ГЛАВА 8. ПОДВЕДЕНИЕ ИТОГОВ

Итак, теперь вы умеете применять важнейшие алгоритмы контролируемого и неконтролируемого машинного обучения, которые позволяют решать широкий спектр задач. Прежде чем оставить вас наедине с этой книгой для самостоятельного исследования всех тех возможностей, которые предлагает машинное обучение, мы хотим дать несколько заключительных советов, рассказать о некоторых дополнительных ресурсах и дать рекомендации по поводу того, как можно дополнительно улучшить навыки в области машинного обучения и науки о данных.

Общий подход к решению задач машинного обучения

Теперь, когда все эти замечательные методы, о которых мы рассказали в этой книге, находятся в вашем распоряжении, возникает соблазн, чтобы сразу, пропустив несколько важных моментов, приступить к решению конкретных задач, просто запустив свой любимый алгоритм. Однако это, как правило, не лучший способ начать анализ. Обычно алгоритм машинного обучения – это лишь небольшая деталь более серьезного процесса анализа данных и принятия решений. Чтобы эффективно использовать машинное обучение, нам нужно сделать шаг назад и рассмотреть задачу в целом. Во-первых, вам стоит подумать о том, на какой вопрос вы хотите ответить. Вы хотите провести разведочный анализ и выяснить, содержат ли данные что-то интересное? Или у вас уже есть конкретная цель? Как правило, вы сначала формулируете цель, например, обнаружение мошеннических транзакций, получение рекомендаций по фильмам или поиск неизвестных планет. Если у вас уже есть такая цель, прежде чем строить систему машинного обучения для ее реализации, вы должны сначала подумать о том, как определить и измерить эффективность достижения цели и какое влияние окажет это эффективное решение на ваши деловые или научные цели. Допустим, ваша цель заключается в том, чтобы обнаружить мошенничество.

Тогда необходимо прояснить следующие вопросы:

- Как определить, что моя система прогнозирования мошенничества на самом деле работает?
- Есть ли у меня подходящие данные для оценки качества алгоритма?
- Если я получил эффективное решение, каким будет его влияние на бизнес?

Как мы уже говорили в главе 5, лучше всего измерять качество алгоритма с использованием бизнес-метрики (увеличение прибыли или снижение убытков). Однако это часто трудно сделать. Вопрос, на который легче будет ответить, звучит так: «Что если я построю

идеальную модель?» Если модель, идеально определяющая любые мошеннические операции, позволит вашей компании экономить 100$ в месяц, вероятно, этих средств будет недостаточно, чтобы оправдать усилия, направленные на разработку алгоритма. С другой стороны, если модель позволит вашей компании экономить десятки тысяч долларов ежемесячно, задача стоит того, чтобы ее решать.

Допустим, вы определили задачу, которую нужно решить, вы знаете, что решение может иметь значительное влияние для вашего проекта, и вы убедились, что у вас есть необходимая информация, позволяющая измерить эффективность решения. Следующие шаги – это получение данных и построение рабочего прототипа. В этой книге мы рассказывали о различных моделях, которыми вы можете воспользоваться, а также о том, как правильно оценить качество модели и настроить ее параметры. Однако экспериментируя с выбором моделей, помните, что модель – это лишь небольшая деталь большого рабочего потока и построение модели, как правило, является частью обратного цикла, включающего сбор новых данных, очистки данных, построения моделей и анализа моделей. Часто анализ ошибок, допущенных моделью, позволяет получить информацию о том, что пропущено в данных, какие дополнительные данные можно еще собрать или как можно переформулировать задачу, чтобы повысить эффективность машинного обучения. Сбор большего количества данных или данных из других источников, незначительное изменение формулировки задачи, возможно, позволят получить гораздо б*о*льшую отдачу, чем запуск бесконечных процедур решетчатого поиска для настройки параметров.

Вмешательство человека в работу модели

Кроме того, вы должны рассмотреть вопрос о том, будут ли люди вмешиваться в работу модели и если да, то как это будет реализовано.[49] Некоторые процессы (например, обнаружение пешеходов беспилотными автомобилями) требуют немедленного принятия решений. Другие процессы, возможно, не требуют немедленной реакции и поэтому человеку предоставляется возможность подтвердить те или иные решения. Например, в медицине, может понадобиться очень высокий

[49] Авторы имеют в виду построение модели «человек-в-системе-управления» (human-in-the-loop). Термин пришел из теории искусственного интеллекта и робототехники. Все роботы обладают автономностью, то есть способны осуществлять какие-либо действия без вмешательства человека. Степень автономности роботов определяется заложенной моделью. Условно можно выделить три типа модели: модель «человек-в-системе-управления» (human-in-the-loop), когда окончательное решение принимает только человек, модель «человек-над-системой-управления» (human-on-the-loop), когда решения принимает система, но человек, выполняющий роль наблюдателя, всегда может вмешаться в процесс, и модель «человек-вне-системы-управления» (human-out-of-the-loop), когда система может принимать решения вообще без человеческого вмешательства. – *Прим. пер.*

уровень точности, которого невозможно добиться с помощью лишь одного алгоритма машинного обучения. Но если алгоритм может принять 90%, 50% или, возможно, даже только 10% решений автоматически, то можно увеличить время отклика или снизить издержки. Как правило, большая часть примеров – это «простые случаи», по которым алгоритм может принять решение, а относительно небольшая доля примеров представляет собой «сложные случаи», которые можно перенаправить человеку.

От прототипа к производству

Инструменты, которые мы обсудили в этой книге? прекрасно подходят для различных проектов машинного обучения и позволяют очень быстро проводить анализ и прототипирование. Кроме того, Python и библиотека `scikit-learn` используется в производственных системах различных организаций, в том числе очень крупных, например, в производственных системах международных банков и глобальных медиакомпаний. Однако многие организации имеют сложную инфраструктуру и интеграция Python в эти системы далеко не всегда является легким процессом. Впрочем, это не является неопреодолимой проблемой. Во многих компаниях команды аналитиков используют языки типа Python и R, которые позволяют осуществить быстрое тестирование идей, тогда как команды, занимающиеся внедрением моделей в производственный процесс, работают с такими языками, как Go, Scala, C++, Java, позволяющими строить надежные, масштабируемые системы. Анализ данных и построение надежно работающих сервисов – совершенно разные задачи с разными требованиями, поэтому использование различных языков для решения этих задач имеет смысл. Относительно распространенное решение – реализовать модель, найденную командой аналитиков, в рамках более крупной платформы, используя высокопроизводительный язык. Это проще, чем встраивание целой библиотеки или языка программирования и бесконечные преобразования данных из одного формата в другой.

Вне зависимости от того, будете ли вы использовать `scikit-learn` в производственной системе или нет, важно помнить, что в отличие от скриптов, использующихся для выполнения одноразового анализа, к производственным системам предъявляются совершенно другие требования. Если алгоритм встроен в более крупную систему, то актуальными становятся такие аспекты программного обеспечения, как надежность, предсказуемость, время запуска и требования к памяти. Простота – ключевый фактор, позволяющий построить эффективные системы машинного обучения. Внимательно исследуйте каждый этап

вашего конвейера, предназначенного для обработки данных и прогнозирования, и спросите себя, в какой мере каждый этап увеличивает сложность, насколько каждый компонент устойчив к изменениям в данных или вычислительной инфраструктуре, и оправдана ли сложность каждого этапа. Если вы строите сложные системы машинного обучения, мы настоятельно рекомендуем прочитать статью Machine Learning: The High Interest Credit Card of Technical Debt, опубликованную специалистами по машинному обучению компании Google. В статье особое внимание уделено созданию и сопровождению программного обеспечения, которое предназначено для применения машинного обучения в крупномасштабных проектах. Проблема «технического долга»[50] особенно актуальна для крупных и долгосрочных проектов, однако уроки, извлеченные из статьи, помогут вам создать более качественное программное обеспечение даже для краткосрочных и небольших систем.

Тестирование производственных систем

В этой книге мы рассказали о том, как оценить качество прогнозов, используя заранее подготовленный тестовый набор. Данная процедура известна как *оценка оффлайн* (*offline evaluation*). Однако если ваша система машинного обучения взаимодействует с пользователем, это лишь первый этап оценки качества алгоритма, Следующим шагом, как правило, является *онлайн-тестирование* (*online testing*) или *тестирование в реальных условиях* (*live testing*), где оцениваются последствия от применения алгоритма в общей системе. Изменение рекомендаций или результатов поиска, отображаемых веб-сайтом, может коренным образом изменить поведение пользователей и привести к неожиданным последствиям. Для защиты от таких сюрпризов, разработчики интерактивных сервисов используют *A/B тестирование* (*A/B testing*), форму слепого тестирования пользователей. В A/B тестировании определенной группе пользователей без их ведома предъявляется веб-сайт или сервис, использующий алгоритм A, а другой группе пользователей предъявляется веб-сайт или сервис, использующий алгоритм B. В обоих группах фиксируются релевантные метрики эффективности за определенный период времени. Затем метрики для алгоритмов A и B сравниваются, выбор оптимального алгоритма

[50] Технический долг или долг кодинга (technical debt) — это метафора-неологизм, обозначающая плохую продуманность структуры системы, непродуманную архитектуру программного обеспечения или некачественную разработку ПО. Долг может рассматриваться в виде работы, которую необходимо проделать, пока задача не сможет считаться выполненной. Если долг не погашается, то он будет продолжать увеличиваться, что усложнит дальнейшую разработку. – *Прим. пер.*

осуществляется согласно этим показателям. Использование A/B тестирования позволяет нам оценить качество алгоритма в реальных условиях и обнаружить различные непредвиденные последствия, когда пользователи взаимодействуют с нашей моделью. Как правило, A – это новая модель, тогда как B – это действующая система. Существуют и более сложные механизмы онлайн-тестирования, выходящие за рамки A/B тестирования, например, *бандитские алгоритмы* (*bandit algorithms*). Замечательным руководством по этой теме является книга издательства O'Reilly *Bandit Algorithms for Website Optimization* за авторством Джона Майлса Уайта.

Создание своего собственного класса Estimator

Эта книга охватывает множество инструментов и алгоритмов библиотеки `scikit-learn`, которые можно использовать для широкого круга задач. Однако часто вам потребуются какие-то конкретные процедуры предварительной обработки, которые не реализованы в `scikit-learn`. Достаточно лишь предварительно обработать данные перед тем как передать их в модель `scikit-learn` или конвейер. Однако, если ваша предварительная обработка зависит от данных и вы хотите применить решетчатый поиск или перекрестную проверку, все становится сложнее. В главе 6 мы обсуждали важность размещения всех процедур обработки, зависящих от данных, внутри цикла перекрестной проверки. Так как же можно использовать собственные процедуры обработки наряду с инструментами `scikit-learn`? Существует простое решение: построить свою собственную модель! Реализация модели, которая будет совместима с интерфейсом `scikit-learn` (и таким образом ею можно будет воспользоваться с помощью классов `Pipeline`, `GridSearch` и функции `cross_val_score`), выглядит довольно просто. Вы можете найти подробные инструкции в документации по `scikit-learn`, здесь же приводится самая суть. Самый простой способ реализовать класс, выполняющий преобразование – воспользоваться наследованием, то есть на основе базовых классов `BaseEstimator` и `TransformerMixin` создать специализированный класс, а затем создать функции `__init__`, `fit` и `predict`, как показано в нижеприведенном программном коде:

```
In[1]:
from sklearn.base import BaseEstimator, TransformerMixin

class MyTransformer(BaseEstimator, TransformerMixin):
    def __init__(self, first_parameter=1, second_parameter=2):
        # Все параметры должны быть заданы в функции __init__
        self.first_parameter = 1
        self.second_parameter = 2

    def fit(self, X, y=None):
        # fit должна принимать в качестве аргументов только X и y
        # Даже если ваша модель является неконтролируемой, вы должны принять аргумент y!

        # Подгонка модели осуществляется прямо здесь
        print("подгоняем модель прямо здесь")
        # fit возвращает self
        return self

    def transform(self, X):
        # transform принимает в качестве аргумента только X

        # Применяем преобразование к X
        X_transformed = X + 1
        return X_transformed
```

Реализация собственного классификатора или регрессора выглядит аналогично, только вместо `TransformerMixin` вам нужно наследовать от классов `ClassifierMixin` или `RegressorMixin`. Кроме того, вместо `transform` вы можете реализовать `predict`.

Как видно из примера, приведенного здесь, реализация своей собственной модели требует незначительного объема программного кода, и большинство пользователей `scikit-learn` со временем создают целый набор собственных моделей.

Куда двигаться дальше

Эта книга представляет собой введение в машинное обучение и позволит вам стать эффективным специалистом. Однако если вы хотите совершенствовать навыки машинного обучения, здесь даются некоторые книги и специализированные ресурсы для более глубокого изучения.

Теория

В этой книге мы попытались дать вам представление о работе наиболее часто используемых алгоритмов машинного обучения, не требуя от вас прочных знаний в области математики или компьютерной науки. Однако многие рассмотренные нами модели используют принципы, взятые из теории вероятностей, линейной алгебры и методов оптимизация. Хотя понимание всех деталей этих алгоритмов не является обязательным, мы считаем, что знание некоторых теорий, лежащих в основе рассмотренных алгоритмов, позволят вам стать хорошим специалистом по анализу данных. По теории машинного обучения написана масса хороших книг и

если бы мы могли заинтересовать вас теми возможностями, которые открывает машинное обучение, мы предложили бы вам выбрать по крайней мере одну из них и уйти в нее с головой. Мы уже упоминали в предисловии книгу Хасти, Тибширани и Фридмана *The Elements of Statistical Learning*, однако стоит повторить эту рекомендацию здесь. Еще одной вполне доступной книгой, к которой прилагаются примеры программного кода Python, является книга Стивена Марсланда *Machine Learning: An Algorithmic Perspective* (издательство Chapman and Hall/CRC). Еще две классические книги, которые настоятельно рекомендуются к прочтению – это книга Кристофера Бишопа *Pattern Recognition and Machine Learning* (издательство Springer), в ней особое внимание уделяется вероятности, и книга Кевина Мерфи *A Probabilistic Persprective* (издательство MIT Press), исчерпывающая научная работа (объемом более 1000 страниц) по методам машинного обучения, которая включает детальное рассмотрение новейших методов и выходит далеко за рамки того, что мы могли охватить в этой книге.

Другие фреймворки и пакеты машинного обучения

Несмотря на то что `scikit-learn` – это наш любимый пакет для машинного обучения[51], а Python является нашим любимым языком машинного обучения, существует еще масса пакетов за пределами среды Python. В силу тех или иных задач, стоящих перед вами, вполне возможна ситуация, когда Python и `scikit-learn` не будут являться оптимальным выбором. Как правило, Python отлично подходит для апробации и оценки качества моделей, но крупные веб-сервисы и приложения чаще всего написаны на Java или C ++ и для развертывания вашей модели может потребоваться ее интеграция в эти системы. Еще одна причина, по которой вы, возможно, не захотите ограничиваться рамками `scikit-learn` – ситуация, когда вас больше будут интересовать не прогнозы, а возможности статистического моделирования и статистического вывода. В этом случае вы должны обратить внимание на питоновский пакет `statsmodel`, в нем реализовано несколько линейных моделей с интерфейсом, в большей степени ориентированным на статистиков. Если вы не являетесь фанатом Python, вы можете также рассмотреть вопрос об использовании R, еще одном универсальном языке, который используют специалисты по работе с данными. R – язык, разработанный специально для статистического анализа, он славится своими превосходными возможностями визуализации и большим количеством пакетов для статистического моделирования (часто узкоспециализированных).

[51] Андреас не может быть полностью объективным в этом вопросе.

Еще популярный пакет для машинного обучения – `vowpal wabbit` (часто называемый `vw`, чтобы не сломать язык), высоко оптимизированный пакет машинного обучения, написанный на C++, с интерфейсом командной строки. `vw` особенно полезен для больших массивов данных и для потоковой передачи данных. Для распределенного запуска алгоритмов машинного обучения на кластере одним из самых популярных решений на момент написания книги была `mllib`, библиотека Scala, реализованная на базе `spark`, среды распределенных вычислений.

Ранжирование, рекомендательные системы и другие виды обучения

Поскольку данная книга является вводной, мы сосредоточили свое внимание на наиболее распространенных задачах машинного обучения: задачах классификации и регрессии для машинного обучения с учителем, а также задачах кластеризации и декомпозиции сигнала для машинного обучения без учителя. Существует масса других видов машинного обучения с различными задачами, которые не были рассмотренны нами в этой книге. Существует две особенно важные темы, которые мы не охватили в этой книге. Первая тема – это *ранжирование* (*ranking*), когда мы хотим получить ответы на конкретный запрос, упорядоченные по их релевантности. Сегодня вы уже наверняка использовали систему ранжирования, она лежит в основе любой поисковой системы. Вы вводите поисковый запрос и получаете отсортированный список ответов, ранжированных по степени их релевантности. Замечательным вводным пособием, посвященным вопросам ранжирования, является книга Мэннинга, Рагхавана и Шютце *Introduction to Information Retrieval*. Вторая тема – *рекомендательные системы* (*recommender systems*), которые делают предложения пользователям в зависимости от их предпочтений. Вы, наверное, уже сталкивались с рекомендательными системами типа «Люди, которых вы можете знать», «Покупатели, которые купили этот товар, также покупают» или «Лучшие предложения для вас». Существует масса литературы по этой теме, и если вы хотите глубже изучить ее, вас, возможно, заинтересует конкурс «Netflix Prize challenge». В рамках этого конкурса потоковый видеосервис Netflix выложил на своем сайте большой набор данных, содержащий кинопредпочтения пользователей, и предложил приз 1 млн $ команде, которая сможет предложить наилучшую рекомендательную систему. Еще одна распространенная задача машинного обучения – это прогнозирование временных рядов (например, цен на акции), по которой также существует целый раздел литературы. Задач машинного обучения

гораздо больше, чем мы можем перечислить здесь, и мы рекомендуем вам обратиться за информацией к книгам, научным статьям и онлайн-сообществам, чтобы найти подходы, которые будут максимально применимы к вашей ситуации.

Вероятностное моделирование, теория статистического вывода и вероятностное программирование

Большинство пакетов машинного обучения предлагают уже готовые модели машинного обучения на базе какого-то одного конкретного алгоритма. Однако многие реальные задачи имеют определенную структуру и при условии, что эта структура будет правильно представлена в модели, мы можем получить прогнозы гораздо лучшего качества. Часто структуру конкретной задачи можно выразить в терминах теории вероятностей. Получение такой структуры становится возможным благодаря наличию математической модели прогнозируемой ситуации. Для пояснения того, что мы подразумеваем под структурированной задачей, рассмотрим следующий пример.

Допустим, вы хотите создать мобильное приложение, которое позволяет очень точно определить положение в открытом пространстве и тем самым помогает пользователям перемещаться по историческим достопримечательностям. Мобильный телефон снабжен множеством датчиков для точного определения местоположения (GPS, акселерометр, компас и т.д.). У вас также есть точная карта местности. Эта задача является хорошо структурированной. Благодаря карте вы знаете, где проходят интересующие вас маршруты и где находятся интересующие вас объекты. Кроме того, у вас есть грубые оценки местоположения, полученные с помощью GPS, а акселерометр и компас, имеющиеся в телефоне, позволяют вам получить относительно точные измерения. Однако загрузить все эти данные в систему машинного обучения, использующую алгоритм «черного ящика», будет не самой лучшей идеей. Это все равно что выбросить всю фактическую информацию, которую вы уже знали. Если компас и акселерометр говорит вам, что пользователь направляется на север, а GPS сообщает, что пользователь направляется на юг, вам, вероятно, не стоит доверять GPS. Если оценка местоположения говорит вам, что пользователь просто прошел сквозь стену, вы также должны быть настроены весьма скептически. Можно представить эту ситуацию с помощью вероятностной модели, а затем использовать машинное обучение или теорию статистического вывода, чтобы выяснить, в какой степени вам следует доверять каждому измерению и затем строить предположения по поводу того, что представляет собой наиболее точная оценка местоположения.

Как только вы дали правильное описание ситуации и создали корректную вероятностную модель взаимодействия различных факторов, к вашим услугам – специальные методы, которые вычисляют прогнозы, непосредственно используя эти модели. Большинство этих методов реализованы с помощью языков вероятностного программирования, которые позволяют очень элегантно и компактно описать изучаемую задачу. Примерами популярных языков вероятностного программирования являются `PyMC` (который можно использовать в Python) и `Stan` (фреймворк, в котором можно использовать несколько языков, в том числе Python). Хотя эти пакеты и требуют определенного понимания теории вероятностей, они значительно упрощают создание новых моделей значительно.

Нейронные сети

Несмотря на то что мы затронули тему нейронных сетей лишь кратко в главах 2 и 7, данное направление является быстро развивающейся областью машинного обучения, в рамках которой мы каждую неделю узнаем о новых достижениях и новых сферах применения нейронных сетей. Последние успехи в области машинного обучения и искусственного интеллекта, например победа программы Alpha Go над чемпионом по игре Go, постоянное улучшение качества распознавания речи, возможность почти мгновенного перевода речи, полностью обусловлен этими достижениями. Хотя прогресс в этой области движется столь быстрыми темпами, что любая ссылка на передовой метод вскоре устаревает, недавно вышедшая книга *Deep Learning* за авторством Яна Гудфеллоу, Йошуа Бенгио и Аарона Курвилля (издательство MIT Press) представляет собой комплексное введение в тему нейронных сетей.[52]

Масштабирование на больших наборах данных

В этой книге принято, что обрабатываемые данные можно хранить в оперативной памяти в виде массива NumPy или разреженной матрицы SciPy. Даже несмотря на то что современные серверы часто оснащены сотнями гигабайт (ГБ) оперативной памяти, память является фундаментальным ограничением, накладываемым на размер обрабатываемых данных. Не каждый может позволить себе купить такой большой сервер, или даже арендовать у провайдера облачных услуг. Однако в большинстве проектов данные, которые используются для построения системы машинного обучения, имеют относительно небольшой объем. Наборов, состоящих из сотен гигабайтов данных и

[52] Препринт книги *Deep Learning* можно посмотреть по адресу http://www.deeplearningbook.org/.

более, довольно мало. Данный факт позволяет считать, что во многих случаях увеличение объема памяти или аренда сервера у провайдера облачных услуг является жизнеспособным решением. Однако если вам необходимо обработать терабайт данных или вам нужно обработать большие объемы данных при ограниченном бюджете, существует две базовые стратегии: *out-of-core learning* (*обучение во внешней памяти*) и *parallelization over a cluster* (*распараллеливание на кластере*).

Обучение во внешней памяти – это обучение на основе данных, которые не могут быть сохранены в основной памяти, но при этом процесс осуществляется на одном компьютере (при этом может использоваться даже одноядерный процессор компьютера). Данные считываются из источника типа жесткого диска или сети либо по одному примеру за проход, либо в виде блоков, состоящих из нескольких примеров, с тем чтобы поместить каждый блок в оперативную память. Затем этот пример или блок примеров обрабатывается и модель обновляется с учетом информации, вычисленной по этим данным.[53] Затем этот блок данных удаляется и считывается следующий поднабор данных. В библиотеке `scikit-learn` обучение во внешней памяти реализовано для некоторых моделей, и вы можете найти подробную информацию о нем в руководстве пользователя. Поскольку обучение во внешней памяти подразумевает обработку всех данных на одном компьютере, то в случае больших наборов данных выполнение подобных вычислений займет много времени. Кроме того, не для всех алгоритмов можно осуществить обучение во внешней памяти.

Еще одна стратегия масштабирования – это распределение данных по нескольким машинам вычислительного кластера, когда каждая машина обрабатывает свою часть данных. Для некоторых моделей эта стратегия может дать гораздо более существенное ускорение и размер обрабатываемых данных ограничен лишь размером кластера. Однако подобные вычисления часто требуют относительно сложной инфраструктуры. На данный момент одной из наиболее популярных платформ распределенных вычислений является платформа `spark`, входящая в экосистему Hadoop. В рамках пакета `MLLib` для платформы `spark` реализованы некоторые алгоритмы машинного обучения. Если ваши данные уже записаны в файловой системе HDFS или вы уже используете `spark` для предварительной обработки данных, описываемый способ вычислений может стать наиболее простым вариантом. Однако если у вас еще нет такой инфраструктуры, установка и интеграция

[53] Например, при оценивании параметров конкретного линейного метода (весов в логистической регрессии) будет инициализировано некоторое начальное значение этих параметров, после чего получая на вход очередной пример или блок примеров из обучающей выборки, веса будут обновляться. – *Прим. пер.*

кластера `spark` могут потребовать очень больших усилий. Ранее упомянутый пакет `vm` предлагает ряд возможностей для осуществления распределенных вычислений и, возможно, будет лучшим решением в данном случае.

Оттачивание навыков

Как и во многих аспектах жизни, только практика позволит вам стать экспертом в вопросах, рассмотренных нами в этой книге. Процедуры выделения признаков, предварительной обработки, визуализации и построения модели могут сильно варьировать в зависимости от различных задач и разных наборов данных. Возможно, в вашем распоряжении уже имеются разные наборы данных с различными задачами. Если вы еще не решаете конкретную задачу, хорошей стартовой площадкой станут конкурсы по машинному обучению, в рамках которых публикуются данные с конкретной задачей и команды соревнуются в получении наилучших прогнозов. Многие компании, некоммерческие организации и университеты проводят такие соревнования. Одна из самых популярных площадок, на которой можно найти эти конкурсы - это Kaggle, веб-сайт, который регулярно проводит соревнования по анализу данных, некоторые из них имеют солидный призовой фонд.

Кроме того, форум сайта Kaggle – это хороший источник информации о новейших инструментах и техниках машинного обучения, на сайте также можно найти различные наборы данных. Еще больше наборов данных с соответствующими задачами можно найти на платформе OpenML, на которой размещено более 20000 наборов данных с более чем 50000 задачами машинного обучения. Работа с этими наборами данных открывает замечательные возможности для оттачивания навыков в области машинного обучения. Недостаток соревнований заключается в том, что решение должно удовлетворять конкретной и заранее заданной метрике оптимизации, и, как правило, данные представляют собой фиксированный предварительно обработанный набор данных. Имейте в виду, постановка задачи и сбор данных также являются важными аспектами, и правильное понимание задачи, возможно, гораздо важнее, чем выжимание максимального процента правильности из классификатора.

Заключение

Мы надеемся, что убедили вас в полезности и легкости применения машинного обучения для решения самых разнообразных задач.

Продолжайте исследовать данные, но не упускайте из виду картину в целом.

Об авторах

Андреас Мюллер получил ученую степень PhD по машинному обучению в Боннском университете. Занимал в течение года должность специалиста по машинному обучению в Amazon, где решал прикладные задачи в области компьютерного зрения. В настоящий момент Андреас работает в Центре изучения данных Нью-Йоркского университета. В течение последних четырех лет Андреас стал куратором и одним из ключевых разработчиков библиотеки `scikit-learn`, инструмента машинного обучения, широко используемого в промышленности и науке. Кроме того, Андреас является автором и разработчиком еще нескольких популярных пакетов машинного обучения. Свою миссию он видит в том, чтобы создавать инструменты с открытым программным кодом, которые убирают препятствия, мешающие более активному использованию машинного обучения в прикладных задачах, содействуют продвижению воспроизводимой науки [54] и упрощают применение высокоточных алгоритмов машинного обучения.

Сара Гвидо – специалист по анализу данных, имеет большой опыт работы в стартапах. Ее сфера интересов – язык Python, машинное обучение, большие объемы данных и мир технологий. Совсем недавно Сара стала ведущим специалистом по анализу данных в компании Bitly, является постоянным спикером конференций по машинному обучению. Кроме того, Сара имеет степень магистра по информатике Мичиганского университета и в настоящее время проживает в Нью-Йорке.

Колофон

Животное, изображенное на обложке книги *Введение в машинное обучение с помощью Python* – аллеганский скрытожаберник (лат. *Cryptobranchus alleganiensis*), амфибия, обитающая в восточной части США (ареал обитания простирается от Нью-Йорка до Джорджии). Это земноводное имеет множество колоритных прозвищ, в их числе «аллеганский аллигатор», «сопли выдры» и «грязь дьявола». Происхождение официального англоязычного названия «hellbender salamander» неясно: согласно одной из теорий, первые поселенцы посчитали внешний вид саламандры отталкивающим и предположили, что это существо из ада, в который оно стремится вернуться.

[54] Воспроизводимая наука (reproducible science) – научная деятельность, в процессе которой открытия осуществляются по заранее известному и отработанному «маршруту», как это, например, происходит в структурной геномике при открытии новых молекул. – *Прим. пер.*

Аллеганский скрытожаберник является представителем семейства гигантских саламандр и может достигать 73 сантиметров в длину. Это третий по величине вид водных саламандр в мире. Саламандра имеет уплощенное тело с толстыми складками кожи по бокам. Хотя саламандры и имеют по одной жаберной щели с каждой стороны, большую часть кислорода они поглощают через складки кожи: газ проникает и выделяется через капилляры, расположенные на поверхности кожи.

В силу этого их идеальная среда обитания – это чистые, быстрые и неглубокие ручьи, где вода хорошо насыщена кислородом. Скрытожаберник прячется под камнями и охотится главным образом с помощью обоняния, хотя способен ощущать малейшие колебания в воде. Его рацион состоит из речных раков, мелких рыб. Иногда скрытожаберник поедает яйца саламандр своего вида. Скрытожаберник является ключевым участником своей экосистемы, будучи добычей для крупной рыбы, черепах и змей.

В течение последних нескольких десятилетий популяция аллеганского скрытожаберника значительно снизилась. Качество воды стало самой большой проблемой, поскольку дыхательная система скрытожаберников делает их очень чувствительными к загрязненной или мутной воде. Активизация человеческой деятельности вблизи мест обитания скрытожаберников означает увеличение количества химических осадков в воде. Пытаясь сохранить этот исчезающий вид, биологи стали разводить амфибий в неволе и выпускать их в естественную среду обитания по достижении ими менее уязвимого возраста.

Многие животные, изображенные на обложках книг издательства O'Reilly, находятся под угрозой исчезновения, все они имеют важное значение для окружающего мира. Чтобы подробнее узнать о том, как вы можете помочь, зайдите на сайт *animals.oreilly.com*.

Изображение для обложки взято из книги *Wood's Animate Creation*. Шрифты обложки – URW Typewriter and Guardian Sans. Шрифт текста – Adobe Minion Pro, шрифт заголовков – Adobe Myriad Condensed, шрифт программного кода – Dalton Maag's Ubuntu Mono.

Обратная сторона обложки

Введение в машинное обучение с помощью Python

Машинное обучение стало неотъемлемой частью различных коммерческих и исследовательских проектов, однако эта область не является прерогативой больших компаний с мощными аналитическими командами. Даже если вы еще новичок в использовании Python, эта книга познакомит вас с практическими способами построения систем машинного обучения. При всем многообразии данных, доступных на сегодняшний день, применение машинного обучения ограничивается лишь вашим воображением.

Вы изучите этапы, необходимые для создания успешного проекта машинного обучения, используя Python и библиотеку scikit-learn. Авторы Андреас Мюллер и Сара Гвидо сосредоточили свое внимание на практических аспектах применения алгоритмов машинного обучения. Знание библиотек NumPy и matplotlib позволит вам извлечь из этой книги еще больше полезной информации.

С помощью этой книги вы изучите:

- Фундаментальные понятия и сферы применения машинного обучения
- Преимущества и недостатки широко используемых алгоритмов машинного обучения
- Способы загрузки данных, обрабатываемых в ходе машинного обучения, включая различные аспекты работы с данными
- Продвинутые методы оценивания модели и тонкая настройка параметров
- Принципы построения конвейеров для объединения моделей в цепочки и инкапсуляции рабочего потока
- Методы работы с текстовыми данными
- Рекомендации по улучшению навыков, связанных с машинным обучением и наукой о данных

Эта книга – фантастический, суперпрактический ресурс для каждого, кто хочет начать использовать машинное обучение в Python – как жаль, что когда я начинала использовать scikit-learn, этой книги не было.

Ханна Уоллок,
старший научный сотрудник Microsoft Research

Андреас Мюллер получил ученую степень PhD по машинному обучению в Боннском университете. Занимал должность специалиста по машинному обучению в Amazon, где занимался разработкой проектов компьютерного зрения. В настоящий момент Андреас работает в Центре изучения данных Нью-Йоркского университета. Кроме того, Андреас – куратор и один из ключевых разработчиков библиотеки scikit-learn.

Сара Гвидо – специалист по анализу данных, имеет большой опыт работы в стартапах, совсем недавно стала ведущим специалистом по анализу данных в компании Bitly, постоянный спикер конференций по машинному обучению. Кроме того, Сара имеет степень магистра по информатике Мичиганского университета.