# Содержание

# Предисловие

Мы живем в потоке данных. Согласно недавним оценкам, ежедневно генерируется 2,5 квинтилиона (1018) байт данных. Это такой огромный объем данных, что более 90% информации, которую мы храним в наши дни, было сгенерировано за все прошедшее десятилетие. К сожалению, люди не способны воспользоваться подавляющей частью этой информации. Данные либо лежат за пределами возможностей стандартных аналитических методов, либо они просто слишком обширны, чтобы наши ограниченные умы смогли их понять.

Благодаря методам машинного обучения мы наделяем компьютеры способностями обрабатывать большие объемы данных, которые в противном случае стояли бы непроницаемой стеной, даем им возможность обучаться на этих данных и извлекать из них практические выводы. От массивных суперкомпьютеров, которые обеспечивают работу поисковых движков компании Google, до смартфонов, которые мы носим в наших карманах, – везде мы опираемся на машинное обучение, которое приводит в действие значительную часть окружающего нас мира, и нередко мы об этом даже не догадываемся.

Являясь первооткрывателями наших дней дивного нового мира больших данных, нам надлежит узнать еще больше о машинном обучении. Что же такое машинное обучение и как оно работает? Каким образом его применять, чтобы заглянуть в неизведанное, привести в действие свой бизнес либо просто узнать, что Интернет в целом думает о моем любимом фильме? Все это и даже больше будет охвачено в следующих главах, созданных моим добрым другом и коллегой Себастьяном Рашка.

Когда Себастьян не занят одомашниванием моей вспыльчивой собаки, он неустанно посвящает свое свободное время сообществу специалистов, работающих с открытым исходным кодом в области машинного обучения. В течение последних нескольких лет Себастьян разработал десятки популярных учебных руководств, в которых затрагиваются различные темы из области машинного обучения и визуализации данных на Python. Он также является автором и соавтором разработок ряда библиотек Python с открытым исходным кодом, и некоторые из них теперь являются составной частью базового потока операций по машинному обучению на Python.

В силу его обширных экспертных познаний в этой области я уверен, что глубокое понимание Себастьяном мира машинного обучения на Python будет неоценимо для пользователей всех уровней квалификации. Я искренне рекомендую эту книгу любому, кто находится в поисках более широкого и более практического понимания принципов машинного обучения.

***Доктор Рандал С. Олсон,***
*исследователь в области искусственного интеллекта*
*и машинного обучения, Университет шт. Пенсильвания, США*

# Об авторе

**Себастьян Рашка** – аспирант докторантуры в Мичиганском университете, США, занимающийся разработкой новых вычислительных методов в области вычислительной биологии. Веб-сайтом Analytics Vidhya (https://www.analyticsvidhya.com/) сообщества увлеченных профессионалов в области науки о данных отмечен первым местом среди наиболее влиятельных аналитиков данных на GitHub. За его плечами многолетний опыт программирования на Python; он также проводит ряд семинаров по практическому применению науки о данных и машинного обучения. Регулярные выступления и публикации на тему науки о данных, машинного обучения и языка Python на деле мотивировали его написать эту книгу, с тем чтобы помочь людям разрабатывать управляемые данными решения без обязательного наличия предварительной квалификации в области машинного обучения.

Он также является активным соавтором проектов с открытым исходным кодом и автором собственных методов, которые теперь успешно применяются в конкурсах по машинному обучению, таких как Kaggle. В свое свободное время он работает над моделями для спортивного прогнозирования, и если не сидит перед компьютером, то любит проводить время, занимаясь спортом.

Хотел бы поблагодарить своих профессоров, Арун Росса и Панг-Нинг Тана, а также многих других, кто вдохновил меня и сформировал у меня огромный интерес к исследованиям в области классификации образов, машинного обучения и добычи данных.

Хотел бы воспользоваться представившейся возможностью и поблагодарить огромное сообщество пользователей и разработчиков библиотек Python с открытым исходным кодом, которые помогли мне создать совершенную среду для научных исследований и науки о данных.

Особую благодарность передаю базовым разработчикам библиотеки scikit-learn. В качестве одного из соавторов этого проекта было приятно работать вместе с замечательными людьми, которые не только превосходно осведомлены в области машинного обучения, но и являются превосходными программистами.

Наконец, хочу поблагодарить всех за проявление интереса к этой книге и искренне надеюсь, что смогу передать весь свой энтузиазм по поводу моего присоединения к огромным сообществам программистов на Python и в области машинного обучения.

# О рецензентах

**Ричард Даттон** начал заниматься программированием компьютера ZX Spectrum в возрасте 8 лет, и с тех пор это увлечение направляет его по противоречивому массиву технологий и ролей в области промышленности и финансов.

Работал в Microsoft и управляющим в Barclays, его текущим увлечением является гибрид из Python, машинного обучения и цепочки блоков транзакций.

Если он не сидит перед компьютером, то его можно найти в спортзале либо дома с бокалом вина перед смартфоном iPhone. Он называет это равновесием.

**Дэйв Джулиан** – ИТ-консультант и преподаватель с 15-летним стажем. Работал техником, проектным инженером, программистом и веб-разработчиком. Его текущие проекты состоят из разработки инструмента для анализа урожайности в составе интегрированных стратегий по борьбе с сельскохозяйственными вредителями в теплицах. Его большой интерес пролегает на пересечении биологии и технологии с уверенностью, что умные машины способны помочь решить самые важные глобальные задачи.

**Вахид Мирджалили** получил звание доктора наук Мичиганского университета по машиностроению, где он разработал новые методы рафинирования белковых структур с использованием молекулярно-динамического имитационного моделирования. Объединив свои знания из областей статистики, добычи данных и физики, разработал мощные управляемые данными подходы, которые помогли ему и его исследовательской группе одержать победу в двух недавних мировых конкурсах по прогнозированию и рафинированию протеиновых структур, CASP, в 2012 и 2014 гг.

Работая над докторской диссертацией, решил присоединиться к факультету информатики и инженерного дела в Мичиганском университете с целью специализации в области машинного обучения. Его текущие исследовательские проекты включают разработку алгоритмов машинного обучения без учителя для добычи массивных наборов данных. Он также является страстным поклонником программирования на Python и делится своими реализациями алгоритмов кластеризации на своем личном веб-сайте http://vahidmirjalili.com.

**Хамидреза Саттари** – ИТ-профессионал, участвовавший в ряде областей, связанных с разработкой программного обеспечения, от программирования до архитектуры и управления. Владеет степенью магистра в области разработки программного обеспечения Университета Хериота-Уатта, Соединенное Королевство, и степенью бакалавра по электротехнике (электронике) Тегеранского университета Азад, Иран. В последние годы его области интереса составляли большие данные и машинное обучение. Является соавтором книги «*Веб-службы Spring 2. Книга рецептов*» (Spring Web Services 2 Cookbook); ведет свой собственный блог по адресу http://justdeveloped-blog.blogspot.com/.

**Дмитрий Тарановский** – разработчик программного обеспечения с заинтересованностью и квалификацией в Python, Linux и машинном обучении. Родом из Киева, Украина, он переехал в США в 1996 г. С раннего возраста страстно увлекался наукой и знаниями, побеждая на конкурсах по физике и математике. В 1999 г. был избран членом команды США по физике. В 2005 г. окончил Мичиганский Технологический институт со специализацией по математике. Позже работал программным

инженером над системой трансформации текста для компьютерных медицинских транскрипций (eScription). Изначально работая на Perl, он по достоинству оценил мощь и ясность Python, который позволил ему масштабировать систему до данных больших объемов. Впоследствии работал инженером программного обеспечения и аналитиком на алгоритмическую трейдинговую фирму. Он также внес значительный вклад в математические основы, в том числе создание и совершенствование расширения языка теории множеств и его связи с аксиомами множеств большой мощности, разработав понятийный аппарат конструктивной истины и создав систему порядковой индексации с реализацией на Python. Он также любит читать, быть за городом и старается сделать мир лучше.

# Введение

Наверное, не стоит и говорить, что машинное обучение стало одной из самых захватывающих технологий современности. Такие крупные компании, как Google, Facebook, Apple, Amazon, IBM, и еще многие другие небезосновательно вкладывают значительный капитал в разработку методов и программных приложений в области машинного обучения. Хотя может показаться, что термин «машинное обучение» сегодня уже набил оскомину, совершенно очевидно, что весь этот ажиотаж не является результатом рекламной шумихи. Эта захватывающая область исследования открывает путь к новым возможностям и стала неотъемлемой частью нашей повседневной жизни. Разговоры с речевым ассистентом по смартфону, предоставление рекомендаций относительно подходящего продукта для клиентов, предотвращение актов мошенничества с кредитными картами, фильтрация спама из входящих сообщений электронной почты, обнаружение и диагностирование внутренних заболеваний – и этот список можно продолжать.

Если вы хотите стать практиком в области машинного обучения, более основательным решателем задач или, возможно, даже обдумываете карьеру в научно-исследовательской области, связанной с машинным обучением, то эта книга для вас! Однако новичка теоретические идеи, лежащие в основании машинного обучения, нередко могут подавлять своей сложностью. И все же многие из опубликованных в последние годы практических изданий способны помочь вам приступить к работе с машинным обучением на основе реализации мощных алгоритмов обучения. По моему мнению, использование практических примеров программного кода служит важной цели. В них идеи иллюстрируются путем приведения изученного материала непосредственно в действие. Однако помните, что огромная мощь влечет за собой большую ответственность! Идеи, лежащие в основании машинного обучения, слишком красивы и важны, чтобы их прятать в черном ящике. Поэтому моя личная миссия состоит в том, чтобы предоставить вам иную книгу: книгу, в которой обсуждаются важные подробности относительно идей и принципов машинного обучения, предлагаются интуитивные и одновременно информативные объяснения по поводу того, каким образом алгоритмы машинного обучения работают, как их использовать и, самое главное, как избежать наиболее распространенных ловушек.

Если в поисковой системе Академия Google[1] в качестве поискового запроса набрать «машинное обучение», то она вернет обескураживающе большое количество публикаций. В английском сегменте их число составляет 1 800 000 публикаций (для сравнения, в русском – 16 400). Разумеется, мы не сможем обсудить мельчайшие подробности всех основных алгоритмов и приложений, которые появились в течение предыдущих 60 лет. Однако в этой книге мы предпримем увлекательное путешествие, которое затронет все существенные темы и понятия, чтобы дать вам преимущество в этой области. Если вы считаете, что ваша потребность в знаниях из области машинного обучения не удовлетворена, то можно воспользоваться много-

[1] Академия Google (Google Scholar) (https://scholar.google.ru/schhp?hl=ru) – бесплатная поисковая система по полным текстам научных публикаций всех форматов и дисциплин. Проект работает с ноября 2004 г. – *Прим. перев.*

численными полезными ресурсами, чтобы отслеживать существенные прорывы в этой области.

Если вы уже подробно изучили теорию машинного обучения, то эта книга покажет вам, каким образом претворить ваши знания на практике. Если вы использовали методы машинного обучения прежде и хотите получить более глубокое понимание того, каким образом машинное обучение работает в действительности, то эта книга для вас! Не переживайте, если вы абсолютно плохо знакомы с машинным обучением; у вас гораздо больше причин испытывать предвкушение. Я обещаю вам, что машинное обучение изменит ваш способ мышления относительно задач, которые вы хотите решать, и покажу вам, как справляться с ними путем высвобождения мощи данных.

Прежде чем мы погрузимся с головой в область машинного обучения, отвечу на ваш сакраментальный вопрос – «почему, собственно, Python?» Ответ прост: он – мощный и одновременно очень доступный. Python стал самым популярным языком программирования для науки о данных, потому что он позволяет забыть об утомительных сторонах программирования и предлагает нам среду, где мы можем быстро набросать наши мысли и привести идеи непосредственно в действие.

Размышляя над тем путем, который я прошел лично, могу сказать вам совершенно искренне, что изучение методов машинного обучения сделало меня более основательным ученым, мыслителем и решателем задач. В этой книге я хочу поделиться этими знаниями с вами. Знание достигается в процессе исследования, ключом к нему является наш энтузиазм, а истинное освоение навыков может быть достигнуто только практикой. Местами продвижение может быть ухабистым, и некоторые темы могут оказаться более сложными для понимания, чем другие, но я надеюсь, что вы воспользуетесь этой возможностью и сконцентрируетесь на вознаграждении. Помните, что мы отправляемся в это путешествие вместе, и по ходу изложения мы будем пополнять ваш арсенал большим количеством мощных методов, которые помогут нам решать даже самые трудноразрешимые задачи на основе управляемого данными подхода.

## О чем эта книга рассказывает

*Глава 1* «*Наделение компьютеров способностью обучаться на данных*» знакомит с основными подобластями машинного обучения, в которых решаются самые разные практические задачи. Кроме того, в ней обсуждаются принципиальные шаги, которые необходимо предпринять, чтобы создать типичную машиннообучаемую модель, и создается конвейер, который будет направлять нас в последующих главах.

*Глава 2* «*Тренировка алгоритмов машинного обучения для задачи классификации*» обращается к истокам области исследований, связанной с машинным обучением, и знакомит с бинарными (двуклассовыми) классификаторами на основе персептрона и адаптивных линейных нейронов. Эта глава является осторожным введением в основополагающие принципы классификации образов, где основное внимание уделено взаимодействию машинного обучения с алгоритмами оптимизации.

*Глава 3* «*Обзор классификаторов с использованием библиотеки scikit-learn*» описывает принципиальные алгоритмы машинного обучения, предназначенные для зада-

чи классификации, и предлагает практические примеры с использованием одной из самых популярных и всеобъемлющих библиотек машинного обучения с открытым исходным кодом scikit-learn.

*Глава 4* «*Создание хороших тренировочных наборов – предобработка данных*» посвящена обсуждению того, как обходиться с наиболее распространенными трудностями, возникающими в работе с исходными наборами данных, такими как пропущенные данные. В ней также обсуждается ряд подходов к идентификации в наборах данных наиболее информативных признаков и будут продемонстрированы способы подготовки переменных различных типов для ввода в алгоритмы машинного обучения.

*Глава 5* «*Сжатие данных путем снижения размерности*» посвящена описанию принципиальных методов, необходимых для сведения числа признаков в наборе данных к наборам меньшего объема при сохранении большей части их полезной и отличительной информации. В ней обсуждается стандартный подход к снижению размерности на основе главных компонент, который сравнивается с методами нелинейного преобразования с учителем.

*Глава 6* «*Изучение наиболее успешных методов оценки моделей и тонкой настройки гиперпараметров*» посвящена обсуждению плюсов и минусов методик оценки качества прогнозных моделей. Кроме того, в ней обсуждаются различные метрики, применяемые для измерения качества моделей, и приемы тонкой настройки алгоритмов машинного обучения.

*Глава 7* «*Объединение моделей для методов ансамблевого обучения*» знакомит с различными принципами эффективного объединения двух и более алгоритмов обучения. В ней будет продемонстрировано создание ансамблей экспертов с целью преодоления недостатков отдельных алгоритмов обучения, которые в результате приводят к более точным и надежным прогнозам.

*Глава 8* «*Применение алгоритмов машинного обучения в анализе мнений*» посвящена обсуждению принципиальных шагов, необходимых для преобразования текстовых данных в содержательные представления для алгоритмов машинного обучения, с целью прогнозирования мнений людей на основе их письменной речи.

*Глава 9* «*Встраивание алгоритма машинного обучения в веб-приложение*» продолжает прогнозную модель из предыдущей главы и проведет вас по основным шагам разработки веб-приложений со встроенными машиннообучаемыми моделями.

*Глава 10* «*Прогнозирование непрерывных целевых величин на основе регрессионного анализа*» посвящена обсуждению принципиальных методов, необходимых для моделирования линейных связей между целевыми переменными и переменной отклика для создания прогнозов на непрерывной шкале. После ознакомления с различными линейными моделями в ней также обсуждаются подходы на основе параболической регрессии и на основе деревьев.

*Глава 11* «*Работа с немаркированными данными – кластерный анализ*» смещает акцент в другую подобласть машинного обучения – обучение без учителя. Мы применим алгоритмы из трех фундаментальных семейств алгоритмов кластеризации с целью нахождения групп объектов, в которых присутствует определенная степень подобия.

*Глава 12* «*Тренировка искусственных нейронных сетей для распознавания изображений*» расширяет принцип градиентной оптимизации, который был впервые представлен в *главе 2* «*Тренировка алгоритмов машинного обучения для задачи*

*классификации*» для создания мощных, многослойных нейронных сетей на основе популярного алгоритма обратного распространения ошибки.

*Глава 13* «*Распараллеливание тренировки нейронных сетей при помощи Theano*» опирается на знания, полученные в предыдущих главах, и предоставляет практическое руководство по более эффективной тренировке нейронных сетей. В центре внимания данной главы находится библиотека Python с открытым исходным кодом Theano, которая предоставит возможность использовать ядра современных многоядерных графических процессоров.

## Что требуется для этой книги

Исполнение прилагаемых к данной книге примеров программ требует установки Python версии 3.4.3 или более поздней в Mac OS X, Linux или Microsoft Windows. На протяжении всей книги мы часто будем использовать ключевые библиотеки Python для научных вычислений, в том числе SciPy, NumPy, scikit-learn, matplotlib и pandas.

В первой главе будут предложены инструкции и полезные подсказки по поводу инсталляции среды Python и указанных ключевых библиотек. Мы пополним наш репертуар дополнительными библиотеками, а инструкции по их инсталляции будут предоставлены в соответствующих главах: библиотека NLTK для обработки естественного языка (*глава 8* «*Применение алгоритмов машинного обучения в анализе мнений*»), веб-платформа Flask (*глава 9* «*Встраивание алгоритма машинного обучения в веб-приложение*»), библиотека seaborn для визуализации статистических данных (*глава 10* «*Прогнозирование непрерывных целевых величин на основе регрессионного анализа*») и Theano для эффективной тренировки нейронных сетей на графических процессорах (*глава 13* «*Распараллеливание тренировки нейронной сети при помощи Theano*»).

## Для кого эта книга

Если вы хотите узнать, как использовать Python, чтобы начать отвечать на критические вопросы в отношении ваших данных, возьмите книгу «*Python и машинное обучение*» – и не важно, хотите вы приступить к изучению науки о данных с нуля или же намереваетесь расширить свои познания в этой области, – это принципиальный ресурс, которого нельзя упускать.

## Условные обозначения

В этой книге вы найдете ряд текстовых стилей, которые выделяют различные виды информации. Вот некоторые примеры этих стилей и объяснение их значения.

Кодовые слова в тексте, имена таблиц баз данных, папок, файлов, расширения файлов, пути, ввод данных пользователем и дескрипторы социальной сети Twitter показаны следующим образом: «И уже установленные пакеты могут быть обновлены при помощи флага `--upgrade`».

Фрагмент исходного кода оформляется следующим образом:

```
import matplotlib.pyplot as plt
import numpy as np
y = df.iloc[0:100, 4].values
y = np.where(y == 'Iris-setosa', -1, 1)
X = df.iloc[0:100, [0, 2]].values
plt.scatter(X[:50, 0], X[:50, 1],
            color='red', marker='x', label='setosa')
plt.scatter(X[50:100, 0], X[50:100, 1],
            color='blue', marker='o', label='versicolor')
plt.xlabel('длина чашелистика')
plt.ylabel('длина лепестка')
plt.legend(loc='upper left')
plt.show()
```

При этом если исходный код содержит результат выполнения, то он предваряется значком консоли интерпретатора Python (>>>):

```
>>>
df.isnull().sum()
A    0
B    0
C    1
D    1
dtype: int64
```

Любой ввод или вывод в командной строке записывается следующим образом:

```
> dot -Tpng tree.dot -o tree.png
```

**Новые термины** и **важные слова** показаны полужирным шрифтом. Слова, которые вы видите на экране, например в меню или диалоговых окнах, выглядят в тексте следующим образом: «После щелчка по кнопке **Dashboard** в верхнем правом углу мы получим доступ к панели управления, показанной в верхней части страницы».

Предупреждения или важные примечания появляются в этом поле.

Подсказки и приемы появляются тут.

Дополнения к тексту оригинала книги.

## Отзывы читателей

Отзывы наших читателей всегда приветствуются. Сообщите нам, что вы думаете об этой книге – что вам понравилось, а что нет. Обратная связь с читателями для нас очень важна, поскольку она помогает нам формировать названия книг, из которых вы действительно получите максимум полезного.

Отзыв по общим вопросам можно отправить по адресу feedback@packtpub.com, упомянув заголовок книги в теме вашего электронного сообщения.

Если же речь идет о теме, в которой вы профессионально осведомлены, и вы интересуетесь написанием либо содействием в написании книги, то обратитесь к нашему перечню авторов по адресу www.packtpub.com/authors.

# Служба поддержки

Теперь, когда вы являетесь довольным владельцем книги издательства «Packt», мы предложим вам ряд возможностей с целью помочь вам получать максимум от своей покупки.

## Скачивание исходного кода примеров

Вы можете скачать файлы с исходным кодом примеров из вашего аккаунта по адресу http://www.packtpub.com/ для всех книг издательства «Packt Publishing», которые вы приобрели. Если вы купили эту книгу в другом месте, то можно посетить http://www.packtpub.com/support и зарегистрироваться там, чтобы получить файлы прямо по электронной почте.

## Опечатки

Несмотря на то что мы приняли все меры, чтобы гарантировать правильность нашего информационного материала, ошибки действительно случаются. Если вы найдете ошибку в одной из наших книг, возможно, в тексте или исходном коде, то мы будем благодарны, если вы сообщите об этом нам. Поступая так, вы можете уберечь других читателей от разочарования и помочь нам улучшать последующие версии этой книги. Если вы найдете какие-либо опечатки, пожалуйста, сообщите о них, посетив страницу http://www.packtpub.com/submit-errata, выбрав вашу книгу, нажав на ссылку в форме **Errata submission form** для отправки информации об опечатке и введя ваши данные об опечатках. Как только ваши данные будут проверены, сообщение будет принято к рассмотрению, и данные об опечатке будут загружены на наш вебсайт или добавлены к любому списку из существующих списков опечаток в разделе **Errata** под соответствующим заголовком.

Чтобы просмотреть ранее предоставленные опечатки, перейдите на страницу https://www.packtpub.com/books/content/support и в поле поиска введите название книги. Запрошенная информация появится под разделом **Errata**.

## Нарушение авторских прав

Пиратство защищенного авторским правом материала в Интернете является хронической проблемой во всех средствах массовой информации. В издательстве «Packt» мы очень серьезно относимся к защите нашего авторского права и лицензий. Если вы сталкиваетесь с какими-либо недопустимыми копиями наших работ в какой-либо форме в Интернете, пожалуйста, просим вас незамедлительно предоставить нам адрес размещения или название веб-сайта, благодаря чему мы можем добиваться правовой защиты.

Пожалуйста, свяжитесь с нами по электронному адресу copyright@packtpub.com с ссылкой на предполагаемый пиратский материал.

Мы ценим вашу помощь в защите наших авторов и в наших усилиях предоставлять вам ценный информационный материал.

### *Вопросы*

Если у вас есть вопрос по каким-либо аспектам этой книги, то вы можете связаться с нами по электронному адресу questions@packtpub.com, и мы приложим все усилия, чтобы решить ваш вопрос.

## Комментарий переводчика

Весь материал книги приведен в соответствие с последними действующими версиями библиотек (время перевода книги – октябрь-ноябрь 2016 г.), дополнен свежей информацией и протестирован в среде Windows 10 и Fedora 24. При тестировании исходного кода за основу взят Python версии 3.5.2.

Большинство содержащихся в книге технических терминов и аббревиатур для удобства кратко определено в сносках, а для некоторых терминов в силу отсутствия единой терминологии приведены соответствующие варианты наименований или пояснения. Книга дополнена «*Глоссарием основных терминов и сокращений*». В приложении к книге особое внимание уделено оценке качества машиннообучаемых моделей.

На веб-сайте Github имеется веб-страница книги, где содержатся обновляемые исходные коды прилагаемых к книге программ, много дополнительных материалов и ссылок, а также обширный раздел часто задаваемых вопросов (https://github.com/rasbt/python-machine-learning-book/tree/master/faq), наиболее интересная, по моему мнению, и лишь незначительная часть ответов на которые сведена в *приложении А* к данной книге.

Прилагаемый к книге адаптированный и скорректированный исходный код примеров должен находиться в подпапке домашней папки пользователя (/home/ваши_проекты_Python или C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ]\ ваши_проекты_Python). Ниже приведена структура папки с прилагаемыми примерами и дополнительной информацией:

| | |
|---|---|
| bonus | Дополнительные материалы, документация и другие примеры. |
| ch01-ch13 | Исходный код примеров в виде блокнотов Jupyter и сценарных файлов Python. |
| data | Наборы данных, используемых в книге и примерах. |
| faq | Копия раздела часто задаваемых вопросов по книге и машинному обучению репозитория Github на английском языке. |
| fonts | Шрифты, используемые для оформления блокнотов, книги и графиков. |

Для просмотра исходного кода примеров лучше всего пользоваться блокнотами Jupyter. Они более читабельны, содержат графики, цветные рисунки и расширенные пояснения.

Для оформления графиков и диаграмм использовался шрифт Ubuntu Condensed, который прилагается к книге. Его можно найти в папке fonts. В Windows для установки шрифта в операционную сиситему следует правой кнопкой мыши нажать на файле шрифта и выбрать из контекстного меню **Установить**.

Далее приведены особенности установки некоторых используемых программных библиотек Python.

### Особенности программных библиотек

**Numpy+MKL** привязана к библиотеке Intel® Math Kernel Library и включает в свой состав необходимые динамические библиотеки (DLL) в каталоге numpy.core. Для работы SciPy и Scikit-learn в Windows требуется, чтобы в системе была установлена библиотека Numpy+MKL. Ее следует скачать из репозитория whl-файлов (http://www.lfd.uci.edu/~gohlke/pythonlibs/) и установить (например, pip3 install numpy-1.11.2+mkl-cp35-cp35m-win_amd64.whl для 64-разрядного компьютера) как whl.

- **NumPy** – основополагающая библиотека, необходимая для научных вычислений на Python.
- **Matplotlib** – библиотека для работы с двумерными графиками. Требует наличия numpy и некоторых других.
- **Pandas** – инструмент для анализа структурных данных и временных рядов. Требует наличия numpy и некоторых других.
- **Scikit-learn** – интегратор классических алгоритмов машинного обучения. Требует наличия numpy+mkl.
- **SciPy** – библиотека, используемая в математике, естественных науках и инженерном деле. Требует наличия numpy+mkl.
- **Jupyter** – интерактивная вычислительная среда.

Факультативно:

- **Spyder** – инструментальная среда программирования на Python.
- **Mlxtend** – библиотека модулей расширения и вспомогательных инструментов для программных библиотек Python, предназначенных для анализа данных и машинного обучения (автор С. Рашка) для решения каждодневных задач в области науки о данных. Используется в приложении и дополнительных материалах к книге (https://github.com/rasbt/mlxtend). Документацию по библиотеке можно найти в папке bonus.
- **PyQt5**, библиотека программ для программирования визуального интерфейса, требуется для работы инструментальной среды программирования Spyder

### Протокол установки программных библиотек

```
python -m pip install --upgrade pip

pip3 install numpy
    либо как whl: pip3 install numpy-1.11.2+mkl-cp35-cp35m-win_amd64.whl
pip3 install matplotlib
pip3 install pandas
pip3 install scikit-learn
    либо как whl: pip3 install scikit_learn-0.18.1-cp35-cp35m-win_amd64.whl
pip3 install scipy
```

либо как whl: `pip3 install scipy-0.18.1-cp35-cp35m-win_amd64-any.whl`

```
pip3 install jupyter
pip3 install theano
pip3 install keras
pip3 install nltk
pip3 install seaborn
pip3 install flask
pip3 install pyprind
pip3 install wtforms
```

факультативно:

```
pip3 install mlxtend
pip3 install pyqt5
pip3 install spyder
```

Примечание: в зависимости от базовой ОС, версий языка Python и версий программных библиотек устанавливаемые вами версии whl-файлов могут отличаться от приведенных выше, где показаны последние на декабрь 2016 г. версии для 64-разрядной ОС Windows и Python версии 3.5.2

### *Установка библиотек Python из whl-файла*

Библиотеки для Python можно разрабатывать не только на чистом Python. Довольно часто библиотеки пишутся на C (динамические библиотеки), и для них пишется обертка Python, или же библиотека пишется на Python, но для оптимизации узких мест часть кода пишется на C. Такие библиотеки получаются очень быстрыми, однако библиотеки с вкраплениями кода на C программисту на Python тяжелее установить ввиду банального отсутствия соответствующих знаний либо необходимых компонентов и настроек в рабочей среде (в особенности в Windows). Для решения описанных проблем разработан специальный формат (файлы с расширением .whl) для распространения библиотек, который содержит заранее скомпилированную версию библиотеки со всеми ее зависимостями. Формат whl поддерживается всеми основными платформами (Mac OS X, Linux, Windows).

Установка производится с помощью менеджера пакетов pip. В отличие от обычной установки командой pip3 install <имя_библиотеки>, вместо имени библиотеки указывается путь к whl-файлу pip3 install <путь/к/whl_файлу>. Например:

```
pip3 install C:\temp\networkx-1.11-py2.py3-none-any.whl
```

Откройте окно командой строки и при помощи команды cd перейдите в каталог, где размещен ваш whl-файл. Просто скопируйте туда имя вашего whl-файла. В этом случае полный путь указывать не понадобится. Например:

```
pip3 install networkx-1.11-py2.py3-none-any.whl
```

При выборе библиотеки важно, чтобы разрядность устанавливаемой библиотеки и разрядность интерпретатора совпадали. Пользователи Windows могут брать whl-файлы на веб-странице http://www.lfd.uci.edu/~gohlke/pythonlibs/ Кристофа Голька из Лаборатории динамики флуоресценции Калифорнийского университета в г. Ирвайн. Библиотеки там постоянно обновляются, и в архиве содержатся все, какие только могут понадобиться.

### Установка и настройка инструментальной среды Spyder

Spyder – это инструментальная среда для научных вычислений для языка Python (Scientific PYthon Development EnviRonment) для Windows, Mac OS X и Linux. Это простая, легковесная и бесплатная интерактивная среда разработки на Python, которая предлагает функционал, аналогичный среде разработки на MATLAB, включая готовые к использованию виджеты PyQt5 и PySide: редактор исходного кода, редактор массивов данных NumPy, редактор словарей, консоли Python и IPython и многое другое.

Чтобы установить среду Spyder в Ubuntu Linux, используя официальный менеджер пакетов, нужна всего одна команда:

```
sudo apt-get install spyder3
```

Чтобы установить с использованием менеджера пакетов pip:

```
sudo apt-get install python-qt5 python-sphinx
sudo pip3 install spyder
```

И чтобы обновить:

```
sudo pip3 install -U spyder
```

Установка среды Spyder в Fedora 24:

```
dnf install python3-spyder
```

Во всех вышеперечисленных случаях речь идет о версии Spyder для Python 3 (на момент инсталляции это был Python 3.5.2).

Установка среды Spyder в Windows:

```
pip3 install spyder
```

Глава 1

# Наделение компьютеров способностью обучаться на данных

По моему убеждению, *машинное обучение* как область практической деятельности и как сфера научных исследований алгоритмов, извлекающих смысл из данных, является самой захватывающей отраслью всей информатики! В наш век переизбытка данных при помощи самообучающихся алгоритмов мы можем превратить эти данные в знание. Благодаря тому что за последние несколько лет были разработаны многочисленные мощные библиотеки с открытым исходным кодом, вероятно, никогда еще не появлялся столь подходящий момент для того, чтобы ворваться в область машинного обучения и научиться использовать мощные алгоритмы, позволяющие обнаруживать в данных повторяющиеся образы и делать прогнозы о будущих событиях.

В этой главе мы узнаем об общих принципах машинного обучения и его типах. Вместе с вводным курсом в соответствующую терминологию мы заложим основу для того, чтобы успешно использовать методы машинного обучения для решения практических задач.

В этой главе мы затронем следующие темы:

- общие принципы машинного обучения;
- три типа обучения и основополагающая терминология;
- базовые элементы успешной разработки систем машинного обучения;
- инсталляция и настройка Python для анализа данных и машинного обучения.

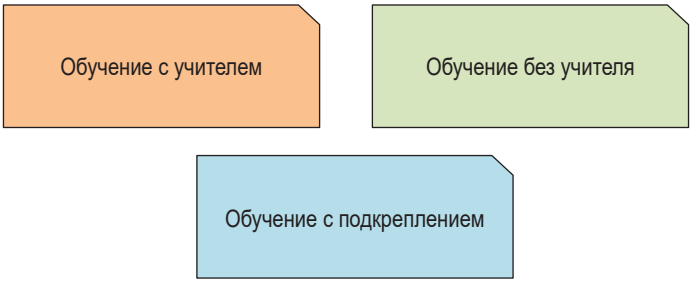
## Построение интеллектуальных машин для преобразования данных в знания

В эпоху современных технологий существует один ресурс, которым мы обладаем в изобилии: большой объем структурированных и неструктурированных данных. Во второй половине XX века машинное обучение развилось в подобласть *искусственного интеллекта*, которая охватывала разработку самообучающихся алгоритмов для приобретения из этих данных знаний с целью выполнения прогнозов. Вместо того чтобы в ручном режиме выявлять правила и строить модели на основе анализа больших объемов данных, методология машинного обучения предлагает для вычленения знаний из данных более действенную альтернативу – постепенное улучшение качества прогнозных моделей и принятие решений, управляемых данными. Помимо того что машинное обучение приобретает все большую значимость в научных исследованиях в области информатики, оно начинает играть все более активную роль в нашей повседневной жизни. Благодаря машинному обучению мы наслаждаемся

удобным программным обеспечением распознавания текста и речи, устойчивыми почтовыми спам-фильтрами, надежными поисковыми системами, амбициозными шахматными программами и, надо надеяться, совсем скоро начнем использовать безопасные и эффективные самоходные автомобили.

## Три типа машинного обучения

В этом разделе мы рассмотрим три типа машинного обучения: *обучение с учителем* (контролируемое), *обучение без учителя* (неконтролируемое, или спонтанное) и *обучение с подкреплением*:

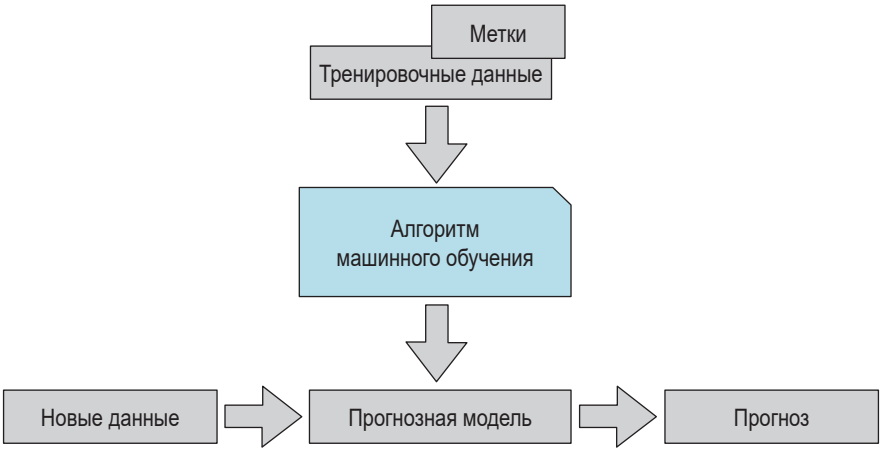



Мы узнаем о принципиальных между ними различиях и, опираясь на концептуальные примеры, разовьем интуитивное понимание того, в каких областях они находят свое практическое применение.

### Выполнение прогнозов о будущем на основе обучения с учителем

Основная задача обучения с учителем состоит в том, чтобы на маркированных *тренировочных данных* извлечь модель, которая позволяет делать прогнозы о ранее не встречавшихся или будущих данных. Здесь термин «*с учителем*» относится к подмножеству образцов, в которых нужные выходные сигналы (метки) уже известны.

В качестве примера рассмотрим фильтрацию почтового спама. При помощи алгоритма машинного обучения с учителем мы можем натренировать модель на корпусе маркированных сообщений электронной почты, где почтовые сообщения правильно помечены как спамные либо неспамные, и затем предсказать, к какому из этих двух категорий принадлежит новое сообщение. Методы обучения с учителем, когда имеются дискретные *метки принадлежности к классу*, такие как в приведенном примере с фильтрацией спамных почтовых сообщений, еще называют методами *классификации*. Другую подкатегорию методов обучения с учителем представляет *регрессия*, где результирующий сигнал – непрерывная величина:




## *Задача классификации – распознавание меток классов*

Задача классификации – это подкатегория методов машинного обучения с учителем, суть которой заключается в идентификации категориальных меток классов для новых экземпляров на основе предыдущих наблюдений[1]. Метка класса представляет собой дискретное, неупорядоченное значение, которое может пониматься как *принадлежность группе* экземпляров. Ранее упомянутый пример с обнаружением почтового спама представляет собой типичный пример задачи *бинарной классификации*, где алгоритм машинного обучения вычисляет серию правил для различения двух возможных классов: спамных и неспамных почтовых сообщений.

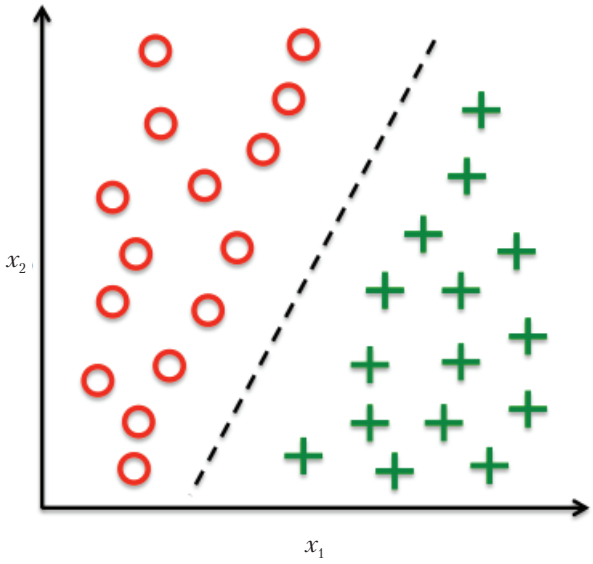
Однако набор меток классов не обязательно должен иметь двоичную природу. Извлеченная алгоритмом обучения с учителем прогнозная модель может присваивать новому, немаркированному экземпляру любую метку класса, которая была определена в тренировочном наборе данных. Типичным примером задачи *многоклассовой (или мультиномиальной) классификации* является рукописное распознавание символов. Здесь можно было бы собрать тренировочный набор данных, состоящий из большого числа рукописных образцов каждой буквы алфавита. Теперь если пользователь через устройство ввода данных предоставит новый рукописный символ, то наша прогнозная модель с определенной степенью соответствия сможет распознать правильную букву алфавита. Однако наша система машинного обучения была бы не в состоянии правильно распознать любую из цифр от нуля до девяти, в случае если они не входили в состав нашего тренировочного набора данных.

Следующий ниже рисунок иллюстрирует принцип работы задачи бинарной классификации при наличии 30 тренировочных образцов[2]: 15 образцов маркированы как *отрицательный класс* (круги) и другие 15 – как *положительный класс* (знаки «плюс»). В этом сценарии наш набор данных является двумерным, то есть каждый образец имеет два связанных с ним значения: $x_1$ и $x_2$. Теперь мы можем применить

[1] Класс – это множество всех объектов с данным значением метки. – *Прим. перев.*

[2] Под образцом, или экземпляром (sample), здесь и далее по тексту подразумевается совокупность признаков, которым поставлены в соответствие конкретные значения. – *Прим. перев.*

алгоритм машинного обучения с учителем для извлечения правила – граница решения представлена черной пунктирной линией, – которое может выделить эти два класса и затем распределить новые данные в каждую из этих двух категорий при наличии значений $x_1$ и $x_2$:




### *Задача регрессии – предсказание значений непрерывной целевой переменной*

В предыдущем разделе мы узнали, что задача классификации заключается в назначении экземплярам категориальных, неупорядоченных меток. Второй тип обучения с учителем представлен предсказанием непрерывных результатов, который еще именуется регрессионным анализом, или методами восстановления зависимости между переменными. В *регрессионном анализе* нам даны несколько *предикторных* (объясняющих) переменных и непрерывная (результирующая) переменная отклика, и мы пытаемся найти между этими переменными связь, которая позволит нам предсказывать результат.

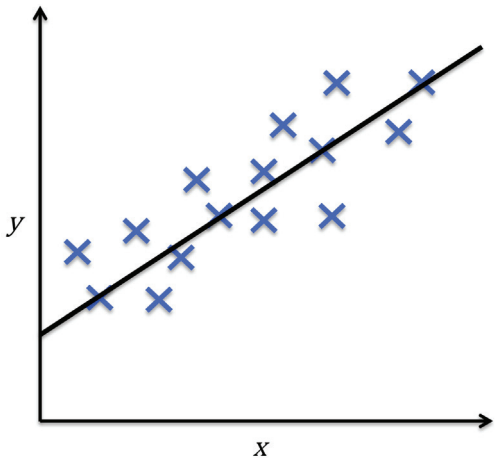
Например, предположим, что нас интересует предсказание оценок студентов за тест по математике SAT Math[1]. Если между затраченным на подготовку к тесту временем и итоговыми оценками существует связь, то мы могли бы воспользоваться ею в качестве тренировочных данных для извлечения модели, в которой время учебы используется для предсказания экзаменационных отметок будущих студентов, планирующих пройти этот тест.

Термин *регрессия* был введен Фрэнсисом Гальтоном в своей статье «*Регрессия к посредственности по наследственному статусу*» (Regression Towards Mediocrity in He-

[1] SAT Math – тест по математике для будущих бакалавров (http://test-sat.ru/sat-math/). – *Прим. перев.*

reditary Stature) в 1886 г. Гальтон описал биологическое явление, заключающееся в том, что дисперсия в росте среди представителей популяции со временем не увеличивается. Он обнаружил, что рост родителей не передается их детям, а рост детей регрессирует к среднеарифметическому росту по популяции.

Следующий ниже рисунок иллюстрирует основную идею *линейной регрессии*. При наличии предикторной переменной $x$ и переменной отклика $y$ мы выполняем под эти данные подгонку прямой, которая минимизирует расстояние – обычно среднеквадратичное – между точками образцоы и подогнанной линией. Далее мы можем воспользоваться полученными из этих данных пересечения оси[1] и наклоном прямой с неким угловым коэффициентом для прогнозирования результирующей переменной в новых данных:



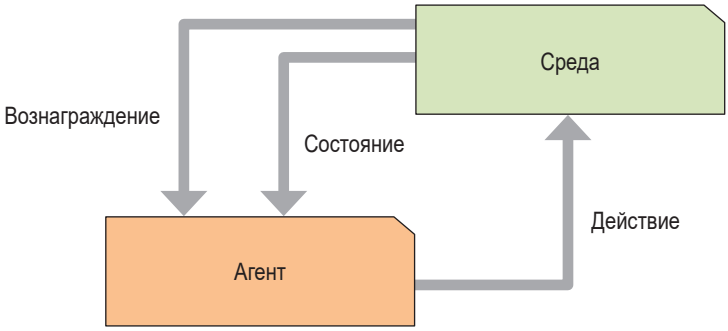

## Решение интерактивных задач на основе обучения с подкреплением

Еще одним типом машинного обучения является обучение с подкреплением. Задача обучения с подкреплением состоит в выработке системы (агента), которая улучшает свое качество на основе взаимодействий со *средой*. Поскольку информация о текущем состоянии среды, как правило, содержит и так называемый сигнал *вознаграждения*, обучение с подкреплением можно представить как область, имеющую отношение к обучению с учителем. Однако в обучении с подкреплением эта обратная связь является не меткой или значением, раз и навсегда определенными в результате прямых наблюдений, а мерой того, насколько хорошо действие было оценено функцией *вознаграждения*. В ходе взаимодействия со средой агент на основе разведочного подхода путем проб и ошибок или совещательного планирования может использовать обучение с подкреплением для вычленения серии действий, которые максимизируют это вознаграждение.

---

[1] Геометрически пересечение (intercept) – это точка, в которой линия регрессии пересекает ось *Y*. В формуле линейной регрессии пересечение – это свободный коэффициент, которому равна зависимая переменная, если предиктор, или независимая переменная, равен нулю. – *Прим. перев.*

Популярным примером обучения с подкреплением является шахматный движок. Здесь агент выбирает серию ходов в зависимости от состояния шахматной доски (среды), где вознаграждение можно задать как *выигрыш* либо *проигрыш* в конце игры:



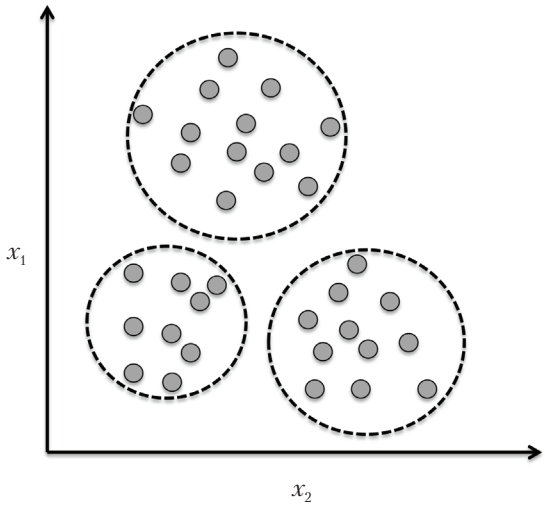

## Обнаружение скрытых структур при помощи обучения без учителя

В обучении с учителем, когда мы тренируем нашу модель, мы знаем *правильный ответ* заранее, а в обучении с подкреплением мы определяем меру *вознаграждения* за выполненные агентом отдельно взятые действия. С другой стороны, в обучении без учителя мы имеем дело с немаркированными данными или данными с *неизвестной структурой*. Используя методы обучения без учителя, мы можем разведать структуру данных с целью выделения содержательной информации без контроля со стороны известной результирующей переменной или функции вознаграждения.

## Выявление подгрупп при помощи кластеризации

*Кластеризация* – это метод разведочного анализа данных, который позволяет организовать груду информации в содержательные подгруппы (*кластеры*), не имея никаких предварительных сведений о принадлежности группе. Каждый кластер, который может появиться во время анализа, обозначает группу объектов, которые обладают определенной степенью подобия и одновременно больше отличаются от объектов в других кластерах, поэтому кластеризацию также иногда называют «классификацией без учителя». Кластеризация незаменима для структурирования информации и получения содержательных связей внутри данных. Например, этот метод позволяет специалистам по маркетингу обнаруживать группы потребителей на основе их интересов с целью разработки конкретных маркетинговых программ.

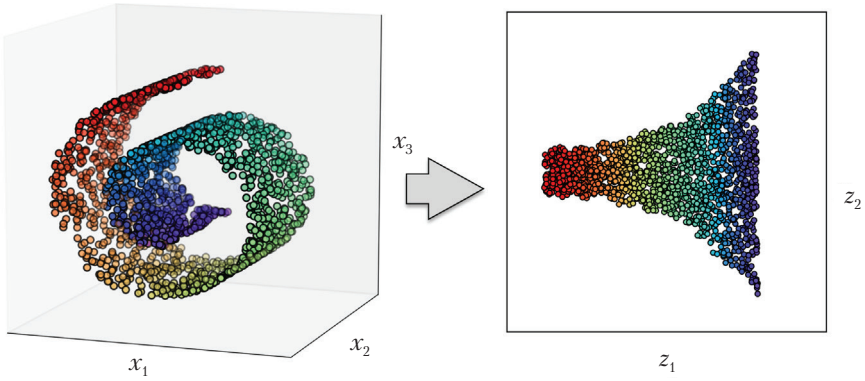
Приведенный ниже рисунок иллюстрирует применение кластеризации для размещения немаркированных данных в три разные группы на основе подобия их признаков $x_1$ и $x_2$:




### *Снижение размерности для сжатия данных*

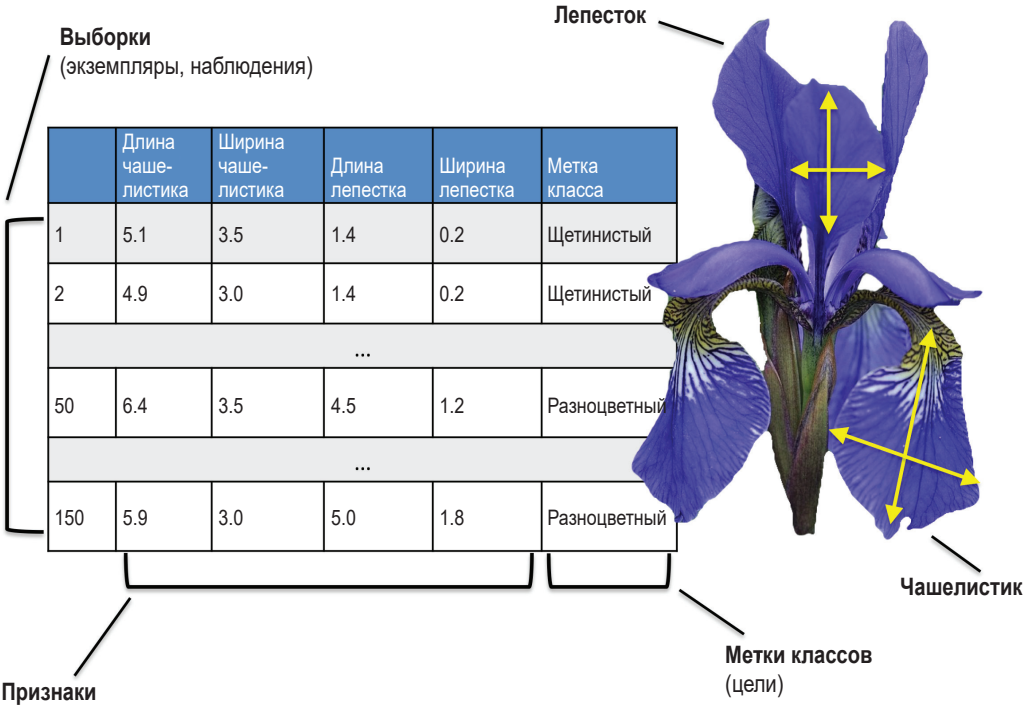
Еще одна подобласть обучения без учителя представлена методом *снижения размерности*. Часто приходится работать с данными высокой размерности – каждое наблюдение состоит из большого числа результатов измерений, которые могут представлять серьезную проблему для ограниченного объема памяти и вычислительной производительности алгоритмов машинного обучения. Подход на основе снижения размерности без учителя широко используется во время предобработки признаков с целью удаления из данных шума, который тоже может ухудшить предсказательную способность некоторых алгоритмов, и для сжатия данных в подпространство меньшей размерности при сохранении большей части релевантной информации.

Кроме того, иногда снижение размерности может быть полезно для визуализации данных – например, высокоразмерный набор признаков может быть спроецирован на одно-, двух- или трехмерные пространства признаков с целью их визуализации при помощи трех- или двухмерных точечных графиков или гистограмм. Рисунок ниже показывает пример использования нелинейного снижения размерности для сжатия трехмерного *швейцарского рулета* (Swiss roll) в новое двумерное подпространство признаков:



# Введение в основополагающую терминологию и систему обозначений

Обсудив три обширные категории машинного обучения – с учителем, без учителя и с подкреплением, теперь обратимся к базовой терминологии, которую мы будем использовать в последующих главах. Следующая ниже таблица изображает выдержку из набора данных *Ирисы Фишера*[1], который представляет собой классический пример из области машинного обучения. Этот набор данных содержит данные измерений 150 цветков ириса трех видов: *ирис щетинистый (Iris setosa), ирис виргинский (Iris virginica) и ирис разноцветный (Iris versicolor)* по четырем характеристикам, или признакам. Здесь каждый образец цветков представляет в нашем наборе данных одну строку, а результаты измерений цветков в сантиметрах хранятся в столбцах, называемых также признаками набора данных:




| | Длина чашелистика | Ширина чашелистика | Длина лепестка | Ширина лепестка | Метка класса |
|---|---|---|---|---|---|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | Щетинистый |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | Щетинистый |
| ... | | | | | |
| 50 | 6.4 | 3.5 | 4.5 | 1.2 | Разноцветный |
| ... | | | | | |
| 150 | 5.9 | 3.0 | 5.0 | 1.8 | Разноцветный |

[1] Ирисы Фишера (Iris) – набор данных для задачи классификации, на примере которого Рональд Фишер в 1936 году продемонстрировал разработанный им метод дискриминантного анализа. Для каждого экземпляра измерялись четыре характеристики/признака (в сантиметрах): длина чашелистика (sepal length), ширина чашелистика (sepal width); длина лепестка (petal length), ширина лепестка (petal width). Этот набор данных уже стал классическим и часто используется в литературе для иллюстрации работы различных статистических алгоритмов (см. https://ru.wikipedia.org/wiki/Ирисы_Фишера). – *Прим. перев.*

Чтобы система обозначений и реализация оставались простыми и одновременно эффективными, мы привлечем несколько базовых элементов *линейной алгебры*. В следующих главах для обозначения наших данных мы воспользуемся *матричной* и *векторной* системами обозначений. Мы будем следовать общепринятому соглашению представлять каждый образец как отдельную строку в матрице признаков $\boldsymbol{X}$, где каждый признак хранится как отдельный столбец.

Тогда набор данных цветков ириса, состоящий из 150 образцов и 4 признаков, можно записать как матрицу размера 150×4, где $X \in \mathbb{R}^{150\times 4}$:

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{bmatrix}.$$

В остальной части этой книги мы будем использовать надстрочный индекс ($i$) для обозначения $i$-й тренировочного образца и подстрочный индекс $j$ для обозначения $j$-й размерности тренировочного набора данных.

Мы используем заглавные (прописные) буквы, набранные жирным шрифтом, для обозначения векторов ($x \in \mathbb{R}^{n\times 1}$) и строчные буквы, набранные жирным шрифтом, для обозначения матриц соответственно ($X \in \mathbb{R}^{n\times m}$). Для обозначения в векторе или матрице отдельных элементов мы пишем буквы курсивом, соответственно $x^{(n)}$ или $x_{(m)}^{(n)}$.

Например, $x_1^{150}$ обозначает первое измерение 150-го образца цветков, *длину чашелистика*. Таким образом, каждая строка в этой матрице признаков представляет один экземпляр цветка и может быть записана как четырехмерный вектор-строка $x^{(i)} \in \mathbb{R}^{1\times 4}$, $x^{(i)} = [x_1^{(i)}\ x_2^{(i)}\ x_3^{(i)}\ x_4^{(i)}]$.

Каждое измерение в признаках – это 150-мерный вектор-столбец $x_j \in \mathbb{R}^{150\times 1}$. Например:

$$x_j = \begin{bmatrix} x_j^{(1)} \\ x_j^{(2)} \\ \vdots \\ x_j^{(150)} \end{bmatrix}.$$

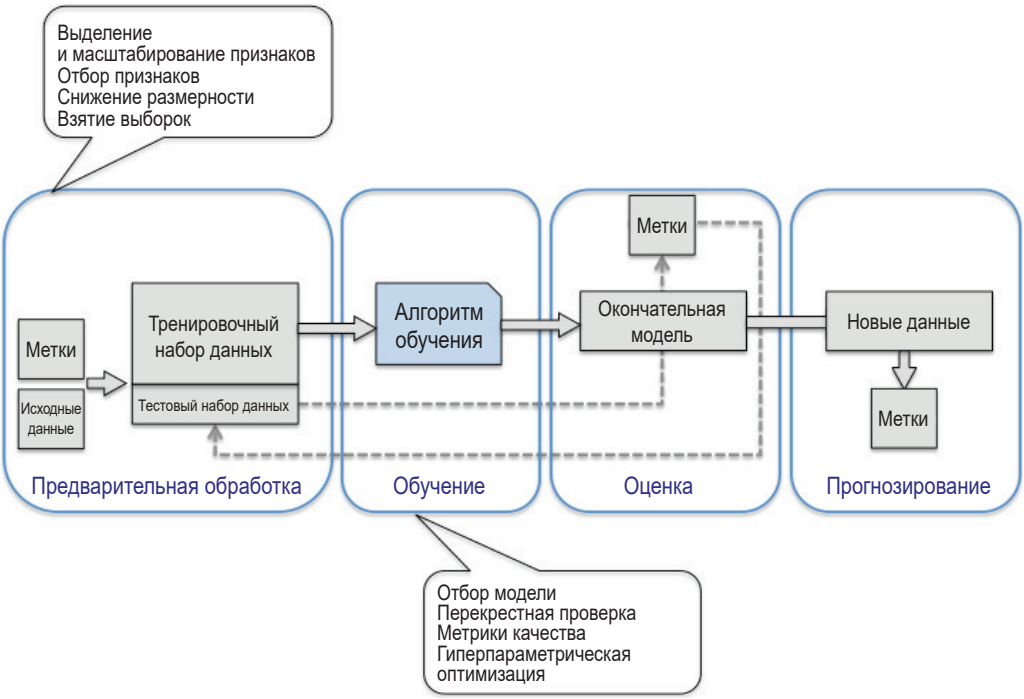
Целевые переменные (в данном случае метки классов) мы храним аналогичным образом, как 150-мерный вектор-столбец:

$$y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(150)} \end{bmatrix} (y \in \{\text{Вергинский, Разноцветный, Щетинистый}\}).$$

# Дорожная карта для построения систем машинного обучения

В предыдущих разделах мы обсудили основные принципы машинного обучения и три разных типа обучения. В этом разделе мы затронем другие важные составляющие системы машинного обучения, сопровождающие алгоритм обучения. Приве-

денная ниже схема показывает типичную диаграмму потока операций при использовании машинного обучения в *прогнозном моделировании*, которое мы обсудим в последующих подразделах:




## Предобработка – приведение данных в приемлемый вид

Исходные (необработанные) данные редко поступают в том виде и в той форме, которые необходимы для оптимальной работы алгоритма обучения. Поэтому *предобработка* данных является одним из самых важных этапов в любом приложении с использованием машинного обучения. Если в качестве примера взять набор данных цветков ириса из предыдущего раздела, то исходные данные можно представить как серию снимков цветков, из которых мы хотим выделить содержательные признаки. Полезными признаками могут быть цвет, оттенок, интенсивность цвета цветков, их высота, длина и ширина. Многие алгоритмы машинного обучения также требуют, чтобы отобранные признаки в целях оптимального качества работы находились в одной и той же шкале, что часто достигается путем приведения признаков к диапазону [0, 1] или стандартному нормальному распределению с нулевым средним значением и единичной дисперсией, как мы убедимся в более поздних главах.

Некоторые отобранные признаки могут высоко коррелироваться и поэтому быть до известной степени избыточными. В этих случаях целесообразно использовать методы снижения размерности в целях сжатия признаков в подпространство более низкой размерности. Преимущество снижения размерности нашего пространства признаков заключается в том, что требуется меньший объем памяти, в результате чего алгоритм обучения может выполняться намного быстрее.

Чтобы подтвердить, что наш алгоритм машинного обучения не только хорошо работает на тренировочном наборе, но и хорошо обобщается на новые данные, нам также понадобится случайным образом подразделить набор данных на два отдельных подмножества: тренировочный и тестовый наборы. Мы используем тренировочный набор, чтобы натренировать и оптимизировать нашу машиннообучаемую модель, в то время как тестовый набор мы храним до самого конца с целью выполнить оценку окончательной модели.

## Тренировка и отбор прогнозной модели

Как мы убедимся в последующих главах, для решения различных практических задач было разработано большое количество разных алгоритмов машинного обучения. Из известной статьи Дэвида Вольперта «*Теоремы "Никаких бесплатных обедов"*» (David Wolpert, No Free Lunch Theorems) можно сделать важный вывод[1] – мы не можем осуществлять машинное обучение «бесплатно» («*Отсутствие априорных различий между алгоритмами обучения*», D. H. Wolpert 1996; «*Теоремы No Free Lunch (NFL-теоремы) для оптимизации*», D. H. Wolpert и W. G. Macready, 1997). Интуитивно эту концепцию можно увязать с популярным высказыванием Абрахама Маслоу «*Предполагаю, что когда у вас в руках всего один инструмент— молоток, то заманчиво рассматривать все вокруг как гвозди*» (1966)[2]. Например, каждый алгоритм классификации имеет присущие им смещения, и никакая модель классификации не обладает превосходством перед другими, при условии что мы не делаем допущений в отношении задачи. Поэтому на практике важно сравнить, по крайней мере, несколько различных алгоритмов, чтобы натренировать и отобрать самую качественную модель. Но прежде чем мы сможем сопоставить различные модели, мы сначала должны принять решение относительно метрики для измерения их качества. Одна из самых распространенных метрик – это верность (accuracy), иногда именуемая правильностью, которая определяется как доля правильно классифицированных экземпляров от общего числа предсказаний[3].

Здесь возникает законный вопрос: *а как узнать, какая модель работает хорошо на окончательном тестовом наборе данных и реальных данных, если мы не используем этот тестовый набор для отбора модели и держим его для окончательной оценки модели?* Для решения заложенной в этот вопрос проблемы можно воспользоваться различными методами перекрестной проверки, или кросс-валидации, где тренировочный набор данных далее подразделяется на тренировочное и *провероч-*

---

[1] Д. Вульперт (1996) показал, что попытки найти алгоритм обучения без смещений бесполезны. Кроме того, он продемонстрировал, что в свободном от шума сценарии, где функция потерь эквивалентна коэффициенту ошибочной классификации, если вы интересуетесь ошибкой вне тренировочного набора, то между алгоритмами обучения нет никаких априорных различий. См. http://www.no-free-lunch.org/. – *Прим. перев.*

[2] Иными словами, когда у вас в руках молоток, все задачи кажутся гвоздями. – *Прим. перев.*

[3] В машинном обучении точность (precision) является мерой статистической изменчивости и описывает случайные ошибки. Верность (accuracy) является мерой статистического смещения и описывает систематические ошибки. Резюмируя, верность модели (алгоритма) – это близость результатов измерений к истинному значению и в силу этого часто переводится как «правильность» и «адекватность»; точность результатов измерений (precision) – это их повторяемость, репродуцируемость. – *Прим. перев.*

*ное* подмножества для оценки *обобщающей способности* модели[1]. Наконец, мы не можем ожидать, что параметры различных алгоритмов обучения, предоставляемых программными библиотеками по умолчанию, оптимальны для нашей конкретной практической задачи. Поэтому в более поздних главах мы часто будем использовать методы *гиперпараметрической оптимизации*, которые помогут выполнить тонкую настройку качества нашей модели. Интуитивно гиперпараметры можно представить как параметры, которые не были извлечены из данных и являются рычагами управления моделью, которые можно изменять с целью улучшения ее качества работы. Их роль станет намного яснее в более поздних главах, когда мы увидим действующие примеры.

### Оценка моделей и прогнозирование на ранее не встречавшихся экземплярах данных

После того как мы отобрали модель, подогнанную под тренировочный набор данных, мы можем воспользоваться тестовым набором данных и оценить, насколько хорошо она работает на этих ранее не встречавшихся ей данных, чтобы вычислить ошибку обобщения. Если мы удовлетворены качеством модели, то теперь мы можем использовать ее для прогнозирования новых будущих данных. Важно отметить, что параметры для таких ранее упомянутых процедур, как масштабирование признаков и снижение размерности, получают исключительно из тренировочного набора данных, и те же самые параметры позже применяют повторно для трансформирования тестового набора данных, а также любых новых образцов данных – в противном случае измеренное на тестовых данных качество модели может оказаться сверхоптимистичной.

## Использование Python для машинного обучения

Python – один из самых популярных языков программирования для науки о данных и потому обладает огромным количеством полезных дополнительных библиотек, разработанных его колоссальным сообществом программистов.

Учитывая, что производительность таких интерпретируемых языков, как Python, для вычислительно-емких задач хуже, чем у языков программирования более низкого уровня, были разработаны дополнительные библиотеки, такие как *NumPy* и *SciPy,* которые опираются на низкоуровневые реализации на Fortran и C для быстродействующих и векторизованных операций на многомерных массивах.

---

[1] Перекрестная проверка, валидация (cross validation, CV) – это метод подтверждения работоспособности моделей, который используется для оценки того, насколько точно прогнозная модель будет работать на практике. Задача перекрестной проверки – выделить из тренировочного набора данных подмножество данных, которое будет использоваться с целью «протестировать» модель на этапе ее тренировки (так называемый перекрестно-проверочный набор), с тем чтобы ограничить такие проблемы, как переобучение, и дать представление о том, как модель будет обобщаться на независимый набор данных (т. е. ранее не встречавшихся данных, например из реальной задачи). – *Прим. перев.*

Для выполнения задач программирования машинного обучения мы главным образом будем обращаться к библиотеке *scikit-learn*, которая на сегодня является одной из самых популярных и доступных библиотек машинного обучения с открытым исходным кодом.

## Установка библиотек Python

Python доступен для всех трех главных операционных систем – Microsoft Windows, Mac OS X и Linux, – и его установщик и документацию можно скачать с официального веб-сайта Python: https://www.python.org.

Эта книга написана для версии Python >= 3.4.3, при этом рекомендуем использовать самую последнюю версию Python 3 (на момент перевода последней была версия 3.5.2), несмотря на то что большинство примеров программ может также быть совместимо с Python >= 2.7.10 (на момент перевода – 2.7.12). Если для выполнения примеров программ вы решили использовать Python 2.7, то удостоверьтесь, что вы знакомы с главными различиями между этими двумя версиями Python. Хорошее краткое описание различий между Python 3.5 и 2.7 приведено на веб-странице https://wiki.python.org/moin/Python2orPython3.

Дополнительные библиотеки, которые мы будем использовать на протяжении всей книги, можно установить при помощи менеджера пакетов *pip*, который входит в состав стандартной библиотеки Python начиная с версии 3.3. Дополнительную информацию о менеджере пакетов pip можно найти на веб-странице https://docs.python.org/3/installing/index.html.

После успешной инсталляции среды Python мы можем запускать менеджер пакетов pip из командной строки в окне терминала для установки дополнительных библиотек Python:

```
pip install НекаяБиблиотека
```

Уже установленные библиотеки можно обновить при помощи опции --upgrade:

```
pip install НекаяБиблиотека --upgrade
```

Настоятельно рекомендуем к использованию альтернативный дистрибутив Python для научных вычислений Anaconda от компании Continuum Analytics. Это бесплатный, включая коммерческое использование, и готовый к использованию в среде предприятия дистрибутив Python, который объединяет все ключевые библиотеки Python, необходимые для работы в области науки о данных, математики и разработки, в одном удобном для пользователя кросс-платформенном дистрибутиве. Установщик Anaconda можно скачать с http://continuum.io/downloads, а краткое вводное руководство Anaconda доступно по прямой ссылке https://store.continuum.io/static/img/Anaconda-Quickstart.PDF.

После успешной установки дистрибутива Anaconda можно установить новые пакеты Python, используя для этого следующую ниже команду:

```
conda install НекаяБиблиотека
```

Существующие пакеты можно обновить при помощи команды:

```
conda upgrade НекаяБиблиотека
```

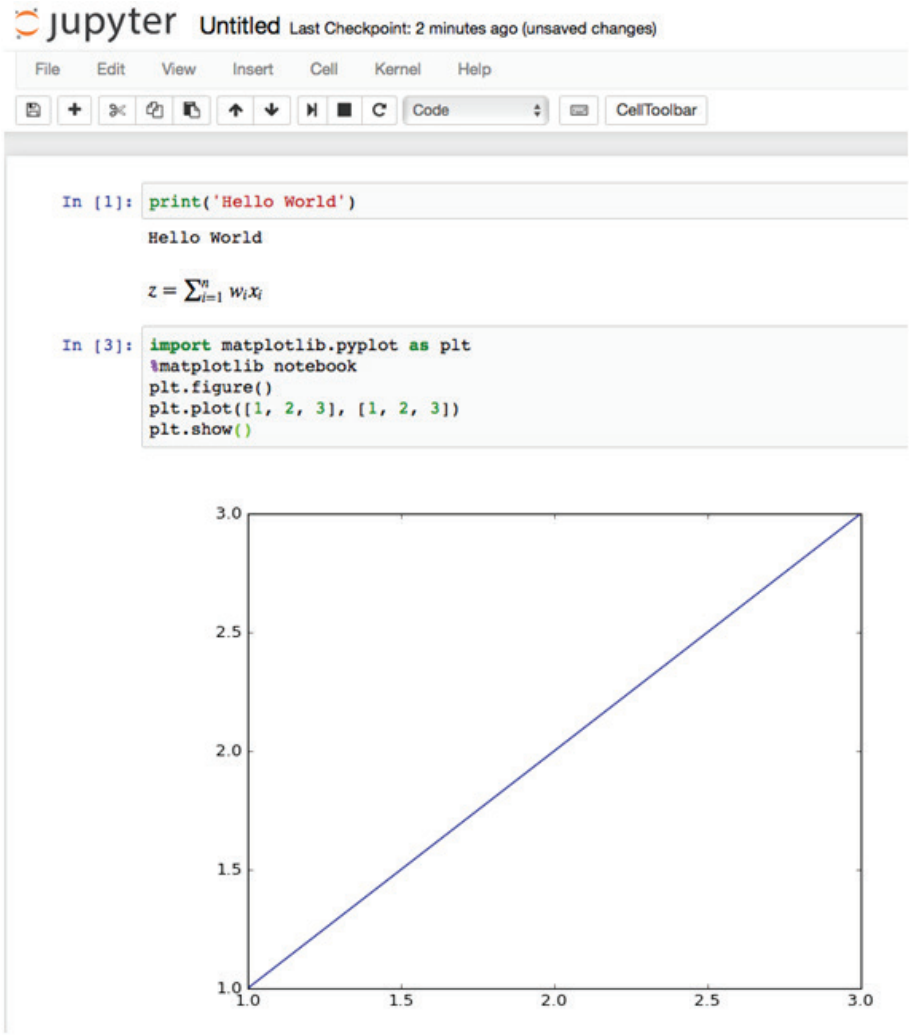
На протяжении всей книги для хранения и управления данными мы будем в основном использовать многомерные массивы *NumPy*. От случая к случаю мы будем пользоваться надстройкой над библиотекой NumPy – библиотекой *pandas*, которая предлагает дополнительные высокоуровневые инструменты управления данными, делая работу с табличными данными еще более удобной. В помощь нашей познавательной деятельности, а также в целях визуализации количественных данных, что порой бывает чрезвычайно полезно делать для интуитивного понимания материала, мы будем пользоваться полностью настраиваемой библиотекой *matplotlib*.

Номера версий основных пакетов Python, которые использовались при написании этой книги, упомянуты ниже. Удостоверьтесь, что номера версий ваших установленных пакетов идентичны приведенным в списке или выше, чтобы примеры программ гарантированно выполнялись корректно:

- NumPy 1.9.1;
- SciPy 0.14.0;
- scikit-learn 0.15.2;
- matplotlib 1.4.0;
- панды 0.15.2.

### Блокноты (записные книжки) Jupyter/IPython

Некоторым читателям, возможно, интересно, что это за файлы с расширением «*.ipynb*» среди примеров программ. Это файлы блокнотов Jupyter (ранее называвшихся записными книжками IPython). Блокнотам Jupyter было отдано предпочтение над простыми сценарными файлами Python «*.py*», потому что, по нашему мнению, они наилучшим образом подходят для проектов в области анализа данных! Блокноты Jupyter позволяют иметь все в одном месте: наш исходный код, результаты выполнения исходного кода, графики данных и документацию, которая поддерживает синтаксис упрощенной разметки Markdown и мощный синтаксис LaTeX!




Записные книжки IPython были переименованы в блокноты Jupyter (http://jupyter.org) в начале 2015 г. Jupyter – это зонтичный проект, который наряду с Python предназначен для поддержки других языков программирования, в том числе Julia, R и многих других (уже более 40 языков). Впрочем, переживать не следует, т. к. для пользователя Python имеется всего лишь различие в терминологии.

Блокнот Jupyter можно установить, как обычно, при помощи pip.

```
$ pip install jupyter notebook
```

Как вариант можно воспользоваться установщиком Conda, в случае если мы инсталлировали Anaconda либо Miniconda:

```
$ conda install jupyter notebook
```

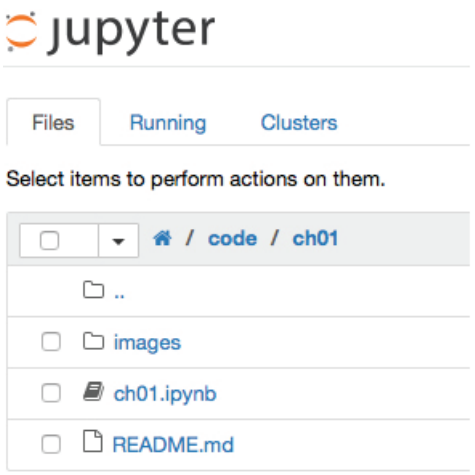
Чтобы открыть блокнот Jupyter, мы при помощи команды cd переходим в каталог с нашими примерами программ, например

```
$ cd ~/code/python-machine-learning-book
```

и запускаем блокнот Jupyter при помощи команды

```
$ jupyter notebook
```

Jupyter запустится в вашем браузере по умолчанию (как правило, по адресу http://localhost:8888/). Теперь мы можем просто выбрать блокнот, который вам нужна, из меню Jupyter.




За дополнительной информацией о проекте Jupyter рекомендуем обращаться к «*Руководству для начинающих по Jupyter*» (http://jupyter-notebook-beginner-guide.readthedocs.org/en/latest/what_is_jupyter.html).

## Резюме

В этой главе мы провели широкомасштабную разведку обширной области информатики – машинного обучения и ознакомились с общей картиной и ее главными концепциями, которые мы собираемся разобрать в следующих главах более подробно.

Мы узнали, что обучение с учителем состоит из двух важных подобластей: задач классификации и регрессии. В отличие от модели классификации, которая позволяет распределять (категоризировать) объекты по известным классам, мы можем применять регрессию с целью предсказания непрерывных результатов для целевых переменных. Обучение без учителя не только предлагает удобные методы обнаружения структур в немаркированных данных, но и может использоваться для сжатия данных на этапе предобработки признаков.

Мы кратко прошлись по типичной дорожной карте применения машинного обучения для решения практических задач, которую мы возьмем за основу при более глубоком обсуждении и рассмотрении живых примеров в последующих главах.

В заключение мы инсталлировали среду Python, установили и обновили требуемые библиотеки, чтобы подготовиться к тому, чтобы увидеть машинное обучение в действии.

В следующей главе мы реализуем несколько самых ранних алгоритмов машинного обучения, используемых для решения задачи классификации, которые подготовят нас к *главе 3* «*Обзор классификаторов с использованием библиотеки scikit-learn*», где при помощи библиотеки машинного обучения с открытым исходным кодом scikit-learn мы охватим более продвинутые алгоритмы машинного обучения. Поскольку алгоритмы машинного обучения учатся на данных, критически важно подавать им нужную информацию, и в *главе 4* «*Создание хороших тренировочных наборов – предобработка данных*» мы рассмотрим важные методы предварительной обработки данных. В *главе 5* «*Сжатие данных путем снижения размерности*» мы узнаем о методах снижения размерности, которые позволят сжимать набор данных в подпространство признаков более низкой размерности, что может позитивно сказаться на вычислительной эффективности. Важным аспектом создания машинно- обучаемых моделей является выполнение оценки их качества и оценки того, насколько хорошо они могут делать прогнозы на новых, ранее не встречавшихся им данных. В *главе 6* «*Изучение наиболее успешных методов оценки моделей и тонкой настройки гиперпараметров*» мы узнаем все о наиболее успешных практических методах тонкой настройки моделей и их оценки. В определенных сценариях мы все еще можем быть не удовлетворены качеством нашей прогнозной модели, несмотря на то что мы, возможно, провели часы или даже дни, тщательно ее настраивая и проверяя. В *главе 7* «*Объединение моделей для методов ансамблевого обучения*» мы узнаем, как объединять различные машиннообучаемые модели для создания еще более мощных систем прогнозирования.

После того как мы охватим все важные концепции типичного конвейера машинного обучения[1], в *главе 8* «*Применение алгоритмов машинного обучения в анализе мнений*» займемся реализацией модели, предназначенной для прогнозирования эмоций в тексте, в *главе 9* «*Встраивание алгоритма машинного обучения в веб-приложение*» мы встроим ее в веб-приложение, чтобы поделиться им с миром. Затем в *главе 10* «*Прогнозирование непрерывных целевых величин на основе регрессионного анализа*» мы воспользуемся алгоритмами машинного обучения для регрессионного анализа, которые позволят прогнозировать непрерывные выходные переменные, и в *главе 11* «*Работа с немаркированными данными – кластерный анализ*» мы применим алгоритмы кластеризации, которые позволят находить в данных скрытые структуры. Последние две главы в этой книге будут касаться искусственных нейронных сетей, позволяющих решать такие сложные задачи, как распознавание изображений и речи. Эти темы в настоящее время являются одними из самых горячих среди научных исследований в области машинного обучения.

---

[1] В информатике конвейер представляет собой набор элементов обработки данных, соединенных последовательно, где выход одного элемента является входом следующего. Элементы конвейера часто выполняются параллельно по принципу квантования по времени. – *Прим. перев.*