель, обученную на меньшем количестве исходных размеченных данных.

Обучение с подкреплением

Обучение с подкреплением представляет собой третью и наиболее продвинутую категорию машинного обучения. В отличие от контролируемого и неконтролируемого обучения оно предполагает построение предсказательной модели на основе обратной связи, получаемой в результате случайных проб и ошибок, а также знаний, приобретенных в ходе предыдущих итераций.

Цель обучения с подкреплением — достижение конкретного выхода путем случайного перебора огромного количества возможных комбинаций входных данных и оценки их эффективности.

Чтобы лучше понять эту концепцию, можно провести аналогию с видеоиграми. По мере продвижения в виртуальном пространстве игры игрок оценивает полезность различных действий в разных условиях и изучает игровой мир. Полученные в ходе этого процесса знания влияют на последующее поведение игрока, и его результативность постепенно улучшается благодаря обучению и наработке опыта.

Аналогичным образом алгоритмы обучения с подкреплением обеспечивают непрерывную тренировку модели. Стандартная модель обучения с подкреплением предусматривает измеримые критерии, по которым оценивается ее эффективность. В случае с беспилотными автомобилями это, например, способность избежать аварии, а в случае с шахматами — способность избежать поражения.

Q-обучение

В качестве конкретного алгоритмического примера обучения с подкреплением можно привести так называемое Q-обучение. При использовании этого подхода мы начинаем с заданной среды состояний, обозначаемой буквой «S». В игре Pac-Man состояниями могут быть опасности, препятствия или пути в лабиринте, например: стена слева, призрак справа и таблетка силы вверху. Набор возможных действий, которые игрок может предпринять в ответ на эти состояния, обозначается буквой «A». В Pac-Man этот набор ограничен движениями влево, вправо, вверх и вниз, а также их многочисленными комбинациями. Третий важный символ «Q» представляет собой исходное значение модели, которое изначально равно «0».

В процессе исследования пространства игры Pac-Man происходят две основные вещи:

1. Значение Q уменьшается вследствие негативных событий, имеющих место после определенного состояния/действия.
2. Значение Q увеличивается вследствие позитивных событий, имеющих место после определенного состояния/действия.

В рамках Q-обучения машина учится выбирать для конкретного состояния такое действие, которое позволяет получить или сохранить наивысшее значение Q. Изначально она выполняет случайные движения (действия) в различных условиях (состояниях). Модель фиксирует полученные результаты (поощрения и наказания) и то, как они влияют на значение Q, а затем использует эти сохраненные значения для оптимизации своих дальнейших действий.

Хотя это звучит довольно просто, реализация требует больших вычислительных ресурсов, а ее описание выходит за рамки вводного курса по машинному обучению. Мы не будем рассматривать алгоритмы обучения с подкреплением, однако вы можете ознакомиться с подробным описанием обучения с подкреплением и Q-обучения на примере игры Pac-Man по адресу:

https://inst.eecs.berkeley.edu/~cs188/sp12/projects/reinforcement/reinforcement.html

4

ИНСТРУМЕНТЫ МАШИННОГО ОБУЧЕНИЯ

При освоении нового навыка бывает полезно представить себе набор инструментов и материалов, необходимых специалистам в этой предметной области. Например, если перед вами стоит задача собрать специальный инструментарий для создания веб-сайта, то в первую очередь вам нужно подумать о языках программирования. В этот набор можно включить языки для разработки фронтенда, такие как: HTML, CSS и JavaScript, один или два языка для разработки бэкенда и, разумеется, текстовый редактор. В дополнение к этому вы можете добавить в инструментарий конструктор сайтов вроде WordPress, веб-хостинг, DNS и, возможно, несколько приобретенных доменных имен.

Несмотря на то что этот перечень не является исчерпывающим, он дает представление о том, какие инструменты вам необходимо освоить на пути к становлению успешным веб-разработчиком.

Теперь давайте заглянем в базовый набор инструментов машинного обучения.

Отделение 1: Данные

В первом отделении ящика с инструментами хранятся исходные данные, необходимые для обучения модели и создания прогнозов. По форме данные могут быть разделены на структурированные и неструктурированные. Новичкам лучше всего начинать с анализа структурированных данных, то есть определенных, организованных и размеченных, как в табл. 3. Изображения, видеоролики, электронные письма и аудиозаписи — это примеры неструктурированных данных, поскольку они не организованы в виде строк и столбцов.

Табл. 3. Цена биткоина в период с 2015 по 2017 год

Дата	Цена биткоина	Количество прошедших дней
19–05–2015	234,31	1
14–01–2016	431,76	240
09–07–2016	652,14	417
15–01–2017	817,26	607
24–05–2017	2358,96	736

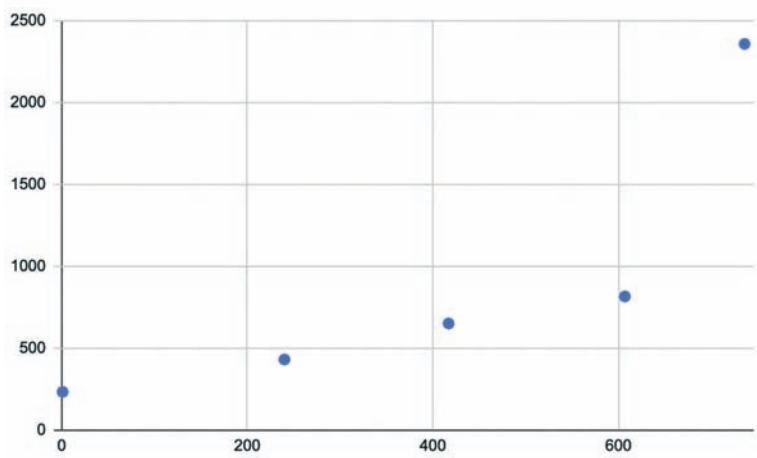
Прежде чем мы продолжим, я хочу ознакомить вас с анатомией табличного набора данных. В таком наборе данные организованы в строки и столбцы. В каждом столбце содержится признак, который также иногда называется переменной, *измерением* или атрибутом. Каждая строка представляет собой одно наблюдение, имеющее отношение к данному признаку/переменной. Иногда строку называют случаем или значением, но в этой книге мы будем использовать термин «строка» (рис. 6).

Рис. 6. Пример табличного набора данных

Рис. 7. Не всегда, но часто значение y содержится в крайнем правом векторе

Каждый столбец также называется вектором. В векторах содержатся значения *X* и y, а несколько векторов (столбцов) принято называть матрицами. В случае контролируемого обучения значения y уже содержатся в наборе данных и используются для выявления взаимосвязей с независимыми переменными (*X*). Как правило, значения y содержатся в последнем векторе, как показано на рис. 7.

Диаграммы рассеяния, включая двумерные, трехмерные и четырехмерные, находятся вместе с данными в первом отделении ящика с инструментами. Двумерная диаграмма рассеяния состоит из вертикальной оси (оси y) и горизонтальной (оси *x*) и представляет собой графический холст для нанесения комбинаций переменных, известных как точки данных (рис. 8). Каждая точка данных на диаграмме рассеяния соответствует наблюдению из набора данных, при этом значения *X* откладываются по оси *x* и значения y — по оси y.



	Независимая переменная (*X*)	Зависимая переменная (y)
Строка 1	**1**	**234,31**
Строка 2	**240**	**431,76**
Строка 3	**417**	**652,14**
Строка 4	**607**	**817,26**
Строка 5	**736**	**2358,96**

Рис. 8. Пример двумерной диаграммы рассеяния. Значение *X* — это количество прошедших дней, а значение y — цена биткоина.

Отделение 2: Инфраструктура

Во втором отделении инструментария находится инфраструктура машинного обучения, которая состоит из платформ и инструментов для обработки данных. Как новичок в этой области, вы, скорее всего, будете использовать веб-приложение (например, Jupyter Notebook) и такой язык программирования, как Python. Существует целый ряд совместимых с Python библиотек машинного обучения, в частности NumPy, Pandas и Scikit-learn. Эти библиотеки представляют собой наборы заранее скомпилированных программных процедур, часто используемых в машинном обучении, которые позволяют манипулировать данными и выполнять алгоритмы с использованием минимального количества кода.

Вам также потребуется машина для обработки данных в виде физического компьютера или виртуального сервера. Кроме того, могут понадобиться специализированные библиотеки для визуализации данных, такие как Seaborn и Matplotlib, или отдельная программа вроде Tableau, которая поддерживает ряд методов визуализации, включая построение диаграмм, графиков и карт.

Подготовив необходимую инфраструктуру, вы можете приступить к построению своей первой модели машинного обучения. Первым делом нужно включить компьютер. Стандартный настольный компьютер или ноутбук вполне подходят для работы с небольшими наборами данных, которые хранятся в центральном местоположении, например в CSV-файле. Затем вам необходимо установить среду разработки, например Jupyter Notebook, и язык программирования, которым для большинства начинающих служит Python.

Вот почему Python — наиболее широко используемый язык программирования в сфере машинного обучения.

1. Он прост в освоении и использовании.
2. Он совместим с целым рядом библиотек машинного обучения.
3. Его можно использовать для решения таких родственных задач, как сбор (веб-скрейпинг) и конвейеризация данных (Hadoop и Spark).

Для машинного обучения подходят и другие языки, например: C и C++. Если вы хорошо ими владеете, можете использовать именно их. Эти языки применяются по умолчанию для продвинутого машинного обучения, поскольку написанный на них код может работать непосредственно

на графическом процессоре (ГП). Код на Python необходимо преобразовать перед запуском на ГП, но об этом и о том, что собой представляет графический процессор, мы поговорим чуть позже.

Затем пользователям Python необходимо импортировать библиотеки NumPy, Pandas и Scikit-learn. NumPy — это бесплатная библиотека с открытым исходным кодом, которая позволяет эффективно загружать большие наборы данных и работать с ними, в том числе объединять их и выполнять операции с матрицами.

Библиотека Scikit-learn предоставляет доступ к ряду популярных алгоритмов неглубокого обучения, включая линейную регрессию, методы кластеризации, деревья решений и машины опорных векторов. Алгоритмы неглубокого обучения предсказывают результаты непосредственно на основе входных признаков. В отличие от них алгоритмы глубокого обучения производят вывод на основе данных, полученных от предшествующих слоев модели (о которых мы поговорим в главе 13 в контексте обсуждения искусственных нейронных сетей)[13].

```
In [6]: # Import library
        import pandas as pd

        # Read in data from CSV as a Pandas dataframe
        df = pd.read_csv('~/Downloads/Melbourne_housing_FULL.csv')

        df.head()
```

Out[6]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Cc
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	1900.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	NaN	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	134.0	150.0	1900.0	

5 rows × 21 columns

Рис. 9. Предварительный просмотр таблицы в Jupyter Notebook с помощью Pandas

Наконец, пакет Pandas позволяет представить данные в виде виртуальной электронной таблицы, которой можно манипулировать с помощью кода (рис. 9). Он напоминает программу Microsoft Excel в том смысле, что позволяет редактировать данные и выполнять вычисления. Название Pandas происходит от термина «panel data» («панельные данные»), что говорит

[13] За исключением искусственных нейронных сетей большинство алгоритмов обучения можно назвать неглубокими.

о способности этой библиотеки создавать серию панелей, напоминающих «листы» в программе Excel. Пакет Pandas также идеально подходит для импорта и извлечения данных из CSV-файлов.

Студенты, ищущие альтернативные варианты инструментов для программирования, помимо Python, C и C++, могут присмотреться к R, MATLAB и Octave.

R — это свободный язык программирования с открытым исходным кодом, оптимизированный для выполнения математических операций, построения матриц и использования статистических функций. Хотя R чаще всего используется для добычи данных, он также может применяться для машинного обучения.

MATLAB — это коммерческий проприетарный язык программирования, который хорошо справляется с решением алгебраических уравнений. Он относительно прост для освоения. MATLAB широко используется в таких областях, как электротехника, гражданское строительство, химическая и авиационная инженерия. Однако ученые-компьютерщики и компьютерные инженеры, как правило, не используют MATLAB, особенно в последние годы, хотя этот язык по-прежнему широко применяется для машинного обучения в академических кругах. Поэтому, увидев MATLAB в программе онлайн-курсов по машинному обучению, особенно на Coursera, имейте в виду, что это вовсе не говорит о его широком применении в этой области. Однако, если вы являетесь выходцем из инженерной среды, то использование MATLAB — вполне логичный выбор.

Наконец, язык Octave представляет собой бесплатную версию MATLAB, созданную сообществом разработчиков ПО с открытым исходным кодом.

Отделение 3: Алгоритмы

После настройки среды разработки и выбора языка программирования и необходимых библиотек вы можете импортировать данные непосредственно из CSV-файла. На сайте **kaggle.com** представлены сотни интересных наборов данных в формате CSV. Чтобы загрузить выбранный набор, вам нужно зарегистрироваться в качестве участника сообщества Kaggle. И регистрация, и сами наборы данных Kaggle абсолютно бесплатны. Выбранный набор данных будет загружен непосредственно на ваш компьютер в виде CSV-файла, а значит, вы сможете использовать программу Microsoft Excel для его открытия и даже применения таких базовых алгоритмов, как линейная регрессия.

В третьем и последнем отделении вашего инструментария находятся алгоритмы машинного обучения. Новички, как правило, начинают с освоения таких простых алгоритмов контролируемого обучения, как линейная регрессия, логистическая регрессия, деревья решений и метод *k*-ближайших соседей, и таких алгоритмов неконтролируемого обучения, как кластеризация методом *k*-средних и алгоритмы снижения размерности.

Визуализация

Какими бы впечатляющими и глубокими ни были открытия, сделанные вами в ходе анализа данных, необходимо донести их до лиц, принимающих решения. Именно здесь вам пригодятся инструменты визуализации данных, позволяющие представить полученные выводы широкой аудитории. Вы можете быстро и легко рассказать о них с помощью визуальной истории, передаваемой через графики, тепловые карты, диаграммы рассеяния и размаха, а также путем представления числовой информации в виде геометрических фигур.

Как правило, чем менее информирована ваша аудитория, тем важнее визуализировать полученные выводы. И наоборот, если ваша аудитория хорошо разбирается в теме, вы можете смело сопроводить визуальные элементы дополнительными деталями и техническими терминами. Для визуализации полученных результатов можно воспользоваться программой вроде Tableau или такой библиотекой Python, как Seaborn, которые хранятся во втором отделении вашего инструментария.

Расширенный набор инструментов

До сих пор мы рассматривали набор инструментов для новичка, а как насчет более опытного пользователя? Как выглядит его набор инструментов? Хотя до того, как вы начнете работать с более продвинутыми инструментами, может пройти некоторое время, вам не помешает взглянуть на них уже сейчас.

Инструментарий продвинутого пользователя включает в себя более широкий спектр инструментов и, конечно же, данных. Одно из самых больших отличий между новичком и экспертом заключается в типе данных, с которыми они имеют дело. Новички работают с небольшими наборами данных, которые легко обрабатывать и загружать прямо на рабочий стол

в виде простого CSV-файла. А вот продвинутые пользователи чаще работают с массивными наборами данных, которые могут храниться в нескольких местах и быть потоковыми (то есть импортироваться и анализироваться в режиме реального времени), а не статичными, что превращает сами данные в движущуюся цель.

Отделение 1: Большие данные

Под большими данными понимаются наборы данных, которые в силу своего многообразия, объема и скорости передачи не могут быть обработаны традиционными методами и без привлечения передовых технологий. У больших данных нет точного определения в плане размера или минимального количества строк и столбцов. В настоящее время большими данными считаются наборы, размер которых измеряется в петабайтах, однако эти массивы становятся все больше по мере нахождения новых и относительно недорогих способов сбора и хранения данных.

Кроме того, такие наборы реже имеют стандартную табличную форму и могут содержать данные разных типов, в том числе структурированные и неструктурированные, то есть изображения, видеоролики, электронные письма и аудиофайлы.

Отделение 2: Инфраструктура

Для работы с петабайтами данных продвинутым специалистам требуется надежная инфраструктура. Вместо того чтобы полагаться на центральный процессор (ЦП) персонального компьютера, эксперты обычно обращаются к распределенным вычислениям и таким облачным провайдерам, как Amazon Web Services (AWS) или Google Cloud Platform, чтобы выполнять обработку данных на виртуальном графическом процессоре (ГП). Будучи специализированными чипами для параллельных вычислений, экземпляры ГП способны выполнять гораздо больше операций над числами с плавающей запятой в секунду, чем ЦП, что позволяет значительно быстрее решать задачи линейной алгебры и статистики.

Изначально чипы ГП встраивались в материнские платы ПК и такие видеоприставки, как PlayStation 2 и Xbox, для игровых целей. Они были разработаны для ускорения рендеринга изображений, состоящих из миллионов пикселей, которые должны были перерисовываться на экране

много раз в секунду. К 2005 году из-за большого объема производства цены на графические процессоры резко снизились, и они практически превратились в товар широкого потребления. Несмотря на популярность этих чипов в индустрии видеоигр, потенциал их применения в области машинного обучения был осознан лишь недавно. В своем романе «Неизбежно. 12 технологических трендов, которые определяют наше будущее» Кевин Келли рассказывает о том, что в 2009 году Эндрю Ын и команда из Стэнфордского университета совершили открытие, позволяющее объединить недорогие ГП в кластеры для запуска нейронных сетей, состоящих из сотен миллионов взаимосвязанных узлов.

По словам Келли, «традиционным процессорам требовалось несколько недель для вычисления всех каскадных возможностей в нейронной сети со ста миллионами параметров. Ын обнаружил, что кластер графических процессоров способен сделать то же самое всего за сутки»[14].

Как уже говорилось, C и C++ — предпочтительные языки для непосредственного редактирования и выполнения математических операций на графическом процессоре. Также можно использовать код Python, преобразуя его в C, в сочетании с библиотекой машинного обучения, такой как TensorFlow от компании Google. Хотя TensorFlow можно запускать и на ЦП, использование ГП обеспечивает примерно 1000-кратный прирост производительности. К огорчению пользователей компьютеров Mac, библиотека TensorFlow совместима только с видеокартой Nvidia, которая больше не поддерживается в Mac OS X. Пользователи Mac по-прежнему могут запускать TensorFlow на своих ЦП, но для использования ГП им придется перенести рабочую нагрузку в облако.

Amazon Web Services, Microsoft Azure, Alibaba Cloud, Google Cloud Platform и другие облачные провайдеры позволяют использовать ресурсы ГП с оплатой по факту потребления, а иногда и бесплатно в рамках пробного периода. В настоящее время по соотношению производительности и цены среди сервисов, предлагающих виртуальные ГП, лидирует Google Cloud Platform. В 2016 году компания Google также объявила о намерении предоставить доступ к своим тензорным процессорам, разработанным специально для работы с TensorFlow, которые уже используются внутри компании.

[14] Кевин Келли, «Неизбежно. 12 технологических трендов, которые определяют наше будущее». Издательство: Манн, Иванов и Фербер, 2021.

Отделение 3: Продвинутые алгоритмы

В завершение этой главы мы заглянем в третье отделение расширенного инструментария, содержащее продвинутые алгоритмы машинного обучения. Для анализа больших наборов данных и решения сложных задач прогнозирования специалисты используют множество различных алгоритмов, включая марковские модели, машины опорных векторов и Q-обучение, а также их комбинации для создания единой модели в рамках так называемого ансамблевого моделирования (о котором мы поговорим в главе 15). Однако чаще всего они работают с искусственными нейронными сетями (описанными в главе 13), которые предусматривают собственный набор продвинутых библиотек машинного обучения.

В то время как Scikit-learn предлагает ряд популярных алгоритмов неглубокого обучения, библиотека TensorFlow предназначена для глубокого обучения и работы с нейронными сетями. Она поддерживает множество передовых методов, включая автоматическое дифференцирование для осуществления обратного распространения ошибки/градиентного спуска. Богатство ресурсов, подробная документации и широкий спектр задач, решаемых с помощью TensorFlow, также делают этот фреймворк очевидным выбором для изучения. Другие популярные библиотеки для работы с нейронными сетями — Torch, Caffe и быстро развивающаяся Keras.

Написанная на языке Python, Keras представляет собой библиотеку глубокого обучения с открытым исходным кодом, которая работает поверх TensorFlow, Theano и других фреймворков, что позволяет пользователям быстро проводить эксперименты, используя меньшее количество строк кода. Как и темам для сайта WordPress, Keras свойственны минимализм, модульность и простота использования. Однако она менее гибкая по сравнению с TensorFlow и другими библиотеками. Поэтому разработчики иногда используют Keras для проверки модели принятия решений, прежде чем переключиться на TensorFlow для построения более настраиваемой модели.

Библиотека Caffe также имеет открытый исходный код и обычно используется для разработки архитектур глубокого обучения, предназначенных для классификации и сегментации изображений. Эта библиотека написана на языке C++, но предусматривает интерфейс Python, который поддерживает ГП-ускорение с использованием чипа Nvidia cuDNN.

Выпущенная в 2002 году библиотека Torch так же хорошо зарекомендовала себя в сфере глубокого обучения и уже используется в Facebook,

Google, Twitter, Нью-Йоркском университете, институте IDIAP, Университете Пердью и множестве других компаний и исследовательских лабораторий[15]. Основанная на языке программирования Lua, библиотека Torch имеет открытый исходный код и предусматривает ряд алгоритмов и функций, используемых для глубокого обучения.

До недавнего времени еще одним конкурентом TensorFlow была библиотека Theano, но в конце 2017 года ее разработка была официально прекращена[16].

[15] "What is Torch?" *Torch*, дата обращения: 20 апреля 2017 г., **http://torch.ch**

[16] Pascal Lamblin, "MILA and the future of Theano," *Google Groups Theano Users Forum*, **https://groups.google.com/forum/#!topic/theano-users/7Poq8BZutbY**

5

ОЧИСТКА ДАННЫХ

Как и большинство сортов фруктов, наборы данных нуждаются в предварительной очистке и других манипуляциях, позволяющих подготовить их к употреблению. В сфере машинного обучения и многих других смежных областях технический процесс доработки набора данных, направленный на то, чтобы сделать его более пригодным для использования, известен как очистка данных. Она может предусматривать изменение и удаление неполных, неправильно отформатированных, неактуальных или дублирующихся данных, а также преобразование текстовых данных в числовые значения и переработку признаков.

От специалистов по работе с данными этот процесс обычно требует наибольших затрат времени и сил.

Отбор признаков

Чтобы получить наилучшие результаты при использовании данных, важно определить, какие переменные наиболее релевантны для вашей гипотезы или цели. На практике это означает, что нужно проявить избирательность в отношении переменных, которые вы включаете в свою модель. Более того, сохранение признаков, которые не сильно коррелируют с выходным значением, может отрицательно сказаться на точности модели. Рассмотрим фрагмент набора данных об умирающих языках, загруженный с сайта **kaggle.com**.

Допустим, наша цель заключается в определении переменных, которые ставят язык под угрозу исчезновения (табл. 4). Учитывая это, значения в столбце *Name in Spanish* («Название языка на испанском»), скорее всего, окажутся для нас бесполезными, поэтому мы можем смело удалить этот вектор (столбец) из набора данных. Это поможет предотвратить чрезмерное усложнение и снижение точности модели, а также повысить общую скорость ее работы.

Табл. 4. Языки, находящиеся под угрозой исчезновения, база данных: **https://www.kaggle.com/the-guardian/extinct-languages**

Name in English	Name in Spanish	Countries	Country Code	Num. of Speakers
South Italian	Napolitano -calabres	Italy	ITA	7500000
Sicilian	Siciliano	Italy	ITA	5000000
Low Saxon	Bajo Sajón	Germany, Denmark, Netherlands, Poland, Russian Federation	DEU, DNK, NLD, POL, RUS	4800000
Belarusian	Bielorruso	Belarus, Latvia, Lithuania, Poland, Russian Federation, Ukraine	BRB, LVA, LTU, POL, RUS, UKR	4000000
Lombard	Lombardo	Italy, Switzerland	ITA, CHE	3500000
Romani	Romaní	Albania, Germany, Austria, Belarus, Bosnia and Herzegovina, Bulgaria, Croatia, Estonia, Finland, France, Greece, Hungary, Italy, Latvia, Lithuania, The former Yugoslav Republic of Macedonia, Netherlands, Poland, Romania, United Kingdom of Great Britain and Northern Ireland, Russian Federation, Slovakia, Slovenia, Switzerland, Czech Republic, Turkey, Ukraine, Serbia, Montenegro	ALB, DEU, AUT, BRB, BIH, BGR, HRV, EST, FIN, FRA, GRC, HUN, ITA, LVA, LTU, MKD, NLD, POL, ROU, GBR, RUS, SVK, SVN, CHE, CZE, TUR, UKR, SRB, MNE	3500000
Yiddish	Yiddish	Israel	ISR	3000000
Gondi	Gondi	India	IND	2713790

Кроме того, в векторах *Countries* («Страны») и *Country Code* («Код страны») этого набора данных содержится дублирующаяся информация. Анализ обоих векторов не позволит нам получить дополнительных полезных сведений, так что мы можем удалить один из них.

Еще один метод уменьшения количества признаков — это объединение нескольких столбцов в один, как показано в следующем примере.

Табл. 5. Пример таблицы с результатами инвентаризации товаров

	Протеиновый коктейль	Кроссовки Nike	Кроссовки Adidas	Браслет Fitbit	Напиток Powerade	Протеиновый батончик	Смарт-часы	Витамины
Покуп. 1	1	1	0	1	0	5	1	0
Покуп. 2	0	0	0	0	0	0	0	1
Покуп. 3	3	0	1	0	5	0	0	0
Покуп. 4	1	1	0	0	10	1	0	0

В табл. 5 содержится список товаров, продаваемых на платформе электронной коммерции. Этот набор данных включает сведения о четырех покупателях и восьми продуктах. Выборка невелика, поскольку бóльшего нам не позволяет

формат книги. Базы данных реальных платформ электронной коммерции содержат гораздо больше столбцов, но мы рассмотрим упрощенный пример.

Для более эффективного анализа данных мы можем сократить количество столбцов, объединив схожие признаки. Например, можно отбросить названия отдельных товаров, заменив их меньшим количеством категорий или подтипов. Поскольку все товары относятся к категории «фитнес», мы можем сгруппировать их по подтипам и сократить количество столбцов с восьми до трех: «Здоровое питание», «Спортивная одежда» и «Цифровые устройства» (табл. 6).

Табл. 6. Результаты инвентаризации синтезированных товаров

	Здоровое питание	Спортивная одежда	Цифровые устройства
Покупатель 1	6	1	2
Покупатель 2	1	0	0
Покупатель 3	8	1	0
Покупатель 4	12	1	0

Такой способ преобразования набора данных позволяет нам сохранить и передать информацию, используя меньшее количество переменных. Недостаток такого преобразования — потеря части сведений о взаимосвязях между конкретными товарами. В результате получаемые пользователями рекомендации будут основаны не на данных об отдельных товарах, а на ассоциациях между их подтипами или на рекомендациях товаров одного и того же подтипа.

Тем не менее такой подход обеспечивает довольно высокий уровень релевантности данных. В зависимости от степени корреляции покупателям будут рекомендованы продукты здорового питания или спортивной одежды при покупке других подобных товаров, а не учебники по машинному обучению, если, конечно, не окажется, что между этими категориями существует сильная корреляция! Но, увы, в представленном выше наборе данных такая переменная/категория отсутствует.

Помните, что сокращение объема данных представляет собой бизнес-решение, так что владельцам бизнеса следует проконсультироваться с командой специалистов по анализу данных, чтобы оценить компромисс между удобством и точностью модели.

Сжатие строк

Помимо отбора признаков вы можете уменьшить количество строк и тем самым сократить общее число точек данных. Для этого можно объединить две или более строк в одну, как показано в следующем наборе данных, где «Тигр» и «Лев» были объединены в категорию «Хищник» (табл. 7).

Табл. 7. Пример объединения строк

До

Животное	Плотоядное	Количество лап	Наличие хвоста	Время преодоления дистанции, мин.
Тигр	Да	4	Да	2:01
Лев	Да	4	Да	2:05
Черепаха	Нет	4	Нет	55:00

После

Животное	Плотоядное	Количество лап	Наличие хвоста	Время преодоления дистанции, мин.
Хищник	Да	4	Да	2:03
Черепаха	Нет	4	Нет	55:00

При объединении двух строк («Тигр» и «Лев») соответствующие значения признаков тоже должны быть агрегированы и записаны в одной строке. В данном случае объединение двух строк возможно, поскольку они содержат одинаковые категориальные значения для всех признаков за исключением времени преодоления дистанции, которое легко агрегировать путем сложения соответствующих значений для тигра и льва и деления полученной суммы на два.

Числовые значения обычно легко агрегировать, если они не категориальные. Например, мы не можем объединить животное с четырьмя лапами и животное с двумя лапами, указав «три» в качестве агрегированного количества лап.

Сжатие строк также может оказаться проблематичным в тех случаях, когда числовые значения недоступны. Например, значения «Япония» и «Аргентина» довольно трудно объединить. С другой стороны, мы вполне можем объединить значения «Япония» и «Южная Корея», так как эти страны относятся

к одному континенту — «Азии» или «Восточной Азии». Однако, если мы добавим в эту же группу «Пакистан» и «Индонезию», то, скорее всего, получим искаженные результаты в силу существенных культурных, религиозных, экономических и прочих различий между этими четырьмя странами.

Короче говоря, содержащиеся в строках нечисловые и категориальные значения может быть сложно объединить с сохранением истинного значения исходных данных. Кроме того, сжатие строк обычно более проблематично, чем сжатие столбцов, особенно в случае с наборами данных, содержащими большое количество признаков.

Прямое кодирование

После того как вы решите, какие признаки и строки будут включены в модель, поищите текстовые значения, которые можно преобразовать в числа. Большинство алгоритмов не совместимы с нечисловыми данными за исключением таких текстовых значений, как Истина/Ложь (которые автоматически преобразуются в «1» и «0» соответственно).

Один из методов преобразования текстовых значений в числовые — прямое кодирование (one-hot encoding), также *известное как* унитарное кодирование или кодирование с одним активным или «горячим» состоянием. При использовании этого метода значения преобразуются в двоичную форму: «1» или «0» («Истина» или «Ложь»). «0» («Ложь») говорит о том, что значение не принадлежит к этому признаку, в то время как «1» («Истина» или «горячее состояние») подтверждает принадлежность к нему.

Ниже приведен еще один фрагмент набора данных об умирающих языках, на примере которого мы посмотрим, как работает прямое кодирование.

Прежде чем мы начнем, обратите внимание на то, что значения, содержащиеся в столбце *No. of Speakers* («Число носителей языка»), не содержат запятых или пробелов, то есть не записаны в виде 7,500,000 или 7 500 000. Хотя подобное форматирование облегчает людям восприятие больших чисел, языки программирования в нем не нуждаются. Форматирование чисел может привести к возникновению синтаксической ошибки или к другому нежелательному результату в зависимости от языка программирования, поэтому для целей программирования оставляйте числа неформатированными. Однако на этапе визуализации данных вы можете смело добавлять пробелы или запятые, чтобы вашей аудитории было легче воспринимать представленную информацию, особенно большие числа.

Табл. 8. Языки, находящиеся под угрозой исчезновения

Название на английском языке	Носители языка	Степень угрозы исчезновения
South Italian	75000000	уязвимый
Sicilian	5000 000	уязвимый
Low Saxon	4800 000	уязвимый
Belarusian	4000 000	уязвимый
Lombard	3500 000	определенно вымирающий
Romani	3500 000	определенно вымирающий
Yiddish	3000 000	определенно вымирающий
Gondi	2713 790	уязвимый
Picard	700000	находящийся под угрозой

В правой части приведенной выше табл. 8 находится вектор, в котором указана степень угрозы исчезновения для девяти языков. Мы можем преобразовать содержимое этого столбца в числовые значения методом прямого кодирования, как показано в следующей табл. 9.

Табл. 9. Пример выполнения прямого кодирования

Название на английском языке	Носители языка	Уязвимый	Определенно вымирающий	Находящийся под угрозой
South Italian	75000000	1	0	0
Sicilian	5000 000	1	0	0
Low Saxon	4800 000	1	0	0
Belarusian	4000 000	1	0	0
Lombard	3500 000	0	1	0
Romani	3500 000	0	1	0
Yiddish	3000 000	0	1	0
Gondi	2713 790	1	0	0
Picard	700000	0	0	1

В результате прямого кодирования набор данных расширился до пяти столбцов, поскольку из исходного признака *Degree of Endangerment* («Степень угрозы исчезновения») мы создали три новых. В каждом из новых столбцов мы указали значение «1» или «0» в зависимости от значения исходного признака. Теперь мы можем подать эти данные на вход нашей модели и использовать более широкий спектр алгоритмов машинного обучения. Увеличение количества признаков в наборе данных может несколько увеличить время обработки, что можно считать недостатком. Обычно это приемлемо, но может представлять проблему при работе с наборами данных, в которых исходные признаки разбиты на большое количество новых.

Один из способов минимизации общего количества признаков заключается в том, чтобы свести бинарные случаи в один столбец. В качестве примера можно привести набор данных о результатах быстрых свиданий с сайта **kaggle.com**, в котором значения признака *Gender* («Пол») указаны в одном столбце с использованием прямого кодирования (табл. 10). Вместо того чтобы создавать отдельные столбцы для значений *Male* («Мужчина») и *Female* («Женщина»), создатели набора объединили эти признаки в один, обозначив женщин как «0», а мужчин как «1». Тот же метод был использован для признаков *Same Race* («Та же раса») и *Match* («Совпадение»).

Табл. 10. Результаты быстрых свиданий, база данных: **https://www.kaggle.com/annavictoria/speed-dating-experiment**

№	Пол	Та же раса	Возраст	Совпадение
1	0	0	27	0
1	0	0	22	0
1	0	1	22	1
1	0	0	23	1
1	0	0	24	1
1	0	0	25	0
1	0	0	30	0

Пол:
Женский = 0
Мужской = 1

Та же раса:
Нет = 0
Да = 1

Совпадение:
Нет = 0
Да = 1

Биннинг

Биннинг (также называемый сегментацией) — это еще один метод конструирования признаков, который используется для преобразования непрерывных числовых значений в несколько бинарных признаков, называемых бинами или сегментами, в соответствии с диапазоном их значений.

Подождите-ка! Разве числовые значения не предпочтительны? Действительно, в большинстве случаев непрерывные числовые значения удобнее в силу своей совместимости с более широким спектром алгоритмов. Однако числовые значения оказываются неуместными в тех случаях, когда они отражают вариации, не имеющие отношения к целям вашего анализа.

В качестве примера возьмем оценку стоимости домов. Точные размеры теннисного корта могут не иметь большого значения при оценке недвижимости; релевантной информацией становится сам факт *наличия* теннисного корта в собственности. Эта же логика применима к гаражу и бассейну, наличие или отсутствие которых обычно имеет большее значение, чем их фактические размеры.

Решением в данном случае будет замена числовых измерений теннисного корта значениями Истина/Ложь или таким категориальным значением, как «маленький», «средний» и «большой». Другой альтернативой может быть применение метода прямого кодирования с использованием значения «0» для домов, в которых нет теннисного корта, а значения «1» — для домов, в которых он *есть*.

Нормализация

Хотя алгоритмы машинного обучения вполне могут работать и без применения следующих двух методов, нормализация и стандартизация помогают повысить точность модели при использовании подходящего алгоритма. Первый метод (нормализация) предполагает изменение исходного диапазона значений признака на диапазон с заданным минимумом и максимумом, например [0, 1] или [–1, 1]. Такой подход позволяет нормализовать дисперсию между признаками в наборе данных, которая в противном случае может оказаться преувеличенной вследствие влияния того или иного фактора. Например, дисперсия признака, измеряемого в сантиметрах, может отвлечь алгоритм от другого признака с аналогичной или более высокой степенью дисперсии, но измеряемого в метрах или других единицах, которые преуменьшают его фактическую дисперсию.

Однако метод нормализации обычно не рекомендуют использовать для масштабирования признаков с экстремальным диапазоном значений, поскольку нормализованный диапазон оказывается слишком узким для того, чтобы подчеркнуть чрезвычайно высокие или низкие значения признака.

Стандартизация

Более эффективный метод подчеркивания высоких или низких значений признака — стандартизация, которая предполагает преобразование единичной дисперсии в стандартное нормальное распределение со средним значением, равным нулю, и стандартным отклонением ($\sum$), равным единице[17]. Это означает, что чрезвычайно высокое или низкое значение будет выражено в виде трех или более стандартных отклонений от среднего (рис. 10).

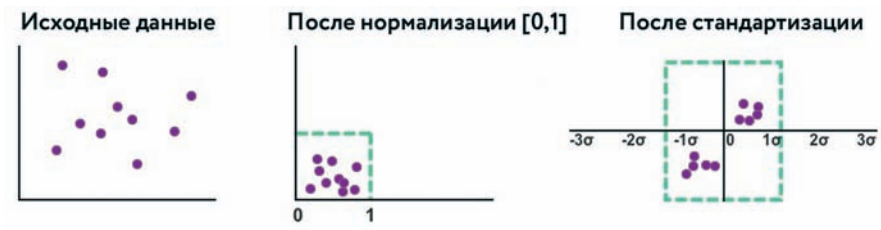


Рис. 10. Результаты масштабирования данных путем нормализации и стандартизации

Как правило, стандартизация оказывается более эффективной по сравнению с нормализацией, когда изменчивость признака отражает форму колоколообразной кривой нормального распределения и часто используется в контексте неконтролируемого обучения. В других ситуациях нормализация и стандартизация могут применяться по отдельности, а их результаты — сравниваться для большей точности.

Стандартизацию рекомендуют выполнять при подготовке данных для последующего использования машин опорных векторов (SVM), метода *k*-ближайших соседей (*k*-NN) и выполнения анализа главных компонент (PCA).

[17] Стандартное отклонение — это мера разброса точек данных. Оно измеряет изменчивость путем вычисления среднеквадратичного расстояния между наблюдениями и средним значением набора данных.

Отсутствующие данные

Сталкиваться с отсутствующими данными всегда неприятно. Представьте, что вы распаковываете пазл и обнаруживаете, что в коробке не хватает 5% кусочков. Отсутствие ряда значений в наборе данных — столь же неприятная ситуация, способная отрицательно сказаться на результатах вашего анализа и прогнозах модели. Однако существуют стратегии, позволяющие минимизировать это негативное влияние.

Один из подходов заключается в аппроксимации отсутствующих значений с помощью моды, которая представляет собой наиболее распространенное значение переменной в наборе данных. Этот подход лучше всего работает с категориальными и двоичными типами переменных, например с системой ранжирования с присвоением от одной до пяти звезд и положительными или отрицательными результатами теста на употребление наркотиков соответственно (рис. 11).

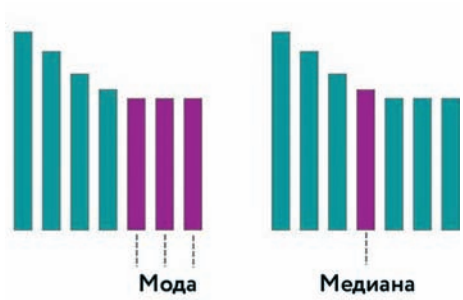


Рис. 11. Наглядный пример моды и медианы

Второй подход заключается в аппроксимации отсутствующих значений с помощью медианы, то есть значения, находящегося в середине набора данных. Этот подход лучше всего работает с непрерывными переменными, имеющими бесконечное число возможных значений, например с ценами на жилье.

В крайнем случае строки с отсутствующими значениями можно просто удалить. Очевидный недостаток такого подхода — сокращение объема анализируемых данных, чреватое недостаточно полным их пониманием.

6

РАЗБИЕНИЕ ДАННЫХ

После очистки набора данных его необходимо разбить на тренировочную и тестовую выборки для выполнения так называемой *сплит-проверки*. Как правило, набор данных делится в пропорции 70/30 или 80/20. Это означает, что, если переменные выражены горизонтально, а экземпляры вертикально (как показано на рис. 12), то на тренировочные данные должно приходиться 70–80% строк набора данных, а на тестовые — оставшиеся 20–30%.

Рис. 12. Разбиение набора данных на тренировочную и тестовую выборки в пропорции 70/30

Хотя данные принято делить в пропорции 70/30 или 80/20, четкого правила для разбиения набора на тренировочную и тестовую выборки не существует. Учитывая постоянно растущий размер современных наборов данных (количество строк в которых иногда превышает миллион), оптимальным вариантом может оказаться менее равномерное разделение, например в пропорции 90/10, поскольку в этом случае у вас будет больше

данных для обучения модели и при этом достаточное количество останется для ее тестирования.

Перед разбиением данные необходимо рандомизировать, то есть произвольным образом изменить порядок следования строк. Это поможет избежать смещения модели, поскольку данные в исходном наборе могут быть упорядочены по алфавиту или по дате их сбора. Если не рандомизировать данные, то можно случайно исключить значительную дисперсию из тренировочной выборки, что может вызвать нежелательные сюрпризы при применении обученной модели к тестовым данным. К счастью, библиотека Scikit-learn предусматривает встроенную команду, позволяющую выполнить перемешивание и рандомизацию данных с помощью всего одной строки кода, как показано в главе 17.

После рандомизации данных вы можете разработать модель и применить ее к тренировочной выборке. Оставшиеся 30% данных или около того следует отложить для последующего тестирования модели на предмет ее точности; крайне важно не тестировать модель на тех же данных, которые использовались для ее тренировки. В рамках контролируемого обучения модель разрабатывается путем подачи машине тренировочных данных и анализа взаимосвязей между входными признаками (*X*) и конечным результатом (*y*).

Следующий шаг — измерение эффективности работы модели. Существует целый ряд показателей эффективности, и выбор подходящего зависит от способа применения модели. Площадь под ROC-кривой (AUC, area under the curve — площадь под кривой; ROC, Receiver Operating Characteristic — рабочая характеристика приемника)[18], матрица ошибок (confusion matrix), полнота (recall) и точность (accuracy) — это четыре показателя эффективности, используемых в рамках таких задач классификации, как выявление спама в электронной почте. Между тем средняя абсолютная ошибка (MAE, mean absolute error) и квадратный корень из средней квадратической ошибки (RMSE, root mean square error) обычно используются для оценки эффективности моделей, которые выдают числовой результат вроде прогнозируемой стоимости дома.

В этой книге мы будем использовать среднюю абсолютную ошибку (MAE) — показатель, который измеряет среднюю величину ошибок в наборе прогнозов по числовой/непрерывной шкале, то есть расстояние

[18] Название этого термина обусловлено его происхождением из области радиолокационной техники.

от регрессионной гиперплоскости до конкретной точки данных. При использовании библиотеки Scikit-learn средняя абсолютная ошибка вычисляется путем подачи на вход модели значений *X* из тренировочной выборки и создания прогноза для каждой строки набора данных. После этого Scikit-learn сравнивает предсказания модели с правильным результатом (*y*) и измеряет точность модели. Низкое значение ошибки на тренировочной и тестовой выборках говорит о том, что модель изучила основные тенденции и закономерности, содержащиеся в наборе данных. Если же среднее значение MAE или RMSE на тестовых данных существенно превышает аналогичный показатель на тренировочных данных, то это обычно свидетельствует о переобучении модели (подробнее об этом мы поговорим в главе 11). Модель, адекватно предсказывающая значения тестовых данных, можно считать готовой к использованию в реальных условиях.

Если модель не может точно предсказать значения тестовой выборки, проверьте, были ли рандомизированы тренировочные и тестовые данные. Далее вам может понадобиться изменить гиперпараметры модели. Каждый алгоритм имеет свои гиперпараметры; это настройки обучения алгоритма (а не самой модели). Если в двух словах, то гиперпараметры контролируют скорость, с которой модель изучает закономерности, а также влияют на то, какие именно закономерности она будет выявлять и анализировать. О гиперпараметрах алгоритма и оптимизации мы поговорим в главах 11 и 18.

Перекрестная проверка

Хотя сплит-проверка может оказаться эффективным подходом к разработке модели на основе существующих данных, возникает вопрос о том, сможет ли эта модель оставаться точной при ее применении к новым данным. Слишком маленький размер существующего набора данных или неправильно выбранная пропорция при его разделении на тренировочную и тестовую выборки может впоследствии привести к плохому качеству прогнозов на реальных данных.

К счастью, эту проблему вполне можно обойти. Вместо того чтобы разбивать данные на два сегмента (один для тренировки, другой для тестирования), вы можете реализовать так называемую перекрестную проверку *(или кросс-валидацию)*, которая максимизирует количество доступных тренировочных данных путем разбиения исходного набора на различные комбинации и тестирования модели на каждой из них.

Для выполнения перекрестной проверки существуют два основных метода. Первый метод под названием исчерпывающая перекрестная проверка предполагает поиск и тестирование всех возможных комбинаций, то есть способов разделения исходного набора данных на тренировочную и тестовую выборки. Альтернативный и более популярный метод — неисчерпывающая или *k-кратная* перекрестная проверка (рис. 13). Этот метод предполагает разбиение набора данных на *k* подмножеств и резервирование одного из них для тестирования модели в каждом раунде проверки.

Для выполнения *k*-кратной перекрестной проверки данные случайным образом разделяются на *k* подмножеств одинакового размера. Одно из них выбирается в качестве тестового и используется для измерения и оценки эффективности модели, построенной на основе остальных (*k – 1*) подмножеств.

Рис. 13. Процесс *k*-кратной перекрестной проверки

Перекрестная проверка выполняется *k* раз. В рамках каждого раунда одно подмножество резервируется для тестирования модели, обученной на остальных подмножествах. Этот процесс повторяется до тех пор, пока все подмножества данных не будут использованы в качестве тренировочной

и тестовой выборки. Затем результаты агрегируются и объединяются для формулирования единой модели.

Благодаря использованию всех доступных данных для тренировки и тестирования модели и дальнейшему усреднению полученных результатов *k*-кратная перекрестная проверка позволяет минимизировать ошибку прогнозирования, обычно возникающую при фиксированном разделении набора данных на тренировочную и тестовую выборки. Однако этот метод работает медленнее, поскольку длительность процесса обучения умножается на количество проверочных подмножеств данных.

Сколько данных мне нужно?

Частый вопрос новичков в области машинного обучения звучит так: «Сколько данных мне нужно для обучения модели?» В целом машинное обучение дает наилучшие результаты тогда, когда набор тренировочных данных содержит полный спектр комбинаций признаков.

Как же выглядит этот спектр? Представьте, что у вас есть набор данных о дата-сайентистах, содержащий следующие признаки:

- университетская степень (*X*);
- более 5 лет профессионального опыта (*X*);
- дети (*X*);
- зарплата (*y*).

Для оценки взаимосвязи первых трех признаков (*X*) с зарплатой специалиста (*y*) нам нужен набор данных, включающий значение *y* для каждой комбинации признаков. Например, нам нужно знать зарплату бездетных дата-сайентистов с высшим образованием и более чем пятилетним опытом работы, а также зарплату специалистов с высшим образованием и опытом работы более пяти лет, у которых есть дети.

Чем больше комбинаций признаков содержится в наборе данных, тем эффективнее модель отражает влияние каждого атрибута на значение *y* (размер заработной платы). Это гарантирует то, что при применении модели к тестовым или реальным данным она не выйдет из строя, столкнувшись с не виденными ранее комбинациями.

Количество точек данных, содержащихся в простейшей модели машинного обучения, должно превышать общее количество признаков как

минимум в десять раз. Так, в случае с небольшим набором данных, включающим в себя пять признаков, тренировочная выборка в идеале должна содержать не менее 50 строк. Однако наборы данных с большим количеством признаков требуют большего количества точек данных, поскольку число комбинаций растет экспоненциально по мере увеличения числа переменных.

Как правило, чем больше релевантных данных вы используете в качестве тренировочных, тем больше комбинаций можете включить в свою предсказательную модель, что способствует получению более точных прогнозов. В некоторых случаях поиск данных, охватывающих все возможные комбинации, может оказаться невозможным или экономически нецелесообразным, и придется довольствоваться тем, что есть. И наоборот, после достижения определенного объема тренировочных данных (достаточно хорошо представляющих проблему) отдача начинает постепенно уменьшаться.

Последний важный момент — соответствие данных алгоритму. Для наборов, содержащих менее 10 000 образцов, высокоэффективными могут оказаться алгоритмы кластеризации и снижения размерности, в то время как алгоритмы регрессионного анализа и классификации больше подходят для наборов данных, содержащих менее 100 000 образцов. Нейронные сети требуют еще больше образцов для эффективной работы. Они более экономичны в плане использования ресурсов при взаимодействии с большими объемами данных.

В библиотеке Scikit-learn есть шпаргалка, позволяющая подобрать алгоритм к различным наборам данных: **http://scikit-learn.org/stable/tutorial/machine_learning_map/**.

> В следующих главах мы рассмотрим конкретные алгоритмы, часто используемые в сфере машинного обучения. *Обратите внимание* на то, что по необходимости я включил в них некоторые уравнения, но постарался сделать их максимально простыми. Многие из методов машинного обучения, обсуждаемых в этой книге, уже предусматривают рабочие реализации на выбранном вами языке программирования и не требуют решения каких-либо уравнений.

Вы также можете найти бесплатные видеоуроки по разработке моделей на языке *Python* с использованием упомянутых в этой книге алгоритмов на сайте*:* ***https://scatterplotpress.teachable.com/p/ml-code-exercises***.

7

ЛИНЕЙНАЯ РЕГРЕССИЯ

Регрессионный анализ, будучи аналогом программы *Hello World* в мире алгоритмов контролируемого обучения, представляет собой простую технику прогнозирования неизвестной переменной на основе известных результатов. Первым делом мы рассмотрим линейную регрессию, которая генерирует прямую линию для описания линейных отношений. Мы начнем с изучения основных компонентов простой модели линейной регрессии с одной независимой переменной, а затем обсудим модель множественной регрессии с несколькими независимыми переменными.

Используя данные о количестве зрителей телесериала *Seinfeld* (табл. 11), мы построим график на основе двух переменных: номер сезона будет выступать в качестве координаты *x*, а число зрителей по сезонам (в миллионах) — в качестве координаты *y*.

Табл. 11. Набор данных *Seinfeld*

Время года (x)	Зрители (y)
1	19,22
2	18,07
3	17,67
4	20,52
5	29,59
6	31,27
7	33,19
8	32,24
9	38,11

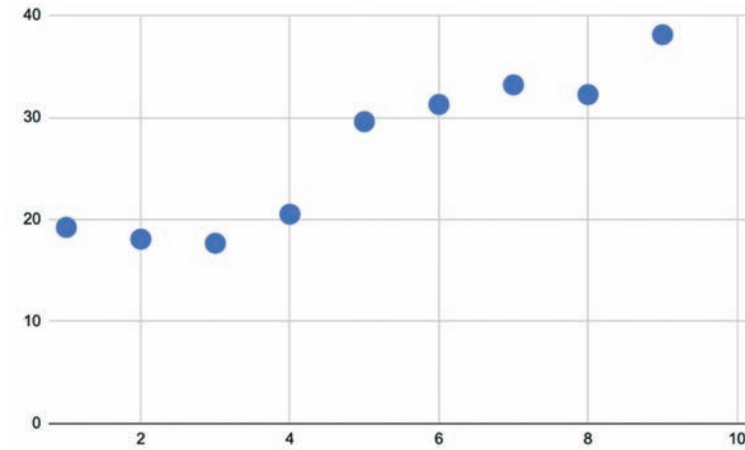


Рис. 14. Набор данных *Seinfeld* на диаграмме рассеяния

Как видно на этой диаграмме рассеяния, тенденция к росту числа зрителей начинается с 4-го сезона, а пик приходится на 9-й сезон (рис. 14).

Теперь давайте определим независимые и зависимые переменные. В этом примере в качестве зависимой переменной будет выступать число зрителей за сезон (которое мы хотим предсказать), а в качестве независимой переменной — номер сезона.

Используя простую модель линейной регрессии, давайте проведем прямую линию, чтобы описать восходящую тенденцию в нашем небольшом наборе данных.

Как видно на рис. 15, линия регрессии четко разделяет совокупность точек данных. На техническом языке линия регрессии называется гиперплоскостью, и вы будете сталкиваться с этим термином на протяжении всего процесса освоения машинного обучения. В двумерном пространстве гиперплоскость выступает в качестве (плоской) линии тренда, — именно так линейная регрессия называется в меню настроек диаграммы рассеяния Google Sheets.

Цель линейной регрессии — разделить данные таким образом, чтобы минимизировать расстояние между гиперплоскостью и наблюдаемыми значениями. Это означает, что если провести перпендикулярную линию (прямую под углом 90°) от гиперплоскости к каждой точке данных на графике, то совокупное расстояние между точками и гиперплоскостью окажется наименьшим из возможных. Расстояние между линией наилучшего соответствия и наблюдаемыми значениями называется остатком или ошибкой, и чем ближе эти значения к гиперплоскости, тем точнее предсказания модели (рис. 16).

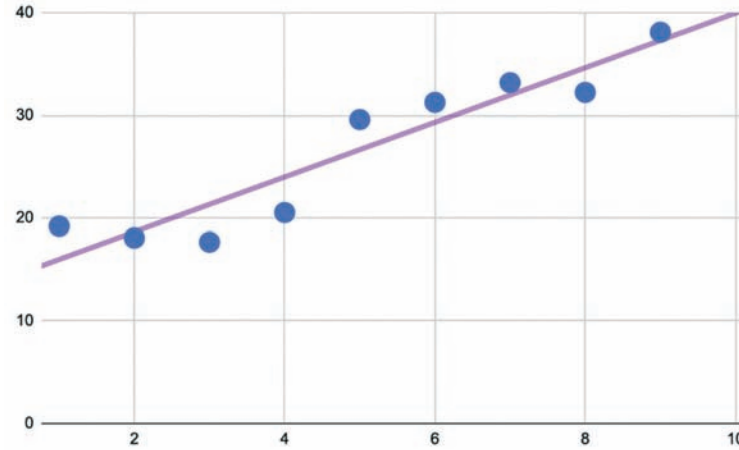


Рис. 15. Гиперплоскость линейной регрессии

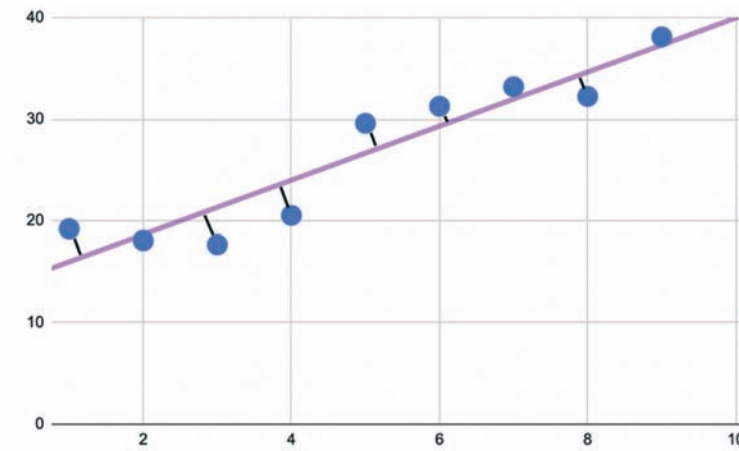


Рис. 16. Ошибка — это расстояние между гиперплоскостью и наблюдаемым значением

Наклон

Важная часть модели линейной регрессии — наклон, который можно рассчитать, обратившись к гиперплоскости, показывающей среднее значение, на которое будет увеличиваться одна переменная при увеличении другой. Таким образом, наклон оказывается полезным для составления прогнозов, например для предсказания количества зрителей потенциального десятого

сезона сериала *Seinfeld*. Для этого мы можем ввести значение 10 в качестве координаты *x* и найти соответствующее значение y, которое в данном случае составляет приблизительно 40 миллионов зрителей (рис. 17).

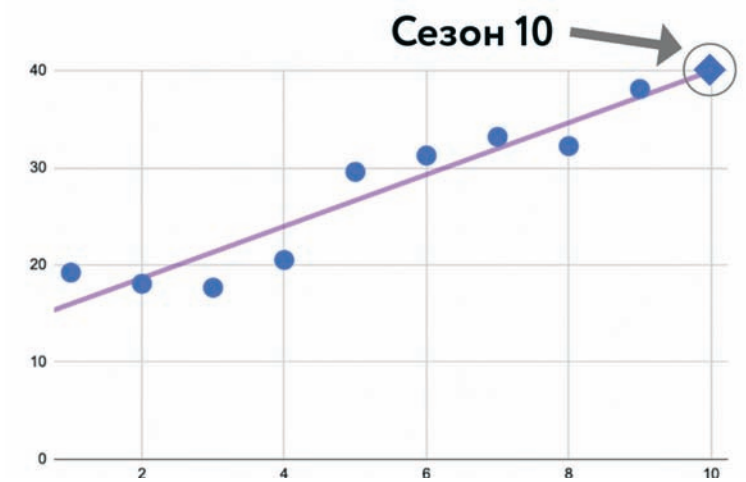


Рис. 17. Использование наклона/гиперплоскости для прогнозирования

Хотя линейная регрессия не безотказный метод прогнозирования, линия тренда предоставляет базовую точку отсчета для предсказания неизвестных или будущих событий.

Формула линейной регрессии

Формула[19] для линейной регрессии выглядит так: $y = bx + a$.

В данном случае «y» представляет собой зависимую переменную, «*x*» — независимую, «*a*» — это точка пересечения гиперплоскости с осью y, то есть значение y при $x = 0$.

Коэффициент «*b*» определяет крутизну наклона линии и объясняет взаимосвязь между *x* и y (то есть предсказывает, насколько изменится y при изменении *x* на 1 единицу).

[19] Хотя в других дисциплинах формула линейной функции может записываться по-разному, в области статистики и машинного обучения предпочтителен формат $y = bx + a$. Эта формула также может быть выражена в виде $y = \beta_0 + \beta_1 x_1 + e$, где β_0 — точка пересечения, β_1 — наклон, а *e* — остаток или ошибка.

Пример расчета

Хотя выбранный вами язык программирования способен выполнить все расчеты автоматически, вам может быть интересно узнать, как работает простая модель линейной регрессии. Давайте разберем приведенную выше формулу на примере следующего набора данных (табл. 12).

Табл. 12. Образец набора данных

	(X)	(Y)	XY	X^2
1	1	3	3	1
2	2	4	8	4
3	1	2	2	1
4	4	7	28	16
5	3	5	15	9
∑ (Итого)	**11**	**21**	**56**	**31**

Последние два столбца таблицы не входят в исходный набор данных и были добавлены для заполнения следующей формулы.

$$\mathbf{a} = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2}$$

$$\mathbf{b} = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

Где:

$\sum$ = Общая сумма
$\sum x$ = Общая сумма всех значений *x* (1 + 2 + 1 + 4 + 3 = 11)
$\sum y$ = Общая сумма всех значений *y* (3 + 4 + 2 + 7 + 5 = 21)
$\sum xy$ = Общая сумма произведений *x* * *y* из каждой строки (3 + 8 + 2 + 28 + 15 = 56)
$\sum x^2$ = Общая сумма произведений *x* * *x* из каждой строки (1 + 4 + 1 + 16 + 9 = 31)
n = Общее количество строк. В данном примере *n* = 5.

$$\mathbf{a} = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2} \qquad \mathbf{a} = \frac{(21)(31) - (11)(56)}{5(31) - (11)^2}$$

$$\mathbf{b} = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2} \qquad \mathbf{b} = \frac{5(56) - (11)(21)}{5(31) - (11)^2}$$

a =
((21 × 31) — (11 × 56)) / (5(31) — 11^2)
(651–616) / (155–121)
35 / 34 = 1,029

b =
(5(56) — (11 × 21)) / (5(31) — 11^2)
(280–231) / (155–121)
49 / 34 = 1,441

Подставим значения *a* и *b* в формулу линейной функции.
$y = bx + a$
$y = 1{,}441x + 1{,}029$

Формула $y = 1{,}441x + 1{,}029$ диктует способ построения гиперплоскости.

Теперь давайте протестируем модель линейной регрессии, найдя координаты для значения $x = 2$.
$y = 1{,}441(x) + 1{,}029$
$y = 1{,}441(2) + 1{,}029$
$y = 3{,}911$
В данном случае прогноз очень близок к фактическому результату 4,0 (рис. 18).

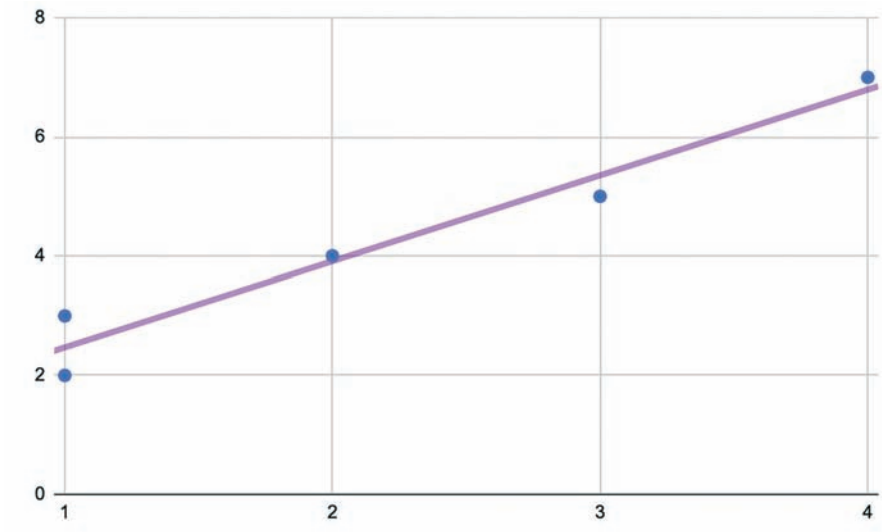


Рис. 18. График функции y = 1,441*x* + 1,029, построенный на диаграмме рассеяния

Множественная линейная регрессия

До этого мы рассмотрели простую модель линейной регрессии с одной независимой переменной. Теперь давайте поговорим о множественной линейной регрессии. Эта методика имеет большее отношение к машинному обучению, поскольку для принятия решений организации используют более одной независимой переменной.

Множественная линейная регрессия представляет собой обычную модель линейной регрессии, но не с одной, а с несколькими независимыми переменными, как показывает следующая формула.

$$y = a + b_1x_1 + b_2x_2 + b_3x_3 + \ldots$$

Точка пересечения с осью y по-прежнему выражается в виде параметра *a*, но теперь у нас есть несколько независимых переменных (x_1, x_2, x_3 и т. д.), каждая из которых предусматривает собственный коэффициент (b_1, b_2, b_3 и т. д.).

Как и в случае с простой линейной регрессией, различные суммы значений *X* и y (включая возведенные в квадрат) из тренировочного набора используются для вычисления значений *a* (точка пересечения с осью y) и *b* (коэффициенты).

После построения модели с использованием значений *X* и y, содержащихся в тренировочном наборе, мы можем использовать формулу множественной линейной регрессии для предсказания значения y на основе значений *X* из тестового набора (для оценки точности модели).

Дискретные переменные

Хотя выходные данные (зависимая переменная) модели линейной регрессии должны быть непрерывными и представлять собой значения с плавающей запятой или целые числа, входные данные (независимые переменные) могут быть как непрерывными, так и категориальными. Значения категориальных переменных (например, пол) должны быть выражены в виде чисел с использованием прямого кодирования (0/1), а не в виде строк (мужчина/женщина).

Выбор переменных

Прежде чем завершить эту главу, важно рассмотреть дилемму выбора переменных и определения соответствующего количества независимых переменных. С одной стороны, использование большего числа переменных помогает учесть больше потенциальных факторов, влияющих на существующие в данных закономерности. С другой стороны, это обоснование справедливо лишь в том случае, если переменные релевантны и имеют определенную корреляцию/линейную связь с зависимой переменной.

Увеличение числа независимых переменных также позволяет рассмотреть больше взаимосвязей. В простой модели линейной регрессии мы рассматривали связь только между двумя переменными по принципу «один к одному», тогда как модель множественной линейной регрессии позволяет рассматривать связи по принципу «многие к одному». В случае множественной линейной регрессии независимые переменные могут быть связаны не только с зависимой переменной, но и друг с другом.

Наличие сильной линейной корреляции между двумя независимыми переменными может привести к такой проблеме, как мультиколлинеарность. Когда две независимые переменные сильно коррелируют, они имеют тенденцию отменять друг друга, предоставляя модели очень мало или вообще никакой уникальной информации (рис. 19).

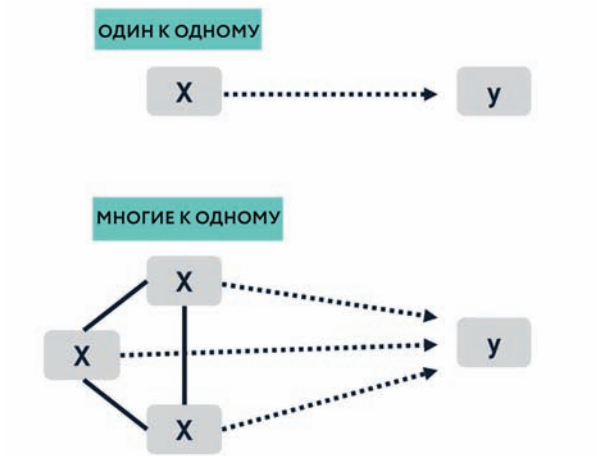


Рис. 19. Простая линейная регрессия (вверху) и множественная линейная регрессия (внизу)

Количество потребленного топлива и количество топлива в баке при прогнозировании дальности полета реактивного самолета можно рассматривать в качестве примеров мультиколлинеарных переменных. Эти независимые переменные непосредственно связаны друг с другом, и в данном случае они отрицательно коррелируют; при увеличении значения одной переменной значение другой уменьшается, и наоборот. Когда обе переменные используются для прогнозирования зависимой переменной — дальности полета самолета, одна из них фактически аннулирует другую. Одну из этих переменных все же стоит включить в модель, однако включение обеих было бы излишним.

Чтобы избежать мультиколлинеарности, необходимо проверить связь между каждой комбинацией независимых переменных с помощью диаграммы рассеяния, парного графика (матрицы, отражающей попарные отношения между переменными) или коэффициента корреляции.

Глядя на парный график, представленный на рис. 20, мы можем проанализировать взаимосвязи между всеми тремя переменными (total_bill, tip и size). Если в качестве зависимой переменной мы выберем tip (чаевые), то нам нужно будет оценить, насколько сильно коррелируют две независимые переменные (total_bill (сумма чека) и size (размер — в данном случае число гостей)). На представленном парном графике мы видим две диаграммы рассеяния, визуализирующие взаимосвязи между переменными

total_bill и size (ряд 1 справа и ряд 3 слева). Эти два графика не идентичны (так как их оси *x* и y инвертированы), но вы можете обратиться к любому из них.

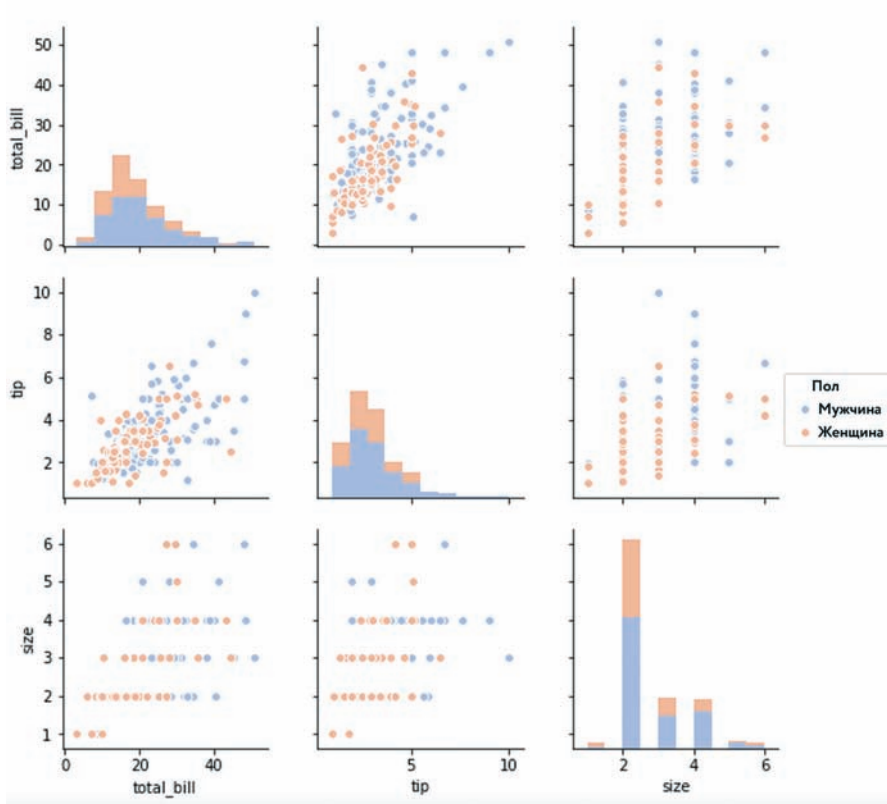


Рис. 20. Парный график с тремя переменными

Восходящая линия тренда говорит о том, что эти две переменные частично коррелированы. Однако, если бы мы построили гиперплоскость линейной регрессии, то по обе стороны от нее обнаружили бы значительные остатки/ошибки, говорящие об отсутствии сильной корреляции между этими двумя переменными, а это значит, что мы вполне можем включить их в свою регрессионную модель.

Тепловая карта, представленная на рис. 21, также свидетельствует о весьма скромном коэффициенте корреляции между переменными total_bill и size, равном 0,6.

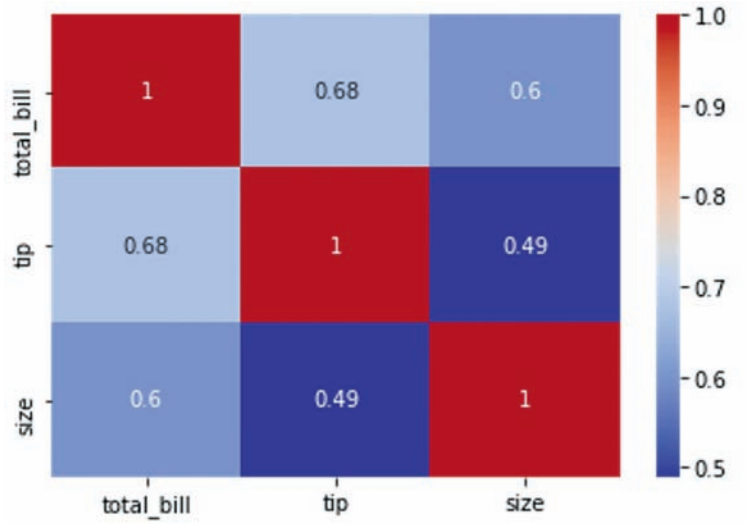


Рис. 21. Тепловая карта с тремя переменными

Мы также можем использовать парный график, тепловую карту или коэффициент корреляции, чтобы проверить, коррелируют ли независимые переменные с зависимой (и, следовательно, имеют ли они отношение к результату прогнозирования). На рис. 21 видно, что переменные total_bill (0,68) и size (0,49) демонстрируют некоторую корреляцию с зависимой переменной tip. (Коэффициент корреляции принимает значения в диапазоне между –1 и 1, причем значение 1 говорит об идеальной положительной корреляции, а значение –1 — об идеальной отрицательной корреляции. Коэффициент, равный 0, означает отсутствие связи между двумя переменными.)

Если вкратце, то цель множественной линейной регрессии заключается в обеспечении того, чтобы все независимые переменные коррелировали с зависимой, но не друг с другом.

Контрольная работа

Используя множественную линейную регрессию, создайте модель для прогнозирования суммы чаевых, которые гости оставят в ресторане при оплате обеда. Обратите внимание на то, что ниже представлен лишь фрагмент набора данных, который содержит 244 строки.

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2

1. Какая(ие) из следующих переменных должна(ы) выступать в качестве зависимой переменной в этой модели?
   А) size;
   Б) total_bill и tip;
   В) total_bill;
   Г) tip.

2. Опираясь только на приведенные выше данные, определите, какая(ие) переменная(ые) имеет(ют) линейную зависимость с переменной tip?
   А) smoker;
   Б) total_bill и size;
   В) time;
   Г) sex.

3. Очень важно, чтобы независимые переменные сильно коррелировали с зависимой переменной, а также с одной или несколькими другими независимыми переменными. Правда или ложь?

Ответы

1. Г.
2. Б (при увеличении значения обеих этих переменных мы наблюдаем увеличение суммы чаевых в большинстве строк. Другие переменные тоже могут быть связаны с переменной tip, но судить об этом только по приведенным 10 строкам не представляется возможным).
3. Ложь (в идеале независимые переменные не должны сильно коррелировать друг с другом).

8

ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

В предыдущей главе мы выяснили, что линейная регрессия — это полезный инструмент количественной оценки отношений между переменными для прогнозирования непрерывного результата. В качестве примеров таких непрерывных переменных мы привели сумму чека и размер (число гостей).

Но что, если нам нужно предсказать значение такой категориальной переменной, как «новый клиент» или «вернувшийся клиент»? В отличие от модели линейной регрессии в данном случае зависимая переменная (*y*) не непрерывна (как общая сумма чаевых), а, скорее, дискретна и категориальна.

Вместо того чтобы количественно оценивать линейную связь между переменными, нам необходимо использовать такой метод классификации, как логистическая регрессия.

Логистическая регрессия — это тоже метод контролируемого обучения, но она дает качественное, а не количественное предсказание. Этот алгоритм часто используется для предсказания одного из двух дискретных классов, например «беременна» или «не беременна». Будучи мощным инструментом бинарной классификации, логистическая регрессия используется для решения множества задач, в том числе для выявления фактов мошенничества, диагностики заболеваний, обнаружения чрезвычайных ситуаций, выявления неплатежей по кредитам и идентификации спама в электронной почте.

Используя сигмоидальную функцию, модель логистической регрессии определяет вероятность того, что независимые переменные (*X*) приведут к получению дискретной зависимой переменной (*y*), такой как «спам» или «не спам».

$$a = \frac{1}{1+e^{-x}}$$

Где:

x — независимая переменная, которую вы хотите преобразовать;

e — число Эйлера, равное 2,718.

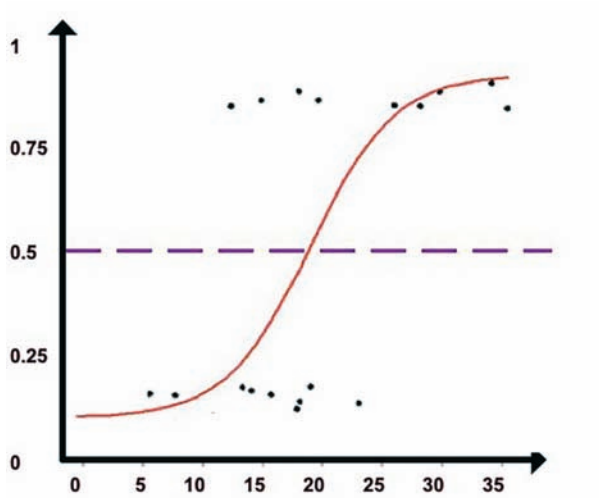


Рис. 22. Сигмоидальная функция, используемая для классификации точек данных

Сигмоидальная функция имеет вид S-образной кривой и преобразует любое число в значение, находящееся в диапазоне от 0 до 1, но никогда не достигающее этих пределов. Используя эту формулу, сигмоидальная функция преобразует независимые переменные в значение вероятности от 0 до 1 относительно зависимой переменной. В случае бинарной классификации значение 0 говорит об отсутствии вероятности наступления события, а 1 гарантирует его наступление. Близость значения к 0 или 1 определяет степень вероятности наступления события.

Опираясь на значения вероятности, вычисленные для независимых переменных, модель логистической регрессии относит каждую точку данных к тому или иному дискретному классу. В случае бинарной классификации (рис. 22) линия отсечения соответствует значению 0,5. Точки данных, находящиеся над этой линией, относятся к классу A, а точки данных под ней — к классу B. Точки данных, значение которых равно 0,5, не поддаются классификации, но встречаются редко из-за математической составляющей сигмоидальной функции.

После логистического преобразования с использованием сигмоидальной функции точки данных оказываются отнесенными к одному из двух классов, как показано на рис. 23.

Как и в случае линейной регрессии, независимые переменные, используемые в качестве входных данных для предсказания зависимой

переменной, могут быть категориальными или непрерывными при условии, что они будут выражены в виде чисел, а не строк. В случае дискретных категориальных переменных для численного представления исходной переменной необходимо выполнить прямое кодирование, чтобы создать новый набор переменных.

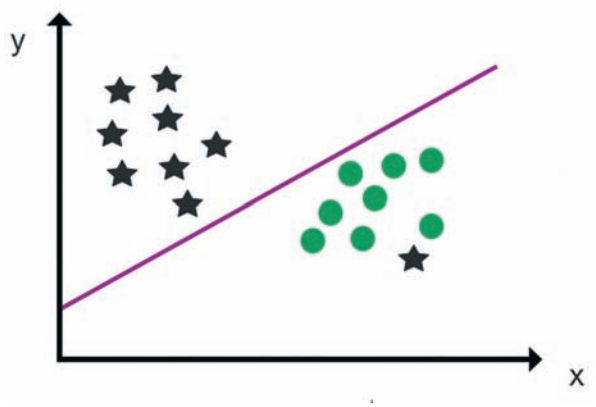


Рис. 23. Пример логистической регрессии

Несмотря на визуальное сходство между логистической и линейной регрессией, логистическая гиперплоскость представляет собой границу классификации/принятия решения, а не прогнозную линию тренда. Таким образом, гиперплоскость используется не для предсказания числового значения, а для разделения набора данных на классы.

Еще одно различие между логистической и линейной регрессией заключается в том, что в случае логистической регрессии зависимая переменная (y) не откладывается по оси y. Вместо этого независимые переменные могут откладываться вдоль обеих осей, а класс зависимой переменной (выход) определяется положением точки данных относительно границы принятия решения. Точки данных, находящиеся по одну сторону от этой границы, относятся к классу A, а точки данных по другую сторону от нее — к классу B.

В случаях классификаций, предусматривающих более двух возможных дискретных исходов, можно использовать мультиномиальную логистическую регрессию, как показано на рис. 24.

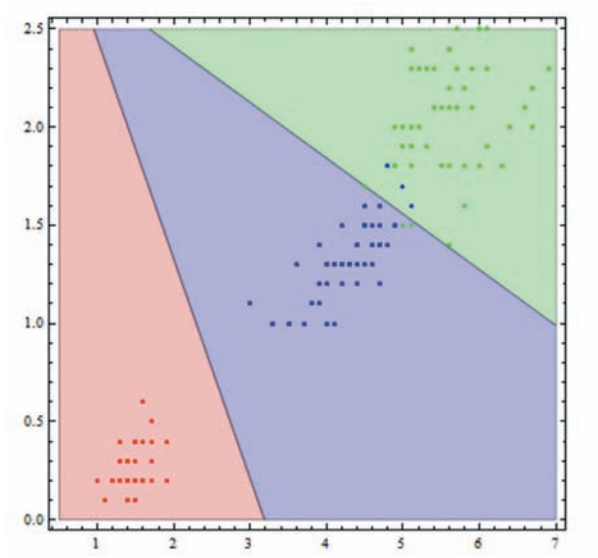


Рис. 24. Пример мультиномиальной логистической регрессии

Будучи еще одним методом классификации, мультиномиальная логистическая регрессия решает задачи многоклассовой классификации с более чем двумя возможными дискретными исходами. Этот метод также может быть применен в случаях с определенным количеством идущих по порядку дискретных исходов, например: бакалавриат, магистратура и аспирантура. Однако имейте в виду то, что основная мощь логистической регрессии заключается в прогнозировании бинарных исходов, а для решения задач многоклассовой классификации предпочтительными могут оказаться другие алгоритмы, в том числе деревья решений и машины опорных векторов.

При использовании логистической регрессии следует позаботиться о том, чтобы в наборе данных не было пропущенных значений, а все независимые переменные не сильно коррелировали друг с другом. Кроме того, для обеспечения высокой точности прогнозов важно иметь достаточно данных для каждой выходной переменной. Лучше ориентироваться на 30–50 точек данных для каждого выходного результата, то есть 60–100 точек данных для бинарной логистической регрессии. В целом логистическая регрессия не очень хорошо работает с большими наборами данных, особенно с такими, которые содержат выбросы, сложные взаимосвязи и пропущенные значения.

Если вы хотите узнать больше о математических основах логистической регрессии, посмотрите серию видеороликов Брэндона Фольца на YouTube[20] под названием *Statistics 101: Logistic Regression*[21].

Контрольная работа

Используя логистическую регрессию, распределите пингвинов по различным классам на основе следующего набора данных. Обратите внимание на то, что в этом наборе содержится 344 строки, а ниже приведен лишь его фрагмент.

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE
6	Adelie	Torgersen	38.9	17.8	181.0	3625.0	FEMALE
7	Adelie	Torgersen	39.2	19.6	195.0	4675.0	MALE
8	Adelie	Torgersen	34.1	18.1	193.0	3475.0	NaN
9	Adelie	Torgersen	42.0	20.2	190.0	4250.0	NaN

1. Какие три переменные (в их нынешней форме) мы могли бы использовать в качестве зависимой переменной для классификации пингвинов?
2. В какой(их) строке (строках) содержатся пропущенные значения?
3. Какая переменная в представленном выше фрагменте набора данных является бинарной (двоичной)?

[20] Соцсеть признана экстремистской и запрещена на территории РФ.

[21] Brandon Foltz, "Logistic Regression," *YouTube*, **https://www.youtube.com/channel/UCFrjdcImgcQVyFbK04MBEhA**

Ответы

1. species, island или sex.
2. В строках 3, 8 и 9 (NaN = отсутствующее значение).
3. sex (переменные species и island тоже могут быть бинарными, но мы не можем судить об этом по приведенному фрагменту набора данных).

9

МЕТОД *K*-БЛИЖАЙШИХ СОСЕДЕЙ

Еще один популярный метод классификации в области машинного обучения — метод *k*-ближайших соседей (*k*-NN, *k*-nearest neighbors). Будучи алгоритмом контролируемого обучения, *k*-NN классифицирует новые точки данных на основе их положения относительно соседних точек.

Во многих отношениях метод *k*-NN напоминает систему голосования или конкурс популярности. Представьте, что вы перешли в новую школу, и вам нужно понять, как одеваться, чтобы вписаться в свой класс. В первый день вы видите, что у шести из девяти учеников, сидящих ближе всего к вам, закатаны рукава. Опираясь на их численное превосходство и близкое соседство, вы решаете на следующий день тоже прийти с закатанными рукавами.

Теперь рассмотрим другой пример.

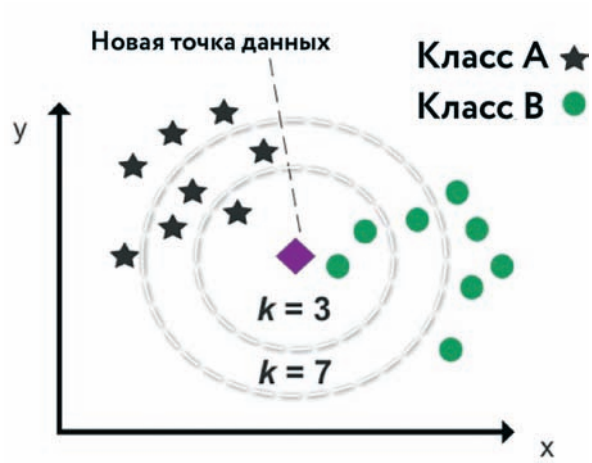


Рис. 25. Пример кластеризации методом *k*-NN, используемой для предсказания класса новой точки данных

На рис. 25 показаны точки данных, разделенные на два класса, и новая точка данных, класс которой еще не известен. Используя метод *k*-NN, мы можем предсказать категорию новой точки на основе ее расположения относительно существующих точек данных.

Однако сначала нам нужно задать значение *k*, определяющее число точек данных, которое мы будем использовать для классификации новой точки. Если мы зададим значение *k* равным 3, то модель *k*-NN проанализирует положение новой точки данных относительно трех ближайших точек (соседей). В результате мы получим две точки данных класса B и одну точку данных класса A. Таким образом, при *k* равном 3 модель отнесет новую точку данных к классу B, к которому принадлежат два из трех ее ближайших соседей.

Выбор числа соседей, определяемого параметром *k*, оказывает решающее влияние на получаемые результаты. На рис. 25 видно, что результат классификации меняется при изменении значения *k* с 3 на 7. Поэтому бывает полезно протестировать несколько значений *k*, чтобы найти наилучший вариант. Кроме того, не следует задавать слишком низкое или слишком высокое значения *k*. Слишком низкое значение *k* может увеличить смещение модели и привести к неправильной классификации, а слишком высокое сопряжено с большими вычислительными затратами. Выбор нечетного значения *k* также помогает исключить возможность попадания в статистический тупик и получения неверного результата. В библиотеке Scikit-learn число соседей при использовании этого алгоритма по умолчанию задается равным пяти.

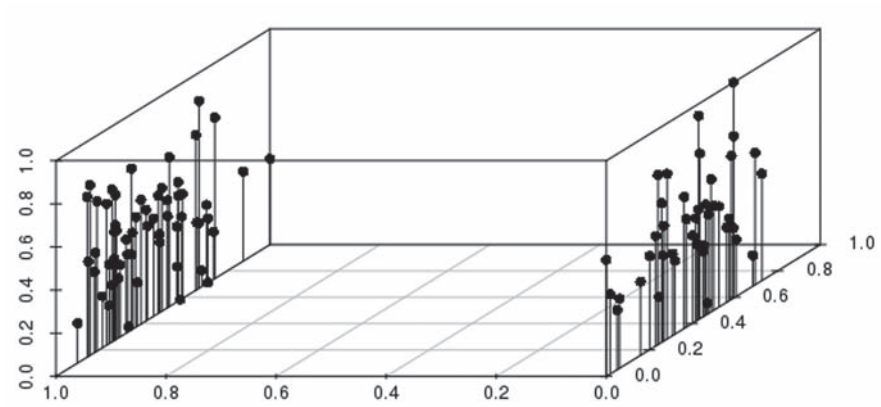


Рис. 26. Одна бинарная переменная и две непрерывные переменные

Учитывая то, что масштаб отдельных переменных оказывает большое влияние на результаты модели *k*-NN, набор данных обычно необходимо масштабировать для стандартизации дисперсии, как было показано в главе 5. Такое преобразование не позволяет одной или нескольким переменным с большим размахом перетянуть на себя внимание модели *k*-NN.

Что касается предпочтительного типа данных, то алгоритм *k*-NN лучше всего работает с непрерывными переменными. Можно использовать и бинарные категориальные переменные, представленные в виде значений 0 и 1, однако результаты, скорее всего, будут определяться бинарным разделением в соответствии с разбросом значений других переменных, как показано на рис. 26.

На представленном выше изображении мы видим, что горизонтальная ось *x* является бинарной (0 или 1), в результате чего данные оказываются сгруппированными по разные стороны. Более того, если мы заменим одну из существующих непрерывных переменных на бинарную (как показано на рис. 27), то увидим, что расстояние между переменными еще больше зависит от итогового значения бинарных переменных.

Если вы все же хотите исследовать бинарные переменные, включайте в модель *k*-NN только самые необходимые из них.

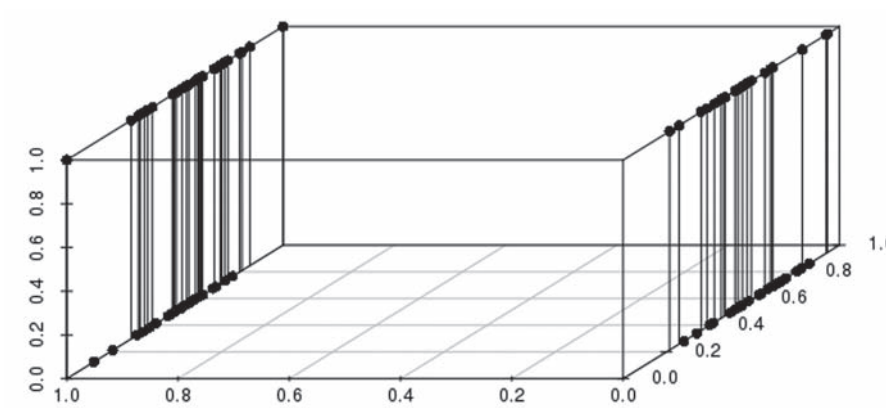


Рис. 27. Две бинарные переменные и одна непрерывная переменная

Хотя в целом метод *k*-NN довольно точен и прост для понимания, хранение всего набора данных и вычисление расстояния между каждой новой точкой данных и всеми существующими точками требует больших

вычислительных ресурсов. Это означает, что число точек данных в наборе пропорционально количеству времени, необходимому для получения одного предсказания. По этой причине метод *k*-NN обычно не рекомендуется использовать для анализа больших наборов данных.

Еще один недостаток связан с потенциальной сложностью применения метода *k*-NN к высокоразмерным данным с большим количеством признаков. Измерение расстояний между множеством точек данных в высокоразмерном пространстве также сопряжено с большими затратами вычислительных ресурсов, что усложняет выполнение точной классификации.

Контрольная работа

Используя метод *k*-ближайших соседей при *k* равном 5, классифицируйте пингвинов по видам. Обратите внимание на то, что в этом наборе содержится 344 строки, а ниже приведен лишь его фрагмент.

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE
6	Adelie	Torgersen	38.9	17.8	181.0	3625.0	FEMALE
7	Adelie	Torgersen	39.2	19.6	195.0	4675.0	MALE
8	Adelie	Torgersen	34.1	18.1	193.0	3475.0	NaN
9	Adelie	Torgersen	42.0	20.2	190.0	4250.0	NaN

1. Какие из следующих переменных нам следует исключить из модели ***k*-NN?**
   А) sex;
   Б) species;
   В) body_mass_g;
   Г) bill_depth_mm.
2. Какой из следующих методов мы можем использовать для ускорения работы нашей модели?

А) увеличение значения *k* с 5 до 10;

Б) уменьшение значения *k* с 10 до 5;

В) повторный запуск модели в надежде на более быстрый результат;

Г) увеличение набора тренировочных данных.

3. Какой метод очистки данных нам следует использовать, чтобы включить в модель переменную «sex»?

Ответы

1. А (бинарные переменные следует использовать только в тех случаях, когда это необходимо для обеспечения точности модели).
2. Б.
3. Прямое кодирование (для преобразования значения переменной в числовой идентификатор 0 или 1).

10

КЛАСТЕРИЗАЦИЯ МЕТОДОМ *K*-СРЕДНИХ

Следующий метод анализа предполагает группировку или кластеризацию точек данных, обладающих схожими признаками, с помощью алгоритма неконтролируемого обучения. Например, некая интернет-компания хочет изучить сегмент клиентов, совершающих покупки в одно и то же время года, и выяснить, какие факторы влияют на их покупательское поведение. Поняв особенности этого кластера покупателей, компания сможет принимать решения о том, какие продукты рекомендовать тем или иным группам клиентов в рамках рекламных акций и персонализированных предложений. Кластеризация может применяться не только при проведении маркетинговых исследований, но и для решения других задач, включая распознавание образов, выявление фактов мошенничества и обработку изображений.

Один из наиболее популярных методов кластеризации — кластеризация методом *k*-средних. Будучи алгоритмом неконтролируемого обучения, этот метод предполагает разделение данных на *k* дискретных групп и является весьма эффективным способом выявления закономерностей. Примеры таких групп — это виды животных, клиенты, обладающие похожими характеристиками, и сегменты рынка недвижимости.

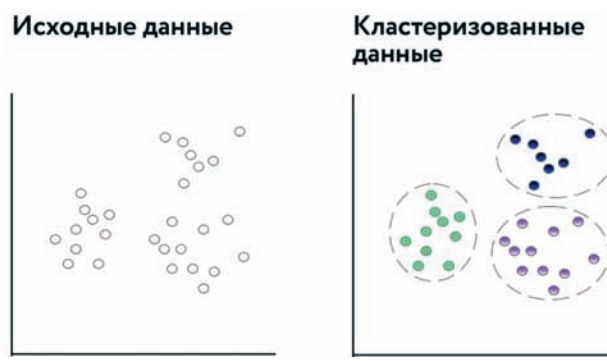


Рис. 28. Сравнение исходных данных с данными, кластеризованными методом *k*-средних

Этот алгоритм кластеризации разбивает данные на *k* кластеров. Если вы хотите разделить набор данных, например на три кластера, то *k* должно быть равно 3. На рис. 28 видно, что исходные данные были разделены на три кластера (*k* = 3). Если бы мы задали параметр *k* равным 4, то в результате получили бы четыре кластера.

Как именно происходит разделение точек данных? Первый этап предполагает изучение некластеризованных данных и выбор центроида для каждого кластера, образующий его центр тяжести.

Центроиды могут выбираться случайным образом, то есть в его качестве может выступать любая точка данных на диаграмме рассеяния. Чтобы сэкономить время, вы можете выбрать в качестве центроидов не соседние, а разбросанные по диаграмме точки. Другими словами, для начала сделайте предположение о том, где могут находиться центроиды каждого кластера. После этого оставшиеся точки данных на диаграмме будут отнесены к ближайшим центроидам путем измерения евклидова расстояния (рис. 29).

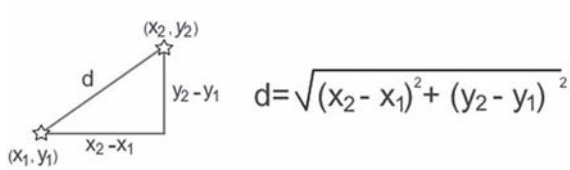


Рис. 29. Вычисление евклидова расстояния

Каждая точка данных может быть отнесена только к одному кластеру, и каждый кластер является дискретным. Это означает, что кластеры не перекрываются и не вкладываются друг в друга. Кроме того, все точки данных, включая аномалии, назначаются центроиду вне зависимости от их влияния на итоговую форму кластера. Однако из-за статистической силы, которая притягивает близлежащие точки данных к центру, кластеры обычно имеют эллиптическую или сферическую форму.

После назначения всех точек данных соответствующим центроидам выполняется агрегирование среднего значения точек в каждом кластере путем вычисления средних значений *x* и *y* содержащихся в кластере точек данных.

Затем необходимо взять среднее значение точек данных в каждом кластере и подставить значения *x* и *y* для обновления координат центроида. Скорее всего, это приведет к одному или нескольким изменениям в его (их)

расположении. Однако общее количество кластеров останется прежним, поскольку мы не создаем новые кластеры, а просто обновляем их положение на диаграмме рассеяния, и оставшиеся точки данных устремляются к ближайшему центроиду, образуя *k* кластеров.

Если в результате изменения положения центроидов какая-то из точек на диаграмме переходит в другой кластер, предыдущий шаг повторяется. Это означает, что мы снова вычисляем среднее значение кластера и обновляем значения *x* и *y* каждого центроида, чтобы отразить средние координаты точек данных, принадлежащих этому кластеру.

Когда точки данных перестанут менять кластеры после обновления координат центроида, работа алгоритма будет завершена, и вы получите окончательный набор кластеров. Весь процесс работы алгоритма представлен на следующих диаграммах (рис. 30-34).

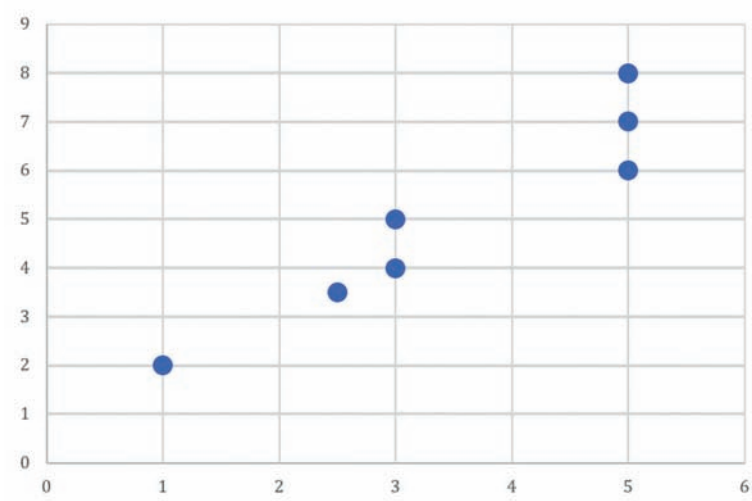


Рис. 30. Точки данных наносятся на диаграмму рассеяния

В этом примере для создания двух кластеров потребовалось выполнить две итерации. Однако метод *k*-средних не всегда позволяет точно определить окончательную комбинацию кластеров. В таких случаях необходимо сменить тактику и использовать другой алгоритм для формулирования модели классификации.

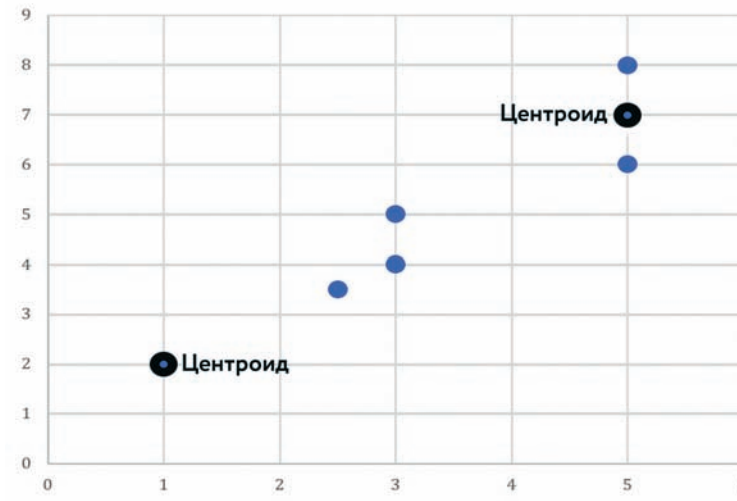


Рис. 31. Две существующие точки данных выбираются в качестве центроидов

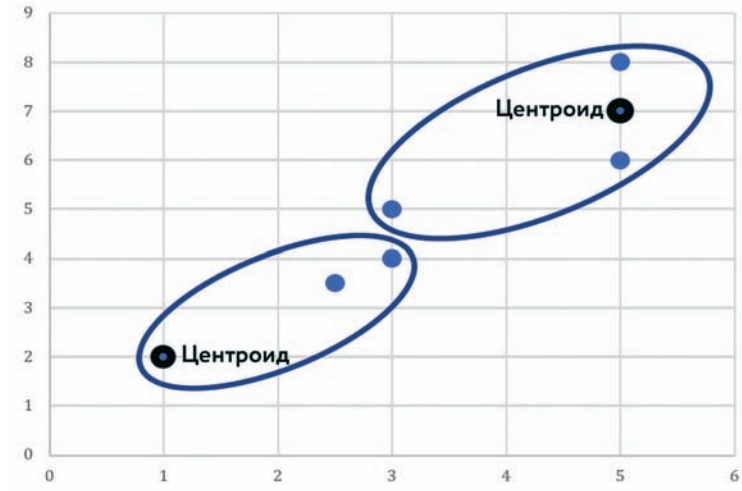


Рис. 32. После вычисления евклидова расстояния от оставшихся точек данных до центроидов формируются два кластера

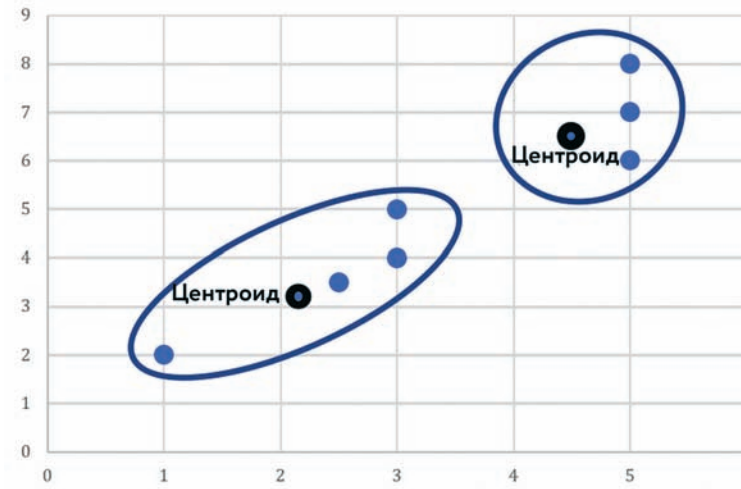


Рис. 33. Координаты центроида каждого кластера обновляются, чтобы отразить среднее значение кластера. Два предыдущих центроида остаются в исходном положении, и на диаграмму рассеяния добавляются два новых центроида. Наконец, поскольку одна точка данных перешла из правого кластера в левый, центроиды обоих кластеров должны быть еще раз обновлены.

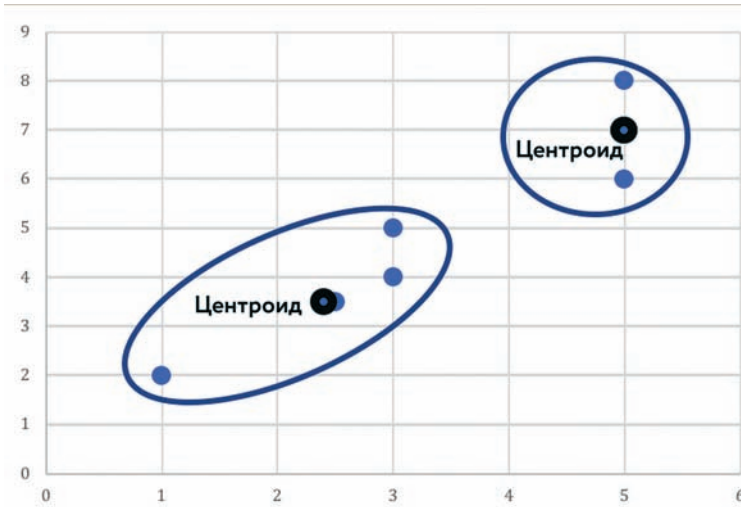


Рис. 34. На основе обновленных центроидов создаются два итоговых кластера

Также имейте в виду, что перед запуском алгоритма кластеризации методом *k*-средних вам может потребоваться выполнить стандартизацию для изменения масштаба входных признаков. Это поможет сохранить истинную форму кластеров и избежать влияния чрезмерной дисперсии на конечный результат (то есть формирования чрезмерно растянутых кластеров).

Выбор значения *k*

Перед заданием параметра *k* для выполнения кластеризации методом *k*-средних важно выбрать правильное количество кластеров. Как правило, с увеличением значения *k* кластеры становятся меньше, а дисперсия снижается. Однако обратная сторона заключается в том, что с увеличением *k* соседние кластеры становится все труднее отличить друг от друга. Если вы зададите значение *k* равным количеству точек данных в наборе, то каждая точка автоматически станет отдельным кластером. И наоборот, если задать значение *k* равным 1, то все точки данных попадут в один большой однородный кластер. Нет нужды говорить о том, что использование любого из этих крайних значений не позволит получить никакой полезной информации.

Для оптимизации значения *k* можно воспользоваться так называемым графиком «каменистой осыпи» (scree plot), который отображает изменение степени рассеяния (дисперсии) внутри кластера по мере увеличения общего числа кластеров. Эти графики, обычно содержащие выраженные перегибы, сравнивают сумму квадратов ошибок (SSE, Sum of Squared Error) для каждого числа кластеров. SSE измеряется как сумма квадратов расстояний между центроидом и другими точками внутри кластера. Если в двух словах, то значение SSE уменьшается по мере увеличения количества кластеров.

В связи с этим возникает вопрос: каково оптимальное количество кластеров? В целом следует выбирать число кластеров, соответствующее такой точке на графике, слева от которой SSE резко снижается, а справа его изменение становится пренебрежимо малым. Например, на рис. 35 наблюдается незначительное изменение SSE для четырех или более кластеров. Выбор любого из этих значений привел бы к формированию слишком маленьких и трудноразличимых кластеров.

Судя по приведенному графику, лучше всего создать два или три кластера, поскольку слева от этих двух точек наблюдается значительный перегиб кривой, обусловленный резким падением значения SSE, некоторое изменение которого наблюдается и справа от них. Это гарантирует то, что два

данных решения отличаются друг от друга и оказывают влияние на результат классификации данных.

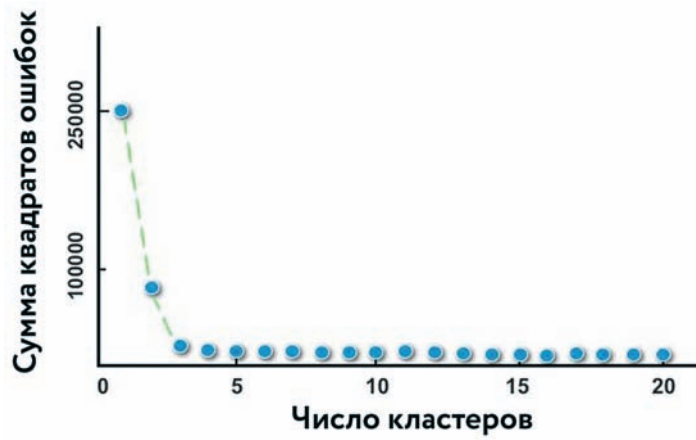


Рис. 35. График «каменистой осыпи»

Еще одна полезная техника выбора количества кластеров предполагает деление общего количества точек данных (*n*) на два и нахождение квадратного корня.

$$\sqrt{\frac{n}{2}}$$

Например, при наличии 200 точек данных рекомендуемое количество кластеров составляет 10, а при наличии 18 точек — 3.

Более простой и не требующий использования математики подход к выбору значения *k* основан на применении знаний о предметной области. Например, при анализе данных о посетителях веб-сайта крупного поставщика ИТ-услуг я, скорее всего, задам значение *k* равным 2. Почему именно два кластера? Потому что я уже знаю о существовании значительного различия в покупательском поведении постоянных и новых посетителей. Новые посетители редко приобретают ИТ-продукты и услуги корпоративного уровня, поскольку такие клиенты обычно проходят через длительный процесс исследования и проверки, прежде чем их закупка будет одобрена.

Основываясь на этих знаниях, я могу использовать метод *k*-средних для создания двух кластеров и проверки своей гипотезы. После создания

кластеров я могу выбрать один из них для дальнейшего изучения с помощью другой техники либо того же метода *k*-средних. Например, я могу разделить возвращающихся пользователей на два кластера (используя метод *k*-средних) для проверки своей гипотезы о том, что пользователи мобильных и настольных компьютеров образуют две разные группы точек данных. Опять же, имеющиеся у меня знания о предметной области говорят о том, что большие предприятия редко совершают крупные покупки через мобильные устройства, и я могу проверить это предположение с помощью кластеризации методом *k*-средних.

Если же я анализирую страницу недорогого товара, например доменного имени стоимостью 4,99 доллара, то вероятность того, что новые и возвращающиеся посетители образуют два разных кластера, будет гораздо ниже. Поскольку цена товара невысока, новые пользователи с меньшей вероятностью будут долго раздумывать перед покупкой. В данном случае я могу задать значение *k* равное 3, взяв за основу три основных источника лидов: органический трафик, платный трафик и email-маркетинг. Эти источники, скорее всего, образуют три отдельных кластера, учитывая, что:

1. **Органический трафик** обычно состоит как из новых, так и из возвращающихся клиентов, имеющих намерение совершить покупку на моем сайте (благодаря предварительному отбору, в частности сарафанному радио или предыдущему опыту покупки).
2. **Платный трафик** нацелен на новых клиентов, которые обычно приходят на сайт с более низким уровнем доверия по сравнению с органическим трафиком. К этой же категории относятся потенциальные клиенты, по ошибке нажавшие на платное рекламное объявление.
3. **Email-маркетинг** охватывает существующих клиентов, которые уже делали покупки на сайте и имеют подтвержденные учетные записи.

Это пример знаний о предметной области, свойственных моей профессии, однако имейте в виду, что эффективность таких знаний резко снижается по мере роста количества кластеров. Другими словами, знаний о предметной области может быть достаточно для определения 2–4 кластеров, но они окажутся менее ценными при выборе между большим количеством кластеров, например 20 или 21.

Контрольная работа

Ваша задача — сгруппировать набор данных об авиарейсах (совершенных в период с 1949 по 1960 год) в дискретные кластеры с помощью метода *k*-средних. Полный набор данных содержит 145 строк.

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121
5	1949	June	135
6	1949	July	148
7	1949	August	148
8	1949	September	136
9	1949	October	119

1. Опираясь только на знания о предметной области/общие знания, определите хорошее начальное число кластеров *k* для обучения модели кластеризации методом *k*-средних, предназначенной для анализа всех трех переменных.
   А) $k = 2$;
   Б) $k = 100$;
   В) $k = 12$;
   Г) $k = 3$.
2. Какой математический метод мы могли бы использовать для нахождения подходящего количества кластеров?
   А) метод локтя;
   Б) средняя абсолютная ошибка;
   В) график «каменистой осыпи»;
   Г) прямое кодирование.
3. Какая из переменных требует применения метода очистки данных?

Ответы

1. B (учитывая, что в году 12 месяцев, в количестве пассажиров, летающих каждый месяц, могут существовать некоторые повторяющиеся закономерности).
2. B.
3. Month (эту переменную необходимо преобразовать в числовой идентификатор, чтобы измерить расстояние между ней и другими переменными).

11

СМЕЩЕНИЕ И ДИСПЕРСИЯ

Выбор алгоритма — важный шаг на пути к пониманию закономерностей, существующих в ваших данных, однако разработка обобщенной модели, точно предсказывающей новые точки данных, может оказаться весьма сложной задачей. Тот факт, что большинство алгоритмов предусматривает множество различных гиперпараметров, также говорит об огромном количестве возможных результатов.

Напомним, что гиперпараметры представляют собой строки кода, которые действуют как настройки алгоритма, подобно регуляторам на приборной панели самолета или рукояткам, используемым для настройки радиочастоты (рис. 36).

```
model = ensemble.GradientBoostingRegressor(
    n_estimators = 150,
    learning_rate = 0.1,
    max_depth = 30,
    min_samples_split = 4,
    min_samples_leaf = 6,
    max_features = 0.6,
    loss = 'huber'
)
```

Рис. 36. Пример гиперпараметров алгоритма градиентного бустинга в Python

Весьма распространенные проблемы в области машинного обучения — недообучение и переобучение, которые описывают то, насколько точно ваша модель улавливает реальные закономерности, существующие в данных. Чтобы понять суть этих концепций, необходимо сначала разобраться с понятиями смещения и дисперсии.

Под смещением понимается разрыв между значением, предсказанным вашей моделью, и фактическим значением. При большом смещении прогнозы модели, скорее всего, будут отклоняться от истинных значений в определенную сторону. Дисперсия описывает степень разброса предсказанных

значений относительно друг друга. Суть концепций смещения и дисперсии визуально представлена на следующем изображении.

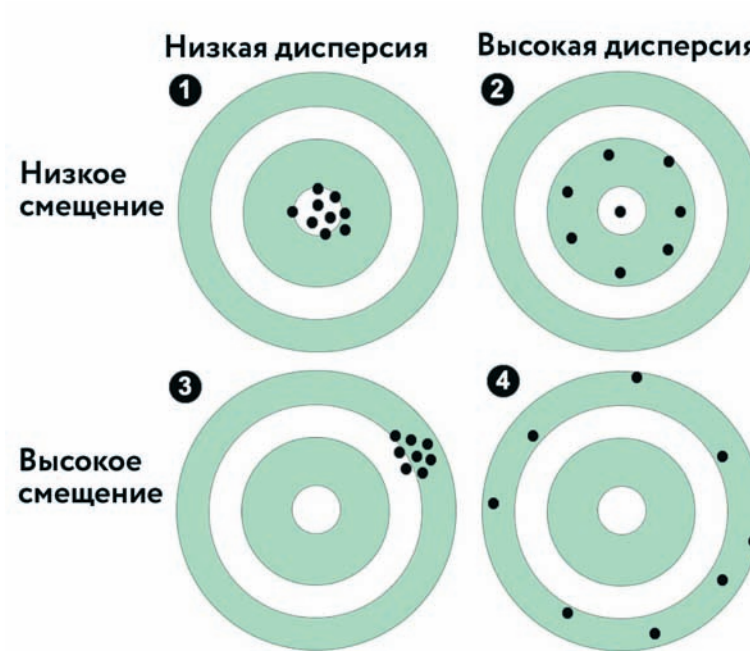


Рис. 37. Мишени для стрельбы, демонстрирующие суть смещения и дисперсии

Аналогия с мишенями, представленная на рис. 37, нечасто используется в области машинного обучения, но в данном случае вполне адекватно передает суть смещения и дисперсии[22].

Представьте, что центр мишени, или «яблочко», соответствует правильно предсказанному значению. Точки на мишени представляют собой отдельные прогнозы модели, основанные на предоставленных ей тренировочных или тестовых данных. В некоторых случаях точки расположены близко к «яблочку», свидетельствуя о том, что предсказания модели близки к фактическим значениям и отражают существующие в данных закономерности. В других случаях прогнозы модели разбросаны по мишени. Чем

[22] Prratek Ramchandani, "Random Forests and the Bias-Variance Tradeoff," *Towards Data Science*, **https://towardsdatascience.com/random-forests-and-the-bias-variance-tradeoff-3b77fee339b4**

больше они отклоняются от центра мишени, тем выше смещение и тем менее надежны предсказания модели.

Первая мишень демонстрирует пример низкого смещения и низкой дисперсии. О низком смещении говорит то, что предсказания модели сгруппированы близко к центру, а о низкой дисперсии свидетельствует их плотное скопление в одном месте.

Вторая мишень (расположенная в первом ряду справа) демонстрирует пример низкого смещения и высокой дисперсии. Хотя в данном случае прогнозы сгруппированы не так близко к «яблочку», как в предыдущем примере, они все равно находятся на довольно небольшом расстоянии от центра мишени, что свидетельствует об относительно низком смещении. Однако на этот раз дисперсия высокая, о чем говорит довольно большой разброс прогнозов.

Третья мишень (во втором ряду слева) демонстрирует высокое смещение и низкую дисперсию, а четвертая (во втором ряду справа) — высокое смещение и высокую дисперсию.

В идеале вам следует стремиться к сочетанию низкой дисперсии и низкого смещения. Однако в реальности между оптимальным смещением и оптимальной дисперсией существует компромисс. Смещение и дисперсия вносят свой вклад в ошибку, однако вам следует минимизировать именно ошибку прогнозирования, а не само смещение или дисперсию.

Как и при обучении езде на велосипеде, поиск оптимального баланса — один из наиболее сложных аспектов машинного обучения. Применить алгоритмы к данным довольно просто, гораздо труднее управлять смещением и дисперсией, поддерживая модель в сбалансированном состоянии.

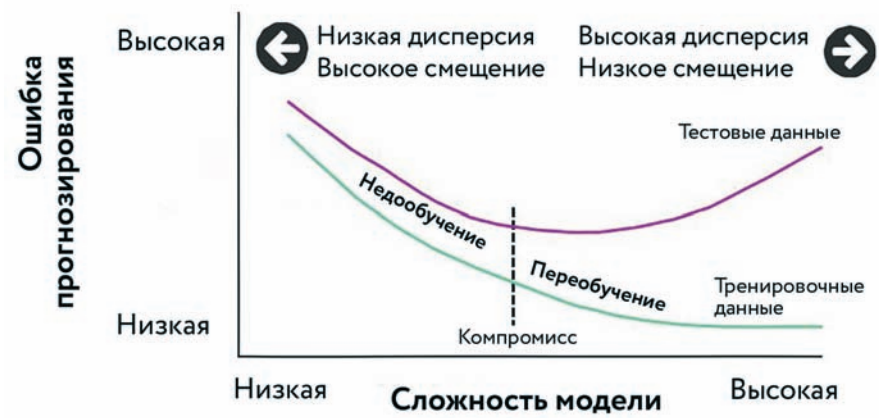


Рис. 38. Зависимость между ошибкой прогнозирования и сложностью модели

Давайте рассмотрим эту проблему на наглядном примере. На рис. 38 мы видим две кривые. Верхняя кривая соответствует тестовым данным, а нижняя — тренировочным. Обе кривые начинаются слева в точке, характеризующейся высокой ошибкой прогнозирования, обусловленной низкой дисперсией и высоким смещением. По мере движения вправо ситуация меняется на противоположную, характеризующуюся высокой дисперсией и низким смещением. При этом мы имеем низкую ошибку прогнозирования на тренировочных данных и высокую ошибку прогнозирования на тестовых данных. Середина графика соответствует точке оптимального баланса между этими ошибками и иллюстрирует компромисс между смещением и дисперсией.

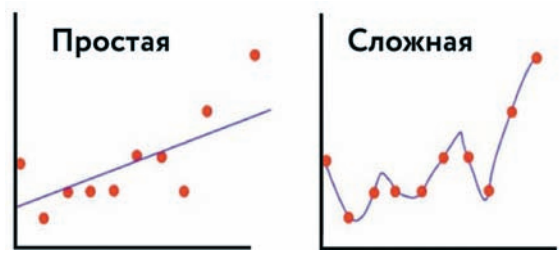


Рис. 39. Недообученная модель слева и переобученная модель справа

Неправильный компромисс между смещением и дисперсией может привести к плохим результатам. Как показано на рис. 39, модель может оказаться слишком простой и негибкой (недообученной) или слишком сложной и чрезмерно гибкой (переобученной).

На левой диаграмме показан случай недообучения (низкая дисперсия, высокое смещение), а справа — случай переобучения (высокая дисперсия, низкое смещение). Для повышения точности модели вы, скорее всего, захотите увеличить ее сложность (как показано справа), но это, в свою очередь, может привести к ее переобучению. Переобученная модель дает точные прогнозы на тренировочных данных, но оказывается менее точной при ее применении к тестовым данным. Переобучение также может иметь место в том случае, если тренировочные и тестовые данные не были рандомизированы до разделения, в результате чего существующие в данных закономерности не были равномерно распределены по этим двум выборкам.

Недообучение имеет место тогда, когда ваша модель оказывается слишком простой, из-за чего ей не удается уловить основные закономерности в данных. Это может привести к получению неточных прогнозов как на тренировочной, так и на тестовой выборке. К распространенным причинам недообучения относится недостаточное количество тренировочных данных для адекватного охвата всех возможных комбинаций, а также ситуации, когда тренировочные и тестовые данные не были должным образом рандомизированы.

Для минимизации риска недообучения и переобучения модели вы можете изменить ее гиперпараметры, чтобы гарантировать их соответствие закономерностям, которые существуют как в тренировочных, так и в тестовых данных, а не только в какой-то одной выборке. Оптимальная модель должна выявлять значительные тенденции в данных и преуменьшать или даже игнорировать их незначительные вариации. Это может потребовать выполнения повторной рандомизации тренировочных и тестовых данных, добавления новых точек данных для лучшего выявления основополагающих закономерностей или смену алгоритмов для достижения компромисса между смещением и дисперсией. Например, линейная регрессия — один из алгоритмов обучения, который редко приводит к переобучению модели (но может приводить к ее недообучению).

Переключение с линейной регрессии на нелинейную также может уменьшить смещение за счет увеличения дисперсии. В качестве альтернативы можно увеличить значение параметра *k* в модели *k*-NN, чтобы минимизировать дисперсию (благодаря усреднению большего числа соседей). Третьим вариантом может быть уменьшение дисперсии путем переключения с одного дерева решений (склонного к переобучению) на случайные леса с большим количеством таких деревьев.

Более продвинутая стратегия борьбы с переобучением заключается в выполнении так называемой регуляризации, которая снижает риск переобучения за счет ограничения степени сложности модели. По сути, этот дополнительный гиперпараметр искусственно преувеличивает ошибку смещения, штрафуя за усложнение модели, и напоминает о необходимости держать дисперсию под контролем в процессе тестирования и оптимизации других гиперпараметров.

Задание высокого значения гиперпараметра регуляризации позволяет избежать переобучения, то есть чрезмерной подгонки модели к тренировочным данным, но может привести к некоторой степени ее недообучения.

В случае линейной регрессии это может означать близкий к нулю наклон гиперплоскости, а в случае с машинами опорных векторов — слишком широкий зазор.

Наконец, еще один метод повышения точности модели — это перекрестная проверка, позволяющая минимизировать расхождения между тренировочными и тестовыми данными. О ней мы говорили в главе 6.

12

МАШИНЫ ОПОРНЫХ ВЕКТОРОВ

Разработанные специалистами по информатике в 1990-х годах машины опорных векторов (SVM, support vector machines) изначально предназначались для прогнозирования числовых и категориальных значений. Однако сегодня SVM в основном используется в качестве метода классификации для предсказания категориальных значений.

Как метод классификации, SVM напоминает логистическую регрессию, поскольку используется для фильтрации данных в бинарную или многоклассовую целевую переменную. Однако, как видно на рис. 40, метод опорных векторов ставит другой акцент при определении местоположения граничной линии классификации.

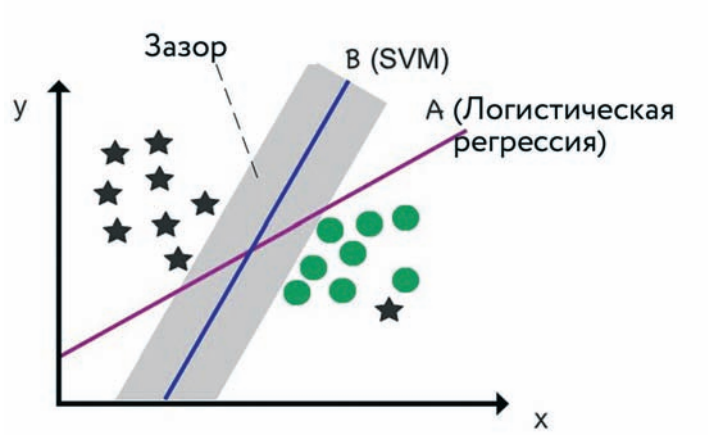


Рис. 40. Сравнение логистической регрессии и метода опорных векторов

Диаграмма рассеяния на рис. 40 содержит 17 точек данных, которые являются линейно разделимыми. Мы видим, что граница принятия решения модели логистической регрессии (A) разделяет точки данных на два класса так, чтобы расстояние между всеми точками данных и границей оказалось

минимальным. Вторая линия, граница SVM (B), тоже разделяет точки на два класса, но при этом максимизирует расстояние между собой и точками данных.

Серым цветом обозначен так называемый зазор (margin), который представляет собой расстояние между границей принятия решения и ближайшей точкой данных, умноженное на два. Зазор — ключевой элемент SVM, обеспечивающий дополнительную опору, которая позволяет справиться с новыми точками данных, способными нарушить границу принятия решения (как в случае с логистической регрессией). Чтобы это проиллюстрировать, добавим на ту же диаграмму рассеяния новую точку данных (рис. 41).

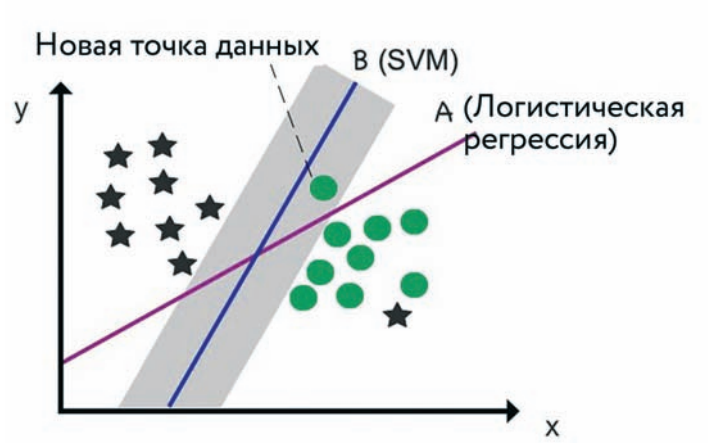


Рис. 41. Добавление новой точки данных на диаграмму рассеяния

Новая точка данных представляется собой круг, но оказывается неправильно расположенной по левую сторону от границы принятия решения модели логистической регрессии (A) (предназначенной для звездочек). Однако эта новая точка данных оказывается правильно расположенной по правую сторону от границы принятия решения SVM (B) благодаря зазору.

Метод опорных векторов также бывает полезен для распутывания сложных взаимосвязей и уменьшения влияния со стороны выбросов и аномалий. Ограничение стандартной логистической регрессии заключается в том, что она старается учесть все выбросы и аномалии (как видно на примере со звездой в правом нижнем углу на рис. 42). Однако метод SVM менее

чувствителен к таким точкам данных и старается минимизировать их влияние на итоговое местоположение граничной линии. На рис. 42 видно, что линия B (SVM) менее чувствительна к аномально расположенной звездочке в правой части диаграммы. Таким образом, SVM можно использовать в качестве метода управления вариациями в данных.

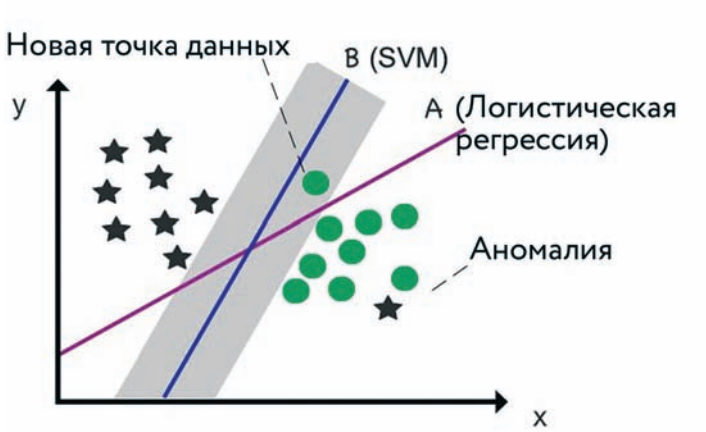


Рис. 42. Уменьшение влияния аномалий

Границу SVM также можно модифицировать с целью игнорирования неверно классифицированных случаев в тренировочных данных, используя гиперпараметр C. Суть машинного обучения обычно сводится к обобщению закономерностей, а не к точной расшифровке тренировочных данных (которые всегда в некоторой степени зашумлены[23]), поскольку некоторые ошибки, полученные при обучении модели, могут позволить ей в дальнейшем гораздо лучше обобщать реальные данные. Поэтому в случае SVM существует компромисс между широким зазором/бóльшим количеством ошибок и узким зазором/меньшим количеством ошибок. Таким образом, главная цель — найти оптимальный уровень строгости, и, изменяя гиперпараметр C, вы можете регулировать то, в какой степени будут игнорироваться неверно классифицированные случаи (то есть случаи, находящиеся с неправильной стороны границы принятия решения).

[23] Шум — это случайная и/или бесполезная информация, которая искажает истинное значение данных.

Увеличение гибкости модели с помощью гиперпараметра C создает так называемый «мягкий зазор», который улучшает обобщающую способность модели за счет игнорирования определенной доли неверно классифицированных случаев. Зазор оказывается широким или мягким при задании низкого значения гиперпараметра C. При C равном 0 неправильно классифицированные случаи вообще не штрафуются. И наоборот, большое значение C[24] делает стоимость неправильной классификации более высокой, тем самым сужая зазор (то есть делая его жестким) в целях предотвращения этой неверной классификации. Это может привести к переобучению модели, то есть к ее чрезмерной подгонке к тренировочной выборке, из-за чего она впоследствии может неправильно классифицировать новые точки данных (рис. 43).

Вы можете бороться с переобучением (при котором модель хорошо работает на тренировочной выборке, но ошибается на новых данных) путем уменьшения значения C, то есть усиления степени регуляризации модели. Оптимальное значение C, как правило, подбирается экспериментально, методом проб и ошибок, который может быть автоматизирован с помощью так называемого поиска по решетке (который мы обсудим в главе 18).

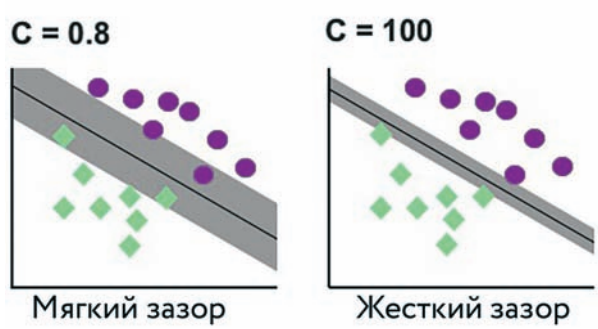


Рис. 43. Сравнение мягкого и жесткого зазора

До сих пор мы рассматривали примеры с двумя признаками, нанесенными на двумерную диаграмму рассеяния, однако истинная мощь SVM

[24] В библиотеке Scikit-learn значение гиперпараметра C по умолчанию равно 1,0, а степень регуляризации (штраф за переобучение) обратно пропорциональна значению C. Это означает, что любое значение, меньшее 1,0, усиливает степень регуляризации модели, а штраф равен квадрату L2 (L2 вычисляется путем извлечения квадратного корня из суммы квадратов элементов вектора).

проявляется в работе с высокоразмерными данными и большим количеством признаков. У SVM есть множество усовершенствованных вариаций, позволяющих классифицировать данные высокой размерности с помощью так называемого ядерного трюка. Это передовое решение позволяет отобразить данные из низкоразмерного в высокоразмерное пространство, когда набор данных не может быть разделен с помощью линейной границы принятия решения в исходном пространстве. Например, переход от двумерного к трехмерному пространству позволяет нам использовать линейную плоскость для разделения данных в трехмерной области. Другими словами, ядерный трюк позволяет классифицировать точки данных с нелинейными характеристиками с помощью линейного классификатора в более высоком измерении (рис. 44).

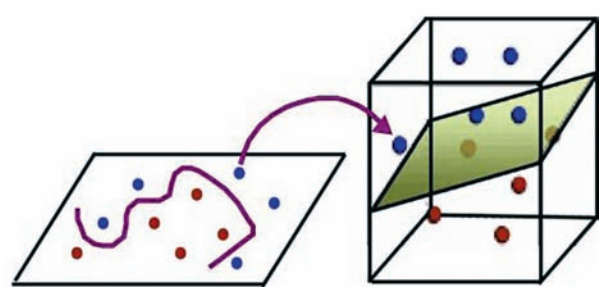


Рис. 44. В этом примере граница принятия решения представляет собой нелинейный разделитель точек данных в двумерном пространстве, который превращается в линейный разделитель при проецировании в трехмерное пространство.

При использовании SVM следует помнить о том, что этот инструмент может быть чувствителен к масштабу признаков, поэтому перед обучением модели вам может потребоваться масштабировать данные[25]. С помощью стандартизации вы можете преобразовать диапазон значений каждого признака в стандартное нормальное распределение со средним значением, равным нулю. В библиотеке Scikit-learn стандартизация реализуется

[25] Хорошей практикой считается двукратное выполнение процесса обучения — со стандартизацией и без нее — и последующее сравнение производительности двух моделей.

с помощью инструмента StandardScaler, документацию по которому можно найти на странице: **http://bit.ly/378pf9Q**.

Наконец, к недостаткам SVM можно отнести большую длительность процесса обучения модели по сравнению с логистической регрессией и другими алгоритмами классификации. В частности, из-за ограниченной скорости и производительности SVM не рекомендуется применять к наборам данных, содержащим малое количество признаков по отношению к строкам. Тем не менее SVM отлично справляется с отсеиванием выбросов из сложных наборов данных малого и среднего размера, а также с управлением высокоразмерными данными.

Контрольная работа

Используя SVM-классификатор, определите исходный остров обитания пингвинов, прибывших на ваш остров. Для получения прогноза вы можете использовать любую или сразу все переменные из набора данных о пингвинах.

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE
6	Adelie	Torgersen	38.9	17.8	181.0	3625.0	FEMALE
7	Adelie	Torgersen	39.2	19.6	195.0	4675.0	MALE
8	Adelie	Torgersen	34.1	18.1	193.0	3475.0	NaN
9	Adelie	Torgersen	42.0	20.2	190.0	4250.0	NaN

1. Какая из следующих переменных могла бы использоваться в качестве зависимой переменной в этой модели?
   А) island;
   Б) species;
   В) sex;
   Г) body_mass_g.

2. Какие из следующих переменных мы могли бы использовать в качестве независимых переменных?
   А) island;
   Б) все переменные;
   В) все переменные, кроме island;
   Г) species.
3. Какие два метода очистки данных обычно используются с этим алгоритмом?

Ответы

1. А.
2. В.
3. Регуляризация и стандартизация.

13

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

В этой предпоследней главе, посвященной алгоритмам машинного обучения, мы поговорим об искусственных нейронных сетях (ИНС), которые подводят нас к теме обучения с подкреплением. Искусственные нейронные сети, или просто нейронные сети, — это популярный метод машинного обучения, который предполагает выполнение анализа данных с помощью сети слоев принятия решений. Название этой техники обусловлено структурным сходством алгоритма с человеческим мозгом. Хотя это не означает, что искусственные нейронные сети виртуальным образом воспроизводят процесс принятия решений в мозге, некоторые общие черты все же существуют.

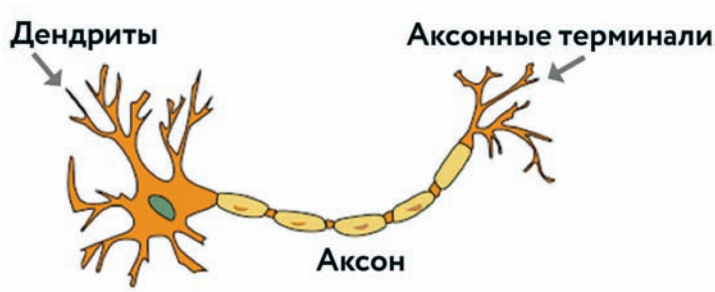


Рис. 45. Анатомия нейрона человеческого мозга

Например, мозг состоит из взаимосвязанных нейронов с дендритами, которые воспринимают входные импульсы (рис. 45). На основе этих импульсов нейрон производит электрический сигнал, который распространяется по аксону и передается через аксонные терминали другим нейронам. Аналогичным образом искусственные нейронные сети состоят из взаимосвязанных функций принятия решений, называемых узлами, которые взаимодействуют друг с другом через аксоноподобные ребра.

Нейронная сеть состоит из организованных в слои узлов и обычно имеет широкое основание. Первый слой состоит из разделенных на узлы необработанных входных данных (таких как числовые значения, текст, пиксели изображения или звук). Каждый входной узел передает информацию узлам следующего слоя через ребра сети (рис. 46).

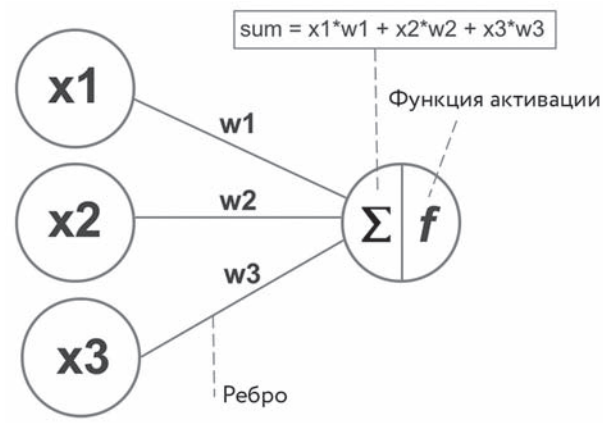


Рис. 46. Узлы, ребра/веса и сумма/функция активации простейшей нейронной сети

Каждое ребро в сети имеет числовой вес, который может изменяться на основе полученного опыта. Если сумма взвешенных сигналов связанных ребер превышает заданное пороговое значение, известное как функция активации, это приводит к активации нейрона в следующем слое. Если эта сумма не превышает порогового значения, функция активации не срабатывает, то есть схема работает по принципу «все или ничего». Более того, веса, присвоенные каждому ребру, уникальны, а это означает, что узлы активируются по-разному, что не позволяет им выдавать одинаковые решения.

При использовании методов контролируемого обучения прогноз модели сравнивается с фактическим выходом (который заведомо верен), и разница между этими двумя результатами представляет собой так называемые затраты. Цель процесса обучения — уменьшение затрат до тех пор, пока предсказание модели не совпадет с правильным выходом. Это достигается путем постепенной настройки весов сети вплоть до получения минимально возможного значения затрат. Этот особый процесс обучения нейронной сети называется методом обратного распространения ошибки. Вместо того чтобы двигаться слева направо, подобно распространяющимся по сети

сигналам, в данном случае процесс идет в обратном направлении — от выходного слоя справа к входному слою слева.

Дилемма «черного ящика»

Один из недостатков модели, созданной на основе сети, — это дилемма «черного ящика». Несмотря на то что сеть способна аппроксимировать точные результаты, отслеживание процесса принятия решений позволяет получить лишь ограниченное представление о том, как конкретные переменные влияют на выдаваемый ею ответ. Например, если мы используем нейронную сеть для прогнозирования результатов кампании на *Kickstarter* (онлайн-платформе для финансирования творческих проектов), сеть может проанализировать множество независимых переменных, включая категорию кампании, валюту, срок реализации проекта, минимальную сумму залога и т. д. Однако модель не может определить связь этих независимых переменных с зависимой переменной, то есть достижением целевого уровня финансирования проекта в рамках проводимой кампании. Между тем такие алгоритмы, как деревья решений и линейная регрессия, вполне прозрачны, поскольку показывают взаимосвязь переменных с заданным результатом. Более того, две нейронные сети с разными топологиями и весами способны выдать одинаковый результат, что еще больше усложняет задачу отслеживания влияния конкретных переменных на выдаваемый сетями ответ.

В связи с этим возникает вопрос: когда следует использовать нейронную сеть (учитывая то, что она представляет собой своего рода «черный ящик»)? Нейронные сети подходят для решения задач прогнозирования с большим количеством входных признаков и сложных закономерностей, особенно для задач, которые представляют большую трудность для компьютера, но с легкостью решаются человеком. Примером такой задачи может служить разгадывание капчи (CAPTCHA, Completely Automated Public Turing test to tell Computers and Humans Apart, полностью автоматизированный публичный тест Тьюринга для различения компьютеров и людей), который позволяет определить, является ли пользователь веб-сайта человеком. Или, например, определение того, собирается ли пешеход выйти на дорогу перед движущимся автомобилем. В обоих случаях получение быстрого и точного прогноза оказывается важнее расшифровки конкретных переменных и их взаимосвязи с конечным результатом.

Построение нейронной сети

Типичная нейронная сеть состоит из входного, скрытого и выходного слоев. Сначала данные поступают на входной слой, где происходит обнаружение признаков. Затем скрытый слой (слои) анализирует и обрабатывает эти входные признаки, а выходной слой отображает конечный результат (рис. 47).

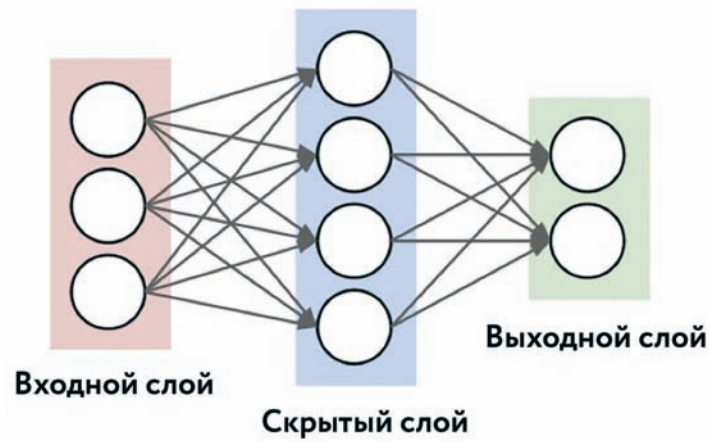


Рис. 47. Три основных слоя нейронной сети

Внутренние слои считаются скрытыми, поскольку, подобно человеческому глазу, в скрытом режиме обрабатывают объекты между входным и выходным слоями. Когда перед нами оказываются четыре линии, соединенные в квадрат, наши глаза мгновенно распознают их как квадрат. Мы не отслеживаем процесс ментальной обработки, которая необходима для регистрации четырех линий (вход) в качестве квадрата (выход).

Нейронные сети работают аналогичным образом: они разбивают данные и обрабатывают их в скрытых слоях для получения конечного результата. Добавление в сеть большего количества скрытых слоев улучшает способность модели анализировать сложные закономерности. Именно поэтому модели с большим количеством слоев часто называют моделями глубокого обучения[26], чтобы подчеркнуть их более выдающиеся способности в плане обработки данных.

[26] В 2006 году Джеффри Хинтон в соавторстве с другими специалистами опубликовал работу, посвященную распознаванию рукописных цифр с помощью глубокой нейронной сети, в которой был введен термин «глубокое обучение».

Хотя существует множество методов организации узлов нейронной сети, самой простой структурой обладает сеть прямого распространения, в которой сигналы идут только в одном направлении, а соединения между узлами не образуют цикл. Легкий пример такой нейронной сети — перцептрон, разработанный в 1950-х годах профессором Фрэнком Розенблаттом.

Рис. 48. Визуальное представление перцептрона

Перцептрон был разработан в качестве функции принятия решений для получения входных данных и генерации двоичного выхода. Он состоит из одного или нескольких входов, процессора и одного выхода. Входные данные подаются в процессор (нейрон) и обрабатываются, после чего генерируется выход (рис. 48).

Перцептрон поддерживает два возможных выходных значения — 0 и 1. Выход, равный 1, запускает функцию активации, а выход, равный 0, — нет. При работе с более крупной нейронной сетью, содержащей дополнительные слои, выход «1» может использоваться для передачи сигнала на следующий слой, а выход «0» — для его игнорирования.

Как метод контролируемого обучения, перцептрон строит предсказательную модель на основе следующих пяти шагов:

1. Входные сигналы поступают в процессор.
2. Перцептрон применяет весовые коэффициенты для оценки значения этих входов.
3. Перцептрон вычисляет величину ошибки, то есть разницу между оценкой и фактическим значением.
4. Перцептрон корректирует свои веса в соответствии с вычисленной ошибкой.
5. Предыдущие четыре шага повторяются до тех пор, пока точность модели не достигнет желаемого уровня. Затем полученная модель обучения может быть применена к тестовым данным.

Чтобы проиллюстрировать этот процесс, давайте представим, что у нас есть перцептрон, состоящий из двух входов:

Вход 1: x1 = 24

Вход 2: x2 = 16

Мы добавляем к этим двум входам случайные веса, после чего они передаются в нейрон для обработки (рис. 49).

Рис. 49. Добавление весовых коэффициентов в перцептрон

Веса

Вход 1: 0,5

Вход 2: –1

Затем мы перемножаем веса с соответствующими входными значениями:

Вход 1: 24 × 0,5 = 12

Вход 2: 16 × (–1) = –16

Хотя перцептрон выдает двоичный выход (0 или 1), существует множество способов настройки функции активации. В этом примере мы используем функцию активации ⩾ 0. Это означает, что, если сумма положительна или равна нулю, то на выходе мы получим 1. Если же сумма отрицательна, то на выходе мы получим 0 (рис. 50).

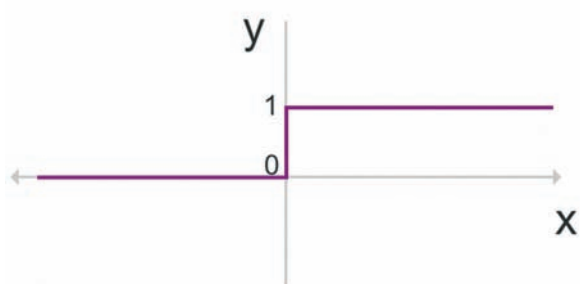


Рис. 50. Функция активации, где выход (y) равен 0 при отрицательном значении *x*, и выход (y) равен 1 при положительном значении *x*

Таким образом:
Вход 1: 24 × 0,5 = 12
Вход 2: 16 × (–1.0) = –16
Сумма (∑): 12 + (–16) = –4

Поскольку в данном случае числовое значение меньше нуля, результатом стал «0», и функция активации перцептрона не срабатывает. Учитывая эту ошибку, перцептрону необходимо скорректировать свои весовые коэффициенты.

Обновленные веса:
Вход 1: 24 × 0,5 = 12
Вход 2: 16 × (–0,5) = –8
Сумма (∑): 12 + (–8) = 4

Поскольку результат положительный, перцептрон выдает значение «1», что запускает функцию активации, а в случае более крупной сети происходит передача сигнала на следующий слой для его дальнейшего анализа (рис. 51).

В этом примере использовалась функция активации ⩾ 0. Однако мы могли бы изменить порог активации в соответствии с другим правилом, например:

x > 3, y = 1
x ⩽ 3, y = 0

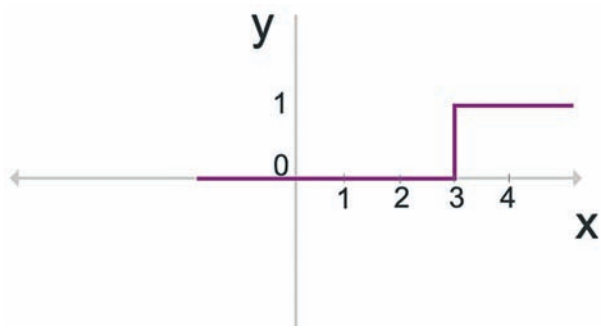


Рис. 51. Функция активации, где выход (y) равен 0 при x ⩽ 3,
и выход (y) равен 1 при *x* > 3

Недостаток перцептрона в том, что, поскольку выходной сигнал двоичный (0 или 1), небольшие изменения в весах или смещении любого отдельного перцептрона в более крупной нейронной сети способны поляризовать результаты. Это может вызвать значительные изменения в сети и обратить конечный результат, что затрудняет получение модели, способной выдавать точный ответ при работе с новыми данными.

Альтернатива перцептрона — сигмоидальный нейрон, который похож на перцептрон, но вместо бинарного фильтра использует сигмоидальную функцию, благодаря чему выдает любое значение в диапазоне от 0 до 1. Это позволяет более гибко реагировать на небольшие изменения весовых коэффициентов ребер без обращения результатов, поскольку выходное значение больше не является двоичным. Другими словами, выход не изменяется на противоположный из-за незначительной корректировки веса ребра или входного значения.

Хотя сигмоидальный нейрон более гибок по сравнению с перцептроном, он не способен генерировать отрицательные значения. Поэтому третий вариант — функция гиперболического тангенса (рис. 52).

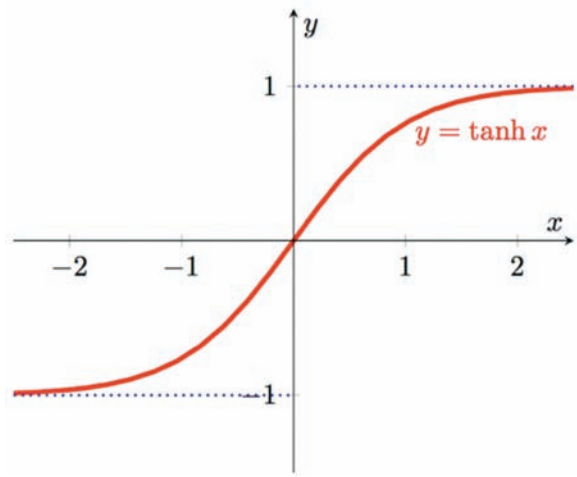


Рис. 52. График функции гиперболического тангенса

До сих пор мы обсуждали простые нейронные сети; для создания более продвинутой нейросети мы можем объединить сигмоидальные нейроны с другими классификаторами, получив сеть с большим количеством

слоев, или объединить несколько перцептронов в так называемый многослойный перцептрон.

Многослойные перцептроны

Многослойный перцептрон (MLP, multilayer perceptron), как и другие методы на основе ИНС, представляет собой алгоритм для предсказания категориальной (классификация) или непрерывной (регрессия) целевой переменной. Мощь многослойных перцептронов обусловлена их способностью объединять несколько моделей в единую предсказательную модель, как показано на рис. 53.

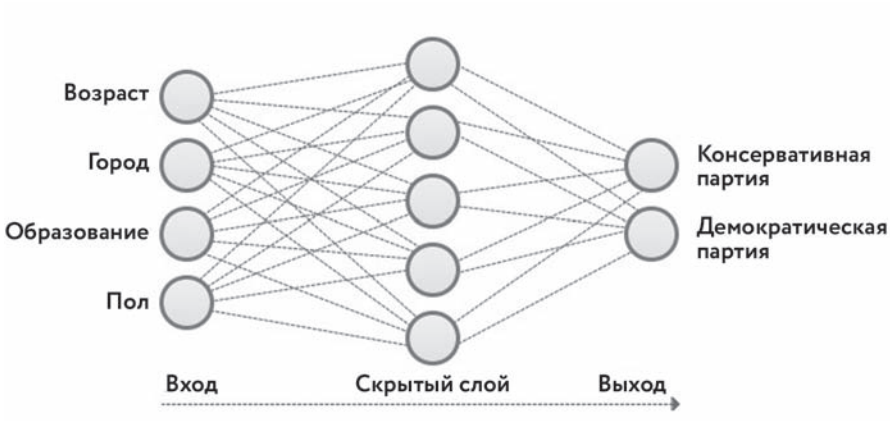


Рис. 53. Многослойный перцептрон, предназначенный для классификации пользователей социальных сетей по политическим предпочтениям

В этом примере MLP-модель разделена на три слоя. Входной слой состоит из четырех узлов, представляющих входные признаки, используемые для предсказания политических предпочтений пользователя социальных сетей: Возраст, Город, Образование и Пол. Затем к каждой входной переменной применяется функция для создания нового слоя узлов, называемого средним или скрытым слоем. Каждый узел скрытого слоя представляет собой функцию, например сигмоидальную, предусматривающую свои уникальные веса/гиперпараметры. Это означает, что каждая входная переменная, по сути, подвергается воздействию пяти различных функций. В свою очередь узлы скрытого слоя одновременно имеют дело со всеми четырьмя признаками.

Выходной слой этой модели состоит из двух дискретных результатов: Консервативная партия и Демократическая партия, которые отражают вероятные политические предпочтения конкретного пользователя. Обратите внимание на то, что количество узлов в каждом слое зависит от количества входных признаков и целевой(ых) переменной(ых).

В целом многослойные перцептроны идеально подходят для интерпретации больших и сложных наборов данных при отсутствии ограничений в плане временны́х и вычислительных ресурсов. Менее ресурсоемкие алгоритмы, такие как деревья решений и логистическая регрессия, более эффективны при работе с небольшими наборами данных. Учитывая многочисленность гиперпараметров, многослойные перцептроны также требуют больше времени и усилий для настройки по сравнению с другими алгоритмами. Что касается времени обработки, то многослойный перцептрон работает дольше, чем большинство алгоритмов поверхностного обучения, включая логистическую регрессию, но в целом быстрее по сравнению с SVM.

Глубокое обучение

Для анализа менее сложных закономерностей можно использовать простой многослойный перцептрон или такие альтернативные алгоритмы классификации, как логистическая регрессия и метод *k*-ближайших соседей. Однако по мере усложнения содержащихся в данных закономерностей, особенно когда речь заходит о большом количестве входов, например пикселей изображения, модель неглубокого обучения перестает быть надежной и способной к сложному анализу, поскольку увеличение количества входов сопровождается экспоненциальным ростом сложности модели. Для интерпретации большого количества входных признаков и разбиения сложных закономерностей на более простые может быть использована нейронная сеть с большим количеством слоев, как показано на рис. 54.

Эта глубокая нейронная сеть для распознавания лиц использует ребра для обнаружения различных физических характеристик, например диагональных линий. Затем результаты работы узлов объединяются подобно строительным блокам, что позволяет классифицировать входные данные, например, лицо человека или морду кошки, после чего сеть приступает к распознаванию индивидуальных особенностей. Этот процесс называется глубоким обучением. «Глубоким» его делает использование по меньшей мере 5–10 слоев, состоящих из узлов.

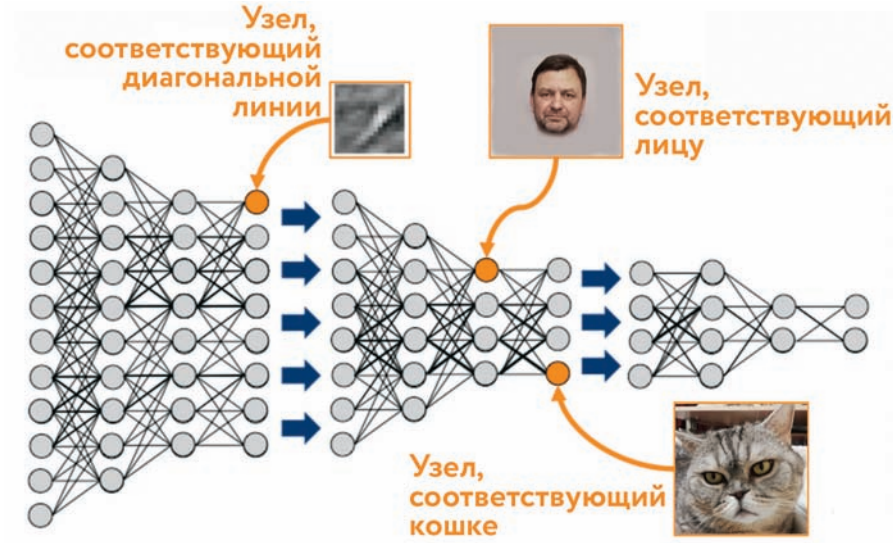


Рис. 54. Распознавание лиц с помощью модели глубокого обучения. Источник: kdnuggets.com

Табл. 13. Типичные сценарии использования различных методов глубокого обучения

	Рекуррентная нейронная сеть	Тензорная рекурсивная нейронная сеть	Сеть глубоких убеждений	Сверточная нейронная сеть	Многослойный перцептрон
Обработка текста	*V*	*V*		*V*	
Распознавание изображений			*V*	*V*	
Распознавание объектов		*V*		*V*	
Распознавание речи	*V*				
Анализ временны́х рядов	*V*				
Классификация			*V*	*V*	*V*

Один из самых распространенных примеров моделей глубокого обучения — это система распознавания объектов в беспилотных автомобилях, предназначенная для распознавания пешеходов и других транспортных средств, в которой используется более 150 слоев. К другим областям применения методов глубокого обучения относится анализ временны́х рядов, направленный на изучение динамики изменения тех или иных показателей за определенные периоды времени или интервалы, распознавание речи и задачи обработки текста, включая анализ настроений, тематическую сегментацию и распознавание именованных сущностей. Другие сценарии и часто применяющиеся в них методы глубокого обучения перечислены в табл. 13.

Как следует из этой таблицы, многослойные перцептроны практически полностью вытеснены такими новыми методами глубокого обучения, как сверточные сети, рекуррентные сети, сети глубоких убеждений и тензорные рекурсивные нейронные сети. Эти продвинутые версии нейросетей могут эффективно использоваться для решения ряда актуальных задач. Самый популярный и мощный из методов глубокого обучения в настоящее время — это сверточные сети, однако благодаря непрерывному развитию в этой сфере постоянно появляются новые методы и вариации.

Контрольная работа

Используя многослойный перцептрон, создайте модель для классификации пингвинов, пострадавших и спасенных во время стихийного бедствия, по полу (male — самец или female — самка). Для обучения модели вы можете использовать только физические характеристики пингвинов. Обратите внимание на то, что этот набор данных содержит 344 строки.

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE
6	Adelie	Torgersen	38.9	17.8	181.0	3625.0	FEMALE
7	Adelie	Torgersen	39.2	19.6	195.0	4675.0	MALE
8	Adelie	Torgersen	34.1	18.1	193.0	3475.0	NaN
9	Adelie	Torgersen	42.0	20.2	190.0	4250.0	NaN

1. Сколько выходных узлов необходимо включить в многослойный перцептрон для прогнозирования зависимой переменной sex (пол)?
2. Какие из семи переменных мы могли бы использовать в качестве независимых переменных, чтобы обучить модель на основе одних лишь физических характеристик пингвина?
3. Какой более прозрачный алгоритм классификации мы могли бы использовать вместо многослойного перцептрона?
   А) Простая линейная регрессия.
   Б) Логистическая регрессия.
   В) Кластеризация методом *k*-средних.
   Г) Множественная линейная регрессия.

Ответы

1. 2 узла (male и female).
2. bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g.
3. А.

14

ДЕРЕВЬЯ РЕШЕНИЙ

Идея о том, что искусственные нейронные сети могут применяться для решения более широкого спектра задач обучения по сравнению с другими методами, побудила некоторых экспертов провозгласить ИНС самым совершенным алгоритмом машинного обучения. Хотя у этого утверждения есть веские основания, это не означает, что ИНС представляет собой универсальный алгоритм. В некоторых случаях нейронные сети не справляются с поставленной задачей, и тогда в качестве альтернативы обычно выступают деревья решений.

Огромный объем входных данных и вычислительных ресурсов, необходимый для обучения нейронной сети, — это основной недостаток при попытке решить все задачи машинного обучения с помощью этой техники. Приложения на основе нейросетей, такие как система распознавания изображений Google, опираются на миллионы размеченных примеров для классификации простых объектов (например, собак), и далеко не каждая организация располагает ресурсами, необходимыми для создания и поддержания работы таких масштабных моделей. Другой важный недостаток нейросетей — дилемма «черного ящика», то есть невозможность понять структуру реализуемого моделью процесса принятия решений. С другой стороны, деревья решений отличаются прозрачностью, и их довольно легко интерпретировать. Они способны работать с меньшим количеством данных и потребляют меньше вычислительных ресурсов. Эти преимущества делают деревья решений популярной альтернативой развертыванию нейронной сети в менее сложных случаях.

Деревья решений в основном используются для решения задач классификации, но могут применяться и в качестве регрессионной модели для прогнозирования числовых результатов. Деревья классификации предсказывают категориальные результаты, используя в качестве входных данных числовые и категориальные переменные, в то время как деревья регрессии предсказывают числовые результаты на основе тех же входных данных. Деревья решений могут применяться в самых разных сценариях, начиная

с выбора получателя стипендии и заканчивая прогнозированием объема продаж интернет-магазина и подбором сотрудников (рис. 55, 56).

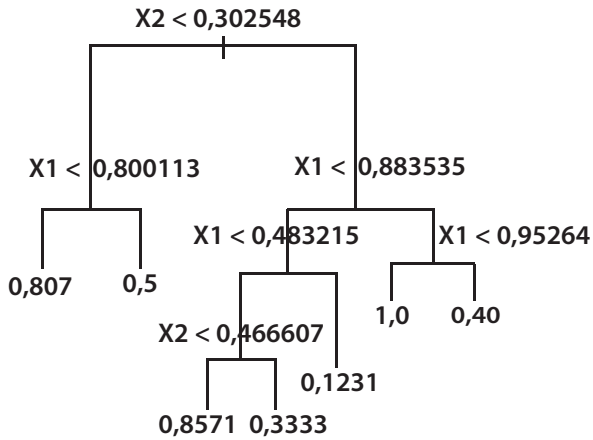


Рис. 55. Пример дерева регрессии

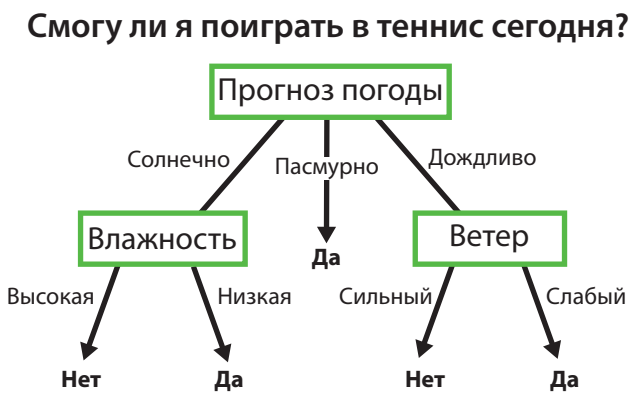


Рис. 56. Пример дерева классификации

Привлекательность деревьев решений отчасти объясняется тем, что их можно представить в графическом виде и легко объяснить неспециалистам. Например, когда клиент спрашивает, почему ему не одобрили кредит на покупку жилья, вы можете показать ему дерево, демонстрирующее процесс принятия решения, что было бы невозможно при использовании метода типа «черного ящика».

Построение дерева решений

Построение дерева решений начинается с корневого узла, от которого отходят ветви, также известные как ребра. В свою очередь ветви соединяются с листьями, также известными как узлы — это точки принятия решений. Процесс повторяется с использованием точек данных, берущихся из каждого нового листа. Окончательная категоризация выполняется тогда, когда лист перестает генерировать новые ветви и образует так называемый конечный узел.

Начиная с корневого узла, деревья решений анализируют данные, разбивая их на подмножества и создавая узел для каждого значения переменной (например: солнечно, пасмурно, дождливо). При этом цель состоит том, чтобы сделать дерево как можно более компактным. Ее можно достичь, выбирая переменную, которая обеспечивает оптимальное разбиение данных на однородные группы, при котором уровень энтропии данных в следующей ветви оказывается минимальным.

Энтропия — это математическая концепция, которая объясняет меру дисперсии в данных между различными классами. Проще говоря, нам нужно, чтобы данные на каждом уровне были более однородными, чем на предыдущем. Поэтому мы стремимся использовать «жадный» алгоритм, способный уменьшить энтропию на каждом уровне дерева. Например, итеративный дихотомизатор 3 (ID3, Iterative Dichotomizer), изобретенный Джоном Россом Куинланом. Это одна из трех реализаций дерева решений, разработанных Куинланом; отсюда и цифра 3 в названии. На каждом уровне алгоритм ID3 определяет переменную (преобразованную в бинарный вопрос), которая обеспечивает наименьшую энтропию на следующем уровне.

Чтобы понять, как это работает, рассмотрим следующий пример.

Табл. 14. Характеристики сотрудников

Сотрудники	Превышение КПЭ	Способность к лидерству	Возраст < 30 лет	Результат
6	6	2	3	Получили повышение
4	0	2	4	Не получили повышения

В табл. 14 мы видим десять сотрудников, три входные переменные (Превышение КПЭ (ключевые показатели эффективности), Способность к лидерству, Возраст < 30 лет) и одну выходную (Результат). Нам нужно предсказать, получит ли сотрудник повышение по службе, опираясь на оценку входных переменных.

Сначала разделим данные на основе переменной 1 (Превышение КПЭ).

- Шесть сотрудников превысили свои КПЭ и получили повышение (Да).
- Четыре сотрудника не превысили свои КПЭ и не получили повышения (Нет).

Переменная 1 создает две однородные группы на следующем уровне дерева.

Черные = Получили повышение, Белые = Не получили повышения

Теперь попробуем переменную 2 (Способность к лидерству), которая дает следующий результат.

- Два продвинутых по службе сотрудника обладают способностью к лидерству (Да).
- Четыре продвинутых по службе сотрудника не обладают способностью к лидерству (Нет).
- Два сотрудника, обладающих способностью к лидерству, не получили повышения (Да).

— Два сотрудника, не обладающих способностью к лидерству, не получили повышения (Нет).

Переменная 2 создает две смешанные группы точек данных.

Черные = Получили повышение, Белые = Не получили повышения

Наконец, у нас есть переменная 3 (возраст до 30 лет), которая дает следующий результат.

— Три сотрудника младше 30 лет получили повышение (Да).
— Три сотрудника старше 30 лет получили повышение (Нет).
— Четыре сотрудника младше 30 лет не получили повышения (Да).

Переменная 3 создает одну однородную группу и одну смешанную группу точек данных.

Черные = Получили повышение, Белые = Не получили повышения

Из этих трех переменных переменная 1 обеспечивает наилучшее разделение данных на две однородные группы. Переменная 3 дает второй лучший результат, образуя один однородный лист. Переменная 2 создает два неоднородных листа. Таким образом, переменная 1 будет выбрана в качестве первого бинарного вопроса для разделения этого набора данных.

Вне зависимости от используемого алгоритма процесс разделения данных на подразделы, известный как рекурсивное разбиение, повторяется вплоть до достижения точки остановки, которая может быть основана на таких критериях:

- все листья содержат менее 3–5 элементов;
- ветвь дает результат, который помещает все элементы в один двоичный лист.

Вычисление энтропии

В этом разделе мы рассмотрим математические расчеты, которые предназначены для поиска переменных, обеспечивающих наименьший уровень энтропии.

Как уже говорилось, построение дерева решений начинается с задания переменной в качестве корневого узла, при этом каждое значение этой переменной, например, «Да» и «Нет», назначается ветви, ведущей к новому узлу. Затем выбирается вторая переменная для дальнейшего разделения данных с целью создания новых ветвей и узлов принятия решений.

Поскольку мы хотим, чтобы в узлах было сосредоточено как можно больше экземпляров одного и того же класса, нам необходимо подходить к выбору каждой переменной стратегически, опираясь на значение энтропии, также известной как мера ценности информации. Энтропия измеряется в битах (при использовании логарифма по основанию 2) и рассчитывается на основе состава точек данных, содержащихся в каждом узле.

Используя следующую логарифмическую формулу, мы вычислим значение энтропии для каждого потенциального разбиения, выраженное в битах и принадлежащее диапазону от 0 до 1.

$$(-p_1 \log p_1 - p_2 \log p_2) / \log 2$$

Имейте в виду, что логарифмические выражения можно быстро вычислить онлайн с помощью калькулятора Google.

Да: p_1[6,6] и p_2[0,6]
Нет: p_1[4,4] и p_2[0,4]

Шаг 1: Вычисление энтропии каждого узла.
$(-p_1 \log p_1 - p_2 \log p_2)$ / log2
Да: (−6/6×log6/6−0/6×log0/6) / log2 = 0
Нет: (−0/4×log0/4−4/4×log4/4) / log2 = 0

Шаг 2: Умножение значений энтропии двух узлов на общее количество точек данных (10).

Да: p_1[2,4] и p_2[2,4]
Нет: p_1[4,6] и p_2[2,6]

Шаг 1: Вычисление энтропии каждого узла.
Да: (−2/4×log2/4−2/4×log2/4) / log2 = 1
Нет: (−4/6×log4/6−2/6×log2/6) / log2 = 0,91829583405

Шаг 2: Умножение значений энтропии двух узлов на общее количество точек данных.

(4/10) × 1 + (6/10) × 0,918

0,4 + 0,5508 = 0,9508

Да: p_1[3,7] и p_2[4,7]

Нет: p_1[3,3] и p_2[0,3]

Шаг 1: Вычисление энтропии каждого узла.

Да: (–3/7 × log3/7–4/7 × log4/7) / log2 = 0,98522813603

Нет: (–3/3 × log3/3–0/3 × log0/3) / log2 = 0

Шаг 2: Умножение значений энтропии двух узлов на общее количество точек данных.

(7/10) × 0,985 + (3/10) × 0

0,6895 + 0 = 0,6895

Результаты

Превышение КПЭ = 0 бит

Способность к лидерству = 0,9508 бит

Возраст < 30 лет = 0,6895 бит

Согласно нашим расчетам, переменная **Превышение КПЭ** обеспечивает идеальную классификацию, а это означает, что нам не нужно продолжать построение дерева после изучения этой переменной. Следующим лучшим кандидатом была переменная **Возраст < 30 лет** с энтропией 0,6895 бит. Переменная **Способность к лидерству** имеет самую высокую энтропию,

составляющую 0,9508 бит, что говорит о повышенном уровне беспорядка и практически полном отсутствии информационного выигрыша. Фактически мы можем рассчитать энтропию данных перед выполнением любого разбиения, чтобы установить необходимость в анализе соответствующей переменной.

Получили повышение 6/10, Не получили повышения 4/10
(−6/10 × log6/10−4/10 × log4/10) / log2 = 0,971
0,971−0,9508 = 0,0202

Итак, вычитание значения энтропии, обеспечиваемой переменной **Способность к лидерству**, из исходной энтропии набора данных приводит к незначительному информационному выигрышу в 0,0202 бит.

Переобучение

Существенный недостаток деревьев решений — их склонность к чрезмерной подгонке модели под тренировочные данные. Опираясь на закономерности, извлекаемые из обучающей выборки, дерево решений точно анализирует и расшифровывает данные, с которыми сталкивается в процессе обучения. Однако то же самое дерево может не справиться с классификацией тестовых данных, например в том случае, если в них содержатся правила, с которыми модель еще не сталкивалась, или если разделение набора данных на тренировочную и тестовую выборки оказалось не репрезентативным. Кроме того, поскольку деревья решений формируются путем многократного разбиения точек данных на подразделы, небольшое изменение в способе разбиения данных в верхней или средней части дерева может кардинально изменить окончательный прогноз модели и привести к образованию совсем другого дерева. Виновником в данном случае будет жадный алгоритм.

Начиная с самого первого разбиения, жадный алгоритм выбирает переменную, которая наилучшим образом разделяет данные на однородные группы. Подобно ребенку, сидящему перед коробкой с кексами, этот алгоритм не задумывается о долгосрочных последствиях своих действий. Переменная, используемая для выполнения первого разделения данных, не гарантирует получения наиболее точной модели в самом конце. Напротив, к получению точной модели может привести менее эффективное

разбиение данных в верхней части дерева. Таким образом, несмотря на то, что деревья решений отличаются наглядностью и способны эффективно классифицировать один набор данных, они также проявляют себя весьма негибкими и подверженными переобучению, особенно в случае с наборами данных, характеризующимися высокой дисперсией.

Бэггинг

Вместо того чтобы стремиться к наиболее эффективному разбиению данных в рамках каждого раунда этого рекурсивного процесса, можно воспользоваться альтернативным методом, который предполагает построение нескольких деревьев и объединение их прогнозов. Один из методов этой техники — так называемый бэггинг, который предполагает выращивание нескольких деревьев решений путем случайного выбора входных данных для каждого дерева и объединения результатов с помощью усреднения выходных данных (в случае задач регрессии) или голосования (в случае задач классификации).

Ключевая особенность бэггинга — бутстрэп-выборка. Для того чтобы несколько деревьев решений могли генерировать уникальные результаты, в каждой модели должен присутствовать элемент вариативности и случайности, так как создавать пять или десять одинаковых моделей было бы совершенно бессмысленно. Бутстрэп-выборка решает эту задачу путем извлечения случайной вариации данных в рамках каждого раунда, а в случае с бэггингом через каждое дерево пропускаются различные варианты тренировочных выборок. Хотя это не устраняет проблему переобучения, зато позволяет гарантировать то, что доминирующие в наборе данных закономерности проявятся в большем количестве деревьев, а также в итоговом классе или прогнозе. Таким образом, бэггинг зарекомендовал себя как эффективный алгоритм для работы с выбросами и снижения степени дисперсии, которая обычно наблюдается при использовании единственного дерева решений.

Метод случайного леса

Тесно связан с бэггингом метод случайного леса. В то время как оба эти метода предполагают формирование нескольких деревьев решений и использование бутстрэп-выборки для рандомизации данных, случайный лес

искусственно сужает выбор переменных, ограничивая количество переменных, рассматриваемых при выполнении каждого разбиения. Другими словами, алгоритм не может рассматривать все *n* переменных перед каждым разбиением.

В случае бэггинга деревья часто оказываются похожими, поскольку используют одну и ту же переменную на начальных этапах процесса принятия решений в попытке уменьшить энтропию. Это означает, что прогнозы деревьев сильно коррелируют и в плане общей дисперсии напоминают прогнозы одного дерева решений. Метод случайного леса позволяет обойти эту проблему, ограничивая подмножество переменных, рассматриваемых перед каждым разбиением, что увеличивает вероятность выбора других переменных. А благодаря усреднению прогнозов уникальных и некоррелированных деревьев итоговая структура процесса принятия решений оказывается менее изменчивой и зачастую более надежной. Поскольку модель обучается на ограниченном подмножестве доступных переменных, случайные леса считаются методом слабо контролируемого обучения (рис. 57).

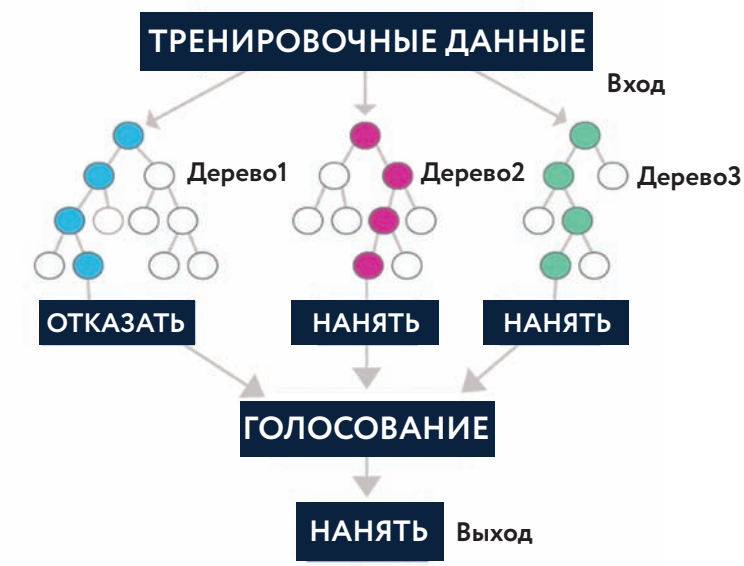


Рис. 57. Пример выращивания случайных деревьев для получения предсказания

В целом метод случайного леса лучше всего работает при использовании большого количества деревьев (более 100), что позволяет

сгладить потенциальное влияние выбросов, однако после определенной точки по мере добавления деревьев эффективность начинает уменьшаться. На определенном уровне дополнительные деревья перестают вносить существенные улучшения в модель и лишь увеличивают время ее работы. Хотя многое зависит от вашего набора данных, рекомендуемой отправной точкой является 100–150 деревьев решений. Автор книг и эксперт по работе с данными Скотт Хартшорн советует сосредоточиться на оптимизации других гиперпараметров перед добавлением в исходную модель новых деревьев, поскольку это позволяет ускорить работу модели в краткосрочной перспективе, а значит, увеличение количества деревьев в дальнейшем сможет принести хоть какую-то дополнительную выгоду[27].

Хотя случайные леса универсальны и хорошо подходят для интерпретации сложных закономерностей в данных, другие методы, в частности, градиентный бустинг, как правило, обеспечивают более высокую точность прогнозирования. Тем не менее случайные леса довольно быстро обучаются и хорошо подходят для быстрого создания эталонной модели.

Бустинг

Бустингом называется еще одно семейство алгоритмов, принцип работы которых основан на объединении большого количества деревьев решений. Основной акцент при этом делается на объединение «слабых» моделей в одну «сильную». Под «слабостью» подразумевается то, что исходная модель — плохой предсказатель, возможно, лишь слегка превосходящий метод случайного угадывания. В то же время под «силой» понимается способность модели надежно предсказывать истинное целевое значение.

Достичь развития сильных моделей на основе слабых можно путем применения весов к деревьям с учетом предыдущих неправильно классифицированных случаев. Это похоже на то, как школьный учитель улучшает успеваемость своего класса, проводя дополнительные занятия с учениками, плохо сдавшими последний экзамен.

Один из наиболее популярных алгоритмов бустинга — градиентный бустинг. Вместо того чтобы выбирать комбинации переменных случайным образом, алгоритм градиентного бустинга выбирает переменные, которые

[27] Scott Hartshorn, “Machine Learning With Random Forests And Decision Trees: A Visual Guide For Beginners,” *Scott Hartshorn*, 2016.

улучшают точность предсказания с каждым новым деревом. Таким образом, деревья решений выращиваются последовательно, поскольку каждое из них создается с использованием информации, полученной от предыдущего дерева, а не независимо от него. Ошибки, допущенные на тренировочных данных, регистрируются и учитываются в рамках следующего раунда обучения. На каждой итерации к обучающим данным применяются веса, значения которых основаны на результатах предыдущей итерации. Более высокий вес применяется к экземплярам, которые были предсказаны неверно, а правильно предсказанным экземплярам уделяется меньше внимания. Таким образом, неудовлетворительные результаты, полученные на более ранних итерациях, могут быть улучшены в рамках последующих итераций. Этот процесс повторяется вплоть до достижения достаточно низкого уровня ошибок. Итоговый результат определяется на основе средневзвешенного значения прогнозов, полученных от каждого дерева решений (рис. 58).

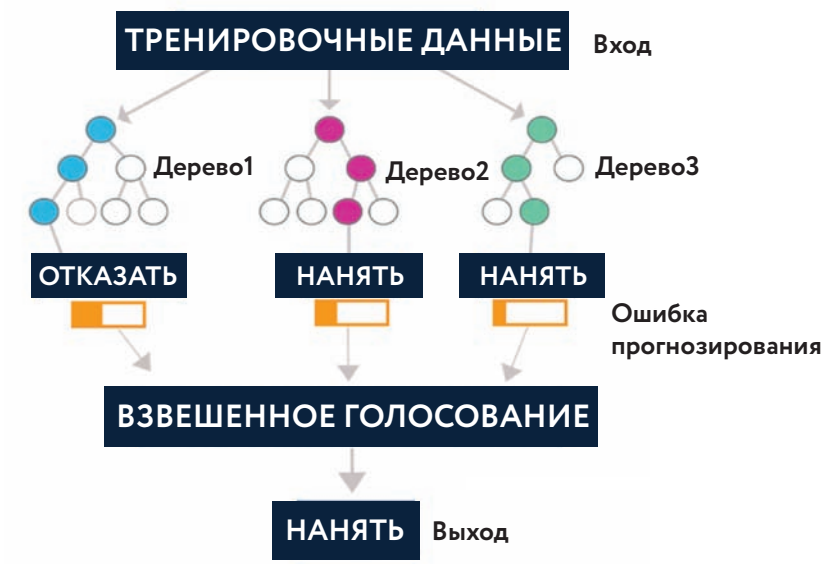


Рис. 58. Пример уменьшения величины ошибки прогнозирования нескольких деревьев для получения предсказания

Помимо всего прочего, бустинг смягчает проблему переобучения, используя при этом меньшее количество деревьев по сравнению с методом случайного леса. Хотя добавление большего числа деревьев в случайный

лес обычно помогает предотвратить переобучение, в случае с алгоритмами бустинга это может стать его причиной, поэтому добавлять деревья следует с осторожностью.

Склонность алгоритмов бустинга к переобучению можно объяснить высокой сфокусированностью их процесса обучения и вниманием к предыдущим ошибкам. Несмотря на то что это обычно обеспечивает более высокую точность прогнозов по сравнению с большинством алгоритмов, это также может привести к получению неоднозначных результатов при работе с данными, содержащими большое количество выбросов. В целом модели машинного обучения не должны слишком близко подгоняться к выбросам, однако алгоритмам бустинга может быть трудно этого избежать, поскольку они постоянно реагируют на ошибки, выявленные в процессе создания модели. При работе со сложными наборами данных, включающими множество выбросов, предпочтительной альтернативной бустингу может служить метод случайного леса.

Еще один важный недостаток бустинга — медленная работа, обусловленная обучением последовательной модели принятия решений. Последовательное обучение деревьев, при котором каждое дерево вынуждено дожидаться окончания работы с предыдущим, ограничивает масштабируемость модели, особенно по мере добавления новых деревьев. С другой стороны, компоненты случайного леса обучаются параллельно, что делает процесс его тренировки более быстрым.

Последний недостаток, который относится как к бустингу, так и к случайным лесам и бэггингу, — это утрата визуальной простоты и легкости интерпретации, свойственных отдельному дереву решений. При использовании сотен деревьев визуализировать и интерпретировать общую структуру процесса принятия решений гораздо сложнее.

Однако, если у вас есть время, ресурсы и подходящий набор данных для обучения бустинг-модели, то конечный результат может оказаться весьма стоящим, так как после развертывания эта модель позволит получать быстрые и точные прогнозы. Также следует отметить, что в настоящее время бустинг — один из самых популярных алгоритмов машинного обучения.

Контрольная работа

Предскажите массу тела пингвинов (body_mass_g), используя следующий набор данных и метод случайного леса.

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE
6	Adelie	Torgersen	38.9	17.8	181.0	3625.0	FEMALE
7	Adelie	Torgersen	39.2	19.6	195.0	4675.0	MALE
8	Adelie	Torgersen	34.1	18.1	193.0	3475.0	NaN
9	Adelie	Torgersen	42.0	20.2	190.0	4250.0	NaN

1. Какие переменные мы могли бы использовать в качестве независимых переменных при обучении нашей модели?
2. Для быстрого обучения эталонной модели градиентный бустинг подходит лучше, чем метод случайного леса. Правда или ложь?
3. Какую из следующих техник можно легко визуализировать?
   А) Деревья решений.
   Б) Градиентный бустинг.
   В) Метод случайного леса.

Ответы

1. Все переменные, кроме body_mass_g (методы на основе деревьев решений хорошо работают как с дискретными, так и с непрерывными переменными в качестве входных данных).
2. Ложь (градиентный бустинг выполняется последовательно, что замедляет процесс тренировки модели. Компоненты случайного леса обучаются одновременно, что ускоряет процесс тренировки).
3. Деревья решений.

15

АНСАМБЛЕВОЕ МОДЕЛИРОВАНИЕ

Перед принятием важного решения мы, как правило, стараемся ознакомиться с несколькими мнениями по интересующему нас вопросу, вместо того чтобы опираться лишь на одну точку зрения или прислушиваться к первому высказавшемуся человеку. Точно так же при выборе модели для анализа данных важно рассмотреть и опробовать несколько алгоритмов. В рамках продвинутого машинного обучения применяется метод, называемый ансамблевым моделированием, который предполагает объединение результатов работы различных алгоритмов или моделей в единую предсказательную модель. Объединяя результаты различных моделей (а не полагаясь лишь на какую-то одну оценку), ансамблевое моделирование помогает достичь консенсуса в отношении значения анализируемых данных. Агрегированные оценки также обычно оказываются более точными по сравнению с результатами применения отдельных методов. Однако во избежание неправильной обработки одних и тех же ошибок крайне важно обеспечить некоторую степень вариативности ансамблевых моделей.

В случае задач классификации результаты работы нескольких моделей объединяются в единый прогноз с помощью системы голосования[28] исходя из частоты ответов, а в случае задач регрессии это делается путем численного усреднения[29,30]. Ансамблевые модели можно разделить на последовательные и параллельные, а также на однородные и неоднородные.

Для начала рассмотрим последовательные и параллельные модели. В первом случае ошибка прогнозирования модели уменьшается путем применения весов к классификаторам, которые ранее классифицировали

[28] Класс, получивший наибольшее количество голосов, выбирается в качестве конечного результата.

[29] Как правило, чем больше голосов или числовых результатов принимается во внимание, тем более точным оказывается итоговый прогноз.

[30] Цель решения задач регрессии — предсказание числового значения, например стоимости дома, а не предсказание дискретного класса (как в случае задач классификации).

данные неправильно. Градиентный бустинг и алгоритм AdaBoost, разработанный для решения задач классификации, — это примеры последовательных моделей. В свою очередь параллельные ансамблевые модели работают одновременно и предполагают уменьшение ошибки путем усреднения. Это, например, случайные леса.

Однородные ансамблевые модели создаются на основе одного метода с многочисленными вариациями, а разнородные — на основе различных методов. Примером однородной ансамблевой модели может служить множество деревьев решений, работающих вместе над формированием единого прогноза (как в случае бэггинга). А примером неоднородного ансамбля — сочетание кластеризации методом *k*-средних или нейронной сети с алгоритмом дерева решений.

Разумеется, очень важно выбирать методы, хорошо дополняющие друг друга. Например, нейронные сети требуют наличия полных данных для анализа, в то время как деревья решений способны работать с отсутствующими значениями[31]. Вместе эти методы обеспечивают дополнительные преимущества по сравнению с использованием однородной модели. Нейронная сеть дает точные прогнозы в большинстве случаев предоставления конкретного значения, а дерево решений гарантирует отсутствие «нулевых» результатов, которые могли бы возникнуть из-за недостающих значений при использовании нейросети.

Хотя производительность ансамблевой модели чаще всего превосходит результаты отдельного алгоритма[32], степень сложности и изощренности может стать ее потенциальным недостатком. Создание ансамблевой модели предполагает тот же компромисс, что и использование множества деревьев решений вместо одного, при котором прозрачность и простота интерпретации приносятся в жертву точности, обеспечиваемой такими более сложными алгоритмами, как: случайный лес, бэггинг или бустинг. В большинстве случаев побеждает производительность модели, однако ее интерпретируемость остается важным фактором, который необходимо учитывать при выборе подходящего алгоритма (алгоритмов) для анализа ваших данных.

---

[31] Деревья решений могут рассматривать отсутствующие значения в качестве отдельной переменной. Например при оценке прогноза погоды точки данных могут распределяться по таким классам, как: *солнечно*, *пасмурно*, *дождливо* или *отсутствует*.

[32] Ian H. Witten, Eibe Frank, Mark A. Hall, "Data Mining: Practical Machine Learning Tools and Techniques," *Morgan Kaufmann*, Third Edition, 2011.

Что касается методов ансамблевого моделирования, то существуют четыре основных: бэггинг, бустинг, «ведро моделей» и стекинг.

«Ведро моделей» (a bucket of models) — это метод создания неоднородного ансамбля, который предполагает обучение нескольких различных алгоритмических моделей на одной и той же тренировочной выборке и выбор той, которая выдает наиболее точные результаты на тестовых данных.

Бэггинг (bagging), как мы уже говорили, — это метод создания однородного ансамбля, который предполагает обучение моделей на случайных выборках данных и усреднение их прогнозов для получения единой модели.

Бустинг (boosting) — это популярный альтернативный метод использования однородного ансамбля, который учитывает ошибки и данные, неверно классифицированные в рамках предыдущей итерации, с целью разработки последовательной модели. Примеры алгоритмов бустинга — градиентный бустинг и AdaBoost.

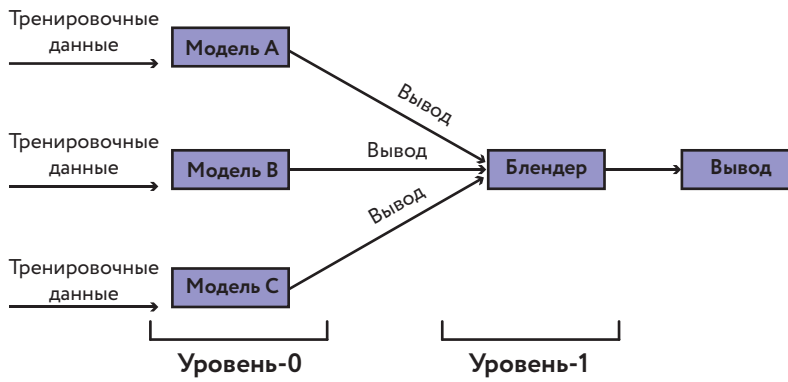


Рис. 59. Алгоритм стекинга

Стекинг (stacking) предполагает одновременное применение к данным нескольких моделей и объединение их результатов для получения итоговой модели (рис. 59). В отличие от бустинга и бэггинга стекинг обычно предполагает объединение результатов различных алгоритмов (неоднородная модель), а не изменение гиперпараметров одного и того же алгоритма (однородная модель). Кроме того, вместо выказывания равного доверия каждой модели с помощью усреднения или голосования, метод стекинга пытается выявить и сделать акцент на наиболее эффективно работающих моделях. Это достигается путем сглаживания коэффициента ошибок

моделей базового уровня (известного как уровень 0) с помощью системы взвешивания, после чего эти результаты передаются в модель уровня 1, где они объединяются в итоговый прогноз.

Хотя стекинг иногда применяется крупными представителями отрасли, выгода от его использования с учетом уровня сложности оказывается незначительной, поэтому организации обычно предпочитают использовать бустинг или бэггинг в силу их простоты и эффективности. Тем не менее стекинг часто применяется на таких соревнованиях по машинному обучению, как Kaggle Challenges и Netflix Prize. Например в рамках конкурса, проводившегося с 2006 по 2009 год, компания Netflix предложила приз за разработку модели машинного обучения, способной значительно улучшить ее систему рекомендации контента. В одной из победивших методик, разработанной командой BellKor's Pragmatic Chaos, была реализована форма линейного стекинга, которая объединяла прогнозы сотен различных моделей, использующих разные алгоритмы.

16

СРЕДА РАЗРАБОТКИ

Теперь, когда мы обсудили статистические основы различных алгоритмов машинного обучения, пришло время поговорить о написании кода и об установке среды разработки.

Хотя для машинного обучения можно использовать множество языков программирования (как было сказано в главе 4), для следующего упражнения был выбран Python, поскольку он прост в изучении и широко используется в промышленности и в рамках учебных онлайн-курсов. Если у вас нет опыта в программировании или написании кода на Python, не отчаивайтесь. Вы можете смело пропустить код и сосредоточиться на текстовых пояснениях, чтобы понять суть выполняемых шагов. Основы программирования на Python вы также можете найти в приложении в конце этой книги.

В качестве среды разработки мы установим Jupyter Notebook, веб-приложение с открытым исходным кодом, позволяющее редактировать код проектов и предоставлять другим людям доступ к нему. Это приложение можно установить с помощью дистрибутива Anaconda или менеджера пакетов Python под названием pip. Опытные пользователи Python, возможно, захотят установить Jupyter Notebook с помощью менеджера pip. Соответствующие инструкции можно найти на странице **http://jupyter.org/install.html**. Новичкам я рекомендую воспользоваться дистрибутивом Anaconda, который позволяет произвести установку простым методом щелчка и перетаскивания (**https://www.anaconda.com/products/individual/**).

Перейдя на веб-сайт Anaconda, вы сможете выбрать программу установки для ОС Windows, macOS или Linux, а также найти инструкции, соответствующие выбранной операционной системе.

После установки Anaconda на свой компьютер вы получите доступ к ряду приложений для работы с данными, включая rstudio, Jupyter Notebook и graphviz, предназначенному для визуализации данных. Для выполнения этого упражнения выберите Jupyter Notebook, щелкнув по кнопке **Launch** на вкладке Jupyter Notebook (рис. 60).

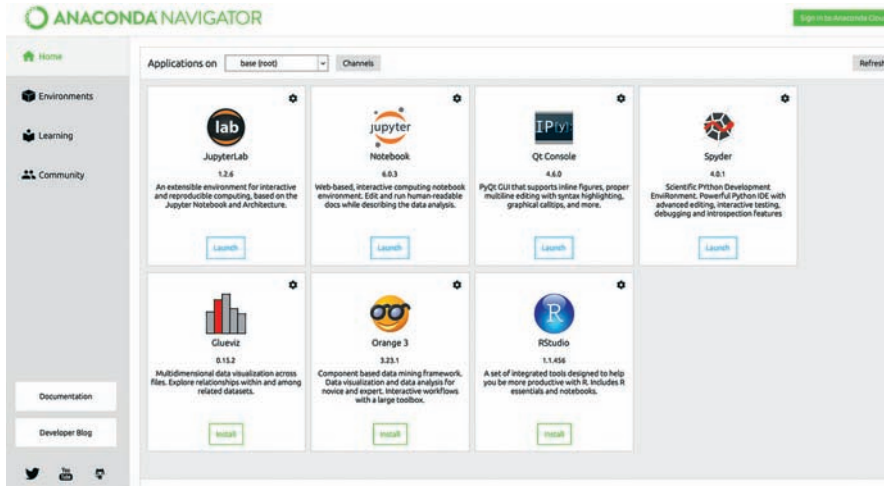


Рис. 60. Портал Anaconda Navigator

Чтобы запустить Jupyter Notebook, введите следующую команду в окно терминала (в ОС Mac/Linux) или в командную строку (в ОС Windows):

```
jupyter notebook
```

После этого в окне терминала/командной строке отобразится URL-адрес наподобие: http://localhost:8888/

Скопируйте и вставьте сгенерированный URL-адрес в адресную строку своего веб-браузера, чтобы загрузить Jupyter Notebook. После того как приложение откроется в браузере, щелкните по кнопке **New** в правом верхнем углу, чтобы создать новый проект, а затем выберите **Python 3**. Теперь вы готовы к написанию кода. Далее мы рассмотрим основы работы в Jupyter Notebook (рис. 61).

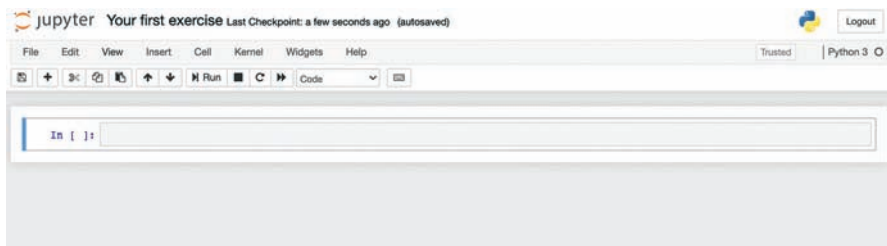


Рис. 61. Снимок экрана с новым проектом

Импорт библиотек

Первым шагом при работе над любым проектом машинного обучения на Python будет установка необходимых библиотек. Их набор отличается от проекта к проекту в зависимости от состава ваших данных и поставленных целей, будь то визуализация данных, ансамблевое моделирование, глубокое обучение или что-то другое (рис. 62).

```
In [ ]: # Import library
        import pandas as pd
```

Рис. 62. Импорт пакета Pandas

Приведенный выше фрагмент кода позволяет импортировать Pandas — популярную библиотеку Python, используемую в области машинного обучения.

Импорт и предварительный просмотр набора данных

Теперь мы можем использовать Pandas для импорта набора данных. Я выбрал бесплатный и общедоступный набор данных с сайта **kaggle.com**, который содержит данные о ценах на дома, квартиры и таунхаусы в Мельбурне, Австралия. Этот набор данных был составлен на основе объявлений, еженедельно публикуемых на сайте **www.domain.com.au**. Полный набор содержит сведения о 34 857 объектах недвижимости и 21 переменную, включая адрес, пригород, размер участка, количество комнат, цену, долготу, широту, почтовый индекс и т. д.

Набор данных Melbourne_housing_FULL можно скачать по ссылке: **https://www.kaggle.com/anthonypino/melbourne-housing-market/**.

После регистрации бесплатной учетной записи и входа на сайт **kaggle.com** загрузите набор данных в виде zip-файла. Затем распакуйте и импортируйте его в Jupyter Notebook. Чтобы загрузить данные в кадр данных Pandas в виде таблицы, вы можете использовать команду `pd.read_csv`.

```
df = pd.read_csv('~/Downloads/Melbourne_housing_FULL.csv')
```

Эта команда импортирует набор данных непосредственно в Jupyter Notebook. Однако обратите внимание на то, что путь к файлу зависит от места сохранения набора данных на вашем компьютере и от вашей операционной системы. Например, если вы сохранили CSV-файл на рабочем столе в ОС Mac, то вам нужно импортировать его с помощью следующей команды:

```
df = pd.read_csv('~/Desktop/Melbourne_housing_FULL.csv')
```

В моем случае набор данных был импортирован из папки «Загрузки» (Downloads). При освоении машинного обучения и науки о данных очень важно сохранять наборы данных и проекты в отдельных именованных папках (рис. 63). Если вы сохраните CSV-файл в той же папке, что и Jupyter Notebook, то вам не придется добавлять имя каталога или фрагмент `~/`.

Рис. 63. Импорт набора данных в виде фрейма данных

При сохранении CSV-файла на рабочем столе в ОС Windows вы можете импортировать его следующим образом:

```
df = pd.read_csv('C:\\Users\\John\\Desktop\\
                  Melbourne_housing_FULL.csv')
```

Затем используйте команду `head()` для предварительного просмотра фрейма данных.

```
df.head()
```

Щелкните правой кнопкой мыши и выберите команду **Run** или перейдите в меню Jupyter Notebook **Cell → Run All** (рис. 64).

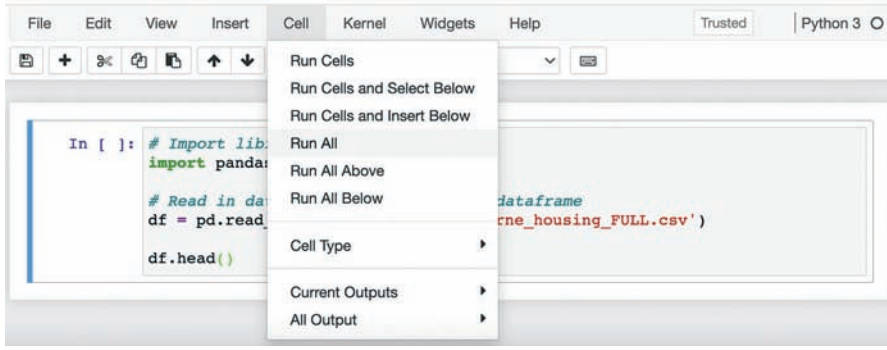
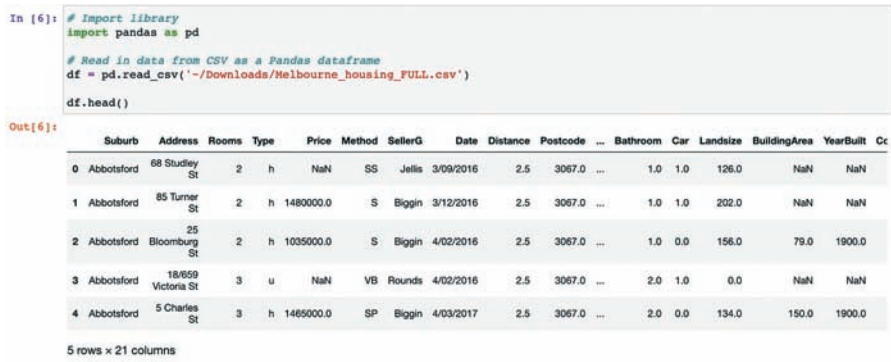


Рис. 64. Выбор команды Run All через навигационное меню

В результате набор данных отобразится в виде фрейма данных Pandas в Jupyter Notebook, как показано на рис. 65.



```
In [6]: # Import library
        import pandas as pd

        # Read in data from CSV as a Pandas dataframe
        df = pd.read_csv('~/Downloads/Melbourne_housing_FULL.csv')

        df.head()
```

Out[6]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Cc
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	1900.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	NaN	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	134.0	150.0	1900.0	

5 rows × 21 columns

Рис. 65. Предварительный просмотр фрейма данных в Jupyter Notebook

По умолчанию количество строк, отображаемых с помощью команды `head()`, равно пяти. Чтобы задать другое количество отображаемых строк, введите нужное число непосредственно внутри круглых скобок, как показано на рис. 66.

```
df.head(10)
```

Теперь фрейм данных содержит 10 строк. Вы также можете заметить, что внизу слева отображается общее количество строк и столбцов (10 строк × 21 столбец).

```
In [2]: # Import library
        import pandas as pd

        # Read in data from CSV as a Pandas dataframe
        df = pd.read_csv('~/Downloads/Melbourne_housing_FULL.csv')

        df.head(10)
```

Out[2]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Co
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	1900.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	NaN	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	134.0	150.0	1900.0	
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	...	2.0	1.0	94.0	NaN	NaN	
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	...	1.0	2.0	120.0	142.0	2014.0	
7	Abbotsford	16 Maugie St	4	h	NaN	SN	Nelson	6/08/2016	2.5	3067.0	...	2.0	2.0	400.0	220.0	2006.0	
8	Abbotsford	53 Turner St	2	h	NaN	S	Biggin	6/08/2016	2.5	3067.0	...	1.0	2.0	201.0	NaN	1900.0	
9	Abbotsford	99 Turner St	2	h	NaN	S	Collins	6/08/2016	2.5	3067.0	...	2.0	1.0	202.0	NaN	1900.0	

10 rows × 21 columns

Рис. 66. Предварительный просмотр фрейма данных с 10 строками

Поиск нужной строки

Хотя команда `head` помогает получить общее представление о форме фрейма данных, найти конкретную информацию в наборах, содержащих сотни, а то и тысячи строк, бывает довольно трудно. Специалистам по машинному обучению часто требуется найти конкретную строку путем сопоставления ее номера с информацией о ней. Например, если наша модель машинного обучения обнаружит, что строка 100 соответствует дому, который лучше всего подходит потенциальному покупателю, нам нужно будет просмотреть информацию, содержащуюся в соответствующем месте фрейма данных.

Для этого можно использовать команду `iloc[]`.

В данном примере код `df.iloc[100]` используется для нахождения строки с индексом 100 во фрейме данных, которая соответствует объекту недвижимости, расположенному в пригороде Airport West. Обратите внимание на то, что индексация строк во фрейме данных Python начинается с 0. Таким образом, объект в Airport West технически будет 101-м объектом, содержащимся во фрейме данных (рис. 67).

```
In [3]: # Import library
        import pandas as pd

        # Read in data from CSV as a Pandas dataframe
        df = pd.read_csv('~/Downloads/Melbourne_housing_FULL.csv')

        df.iloc[100]

Out[3]: Suburb                          Airport West
        Address                         180 Parer Rd
        Rooms                                      3
        Type                                       h
        Price                                 830000
        Method                                     S
        SellerG                               Barry
        Date                             16/04/2016
        Distance                               13.5
        Postcode                               3042
        Bedroom2                                  3
        Bathroom                                  1
        Car                                       2
        Landsize                                971
        BuildingArea                            113
        YearBuilt                              1960
        CouncilArea      Moonee Valley City Council
        Lattitude                          -37.7186
        Longtitude                          144.876
        Regionname            Western Metropolitan
        Propertycount                          3464
        Name: 100, dtype: object
```

Рис. 67. Поиск строки с помощью команды `.iloc[]`

Вывод на экран названий столбцов

Последний фрагмент кода, который я хотел бы вам представить, — это команда `columns`, которая выводит на экран названия столбцов, содержащихся в наборе данных. Она может пригодиться при решении вопросов о том, какие признаки следует выбрать, изменить или удалить из модели (рис. 68).

```
df.columns
```

```
In [5]: # Import library
        import pandas as pd

        # Read in data from CSV as a Pandas dataframe
        df = pd.read_csv('~/Downloads/Melbourne_housing_FULL.csv')

        df.columns

Out[5]: Index(['Suburb', 'Address', 'Rooms', 'Type', 'Price', 'Method', 'SellerG',
               'Date', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom', 'Car',
               'Landsize', 'BuildingArea', 'YearBuilt', 'CouncilArea', 'Lattitude',
               'Longtitude', 'Regionname', 'Propertycount'],
              dtype='object')
```

Рис. 68. Вывод на экран названий столбцов

Снова запустите код, выбрав команду **Run**, чтобы отобразить названия 21 столбца и их тип данных, которым в данном случае является объект (`dtype='object'`). Вы можете заметить, что некоторые из названий столбцов написаны неправильно. Эту проблему мы обсудим в следующей главе.

17

ПОСТРОЕНИЕ МОДЕЛИ НА ЯЗЫКЕ PYTHON

В этом разделе мы разработаем полноценную модель машинного обучения на основе кода, представленного в предыдущей главе.

В рамках этого упражнения мы создадим систему оценки стоимости объектов недвижимости на основе алгоритма градиентного бустинга, выполнив следующие шесть шагов.

1. Импорт библиотек.
2. Импорт набора данных.
3. Очистка набора данных.
4. Разбиение данных на тренировочную и тестовую выборки.
5. Выбор алгоритма и настройка его гиперпараметров.
6. Оценка результатов.

Импорт библиотек

Перед построением модели нам необходимо импортировать пакет Pandas и ряд функций из библиотеки Scikit-learn, включая градиентный бустинг (`ensemble`) и среднюю абсолютную ошибку для оценки производительности.

Импортируйте библиотеки, введя в Jupyter Notebook следующие команды:

```
#Импорт библиотек
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import ensemble
from sklearn.metrics import mean_absolute_error
```

Не волнуйтесь, если названия библиотек Scikit-learn, указанные в приведенном выше фрагменте кода, кажутся вам незнакомыми. Мы будем ссылаться на них при выполнении следующих шагов.

Импорт набора данных

Используйте команду `pd.read_csv` для загрузки набора данных Melbourne Housing Market во фрейм данных Pandas (как мы это делали в предыдущей главе).

```
df = pd.read_csv('~/Downloads/Melbourne_housing_FULL.csv')
```

Обратите внимание на то, что стоимость объектов недвижимости, перечисленных в этом наборе данных, выражена в австралийских долларах, а по состоянию на 2017 год один австралийский доллар был равен примерно 0,77 доллара США (табл. 15).

Табл. 15. Переменные из набора данных об объектах недвижимости в Мельбурне

Feature	Data Type	Continuous/Discrete
Suburb	String	Discrete
Address	String	Discrete
Rooms	Integer	Continuous
Type	String	Discrete
Price	Integer	Continuous
Method	String	Discrete
SellerG (seller's name)	String	Discrete
Date	TimeDate	Discrete
Distance	Floating-point	Continuous
Postcode	Integer	Discrete
Bedroom2	Integer	Continuous
Bathroom	Integer	Continuous
Car	Integer	Continuous
Landsize	Integer	Continuous
BuildingArea	Integer	Continuous
YearBuilt	TimeDate	Discrete
CouncilArea	String	Discrete
Lattitude	String	Discrete
Longtitude	String	Discrete
Regionname	String	Discrete
Propertycount (in that suburb)	Integer	Continuous

Очистка набора данных

Следующий шаг сводится к очистке набора данных, под которой понимается процесс его уточнения, в частности, изменение или удаление неполной, неактуальной или повторяющейся информации, а в некоторых случаях — преобразование текстовых данных в числовые значения и пересмотр признаков.

Стоит отметить, что некоторая очистка набора данных может выполняться до его импорта в среду разработки. Например, создатель набора данных Melbourne Housing Market неправильно написал слова Longitude (долгота) и Latitude (широта) в названиях столбцов. Поскольку мы не будем рассматривать эти две переменные в своей модели, нет необходимости вносить какие-либо изменения. Однако, если бы мы все же решили включить эти две переменные в нашу модель, нам стоило бы исправить эту ошибку в исходном файле.

В плане программирования орфографические ошибки, содержащиеся в названиях столбцов, не представляют проблемы при условии использования того же варианта написания в коде. Однако неправильное наименование столбцов может привести к человеческим ошибкам, особенно если вы работаете над своим кодом совместно с другими членами команды. Чтобы избежать путаницы, лучше всего исправлять орфографические и другие простые ошибки в исходном файле перед импортом набора данных в Jupyter Notebook или другую среду разработки. Вы можете сделать это, открыв CSV-файл в приложении Microsoft Excel (или аналогичной программе), отредактировав набор данных и снова сохранив его в виде CSV-файла.

В то время как простые ошибки вполне можно исправлять в исходном файле, такие серьезные структурные изменения набора данных, как удаление переменных или недостающих значений, лучше всего выполнять в среде разработки в целях обеспечения дополнительной гибкости и сохранения исходного набора данных для дальнейшего использования. Изменения набора данных в среде разработки оказываются менее долговечными и, как правило, выполняются проще и быстрее, чем при работе с исходным файлом.

Процесс очистки

Давайте удалим столбцы, которые мы не собираемся включать в модель, введя команду `del` и названия ненужных нам векторов (столбцов).

```
# Здесь сохранены неправильные варианты написания слов
# "Longitude" и "Latitude"
del df['Address']
del df['Method']
del df['SellerG']
del df['Date']
del df['Postcode"']
del df['Lattitude']
del df['Longtitude']
del df['Regionname']
del df['Propertycount']
```

Мы удалили столбцы Address (адрес), Regionname (название региона), Postcode (почтовый индекс), Latitude (широта) и Longitude (долгота), поскольку местоположение объектов недвижимости содержится и в других столбцах (Suburb (пригород) и CouncilArea (административный район)). Я предполагаю, что пригород и район имеют большее значение для покупателей, чем почтовый индекс, широта и долгота, хотя адрес тоже заслуживает упоминания.

Столбцы Method (метод), SellerG (продавец), Propertycount (количество объектов недвижимости) и Date (дата) также были удалены, поскольку они имеют меньшую значимость по сравнению с другими переменными. Это не означает, что эти переменные не влияют на стоимость объекта недвижимости, но для построения исходной модели нам вполне достаточно использовать остальные 11 независимых переменных. Мы можем добавить любую из этих переменных позднее, а при желании вы можете использовать их в собственной модели.

Вот 11 независимых переменных, оставшихся в нашем наборе данных: Suburb (пригород), Rooms (комнаты), Type (тип), Distance (удаленность), Bedroom2 (спальня2), Bathroom (ванная комната), Car (автомобиль), Landsize (площадь участка), BuildingArea (площадь здания), YearBuilt (год постройки) и CouncilArea (административный район). Двенадцатая зависимая переменная — Price (цена). Как уже было сказано, модели на основе деревьев решений (включая градиентный бустинг и случайные леса) отлично справляются с большими и высокоразмерными наборами данных, содержащими множество входных переменных.

Следующим этапом очистки набора данных будет удаление пропущенных значений. Хотя для решения проблемы, связанной с такими значениями, существует целый ряд методов (в частности, заполнение пустых ячеек средним или медианным значением набора данных, а также полное удаление отсутствующих значений), в этом упражнении мы хотим сохранить набор данных как можно более простым, поэтому не будем исследовать строки с пропущенными значениями. Очевидный недостаток такого подхода — сокращение объема анализируемых данных.

Новичку следует научиться работать с полными наборами данных, прежде чем вносить дополнительный элемент сложности, связанный с пропущенными значениями. К сожалению, в нашем наборе таких значений довольно много! Тем не менее после их удаления у нас останется достаточно строк для построения модели.

Для удаления строк с пропущенными значениями можно использовать метод Pandas `dropna`. Для получения дополнительной информации об этом методе и его параметрах обратитесь к табл. 16 или документации Pandas[33].

```
df.dropna(axis = 0, how = 'any', thresh = None,
          subset = None, inplace = True)
```

Табл. 16. Параметры метода Dropna

Параметр	Аргумент	Объяснение	Значение по умолчанию
`axis`	`0`	Удаляет строки с пропущенными значениями.	*V*
	`1`	Удаляет столбцы с пропущенными значениями.	
`how`	`any`	Удаляет строки или столбцы с любым количеством пропущенных значений.	*V*
	`all`	Удаляет строки или столбцы, все значения которых пропущены.	

[33] "Dropna", *Pandas*, **https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.dropna.html**

Окончание табл. 16

Параметр	Аргумент	Объяснение	Значение по умолчанию
`thresh`	`integer`	Задает целочисленное пороговое значение для удаления столбцов/строк, например, выбор значения «4» приводит к удалению строк или столбцов с 4 и более отсутствующими значениями.	
	`None`	Выберите «None», если не хотите задавать пороговое значение.	
`subset`	`variable`	Определяет, в каких столбцах следует искать отсутствующие значения, например 'genre'.	
	`None`	Выберите «None», если не хотите указывать подмножество.	
`inplace`	`True`	При выборе значения True операция выполняется на месте (то есть выполняется обновление, а не замена)	
	`False`		*V*

Помните о том, что строки с отсутствующими значениями следует отбрасывать после применения команды `delete` для удаления столбцов (как было показано в предыдущем шаге). В некоторых случаях это позволяет сохранить больше строк из исходного набора данных. Представьте, что вы отбрасываете целую строку только потому, что в ней отсутствует значение переменной, которая позднее будет удалена, например почтовый индекс!

Теперь давайте преобразуем столбцы, содержащие не числовые данные, в числовые значения с помощью прямого кодирования. В Pandas такое кодирование можно выполнить с помощью метода `pd.get_dummies`.

```
df = pd.get_dummies(df, columns = ['Suburb',
'CouncilArea', 'Type'])
```

Эта команда преобразует значения столбцов Suburb, CouncilArea и Type в числа путем прямого кодирования.

Наконец, давайте назначим зависимые и независимые переменные, указав переменную Price (цена) в качестве y, а оставшиеся 11 переменных в качестве *X* (при этом исключив Price из массива данных с помощью метода `drop`).

```
X = df.drop('Price', axis=1)
y = df['Price']
```

Разбиение набора данных

Теперь нам нужно разбить данные на тренировочную и тестовую выборки. В этом упражнении мы используем стандартную пропорцию 70/30, выполнив следующую команду Scikit-learn с параметром `test_size = 0.3` и перетасовав набор данных.

```
X_train, X_test, y_train,
        y_test = train_test_split(X, y, test_size = 0.3,
                                  shuffle = True)
```

Выбор алгоритма и настройка его гиперпараметров

Теперь нам нужно задать выбранный алгоритм (регрессор на основе градиентного бустинга) в качестве новой переменной (`model`) и настроить его гиперпараметры, как показано ниже.

```
model = ensemble.GradientBoostingRegressor(
    n_estimators = 150,
    learning_rate = 0.1,
    max_depth = 30,
    min_samples_split = 4,
    min_samples_leaf = 6,
    max_features = 0.6,
    loss = 'huber'
)
```

В первой строке кода указан сам алгоритм (градиентный бустинг), а в остальных строках — его гиперпараметры.

n_estimators задает количество деревьев решений. Напомним, что использование большого количества деревьев обычно повышает точность модели (до определенного момента), но при этом замедляет ее работу. В качестве отправной точки я выбрал 150 деревьев решений.

learning_rate определяет степень влияния дополнительных деревьев на общий прогноз, то есть, по сути, уменьшает вклад каждого дерева на заданное значение. Использование низкого значения, например 0,1, должно способствовать повышению точности модели.

max_depth определяет максимальное количество уровней (глубину) каждого дерева решений. При выборе значения `None` глубина дерева увеличивается до тех пор, пока все листья не станут однородными или пока количество образцов во всех листьях не окажется меньше значения `min_samples_leaf`. Здесь в качестве максимального количества уровней я задал довольно высокое значение (30), что, как мы вскоре увидим, оказывает существенное влияние на конечный результат.

min_samples_split определяет минимальное количество образцов, необходимое для выполнения очередного бинарного разделения. Например `min_samples_split = 10` означает, что для создания новой ветви в узле должно содержаться десять образцов.

min_samples_leaf определяет минимальное количество образцов, которые должны появиться в каждом дочернем узле (листе), прежде чем будет создана новая ветвь. Это помогает смягчить влияние выбросов и аномалий в виде низкого количества образцов, оказавшихся в одном листе в результате бинарного разделения. Например, значение `min_samples_leaf = 4` говорит о том, что для создания новой ветви в каждом листе должно содержаться не менее четырех образцов.

max_features определяет общее количество признаков, предъявляемых модели при определении наилучшего разбиения. Как было сказано в главе 14, случайные леса и градиентный бустинг ограничивают количество признаков, подаваемых каждому отдельному дереву с целью получения нескольких результатов, которые впоследствии будут поставлены на голосование.

Если в качестве значения `max_features` задано целое число, то модель будет рассматривать соответствующее количество признаков при каждом разбиении (ветвлении). Если в качестве значения задано число с плавающей запятой (например, 0,6), то оно будет определять процент признаков, случайным образом отобранных из их общего количества. Хотя

этот гиперпараметр задает максимальное количество признаков, учитываемых при определении наилучшего разбиения, общее количество признаков может превысить установленный предел в тех случаях, когда разбиение изначально не может быть произведено.

loss вычисляет коэффициент ошибок модели. В этом упражнении мы используем функцию потерь Хьюбера (`huber`), которая позволяет защититься от выбросов и аномалий. Альтернативные варианты — `ls` (регрессия методом наименьших квадратов), `lad` (наименьшие абсолютные отклонения) и `quantile` (квантильная регрессия). Функция потерь Хьюбера фактически представляет собой комбинацию регрессии методом наименьших квадратов и наименьших абсолютных отклонений.

Дополнительную информацию о гиперпараметрах алгоритма градиентного бустинга вы можете найти в документации Scikit-learn[34].

После настройки гиперпараметров модели мы применим к ней функцию `fit()` библиотеки Scikit-learn, чтобы связать тренировочные данные с алгоритмом обучения, хранящимся в переменной (`model`), тем самым произведя подгонку нашей предсказательной модели.

```
model.fit(X_train, y_train)
```

Оценка результатов

После обучения модели мы можем использовать функцию `predict()` из библиотеки Scikit-learn, чтобы применить модель к данным `X_train` и оценить ее производительность, сравнив полученные прогнозы с фактическими данными `y_train`. Как было сказано ранее, в этом упражнении для оценки точности модели мы используем среднюю абсолютную ошибку.

```
mae_train = mean_absolute_error(y_train,
                                model.predict(X_train))
print ("Training Set Mean Absolute Error: %.2f" %
       mae_train)
```

[34] "Gradient Boosting Regressor," *Scikit-learn*, **http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html**

Здесь мы вводим значения `y_train`, которые представляют собой правильные результаты из тренировочного набора данных. Функция `predict()` применяется к набору `X_train` и генерирует прогнозы. Затем функция `mean_absolute_error` определяет разницу между фактическими значениями и прогнозами модели. Вторая строка кода выводит на экран результаты с точностью до двух десятичных знаков вместе с текстовой строкой `"Training Set Mean Absolute Error:"`. Тот же процесс повторяется и с тестовыми данными.

```
mae_test = mean_absolute_error(y_test,
                               model.predict(X_test))
print ("Test Set Mean Absolute Error: %.2f" % mae_test)
```

Теперь запустите модель, щелкнув правой кнопкой мыши и выбрав команду **Run** или перейдя в меню Jupyter Notebook: **Cell → Run All**.

Подождите 30 секунд или чуть больше, пока компьютер обработает код модели обучения. Затем в нижней части блокнота отобразятся представленные ниже результаты.

```
Training Set Mean Absolute Error: 27834.12
Test Set Mean Absolute Error: 168262.14
```

Величина средней абсолютной ошибки нашей модели на тренировочной выборке составляет 27 834,12, а на тестовой выборке — 168 262,14. Это означает, что при расчете стоимости объекта недвижимости на основе тренировочного набора модель в среднем ошибалась на 27 834,12 доллара, а при расчете стоимости на основе тестового набора она ошибалась на 168 262,14 доллара.

Таким образом, наша модель почти точно предсказала фактическую стоимость объектов недвижимости из тренировочной выборки. Хотя 27 834,12 доллара может показаться большой суммой, такое среднее значение ошибки можно назвать низким, учитывая то, что максимальный диапазон значений в нашем наборе данных составляет 8 миллионов долларов. Поскольку стоимость бóльшего количества объектов в наборе данных выражается семизначным числом (то есть превышает 1 000 000 долларов), величину ошибки в 27 834,12 можно считать относительно небольшой.

На тестовой выборке модель дала менее точные прогнозы, а ее средняя ошибка составила 168 262,14 доллара. Большая разница в результативности модели на тренировочных и тестовых данных обычно говорит о ее переобучении. Поскольку наша модель была подогнана к закономерностям в тренировочных данных, она допустила больше ошибок при прогнозировании на тестовых данных, которые, скорее всего, содержат закономерности, а также выбросы и аномалии, с которыми модель еще не сталкивалась.

Однако в данном случае разница в производительности на тренировочных и тестовых данных усугубляется тем, что мы настроили свою модель на чрезмерную подгонку к тренировочной выборке, задав значение 30 для параметра `max_depth` (максимальная глубина). Хотя большое значение этого параметра повышает вероятность выявления закономерностей, содержащихся в тренировочных данных, это также приводит к переобучению модели.

Наконец, имейте в виду, что данные для тренировки и тестирования модели перемешиваются, а затем подаются деревьям решений в случайном порядке, поэтому при воспроизведении этой модели на вашем компьютере ее прогнозы могут немного отличаться.

Видеоверсия этой главы доступна в виде миникурса по адресу:

https://scatterplotpress.teachable.com/p/house-prediction-model. Это бесплатно и позволит шаг за шагом отследить весь процесс, описанный в этой главе.

18

ОПТИМИЗАЦИЯ МОДЕЛИ

В предыдущей главе мы разработали свою первую модель контролируемого обучения. Теперь нам нужно повысить точность ее прогнозов на новых данных и уменьшить последствия переобучения. Начнем с настройки гиперпараметров и изменим значение максимальной глубины (`max_depth`) с 30 до 5. Теперь наша модель генерирует следующие результаты:

```
Training Set Mean Absolute Error: 135283.69
```

Несмотря на возросшую величину средней абсолютной ошибки на тренировочной выборке, это должно помочь смягчить проблему переобучения и улучшить производительность модели. Оптимизации модели способствует также добавление деревьев. Если мы зададим значение 250 для параметра `n_estimators`, то получим следующие результаты:

```
Training Set Mean Absolute Error: 124469.48
Test Set Mean Absolute Error: 161602.45
```

Это уменьшает среднюю абсолютную ошибку на тренировочной выборке примерно на 11000, а также сокращает разрыв между величинами средней абсолютной ошибки на тренировочных и тестовых данных[35].

[35] В машинном обучении тестовые данные используются исключительно для оценки эффективности модели, а не для ее оптимизации. Поскольку тестовая выборка не может использоваться для построения и оптимизации модели, специалисты по работе с данными обычно используют третий независимый набор, называемый *проверочным*. После построения исходной модели на основе тренировочной выборки она может быть протестирована на проверочной выборке, а полученные результаты использованы для оптимизации ее гиперпараметров. Затем для оценки ошибки прогнозирования окончательной модели может быть использована тестовая выборка.

Вместе эти два шага оптимизации подчеркивают важность понимания влияния отдельных гиперпараметров. Если вы решите воспроизвести эту модель контролируемого обучения, я рекомендую изменять гиперпараметры по отдельности и анализировать их влияние на величину средней абсолютной ошибки модели на тренировочной выборке. Помимо всего прочего, при корректировке значений гиперпараметров вы заметите изменения во времени обработки. Например, уменьшение максимального количества уровней дерева (`max_depth`) с 30 до 5 приведет к значительному сокращению общего времени обработки. Скорость работы модели и расход ресурсов становятся важными факторами при обработке больших наборов данных.

Еще один важный метод оптимизации — отбор признаков. Ранее мы удалили из набора данных девять признаков, но сейчас пришло время их пересмотреть, чтобы оценить степень их влияния на точность предсказаний модели. Признак SellerG стоило бы добавить в модель, поскольку риелторская компания, продающая недвижимость, может оказать некоторое влияние на конечную цену продажи.

С другой стороны, исключение признаков из текущей модели может ускорить ее работу, не оказав существенного влияния на точность или даже повысив ее. При отборе признаков их следует изменять по отдельности и анализировать полученные результаты, вместо того чтобы вносить несколько изменений сразу.

Хотя оценить влияние выбора тех или иных переменных и гиперпараметров можно вручную методом проб и ошибок, существуют и автоматизированные методы оптимизации модели, в частности, поиск по решетке. Этот метод позволяет вам перечислить ряд конфигураций для каждого гиперпараметра и методично их протестировать, а затем выбрать оптимальную модель с помощью автоматизированного процесса голосования. Поскольку модель должна исследовать каждую возможную комбинацию гиперпараметров, поиск по решетке занимает много времени![36] Иногда бывает полезно сначала выполнить грубый поиск по решетке с использованием последовательных степеней числа 10 (например, 0,01, 0,1, 1, 10), а после выявления наилучшего значения произвести уточняющий поиск[37]. Пример кода для

[36] Большинство читателей этой книги сообщают о том, что время работы модели достигает 30 минут.

[37] Орельен Жерон, «Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем». Диалектика-Вильямс, 2020.

выполнения поиска по решетке с помощью библиотеки Scikit-learn приведен в конце этой главы.

Еще один способ оптимизации гиперпараметров алгоритма — это метод рандомизированного поиска, который реализован в Scikit-learn с помощью класса RandomizedSearchC V. За один раунд он позволяет протестировать гораздо больше гиперпараметров, чем поиск по решетке (при котором в рамках раунда изменяется лишь один гиперпараметр), поскольку использует случайное значение для каждого гиперпараметра. Кроме того, рандомизированный поиск позволяет задать количество раундов тестирования и контролировать расход вычислительных ресурсов. Поиск по решетке, с другой стороны, предполагает полный перебор комбинаций гиперпараметров, что не очевидно при анализе кода, поэтому может занять больше времени, чем вы ожидаете.

Наконец, если вместо градиентного бустинга вы хотите использовать другой алгоритм контролируемого обучения, вам все равно пригодится бóльшая часть кода, приведенного в этом разделе. Например, тот же код можно использовать для импорта нового набора данных, предварительного просмотра фрейма данных, удаления признаков (столбцов) и строк, разбиения и перемешивания данных, а также для вычисления средней абсолютной ошибки. Официальный сайт **http://scikit-learn.org** может стать отличным ресурсом для получения дополнительной информации о других алгоритмах, в том числе об использованном в этом упражнении градиентном бустинге.

Чтобы узнать о том, как применить созданную нами модель для оценки стоимости отдельного объекта недвижимости, вы можете обратиться к расширенному руководству, доступному по адресу: **https://scatterplotpress.teachable.com/p/house-prediction-model**.

Если у вас возникнут трудности с реализацией модели на основе кода, приведенного в этой книге, вы можете обратиться за помощью к автору по электронной почте oliver.theobald@scatterplotpress.com.

Код оптимизированной модели

```
# Импорт библиотек
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import ensemble
from sklearn.metrics import mean_absolute_error
```

```
# Считывание данных из CSV-файла
df = pd.read_csv('~/Downloads/Melbourne_housing_FULL.csv')

# Удаление ненужных столбцов
del df['Address']
del df['Method']
del df['SellerG']
del df['Date']
del df['Postcode']
del df['Lattitude']
del df['Longtitude']
del df['Regionname']
del df['Propertycount']

# Удаление строк с отсутствующими значениями
df.dropna(axis = 0, how = 'any', thresh = None,
          subset = None, inplace = True)

# Преобразование нечисловых данных методом прямого
# кодирования
df = pd.get_dummies(df, columns = ['Suburb',
'CouncilArea', 'Type'])

# Объявление переменных X и y
X = df.drop('Price', axis=1)
y = df['Price']

# Разбиение данных на тренировочную и тестовую
# выборки (70/30) и перемешивание
X_train, X_test, y_train, y_test = train_test_split(X, y,
                            test_size = 0.3, shuffle = True)

# Настройка параметров алгоритма
model = ensemble.GradientBoostingRegressor(
    n_estimators = 250,
    learning_rate = 0.1,
    max_depth = 5,
```

```
    min_samples_split = 4,
    min_samples_leaf = 6,
    max_features = 0.6,
    loss = 'huber'
)

# Применение модели к тренировочным данным
model.fit(X_train, y_train)

# Оценка точности модели (до двух знаков после запятой)
mae_train = mean_absolute_error(y_train,
                                model.predict(X_train))
print ("Training Set Mean Absolute Error: %.2f" % mae_train)

mae_test = mean_absolute_error(y_test, model.predict(X_test))
print ("Test Set Mean Absolute Error: %.2f" % mae_test)
```

Код для выполнения поиска по решетке

```
# Импорт библиотек, включая GridSearchCV
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import ensemble
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import GridSearchCV

# Считывание данных из CSV-файла
df = pd.read_csv('~/Downloads/Melbourne_housing_FULL.csv')

# Удаление ненужных столбцов
del df['Address']
del df['Method']
del df['SellerG']
del df['Date']
del df['Postcode']
del df['Lattitude']
del df['Longtitude']
```

```
del df['Regionname']
del df['Propertycount']

# Удаление строк с отсутствующими значениями
df.dropna(axis = 0, how = 'any', thresh = None, subset =
None, inplace = True)

# Преобразование нечисловых данных методом прямого
# кодирования
df = pd.get_dummies(df, columns = ['Suburb',
'CouncilArea', 'Type'])

# Объявление переменных X и y
X = df.drop('Price', axis=1)
y = df['Price']

# Разбиение данных на тренировочную и тестовую
# выборки (70/30) и перемешивание
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.3, shuffle = True)

# Выбор алгоритма
model = ensemble.GradientBoostingRegressor()

# Задайте конфигурации, которые хотите протестировать.
# Чтобы минимизировать время обработки, ограничьте число
# переменных или экспериментируйте с каждым
# гиперпараметром по отдельности.
hyperparameters = {
    'n_estimators': [200, 300],
    'max_depth': [4, 6],
    'min_samples_split': [3, 4],
    'min_samples_leaf': [5, 6],
    'learning_rate': [0.01, 0.02],
    'max_features': [0.8, 0.9],
    'loss': ['ls', 'lad', 'huber']
}
```

```
# Определение алгоритма поиска по решетке. Если применимо,
# запустите поиск параллельно на четырех процессорах.
grid = GridSearchCV(model, hyperparameters, n_jobs = 4)

# Выполнение поиска по решетке на тренировочных данных
grid.fit(X_train, y_train)

# Возврат оптимальных гиперпараметров
grid.best_params_

# Проверка точности модели с использованием оптимальных
# гиперпараметров
mae_train = mean_absolute_error(y_train,
                                grid.predict(X_train))
print ("Training Set Mean Absolute Error: %.2f" % mae_train)

mae_test = mean_absolute_error(y_test, grid.predict(X_test))
print ("Test Set Mean Absolute Error: %.2f" % mae_test)
```

ДАЛЬНЕЙШИЕ ШАГИ

Видеоуроки

Чтобы помочь вам сделать следующий шаг в освоении машинного обучения, я подготовил шесть видеоуроков, которые ознакомят вас с процессом создания предсказательных моделей на языке Python на основе бесплатных и доступных онлайн наборов данных. После выполнения этих упражнений вы будете готовы к созданию собственных моделей и освоению более сложных ресурсов.

- Линейная регрессия.
- Логистическая регрессия.
- Машины опорных векторов.
- Метод *k*-ближайших соседей.
- Кластеризация методом *k*-средних.
- Деревья решений.

Вы можете найти эти бесплатные видеоуроки по адресу:
https://scatterplotpress.teachable.com/p/ml-code-exercises.

Построение модели для прогнозирования стоимости домов на Python

Кроме того, помните, что в Интернете доступна бесплатная бонусная глава, в которой представлен код и процесс оценки отдельного дома с помощью модели, которую мы разработали в главе 17.

Вы можете найти эту бесплатную главу в видеоформате по адресу:
https://scatterplotpress.teachable.com/p/house-prediction-model.

Прочие ресурсы

Для более глубокого погружения в тему машинного обучения я настоятельно рекомендую пройти бесплатный курс Эндрю Ына (Andrew Ng) *Machine Learning* на сайте Coursera, а также ознакомиться с серией подкастов OCDevel: *Machine Learning Guide*, который станет отличным аудиоресурсом для начинающих.

Если вам понравился этот вводный курс, посвященный машинному обучению, я рекомендую вам две следующие книги из этой серии: ***Machine Learning with Python for Beginners*** и ***Machine Learning: Make Your Own Recommender System***. Они помогут вам расширить знания о машинном обучении, полученные в ходе чтения этого руководства, с помощью практических упражнений по написанию кода на языке Python.

Благодарность читателю

Благодарю вас за покупку этой книги. Теперь вы имеете базовое представление о ключевых понятиях из области машинного обучения и готовы приступить к серьезному изучению этого сложного предмета, в том числе к освоению такого важнейшего его компонента, как программирование.

Вы также можете ознакомиться с бесплатными обучающими материалами и видеоуроками на моем канале в Instagram[38] **machinelearning_beginners** и подписаться на мой ежемесячный информационный бюллетень: **http://eepurl.com/gKjQij.**

Свои отзывы, как положительные, так и отрицательные, а также предложения по улучшению этой книги вы можете прислать на мою электронную почту oliver.theobald@scatterplotpress.com. Я очень ценю подобную обратную связь и буду рад вашим письмам.

Наконец, я хочу выразить благодарность своим коллегам Джереми Педерсену и Руи Сюнгу за предоставление практических советов и фрагментов кода, использованных в этой книге, а также моим редакторам Крису Дино (Red to Black Editing) и Джереми Педерсену.

[38] Соцсеть признана экстремистской и запрещена на территории РФ.

Программа Bug Bounty

Мы предлагаем денежное вознаграждение читателям за обнаружение ошибок и других недочетов в книге. Некоторые очевидные ошибки могут быть допущены при интерпретации диаграммы или выполнении приведенного в книге кода, поэтому мы предлагаем всем читателям сначала связаться с автором для получения разъяснений и возможного вознаграждения, прежде чем оставлять в Интернете отрицательный отзыв о книге! Просто отправьте письмо по адресу **oliver.theobald@scatterplotpress.com** с описанием ошибки или неточности, с которой вы столкнулись.

Таким образом, мы можем предоставить вам дополнительные пояснения и примеры по электронной почте, чтобы помочь в понимании материала, а если вы окажетесь правы, мы вручим вам денежное вознаграждение через систему PayPal или в виде подарочной карты Amazon. Так вы сможете получить кругленькую сумму за свой отзыв, а мы — улучшить содержание книги для будущих читателей.

Дополнительные ресурсы

В этом разделе перечислены учебные материалы для читателей, желающих продвинуться в освоении области машинного обучения. Обратите внимание на то, что некоторые сведения, приведенные в этом разделе, включая цены, в будущем могут измениться.

I Машинное обучение I

Machine Learning

Формат: Бесплатный курс на Coursera.

Докладчик: Эндрю Ын (Andrew Ng).

Целевая аудитория: Начинающие (особенно те, кто предпочитает использовать MATLAB).

Бесплатный и качественный вводный курс от адъюнкт-профессора Эндрю Ына, одной из самых влиятельных фигур в области машинного обучения. Это настоящий обряд посвящения для всех заинтересованных.

Project 3: Reinforcement Learning

Формат: Онлайн-руководство в форме блога.

Автор: EECS Berkeley.

Целевая аудитория: Пользователи от среднего до продвинутого уровня.

Практическая демонстрация метода обучения с подкреплением, в частности Q-обучения, на примере игры Pac-Man.

| Базовые алгоритмы |

Machine Learning With Random Forests And Decision Trees: A Visual Guide For Beginners

Формат: Электронная книга.

Автор: Скотт Хартшорн (Scott Hartshorn).

Целевая аудитория: Начинающие.

Краткая, доступная и увлекательная книга о деревьях решений и случайных лесах с подробными наглядными примерами, полезными практическими советами и четкими инструкциями.

Linear Regression And Correlation: A Beginner's Guide

Формат: Электронная книга.

Автор: Скотт Хартшорн (Scott Hartshorn).

Целевая аудитория: Все.

Хорошо изложенное и доступное введение в темы линейной регрессии и корреляции.

| Будущее искусственного интеллекта |

The Inevitable: Understanding the 12 Technological Forces That Will Shape Our Future

Неизбежно. 12 технологических трендов, которые определяют наше будущее

Формат: Электронная книга, печатная книга, аудиокнига.

Автор: Кевин Келли (Kevin Kelly).

Целевая аудитория: Все, кого интересует будущее.

Хорошо проработанный взгляд в будущее с акцентом на искусственный интеллект и машинное обучение от автора бестселлера *The New York Times* Кевина Келли. В этой книге описано 12 технологических трендов, которые будут определять следующие 30 лет.

Homo Deus: A Brief History of Tomorrow
Homo Deus. Краткая история будущего

Формат: Электронная книга, печатная книга, аудиокнига.

Автор: Юваль Ной Харари (Yuval Noah Harari).

Целевая аудитория: Все, кого интересует будущее.

Автор успешной книги «Sapiens. Краткая история человечества» Юваль Ной Харари рассматривает возможные сценарии будущего, посвящая большие разделы машинному сознанию, приложениям на основе искусственного интеллекта, а также мощи данных и алгоритмов.

| Программирование |

Learning Python, 5th Edition
Изучаем Python, 5-е издание

Формат: Электронная книга, печатная книга.

Автор: Марк Лутц (Mark Lutz).

Целевая аудитория: Все, кто хочет освоить язык Python.

Исчерпывающее введение в Python, опубликованное издательством O'Reilly Media.

Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems
Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем

Формат: Электронная книга, печатная книга.

Автор: Орельен Жерон.

Целевая аудитория: Все, кто хочет освоить программирование на Python, а также библиотеки Scikit-Learn и TensorFlow.

Эта популярная книга O'Reilly Media, написанная консультантом по машинному обучению Орельеном Жероном, — отличный ресурс для тех, кто уже достаточно хорошо освоил машинное обучение и компьютерное программирование.

| Рекомендательные системы |

The Netflix Prize and Production Machine Learning Systems: An Insider Look

Формат: Блог.

Автор: Mathworks.

Целевая аудитория: Все.

Очень интересная статья в блоге, в которой рассказывается о том, как компания Netflix применяет машинное обучение для составления рекомендаций фильмов.

Recommender Systems

Формат: Курс на Coursera.

Докладчик: Университет Миннесоты.

Стоимость: Вы можете воспользоваться бесплатной 7-дневной пробной версией или получить курс вместе с подпиской на Coursera за 49 долларов США.

Целевая аудитория: Все.

Курс, разработанный Университетом Миннесоты, посвящен фундаментальным методам создания рекомендательных систем, в том числе контентной и коллаборативной фильтрации, а также неперсонализированным и проектно-ассоциативным рекомендательным системам.

| Глубокое обучение |

Deep Learning Simplified

Формат: Блог.

Канал: DeepLearning.TV.

Целевая аудитория: Все.

Небольшая серия видеороликов для ознакомления с темой глубокого обучения. Доступна бесплатно на YouTube.

Deep Learning Specialization: Master Deep Learning, and Break into AI

Формат: Курс на Coursera.

Докладчик: deeplearning.ai и NVIDIA.

Стоимость: Вы можете воспользоваться бесплатной 7-дневной пробной версией или получить курс вместе с подпиской на Coursera за 49 долларов США.

Целевая аудитория: Пользователи от среднего до продвинутого уровня, обладающие опытом работы с Python.

Насыщенная учебная программа для тех, кто хочет научиться строить нейронные сети с помощью языка Python и библиотеки TensorFlow, дающая возможность получить советы по поводу построения карьеры и узнать, как теория глубокого обучения применяется в индустрии.

Deep Learning Nanodegree

Формат: Курс на Udacity.

Докладчик: Udacity.

Стоимость: 599 долларов США.

Целевая аудитория: Пользователи от начального до продвинутого уровня, обладающие базовым опытом работы с Python.

Четырехмесячный онлайн-курс представляет собой всестороннее введение в сверхточные нейронные сети, рекуррентные нейронные сети и глубокое обучение с подкреплением, а также предполагает выполнение таких практических упражнений, как создание классификатора пород собак, генерирование телевизионных сценариев, генерирование лиц и управление полетом квадрокоптера.

| Профессии будущего |

Will a Robot Take My Job?

Формат: Онлайн-статья.

Автор: BBC.

Целевая аудитория: Все.

Проверьте вероятность того, что ИИ заменит вас на рабочем месте до 2035 года.

So You Wanna Be a Data Scientist? A Guide to 2015's Hottest Profession

Формат: Блог.

Автор: Тодд Вассерман (Todd Wasserman).

Целевая аудитория: Все.

Отличный ресурс, посвященный тому, как стать специалистом по работе с данными.

ПРИЛОЖЕНИЕ: ВВЕДЕНИЕ В PYTHON

Язык Python был разработан Гвидо ван Россумом в Национальном исследовательском институте математики и информатики в Нидерландах в конце 1980-х — начале 1990-х годов. Созданный на основе интерпретатора командной строки Unix и других языков программирования, включая C и C++, он был призван дать разработчикам возможность писать программы с использованием минимального количества строк кода[39]. В отличие от других языков программирования Python предусматривает множество английских ключевых слов для того, что в других языках выражается знаками пунктуации.

Интерпретатор Python считывает введенный код и выводит результат. Любые ошибки, включая некорректное форматирование, неправильно написанные функции или символы, оказавшиеся в вашем сценарии случайно, воспринимаются интерпретатором Python в качестве синтаксической ошибки.

В этом приложении мы обсудим базовый синтаксис и концепции, которые помогут вам писать гибкий и эффективный код на языке Python 3.

Комментарии

Добавление комментариев, проясняющих назначение и содержание кода, — хорошая практика в компьютерном программировании. В Python комментарии можно добавлять с помощью хеш-символа #. Все, что следует за этим символом (в строке кода), игнорируется интерпретатором.

Пример:

```
# Импорт набора данных Melbourne Housing из папки Загрузки
dataframe = pd.read_csv('~/Downloads/
                         Melbourne_housing_FULL.csv')
```

[39] Mike McGrath, "Python in easy steps: Covers Python 3.7," *In Easy Steps Limited*, Second Edition, 2018.

В данном случае будет выполнена только вторая строка кода, а первую интерпретатор Python проигнорирует.

Типы данных в Python

В табл. 17 перечислены распространенные типы данных в языке Python.

Табл. 17. Распространенные типы данных в языке Python

Название	Описание	Ключевая особенность	Пример
Целое число	Целые числа	Отсутствие дробной части	50
Число с плавающей точкой	Числа с дробной частью	Наличие дробной части	50.1
Строка	Слова и символы	Одинарные/двойные кавычки	"Fifty5" или ‹'Fifty5'
Список	Упорядоченная последовательность объектов	Квадратные скобки	[1, 2, 3, 4, 'fifty']
Кортеж	Упорядоченная и неизменяемая последовательность объектов. Похож на список, но не позволяет манипулировать значениями, что гарантирует целостность данных за счет предотвращения внесения случайных изменений в сложные фрагменты кода.	Круглые скобки	(1, 2, 3, 4)
Словарь	Пара ключ-значение. Ключ представляет собой строку, например имя файла, и связан со значением, например с изображением или текстом.	Фигурные скобки, двоеточие и кавычки	{"name": "john", "gender": "male"}
Множество	Неупорядоченная коллекция уникальных объектов	Фигурные скобки	{"1", "2", "a"}
Логический тип	Двоичные значения	Заглавная первая буква (T/F)	**True** или **False**

Занимаясь машинным обучением, вы будете часто работать со списками, содержащими строки, целочисленные значения или числа с плавающей точкой. Строковые переменные также называются символьными или буквенно-цифровыми и могут содержать буквы, цифры и такие символы, как хеш-символ (#) или символ нижнего подчеркивания (_).

Отступы и пробелы

В отличие от других языков программирования для группировки таких операторов, как функции и циклы, Python использует отступы, а не ключевые слова или пунктуацию.

Пример:

```
new_user = [
    66.00, #Время, проведенное на сайте в течение дня
    48, #Возраст
    24593.33, #Доход
    131.76, #Ежедневное использование Интернета
    1, #Мужчина
    1, #Страна_ Албания
    0, #Страна_ Алжир
]
```

Пробелы в выражениях игнорируются интерпретатором Python (например, выражение 8+4=12 эквивалентно 8 + 4 = 12), но могут добавляться для облегчения восприятия кода человеком.

Арифметические операторы в Python

Арифметические операторы, часто используемые в Python, представлены в табл. 18.

Табл. 18. Арифметические операторы, часто используемые в Python

Оператор	Описание	Пример ввода	Вывод
+	Сложение	2 + 2	4
–	Вычитание	2–2	0
*	Умножение	2 * 2	4
/	Деление	5 / 2	2.5
%	Деление по модулю (возвращает остаток от деления)	5 % 2	1
//	Целочисленное деление (возвращает целую часть частного)	5 // 2	2
**	Возведение в степень	2 ** 3	8

В Python применяется стандартный математический порядок операций, то есть умножение или деление выполняется перед сложением или вычитанием.

Пример:

```
2 + 2 * 3
```

Результатом вычисления этого выражения будет 8 ((2 * 3) + 2).

Как и в обычной арифметике, для изменения последовательности операций можно использовать скобки.

Пример:

```
(2 + 2) * 3
```

Результатом вычисления этого выражения будет 12 (4 * 3).

Объявление переменных

Роль переменной в компьютерном программировании заключается в хранении значения в памяти компьютера для последующего использования. Это позволяет интерпретатору Python ссылаться на более ранние фрагменты кода и манипулировать ими, используя имя соответствующей переменной. Выбранное имя переменной должно соответствовать следующим правилам:

- содержит только буквы, цифры и символы подчеркивания (A–Z, 0–9, _);
- начинается с буквы или символа подчеркивания, но не с цифры;
- не совпадает с ключевыми словами языка Python, например «return».

Имейте в виду, что имена переменных чувствительны к регистру, например `dataframe` и `Dataframe` считаются двумя разными переменными.

В Python переменные объявляются с помощью оператора =.

Пример:

```
dataset = 8
```

Однако Python не поддерживает пробелы между словами в имени переменных, поэтому для их связывания необходимо использовать символ подчеркивания.

Пример:

```
my_dataset = 8
```

Теперь на сохраненное значение (8) можно сослаться, вызвав переменную с именем `my_dataset`. Переменные также обладают «переменчивой» природой в том смысле, что мы можем изменять хранящиеся в них значения, например:

Пример:

```
my_dataset = 8 + 8
```

Теперь в переменной `my_dataset` хранится значение 16.

Важно отметить, что оператор равенства (=) в Python выполняет не ту же функцию, что в математике. В Python он используется для присвоения значения переменной, но не следует математической логике. Если вы хотите вычислить значение арифметического выражения в Python, вы можете просто выполнить код без добавления в него оператора равенства.

Пример:

```
2 + 2
```

В данном случае будет возвращено значение 4.

Если вы хотите проверить, является ли некоторое математическое отношение истинным или ложным, вы можете использовать оператор ==.

Пример:

```
2 + 2 == 4
```

В данном случае будет возвращено значение `True` (истина).

Импорт библиотек

Язык Python позволяет заниматься множеством вещей, от веб-скрейпинга до создания игровых приложений, однако написание кода с нуля — это сложный и трудоемкий процесс. Именно здесь на помощь приходят библиотеки, представляющие собой набор готовых фрагментов кода и стандартных процедур. Вместо того чтобы писать десятки строк кода для построения простого графика или извлечения данных с веб-сайтов, вы можете выполнить уже готовую функцию из соответствующей библиотеки, используя всего одну строку кода.

Существует огромное количество бесплатных библиотек, предназначенных для веб-скрейпинга, визуализации и анализа данных и решения других задач. Наиболее популярные библиотеки для машинного обучения — Scikit-learn, Pandas и NumPy. Библиотеки NumPy и Pandas можно импортировать с помощью одной строки кода, в то время как для импорта алгоритмов или функций Scikit-learn вам придется указать их по отдельности в нескольких строках кода.

Пример:

```
import numpy as np
import pandas as pd
from sklearn.neighbors import NearestNeighbors
```

Этот фрагмент позволяет вам использовать функции из библиотек NumPy и Pandas, а также алгоритм *k*-ближайших соседей из Scikit-learn, путем вызова `np`, `pd` и `NearestNeighbors` в любом разделе кода, расположенном ниже. Команды импорта других алгоритмов Scikit-learn и различных библиотек вы можете найти, обратившись к соответствующей документации, доступной в Интернете.

Импорт набора данных

Набор данных в формате CSV можно импортировать в среду разработки Python в виде фрейма данных Pandas (в виде таблицы) из хост-файла

с помощью команды Pandas `pd.read_csv()`. Обратите внимание, что имя хост-файла должно быть указано внутри круглых скобок и заключено в одинарные или двойные кавычки.

Вам также нужно присвоить набору данных переменную с помощью оператора равенства, чтобы получить возможность обращаться к этому набору в других разделах кода. Это означает, что при каждом вызове, например `dataframe`, интерпретатор Python будет знать, что вы обращаетесь к набору данных, импортированному и сохраненному в переменной с соответствующим именем.

Пример:

```
dataframe = pd.read_csv('~/Downloads/
                         Melbourne_housing_FULL.csv')
```

Вывод данных на экран

Функция `print()`, предназначенная для вывода на экран сообщения, заключенного в круглые скобки, — одна из самых часто используемых в языке Python. Из-за своего простого назначения она может показаться не особенно важной с точки зрения программирования. Но это ошибочное впечатление.

Во-первых, функция `print()` полезна для отладки (поиска и исправления ошибок в коде). Например, с помощью этой функции можно проверить значение переменной после его изменения.

Ввод:

```
my_dataset = 8
my_dataset = 8 + 8
print(my_dataset)
```

Вывод:

```
16
```

Еще один распространенный случай использования этой функции — вывод на экран не подлежащей обработке информации в виде строки. Это означает, что выражение/строка, заключенная в круглые скобки, просто выводится на экран и не взаимодействует с другими элементами кода.

Это позволяет добавить контекст и прояснить фрагменты кода с помощью подписей, что весьма полезно, учитывая то, что комментарии (#) на экран не выводятся.

Ввод:

```
print ("Training Set Mean Absolute Error: %.2f" % mae_train)
```

Вывод:

```
Training Set Mean Absolute Error: 27834.12
```

В этом примере оператор `print` информирует пользователя о том, что именно было обработано интерпретатором Python и выведено на экран. Без фрагмента `print("Test Set Mean Absolute Error:")` после выполнения кода мы бы увидели лишь числа без каких-либо подписей.

Учтите, что строка внутри круглых скобок должна быть заключена в двойные (" ") или одинарные (' ') кавычки. Смешивание кавычек считается ошибкой, то есть нельзя начинать строку, например, с одинарной кавычки и завершать ее двойной. Оператор `print` автоматически удаляет кавычки после выполнения кода. Если вы хотите включить кавычки в вывод, можно использовать одинарные кавычки внутри двойных кавычек, как показано ниже:

Ввод:

```
print("'Test Set Mean Absolute Error'")
```

Вывод:

```
'Test Set Mean Absolute Error'
```

Индексирование

Индексирование — это метод выбора отдельного элемента, например, из списка или строки. Каждый элемент имеет числовой индекс, начиная с 0, и к нему можно обратиться, указав соответствующее число в квадратных скобках.

Пример:

```
my_string = "hello_world"
my_string[1]
```

В этом примере будет возвращено значение **e**.

Элемент: h e l l o _ w o r l d
Индекс: 0 1 2 3 4 5 6 7 8 9 10

Пример:

```
my_list = [10, 20, 30, 40]
my_list[0]
```

В этом примере будет возвращено значение **10**.

Элемент: 10 20 30 40
Индекс: 0 1 2 3

Нарезка

Вместо извлечения отдельного элемента из коллекции вы можете выделить определенный подраздел с помощью двоеточия (:).

Пример:

```
my_list = [10, 20, 30, 40]
my_list[:3]
```

В этом примере подраздел заканчивается элементом с индексом 3, но не включает его, поэтому возвращаются значения **10**, **20** и **30**.

Пример:

```
my_list = [10, 20, 30, 40]
my_list[1:3]
```

Здесь подраздел начинается с элемента с индексом 1, а заканчивается элементом с индексом 3, но не включает его, поэтому возвращаются значения **20** и **30**.

ДРУГИЕ КНИГИ АВТОРА

Python for Absolute Beginners
Научиться писать код непросто. Если вы хотите познакомиться с языком Python, эта книга станет для вас идеальным пошаговым руководством.

Machine Learning with Python for Beginners
Поднимитесь на новый уровень в машинном обучении, научившись создавать собственные предсказательные модели и решать реальные проблемы с помощью языка Python.

Machine Learning: Make Your Own Recommender System
Создайте собственную систему рекомендаций за один день с помощью методов машинного обучения и языка Python.

Data Analytics for Absolute Beginners
Научитесь принимать более взвешенные решения, используя все имеющиеся переменные с помощью этого руководства по анализу данных.

Statistics for Absolute Beginners
Освойте основы инференциальной и описательной статистики с помощью практических демонстраций и наглядных примеров.

Курс на платформе Skillshare

Introduction to Machine Learning Concepts for Absolute Beginners
Этот курс представляет основы машинного обучения в видеоформате. После его прохождения вы сможете перейти к более сложным видеокурсам, доступным на платформе Skillshare.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Производственно-практическое издание

МИРОВОЙ КОМПЬЮТЕРНЫЙ БЕСТСЕЛЛЕР

Оливер Теобальд

МАШИННОЕ ОБУЧЕНИЕ ДЛЯ АБСОЛЮТНЫХ НОВИЧКОВ

ВВОДНЫЙ КУРС, ИЗЛОЖЕННЫЙ ПРОСТЫМ ЯЗЫКОМ

Главный редактор *Р. Фасхутдинов*
Руководитель направления *В. Обручев*
Ответственный редактор *Л. Салихова*
Научный редактор *Н. Белявская*
Литературный редактор *Н. Домнина*
Младший редактор *П. Смирнов*
Художественный редактор *В. Брагина*
Компьютерная верстка *Э. Брегис*
Корректоры *А. Баскакова*, *Л. Макарова*

Страна происхождения: Российская Федерация
Шығарылған елі: Ресей Федерациясы

ООО «Издательство «Эксмо»
123308, Россия, г. Москва, ул. Зорге, д. 1, стр. 1, эт. 20, каб. 2013. Тел.: 8 (495) 411-68-86.
Home page: www.eksmo.ru E-mail: info@eksmo.ru
Өндіруші: «Издательство «Эксмо» ЖШҚ
123308, Ресей, Мәскеу қаласы, Зорге көшесі, 1-үй, 1-құрылыс, 20 қабат, 2013-каб.
Тел.: 8 (495) 411-68-86. Home page: www.eksmo.ru E-mail: info@eksmo.ru.
Тауар белгісі: «Эксмо»

Интернет-магазин : www.book24.ru
Интернет-магазин : www.book24.kz
Интернет-дукен : www.book24.kz

Импортёр в Республику Казахстан ТОО «РДЦ-Алматы».
Қазақстан Республикасына импорттаушы «РДЦ-Алматы» ЖШС.

Дистрибьютор и представитель по приему претензий на продукцию в Республике Казахстан: ТОО «РДЦ-Алматы»
Дистрибьютор және Қазақстан Республикасында өнімге шағымдар қабылдау жөніндегі өкіл: «РДЦ-Алматы» ЖШС.
Алматы қ., Домбровский көш., 3 «а», литер Б, офис 1.
Тел.: 8 (727) 251-59-90/91/92. E-mail: RDC-Almaty@eksmo.kz

Сведения о подтверждении соответствия издания согласно законодательству РФ о техническом регулировании можно получить на сайте Издательства «Эксмо»: www.eksmo.ru/certification

Техникалық реттеу туралы РФ заңнамасына сай басылымның сәйкестігін растау туралы мәліметтерді мына адрес бойынша алуға болады: http://eksmo.ru/certification/

Произведено в Российской Федерации
Ресей Федерациясында өндірілген

Сертификаттауға жатпайды

12+

Дата изготовления / Подписано в печать 27.04.2024. Формат 70x100 1/16.
Печать офсетная. Усл. печ. л. 16,85.
Тираж экз. Заказ

БОМБОРА
ИЗДАТЕЛЬСТВО

БОМБОРА – лидер на рынке полезных и вдохновляющих книг.
Мы любим книги и создаем их, чтобы вы могли творить, открывать мир, пробовать новое, расти. Быть счастливыми. Быть на волне.

bombora.ru bomborabooks bombora

ISBN 978-5-04-190305-3

Литрес
Я ТАК ЧИТАЮ

ЧИТАЙ ГОРОД

Москва. ООО «Торговый Дом «Эксмо»
Адрес: 123308, г. Москва, ул. Зорге, д.1, строение 1.
Телефон: +7 (495) 411-50-74. **E-mail:** reception@eksmo-sale.ru

По вопросам приобретения книг «Эксмо» зарубежными оптовыми покупателями обращаться в отдел зарубежных продаж ТД «Эксмо»
E-mail: **international@eksmo-sale.ru**

International Sales: International wholesale customers should contact Foreign Sales Department of Trading House «Eksmo» for their orders.
international@eksmo-sale.ru

По вопросам заказа книг корпоративным клиентам, в том числе в специальном оформлении, обращаться по тел.: +7 (495) 411-68-59, доб. 2151.
E-mail: **borodkin.da@eksmo.ru**

Оптовая торговля бумажно-беловыми и канцелярскими товарами для школы и офиса «Канц-Эксмо»:
Компания «Канц-Эксмо»: 142702, Московская обл., Ленинский р-н, г. Видное-2, Белокаменное ш., д. 1, а/я 5. Тел./факс: +7 (495) 745-28-87 (многоканальный).
e-mail: **kanc@eksmo-sale.ru**, сайт: www.**kanc-eksmo.ru**

Филиал «Торгового Дома «Эксмо» в Нижнем Новгороде
Адрес: 603094, г. Нижний Новгород, улица Карпинского, д. 29, бизнес-парк «Грин Плаза»
Телефон: +7 (831) 216-15-91 (92, 93, 94). **E-mail**: reception@eksmonn.ru

Филиал ООО «Издательство «Эксмо» в г. Санкт-Петербурге
Адрес: 192029, г. Санкт-Петербург, пр. Обуховской обороны, д. 84, лит. «Е»
Телефон: +7 (812) 365-46-03 / 04. **E-mail**: server@szko.ru

Филиал ООО «Издательство «Эксмо» в г. Екатеринбурге
Адрес: 620024, г. Екатеринбург, ул. Новинская, д. 2щ
Телефон: +7 (343) 272-72-01 (02/03/04/05/06/08)

Филиал ООО «Издательство «Эксмо» в г. Самаре
Адрес: 443052, г. Самара, пр-т Кирова, д. 75/1, лит. «Е»
Телефон: +7 (846) 207-55-50. **E-mail**: RDC-samara@mail.ru

Филиал ООО «Издательство «Эксмо» в г. Ростове-на-Дону
Адрес: 344023, г. Ростов-на-Дону, ул. Страны Советов, 44А
Телефон: +7(863) 303-62-10. **E-mail**: info@rnd.eksmo.ru

Филиал ООО «Издательство «Эксмо» в г. Новосибирске
Адрес: 630015, г. Новосибирск, Комбинатский пер., д. 3
Телефон: +7(383) 289-91-42. E-mail: eksmo-nsk@yandex.ru

Обособленное подразделение в г. Хабаровске
Фактический адрес: 680000, г. Хабаровск, ул. Фрунзе, 22, оф. 703
Почтовый адрес: 680020, г. Хабаровск, А/Я 1006
Телефон: (4212) 910-120, 910-211. **E-mail**: eksmo-khv@mail.ru

Республика Беларусь: ООО «ЭКСМО АСТ Си энд Си»
Центр оптово-розничных продаж Cash&Carry в г. Минске
Адрес: 220014, Республика Беларусь, г. Минск, проспект Жукова, 44, пом. 1-17, ТЦ «Outleto»
Телефон: +375 17 251-40-23; +375 44 581-81-92
Режим работы: с 10.00 до 22.00. **E-mail:** exmoast@yandex.by

Казахстан: «РДЦ Алматы»
Адрес: 050039, г. Алматы, ул. Домбровского, 3А
Телефон: +7 (727) 251-58-12, 251-59-90 (91,92,99). E-mail: RDC-Almaty@eksmo.kz

Полный ассортимент продукции ООО «Издательство «Эксмо» можно приобрести в книжных магазинах «Читай-город» и заказать в интернет-магазине: www.chitai-gorod.ru.
Телефон единой справочной службы: 8 (800) 444-8-444. Звонок по России бесплатный.

Интернет-магазин ООО «Издательство «Эксмо»
www.eksmo.ru
Розничная продажа книг с доставкой по всему миру.
Тел.: +7 (495) 745-89-14. E-mail: **imarket@eksmo-sale.ru**

eksmo.ru
Официальный интернет-магазин издательства «Эксмо»

Хочешь стать автором «Эксмо»?

ТЕРИТОРИЯ КНИЖНЫЙ МАГАЗИН
Официальная франшиза издательства «Эксмо»

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

Простые и понятные объяснения и отсутствие необходимости опыта программирования делают эту книгу прекрасной альтернативой академическому учебнику. Здесь представлены основные алгоритмы машинного обучения (ML), которые сопровождаются наглядными примерами и практическими работами. Также вы узнаете про перекрестную проверку, ансамблевое моделирование, поиск по сетке для настройки моделей, проектирование функций, горячее кодирование и многое другое.

Для разработки интеллектуальных машин в первую очередь надо понять классическую статистику, так как алгоритмы на ее основе — это сердце машинного обучения. Написание кода — еще одна неотъемлемая часть ML, которая предусматривает управление данными. Однако материал этого руководства можно освоить даже без навыков программирования.

Возможно, с чтения этой книги начнется ваш путь к получению работы в области машинного обучения, а может быть, она просто удовлетворит ваше любопытство.

Самое главное:

- Загрузка бесплатных наборов данных
- Методы очистки данных, включая горячее кодирование, группирование и обработку недостающих данных
- Подготовка данных для анализа
- Линейный регрессионный анализ
- Кластеризация, включая кластеризацию k-средних
- Основы работы нейронных сетей
- Смещение/дисперсия для улучшения модели машинного обучения
- Деревья решений для декодирования классификации
- Ваша первая модель машинного обучения с помощью Python

Об авторе

ОЛИВЕР ТЕОБАЛЬД — технический писатель, специализирующийся на темах искусственного интеллекта, финансовых технологий и облачных вычислений. Автор книг Python for Absolute Beginners, Machine Learning with Python for Beginners, Data Analytics for Absolute Beginners и др.

ISBN 978-5-04-190305-3

БОМБОРА
ИЗДАТЕЛЬСТВО

БОМБОРА — лидер на рынке полезных и вдохновляющих книг.
Мы любим книги и создаем их, чтобы вы могли творить, открывать мир, пробовать новое, расти. Быть счастливыми. Быть на волне.