



# ОГЛАВЛЕНИЕ

<b>Об авторах .....</b>	<b>11</b>
<b>О рецензентах.....</b>	<b>13</b>
<b>Предисловие .....</b>	<b>15</b>
О содержании книги .....	15
Что необходимо для чтения этой книги.....	17
На кого рассчитана эта книга .....	17
Графические выделения.....	17
Отзывы.....	18
Поддержка клиентов .....	18
Загрузка кода примеров .....	19
Опечатки.....	19
Нарушение авторских прав .....	19
Вопросы.....	20
<b>Глава 1. Введение в машинное обучение на языке Python .....</b>	<b>21</b>
Машинное обучение и Python – команда мечты .....	22
Что вы узнаете (и чего не узнаете из этой книги) .....	23
Что делать, если вы застряли .....	25
Приступая к работе .....	26
Введение в NumPy, SciPy и matplotlib .....	26
Установка Python .....	27
NumPy как средство эффективной и SciPy как средство интеллектуальной обработки данных .....	27
Изучаем NumPy.....	27
Изучаем SciPy.....	32
Наше первое (простенькое) приложение машинного обучения .....	33
Чтение данных .....	33
Предварительная обработка и очистка данных .....	35
Выбор подходящей модели и обучающего алгоритма.....	36
Резюме .....	46

<b>Глава 2. Классификация в реальной жизни.....</b>	<b>47</b>
Набор данных Iris.....	48
Визуализация – первый шаг к цели .....	48
Построение первой модели классификации .....	50
Оценка качества – резервирование данных и перекрестная проверка.....	53
Построение более сложных классификаторов.....	57
Более сложный набор данных и более сложный классификатор ...	58
Набор данных Seeds .....	58
Признаки и подготовка признаков.....	59
Классификация по ближайшему соседу .....	60
Классификация с помощью scikit-learn .....	61
Решающие границы .....	62
Бинарная и многоклассовая классификация .....	65
Резюме .....	66
<b>Глава 3. Кластеризация – поиск взаимосвязанных сообщений .....</b>	<b>68</b>
Измерение сходства сообщений .....	69
Как не надо делать .....	69
Как надо делать .....	70
Предварительная обработка – количество общих слов как мера сходства .....	71
Преобразование простого текста в набор слов .....	71
Развитие концепции стоп-слов .....	80
Чего мы достигли и к чему стремимся.....	81
Кластеризация .....	82
Метод K-средних .....	83
Тестовые данные для проверки наших идей .....	85
Кластеризация сообщений .....	87
Решение исходной задачи.....	88
Другой взгляд на шум .....	90
Настройка параметров .....	92
Резюме .....	92
<b>Глава 4. Тематическое моделирование.....</b>	<b>93</b>
Латентное размещение Дирихле .....	93
Построение тематической модели .....	95
Сравнение документов по темам.....	100
Моделирование всей википедии.....	103
Выбор числа тем .....	106

Резюме .....	107
<b>Глава 5. Классификация – выявление плохих ответов .....</b>	<b>109</b>
План действий.....	109
Учимся классифицировать классные ответы .....	110
Подготовка образца.....	110
Настройка классификатора.....	110
Получение данных .....	111
Сокращение объема данных .....	112
Предварительная выборка и обработка атрибутов .....	112
Что считать хорошим ответом? .....	114
Создание первого классификатора .....	115
Метод k ближайших соседей.....	115
Подготовка признаков .....	116
Обучение классификатора .....	117
Измерение качества классификатора .....	117
Проектирование дополнительных признаков .....	118
Как поправить дело? .....	121
Дилемма смещения-дисперсии .....	122
Устранение высокого смещения .....	122
Устранение высокой дисперсии .....	123
Низкое или высокое смещение? .....	123
Логистическая регрессия.....	125
Немного математики на простом примере.....	126
Применение логистической регрессии к задаче классификации .....	128
Не верностью единой – точность и полнота .....	129
Упрощение классификатора.....	133
К поставке готов!.....	134
Резюме .....	135
<b>Глава 6. Классификация II – анализ эмоциональной окраски.....</b>	<b>136</b>
План действий.....	136
Чтение данных из Твиттера.....	137
Введение в наивный байесовский классификатор.....	137
О теореме Байеса .....	138
Что значит быть наивным .....	139
Использование наивного байесовского алгоритма для классификации .....	140
Учет ранее не встречавшихся слов и другие тонкости .....	143
Борьба с потерей точности при вычислениях .....	144

Создание и настройка классификатора .....	147
Сначала решим простую задачу .....	147
Использование всех классов .....	150
Настройка параметров классификатора .....	153
Очистка твитов .....	157
Учет типов слов .....	159
Определение типов слов .....	159
Удачный обмен с помощью SentiWordNet .....	162
Наш первый оценщик .....	164
Соберем все вместе .....	166
Резюме .....	167
<b>Глава 7. Регрессия .....</b>	<b>168</b>
Прогнозирование стоимости домов с помощью регрессии .....	168
Многомерная регрессия .....	172
Перекрестная проверка для регрессии .....	173
Регрессия со штрафом, или регуляризованная регрессия .....	174
Штрафы L1 и L2 .....	175
Lasso и эластичная сеть в библиотеке scikit-learn .....	176
Визуализация пути в Lasso .....	177
Сценарии Р-больше-N .....	178
Пример, основанный на текстовых документах .....	179
Объективный подход к заданию гиперпараметров .....	181
Резюме .....	185
<b>Глава 8. Рекомендации .....</b>	<b>186</b>
Прогноз и рекомендование оценок .....	186
Разделение данных на обучающие и тестовые .....	188
Нормировка обучающих данных .....	189
Рекомендование на основе ближайших соседей .....	191
Регрессионный подход к рекомендованию .....	195
Комбинирование нескольких методов .....	196
Анализ корзины .....	199
Получение полезных прогнозов .....	200
Анализ корзины покупок в супермаркете .....	201
Поиск ассоциативных правил .....	204
Более сложный анализ корзины .....	206
Резюме .....	207
<b>Глава 9. Классификация по музыкальным жанрам .....</b>	<b>208</b>
План действий .....	208
Получение музыкальных данных .....	209
Преобразование в формат WAV .....	209

Взгляд на музыку .....	210
Разложение на синусоидальные волны .....	211
Применение БПФ для построения первого классификатора .....	213
Повышение гибкости эксперимента.....	213
Обучение классификатора .....	215
Применение матрицы неточностей для измерения верности в многоклассовых задачах .....	215
Альтернативный способ измерения качества классификатора с помощью рабочей характеристики приемника .....	218
Повышение качества классификации с помощью мел-частотных кепстральных коэффициентов.....	220
Резюме .....	225
<b>Глава 10. Машинное зрение .....</b>	<b>227</b>
Введение в обработку изображений.....	227
Загрузка и показ изображения.....	228
Бинаризация.....	230
Гауссово размывание.....	231
Помещение центра в фокус.....	233
Простая классификация изображений .....	235
Вычисление признаков по изображению.....	236
Создание собственных признаков .....	237
Использование признаков для поиска похожих изображений .....	239
Классификация на более трудном наборе данных.....	240
Локальные представления признаков .....	242
Резюме .....	246
<b>Глава 11. Понижение размерности .....</b>	<b>248</b>
План действий.....	249
Отбор признаков .....	249
Выявление избыточных признаков с помощью фильтров .....	250
Применение оберток для задания модели вопросов о признаках.....	257
Другие методы отбора признаков .....	259
Выделение признаков .....	260
Об анализе главных компонент .....	260
Ограничения PCA и чем может помочь LDA .....	263
Многомерное шкалирование.....	264
Резюме .....	267
<b>Глава 10. Когда данных больше .....</b>	<b>269</b>
Что такое большие данные .....	269
Использование <code>jug</code> для построения конвейера задач .....	270
Введение в задачи <code>jug</code> .....	271

Заглянем под капот .....	273
Применение <code>jug</code> для анализа данных .....	275
Повторное использование частичных результатов .....	278
Работа с Amazon Web Services .....	279
Создание виртуальной машины .....	281
Установка Python-пакетов на Amazon Linux .....	285
Запуск <code>jug</code> на облачной машине .....	286
Автоматизированная генерация кластеров с помощью <code>StarCluster</code> .....	287
Резюме .....	291
<b>Где получить дополнительные сведения</b>	
<b>о машинном обучении .....</b>	<b>293</b>
Онлайновые курсы .....	293
Книги .....	293
Вопросно-ответные сайты .....	294
Блоги .....	294
Источники данных .....	295
Участие в конкурсах .....	295
Что не вошло в книгу .....	295
Резюме .....	296
<b>Предметный указатель .....</b>	<b>297</b>



## ОБ АВТОРАХ

**Луис Педро Коэльо** (Luis Pedro Coelho) – специалист по компьютерной биологии, то есть человек, применяющий компьютеры для изучения биологических систем. Конкретно Луис анализирует ДНК микробных сообществ, чтобы охарактеризовать их поведение. Луис также много работал в области обработки биологических изображений – приложения методов машинного обучения к анализу изображений биологических образцов. Сфера его научных интересов – обработка и агрегирование больших наборов данных.

Луис защитил докторскую диссертацию в Университете Карнеги-Меллон, одном из ведущих учебных заведений в области машинного обучения. Он автор нескольких научных публикаций.

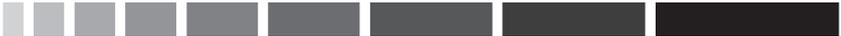
Разработкой программного обеспечения с открытым исходным кодом Луис начал заниматься в 1998 году, поставив целью применить на практике методы кодирования, которые он изучал на курсах информатики в Лиссабонском техническом университете. В 2004 году он начал программировать на Python и внес свой вклад в несколько открытых библиотек на этом языке. Он ведущий разработчик популярного пакета программ машинного зрения для Python mahotas, а также соавтор ряда программ машинного обучения.

В настоящее время Луис живет то в Люксембурге, то в Гейдельберге.

*Благодарю свою жену Риту за любовь и поддержку и дочь Анну за то, что она лучше всех.*

**Вили Ричарт** (Willi Richert) имеет степень доктора наук по машинному обучению и робототехнике. Он применял методы обучения с подкреплением, скрытые марковские модели и байесовские сети для обучения гетерогенных роботов путем подражания. В настоящее время работает в корпорации Microsoft в группе основ релевантности в Bing, где занимается различными приложениями машинного обучения, в том числе активным обучением, статистическим машинным переводом и решающими деревьями.

*Эта книга не была бы написана без поддержки моей жены Натали и моих сыновей, Линуса и Морица. Я благодарю за полезные дискуссии моих предыдущих руководителей Андреаса Бодера (Andreas Bode), Клеменса Маршнера (Clemens Marschner), Хон Янь Чжу (Hongyan Zhou) и Эрика Крестана (Eric Crestan), а также коллег и друзей Томаша Марциняка (Tomasz Marciniak), Кристиана Эйгеля (Cristian Eigel), Оливера Нихёрстера (Oliver Niehoerster) и Филиппа Эделта (Philipp Adelt). Интересные идеи, скорее всего, исходят от них, ошибки же целиком на моей совести.*



## О РЕЦЕНЗЕНТАХ

**Мэтью Бруше** получил диплом инженера во французской Высшей школе электрики (Ecole Supérieure d'Electricité) (факультет информации, обработки и измерения сигналов), а степень доктора наук по обучению без учителя на многообразиях – в Страсбургском университете, Франция. В настоящее время он занимается высокопроизводительными вычислениями в нефтяной компании и работает над программой моделирования резервуаров следующего поколения.

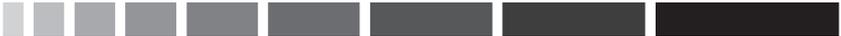
**Морис НТ Линь** программирует на Python с 2003 года. Он получил степень доктора наук по биоинформатике и диплом с отличием бакалавра по молекулярной и клеточной биологии в Университете Мельбурна. В настоящее время работает научным сотрудником в Наньянском технологическом университете, Сингапур, и почетным научным сотрудником в Университете Мельбурна, Австралия. Морис – главный редактор журнала «Computational and Mathematical Biology» и соредактор журнала «The Python Papers». Недавно Морис основал первый стартап по синтетической биологии в Сингапуре, AdvanceSyn Pte. Ltd., и занимает в нем должность директора и технического руководителя. Область его научных интересов – изучение различных аспектов жизни – биологической, искусственной и искусственного интеллекта – путем применения методов информатики и статистики. В свободное время Морис любит читать, смаковать чашечку кофе, писать в свой персональный журнал или размышлять о многообразных проявлениях жизни. Его сайт находится по адресу <http://maurice.vodien.com>, а страница в LinkedIn – по адресу <http://www.linkedin.com/in/mauriceling>.

**Ражим Ржезоржек** – технический спец и разработчик по призванию души. Он основал и возглавлял исследовательский отдел в компании Seznam.cz, ведущей поисковой системе в Центральной Европе. Защитив докторскую диссертацию, он решил не останавливаться на достигнутом, а создать для удовлетворения своей страсти к машин-

ному обучению собственную исследовательскую компанию RaRe Consulting Ltd. Компания специализируется на заказных решениях в области добычи данных и разработке передовых систем для клиентов самого разного ранга: от многонациональных корпораций до только что основанных стартапов.

Радим – также автор ряда популярных проектов с открытым исходным кодом, в том числе `gensim` и `smart_open`.

Большой любитель исследования разных культур, Радим в последние годы жил со своей женой в разных странах и намеревается перебраться к Южной Корее. Но где бы Радим ни находился, он вместе со своей командой стремится всячески популяризировать методы обработки данных и помогать компаниям по всему миру извлекать максимум пользы из машинного обучения.



# ПРЕДИСЛОВИЕ

Кто-то скажет, что вы держите эту книгу в руках (или в своей читалке) лишь благодаря счастливому совпадению. В конце концов, каждый год выходят миллионы книг, и их читают миллионы людей. А эту книгу читаете вы. А кто-то возразит, что в вашей встрече с этой книгой – или книги с вами – сыграли роль кое-какие алгоритмы машинного обучения. И мы, авторы, очень рады, что вы захотите подробнее узнать, как и почему.

В основном, эта книга отвечает на вопрос «как». Как следует обрабатывать данные, чтобы алгоритмы машинного обучения могли извлечь из них максимум полезной информации? Как выбрать подходящий алгоритм для решения задачи?

Но иногда мы задаемся также вопросом «почему». Почему важно правильно производить измерения? Почему в конкретной ситуации один алгоритм оказывается лучше другого?

Мы знаем, что для превращения в специалиста в этой области нужно знать гораздо больше. Ведь мы смогли затронуть лишь несколько «как» и уж совсем немного «почему». Но мы надеемся, что эта малость поможет вам быстро встать на ноги и двигаться дальше самостоятельно.

## О содержании книги

В главе 1 «Введение в машинное обучение на языке Python» читатель знакомится с основной идеей машинного обучения на очень простом примере. Но, несмотря на простоту, в этом примере имеет место опасность переобучения.

В главе 2 «Классификация в реальной жизни» мы используем реальные данные, чтобы продемонстрировать классификацию и научить компьютер различать различные классы цветов.

В главе 3 «Кластеризация – поиск взаимосвязанных сообщений» мы узнаем об эффективности модели набора слов, с помощью которой сумеем найти похожие сообщения, не «понимая» их смысла.

В главе 4 «Тематическое моделирование» мы не станем ограничиваться отношением сообщения только к одному кластеру, а свяжем с ним несколько тем, поскольку политематичность характерна для реальных текстов.

В главе 5 «Классификация – выявление плохих ответов» мы узнаем, как применить дилемму смещения-дисперсии к отладке моделей машинного обучения, хотя эта глава посвящена в основном использованию логистической регрессии для оценки того, хорош или плох ответ пользователя на заданный вопрос.

В главе 6 «Классификация II – анализ эмоциональной окраски» объясняется принцип работы наивного байесовского классификатора и описывается, как с его помощью узнать, несет ли твит положительный или отрицательный эмоциональный заряд.

В главе 7 «Регрессия» объясняется, как использовать классический, но не утративший актуальности метод – регрессию – при обработке данных. Вы узнаете и о более сложных методах регрессии, в частности Lasso и эластичных сетях.

В главе 8 «Рекомендование» мы построим систему рекомендаций на основе выставленных потребителями оценок. Мы также узнаем, как формировать рекомендации, имея только данные о покупках, безо всяких оценок (которые пользователи выставляют далеко не всегда).

В главе 9 «Классификация по музыкальным жанрам» мы предположим, что кто-то сознательно внес хаос в нашу огромную коллекцию музыкальных произведений, и единственная надежда навести порядок – поручить машине их классификацию. Как выяснится, иногда лучше довериться чужому опыту, чем создавать признаки самостоятельно.

В главе 10 «Машинное зрение» мы применим методы классификации к обработке изображений, выделяя признаки из данных. Мы также увидим, как с помощью этих методов можно находить похожие изображения в наборе.

Из главы 11 «Понижение размерности» мы узнаем о методах, позволяющих уменьшить объем данных, чтобы алгоритмы машинного обучения могли с ними справиться.

В главе 12 «Когда данных больше» мы рассмотрим некоторые подходы, позволяющие успешно обрабатывать большие наборы данных, задействуя несколько ядер или вычислительные кластеры. Мы также познакомимся с основами облачных вычислений (на примере служб Amazon Web Services).

В приложении «Где получить дополнительные сведения о машинном обучении» перечислены многочисленные полезные ресурсы, посвященные этой теме.

## Что необходимо для чтения этой книги

Предполагается, что вы знаете язык Python и умеете устанавливать библиотеки с помощью программы `easy_install` или `pip`. Мы не прибегаем к сложной математике типа математического анализа или матричной алгебры.

В этой книге используются следующие версии программ, хотя более поздние тоже должны подойти:

- Python 2.7 (весь код совместим также с версиями 3.3 и 3.4);
- NumPy 1.8.1;
- SciPy 0.13;
- scikit-learn 0.14.0.

## На кого рассчитана эта книга

Книга рассчитана на программистов, пишущих на Python и желающих узнать о построении систем машинного обучения с помощью библиотек с открытым исходным кодом. Мы рассматриваем основные модели машинного обучения на примерах, взятых из реальной жизни.

Эта книга будет полезна также специалистам по машинному обучению, желающим использовать Python для создания своих систем. Python – гибкий язык для быстрой разработки прототипов, сами же базовые алгоритмы написаны на C или C++ и хорошо оптимизированы. Поэтому получается быстрый и надежный код, пригодный для производственных систем.

## Графические выделения

В этой книге тип информации обозначается шрифтом. Ниже приведено несколько примеров с пояснениями.

Фрагменты кода внутри абзаца, имена таблиц базы данных, папок и файлов, URL-адреса, данные, которые вводит пользователь, и

адреса в Твиттере выделяются следующим образом: «Далее мы с помощью метода `polyld()` строим модельную функцию по параметрам модели».

Кусок кода выглядит так:

```
[aws info]
AWS_ACCESS_KEY_ID = AAKIIT7HNF6IUSN3OCAA
AWS_SECRET_ACCESS_KEY = <ваш секретный ключ>
```

Входная и выходная информация командных утилит выглядит так:

```
>>> import numpy
>>> numpy.version.full_version
1.8.1
```

**Новые термины и важные фрагменты** выделяются полужирным шрифтом. Например, элементы графического интерфейса в меню или диалоговых окнах выглядят в книге так: «После остановки машины становится доступной кнопка **Change instance type**».



---

Предупреждения и важные примечания выглядят так.

---



---

Советы и рекомендации выглядят так.

---

## Отзывы

Мы всегда рады отзывам читателей. Расскажите нам, что вы думаете об этой книге – что вам понравилось или, быть может, не понравилось. Читательские отзывы важны для нас, так как помогают выпускать книги, из которых вы черпаете максимум полезного для себя.

Чтобы отправить обычный отзыв, просто пошлите письмо на адрес [feedback@packtpub.com](mailto:feedback@packtpub.com), указав название книги в качестве темы. Если вы являетесь специалистом в некоторой области и хотели бы стать автором или соавтором книги, познакомьтесь с инструкциями для авторов по адресу [www.packtpub.com/authors](http://www.packtpub.com/authors).

## Поддержка клиентов

Счастливым обладателям книг Packt мы можем предложить ряд услуг, которые позволят извлечь из своего приобретения максимум пользы.

## Загрузка кода примеров

Вы можете скачать код примеров ко всем книгам издательства Packt, купленным на сайте <http://www.packtpub.com>. Если книга была куплена в другом месте, зайдите на страницу <http://www.packtpub.com/support>, зарегистрируйтесь, и мы отправим файлы по электронной почте.

Код к этой книге имеется также на странице сайта GitHub по адресу <https://github.com/luispedro/BuildingMachineLearningSystemsWithPython>. Этот репозиторий поддерживается в актуальном состоянии, то есть включены исправления замеченных ошибок и обновления для новых версий Python и использованных в книге пакетов.

## Опечатки

Мы проверяли содержимое книги со всем тщанием, но какие-то ошибки все же могли проскользнуть. Если вы найдете в нашей книге ошибку, в тексте или в коде, пожалуйста, сообщите нам о ней. Так вы избавите других читателей от разочарования и поможете нам сделать следующие издания книги лучше. При обнаружении опечатки просьба зайти на страницу <http://www.packtpub.com/support>, выбрать книгу, щелкнуть по ссылке **Errata Submission Form** и ввести информацию об опечатке. Проверив ваше сообщение, мы поместим информацию об опечатке на нашем сайте или добавим ее в список замеченных опечаток в разделе Errata для данной книги.

Список ранее отправленных опечаток можно просмотреть, выбрав название книги на странице <http://www.packtpub.com/books/content/support>. Запрошенная информация появится в разделе **Errata**.

Есть еще один отличный способ – зайти на сайт [www.TwoToReal.com](http://www.TwoToReal.com), где авторы стараются предоставлять техническую поддержку читателям и отвечать на их вопросы.

## Нарушение авторских прав

Незаконное размещение защищенного авторским правом материала в Интернете – проблема для всех носителей информации. В издательстве Packt мы относимся к защите прав интеллектуальной собственности и лицензированию очень серьезно. Если вы обнаружите незаконные копии наших изданий в любой форме в Интернете, пожалуйста, незамедлительно сообщите нам адрес или название веб-сайта, чтобы мы могли предпринять соответствующие меры.

Просим отправить ссылку на вызывающий подозрение в пиратстве материал по адресу [copyright@packtpub.com](mailto:copyright@packtpub.com).

Мы будем признательны за помощь в защите прав наших авторов и содействие в наших стараниях предоставлять читателям полезные сведения.

## Вопросы

Если вас смущает что-то в этой книге, вы можете связаться с нами по адресу [questions@packtpub.com](mailto:questions@packtpub.com), и мы сделаем все возможное для решения проблемы.



# ГЛАВА 1.

## Введение в машинное обучение на языке Python

Машинное обучение – это наука о том, как научить машину самостоятельно решать задачи. Вот так все просто. Но дьявол кроется в деталях, и именно поэтому вы читаете эту книгу.

Быть может, данных слишком много, а априорных знаний о них слишком мало. И вы надеетесь, что алгоритмы машинного обучения помогут справиться с этой проблемой, а потому начинаете в них разбираться. Но через некоторое время впадаете в ступор: какой же из множества алгоритмов выбрать?

Или, быть может, вас заинтересовали общие вопросы машинного обучения, и вы начали читать блоги и статьи на эту тему. Все это показалось вам таким крутым и волшебным, что вы приступили к собственным исследованиям и загрузили простенькие данные в решающее дерево или в машину опорных векторов. Но вот эксперимент завершился, и возникает вопрос: а правильно ли он был поставлен? Насколько оптимальны полученные результаты? И как узнать, не существует ли алгоритма получше? Да и вообще – правильно ли были подобраны данные?

Вы не одиноки! Мы оба (авторы) проходили через это – искали сведения о том, как на практике выглядит то, о чем пишут в учебниках по машинному обучению. Оказалось, что очень многое – «черная магия», о которой авторы стандартных учебников забывают упомянуть. Так что в некотором роде эта книга написана нами для нас же, только на несколько лет моложе. Здесь вы найдете не только краткое введение в методы машинного обучения, но и уроки, которые мы извлекли, идя по этому пути. Мы надеемся, что благодаря этой книге дорога, ведущая в самые захватывающие области Информатики, станет чуть более гладкой.

# Машинное обучение и Python – команда мечты

Цель машинного обучения – научить машину (точнее, программу) решать задачу, предъявив ей несколько примеров (с правильными и неправильными решениями). Предположим, что каждое утро, включив компьютер, вы делаете одно и то же: сортируете пришедшую почту, раскладывая письма в папки по темам. Через какое-то время эта работа вам надоест, и вы захотите ее автоматизировать. Один из возможных подходов – проанализировать собственное поведение и выписать правила, которыми вы руководствуетесь при сортировке писем. Однако это громоздкое и отнюдь не совершенное решение. Некоторые правила вы упустите из виду, другие сформулируете излишне детально. Гораздо лучше выбрать какой-то набор метаданных о письмах, задать пары (тело письма, имя папки) и поручить алгоритму вывести наилучший набор правил. Такие пары называются обучающими данными, а получившийся набор правил можно будет применить к будущим письмам, которых алгоритм еще не видел. Это и есть машинное обучение в простейшем виде.

Разумеется, машинное обучение (его еще часто называют добычей данных или прогностическим анализом) – не новая дисциплина. Наоборот, своими успехами в последние годы она обязана практическому применению проверенных временем методов и идей из других областей знания, в частности математической статистики. Цель человека – извлечь полезную информацию из данных, например, выявить скрытые закономерности и взаимосвязи. Читая об успешных применениях машинного обучения (Вы ведь уже открыли для себя сайт [www.kaggle.com](http://www.kaggle.com), правда?), вы убедитесь, что прикладная статистика всюду используется специалистами.

Ниже вы увидите, что процесс поиска подходящего подхода к машинному обучению вовсе не линеен. Напротив, приходится многократно возвращаться назад, пробуя другие сочетания исходных данных и алгоритмов. Именно изыскательская природа этого процесса делает применение Python чрезвычайно уместным. Ведь Python, будучи интерпретируемым высокоуровневым языком программирования, как будто специально придуман для опробования разных вариантов. К тому же, он работает быстро. Конечно, он медленнее C и подобных ему статически типизированных языков. Но при наличии огромного числа простых в использовании библиотек, зачастую написанных на C, вам не придется жертвовать скоростью ради гибкости.

## Что вы узнаете (и чего не узнаете из этой книги)

В этой книге вы найдете общий обзор обучающих алгоритмов, которые чаще всего применяются в различных отраслях машинного обучения, и узнаете, на что обращать внимание при их применении. Но по своему опыту мы знаем, что такие «интересные» вещи, как использование и настройка отдельных алгоритмов, например метода опорных векторов, классификации по ближайшим соседям, или их ансамблей, – занимают лишь малую часть рабочего времени специалиста по машинному обучению. А основное время тратится на довольно скучную работу:

- чтение и очистка данных;
- изучение исходных данных и попытки понять их;
- размышления о том, как лучше подать данные на вход алгоритма обучения;
- выбор подходящей модели и алгоритма;
- правильное измерение качества работы алгоритма.

В процессе изучения и осмысления исходных данных нам понадобится статистика и не очень сложная математика. И, как вы убедитесь, методы, казавшиеся такими скучными на занятиях по математике, могут стать по-настоящему увлекательными, когда применяются для анализа интересных данных.

Наше путешествие начинается с чтения данных. Вы поймете, что поиск ответа на вопрос, как быть с некорректными или отсутствующими данными, – скорее искусство, чем точная наука. Правильное решение воздастся сторицей, потому что позволит применить к данным больше обучающих алгоритмов, а, значит, повысит шансы на успех.

Когда данные будут представлены в виде структур данных в программе, вы захотите понять, с чем же все-таки работаете. Достаточно ли данных для ответа на интересующие вас вопросы? Если нет, то как добыть дополнительные данные? А, быть может, данных слишком много? Тогда нужно подумать, как лучше всего произвести выборку из них.

Часто бывает, что данные не стоит подавать сразу на вход алгоритма машинного обучения, а надо предварительно улучшить их. Алгоритм с благодарностью ответит на это более качественными результатами. Иногда даже оказывается, что простой алгоритм на предварительно обработанных данных работает лучше, чем очень изощренный алгоритм на данных в первоизданном виде. Эта часть

работы, называемая **подготовкой признаков** (feature engineering), чаще всего оказывается безумно увлекательной задачей, приносящей немедленные плоды. Вы сразу же видите результаты творческого нестандартного подхода.

Таким образом, чтобы выбрать подходящий обучающий алгоритм, недостаточно просто ткнуть наугад в один из трех-четырёх имеющихся в вашем арсенале (как вы скоро увидите, их вообще-то больше). Это вдумчивый процесс взвешивания различных критериев – от качества работы до функциональных требований. Вы хотите получить результат быстро, даже в ущерб качеству? Или предпочитаете потратить больше времени, но получить наилучший возможный результат? У вас есть отчетливое представление о будущих данных или лучше бы не делать слишком ограничительных предположений на этот счет?

Наконец, измерение качества работы – то место, где начинающий изучать машинное обучение может наделать больше всего ошибок. Есть ошибки простые, например, контроль результатов на тех же данных, на которых производилось обучение. А есть и посложнее, например, подача на вход несбалансированного обучающего набора. Как и раньше, именно данные определяют успех или неудачу всего предприятия.

Как мы видим, лишь четвертый пункт в нашем списке относится собственно к возне с алгоритмами. И тем не менее, мы надеемся убедить вас в том, что и остальные четыре задачи могут быть не менее увлекательными. Нам хотелось бы, чтобы, прочитав книгу до конца, вы влюбились не столько в алгоритмы обучения, сколько в сами данные.

Поэтому мы не станем нагружать вас теоретическими вопросами разнообразных алгоритмов машинного обучения, поскольку на эту тему существует немало отличных книг (их перечень вы найдете в приложении), а вместо этого попытаемся развить интуицию настолько, чтобы вы поняли идею и смогли сделать первые шаги. Так что эту книгу ни в коем случае нельзя считать *авторитетным руководством* по машинному обучению. Это скорее учебник для начинающих. Мы надеемся возбудить в вас любопытство настолько, чтобы вам захотелось продолжить изучение этой интереснейшей области знаний.

Далее в этой главе мы познакомимся с основами библиотек NumPy и SciPy для Python и обучим наш первый алгоритм с помощью библиотеки scikit-learn. По ходу дела мы введем основные понятия машинного обучения, которые будут использоваться на протяжении всей книги. В последующих главах мы подробно рассмотрим все

пять вышеперечисленных шагов, демонстрируя различные аспекты машинного обучения на примерах из разных областей.

## Что делать, если вы застряли

Мы старались как можно понятнее изложить все идеи, необходимые, чтобы воспроизвести описанные в книге примеры. Тем не менее, не исключено, что вы зайдете в тупик. Причины могут быть различными: опечатки, непредвиденные комбинации версий пакетов или недопонимание.

В таком случае обратиться за помощью можно в разные места. Скорее всего, кто-то уже сталкивался с вашей проблемой, и решение имеется на одном из следующих сайтов вопросов и ответов.

- <http://metaoptimize.com/qa>: этот сайт посвящен исключительно машинному обучению. Почти на каждый вопрос имеются квалифицированные ответы от специалистов. Даже если у вас нет никаких вопросов, полезно взять за правило время от времени заходить сюда и читать ответы.
- <http://stats.stackexchange.com>: этот сайт носит название Cross Validated, он похож на MetaOptimize, но посвящен больше вопросам из области статистики.
- <http://stackoverflow.com>: этот сайт во многом напоминает предыдущий, но диапазон рассматриваемых вопросов программирования гораздо шире. Так, на нем задается больше вопросов о некоторых используемых в этой книге пакетах, в частности, SciPy и matplotlib.
- Канал #machinelearning на сайте <https://freenode.net/>: это IRC-канал, посвященный проблемам машинного обучения. Здесь собирается небольшое, но очень активное и всегда готовое оказать помощь сообщество специалистов по машинному обучению.
- <http://www.TwoToReal.com>: это сайт мгновенных вопросов и ответов, созданный авторами книги для оказания помощи по вопросам, не попадающим ни в одну из описанных выше категорий. Если вы зададите здесь вопрос, то один из авторов немедленно получит сообщение, если находится в сети, и будет готов вступить с вами в беседу.

Как уже было сказано в начале книги, наша цель – помочь вам быстро освоить азы машинного обучения и дальше двигаться уже самостоятельно. Поэтому мы горячо рекомендуем составить свой список

блогов, относящихся к машинному обучению, и регулярно заглядывать в них. Это лучший способ узнать, что работает, а что – нет.

Единственный блог, о котором мы хотели бы упомянуть прямо сейчас (и подробнее рассказать в приложении) – <http://blog.kaggle.com>, блог компании Kaggle, которая проводит конкурсы по машинному обучению. Обычно победителям предлагается рассказать о своих подходах к конкурсным задачам, о том, какие стратегии не сработали, и как они пришли к победной стратегии. Этот блог обязателен к прочтению, пусть даже он окажется единственным.

## Приступая к работе

В предположении, что Python уже установлен (годится любая версия, начиная с 2.7), нам нужно еще установить пакеты NumPy и SciPy для численных расчетов и matplotlib для визуализации.

### ***Введение в NumPy, SciPy и matplotlib***

Прежде чем говорить о конкретных алгоритмах машинного обучения, следует определиться с тем, как лучше хранить данные, которые мы будем обрабатывать. Это важно, потому что даже самый изощренный обучающий алгоритм ничем не поможет, если он не завершается. А так может случиться просто потому, что доступ к данным слишком медленный. Или потому, что выбранное представление данных заставляет операционную систему постоянно пробуксовывать. Добавьте сюда тот факт, что Python все-таки интерпретируемый язык (хотя и хорошо оптимизированный), слишком медленный для численных расчетов по сравнению с C или FORTRAN. Но тогда возникает вопрос – почему же так много ученых и компаний ставят свое благополучие на Python, даже в задачах, требующих очень интенсивных вычислений?

Дело в том, что, работая на Python, очень просто перепоручить численные расчеты низкоуровневым расширениям, реализованным на C или FORTRAN. Именно так устроены библиотеки NumPy и SciPy (<http://scipy.org/Download>). В этом тандеме NumPy отвечает за высоко оптимизированные многомерные массивы – основную структуру данных для большинства современных алгоритмов. А SciPy на базе этих массивов реализует быстрые численные алгоритмы. Наконец, matplotlib (<http://matplotlib.org/>) – пожалуй, самая удобная и функционально развитая библиотека для построения высококачественных графиков на Python.

## **Установка Python**

По счастью, для всех популярных операционных систем, то есть Windows, Mac и Linux, существуют готовые инсталляторы NumPy, SciPy и matplotlib. Если вы не уверены, сможете ли справиться с их установкой, то установите дистрибутив Anaconda Python (его можно скачать с сайта <https://store.continuum.io/cshop/anaconda/>), созданный Трэвисом Олифантом (Travis Oliphant), основателем и активным автором проекта SciPy. От других дистрибутивов, например Enthought Canopy (<https://www.enthought.com/downloads/>) или Python(x,y) (<http://code.google.com/p/pythonxy/wiki/Downloads>), Anaconda отличается полной совместимостью с Python 3 – версией Python, которой мы будем пользоваться в этой книге.

## **NumPy как средство эффективной и SciPy как средство интеллектуальной обработки данных**

Давайте разберем несколько простых примеров применения NumPy, а затем посмотрим, что SciPy дает сверх того. Попутно мы сделаем первую попытку построить графики с помощью чудесного пакета Matplotlib.

Желающие углубиться в тему могут ознакомиться с интересными примерами применения NumPy, приведенными в пособии по адресу [http://www.scipy.org/Tentative\\_NumPy\\_Tutorial](http://www.scipy.org/Tentative_NumPy_Tutorial).

Весьма полезна также книга Ivan Idris «NumPy Beginner's Guide» (второе издание), вышедшая в издательстве Packt Publishing. Дополнительные учебные материалы можно найти на сайте <http://scipy-lectures.github.com>, а также в официальном руководстве по SciPy по адресу <http://docs.scipy.org/doc/scipy/reference/tutorial>.



---

В этой книге мы пользуемся версиями NumPy 1.8.1 и SciPy 0.14.0.

---

## **Изучаем NumPy**

Итак, импортируем NumPy и немного поэкспериментируем. Для этого нужно запустить интерактивную оболочку Python:

```
>>> import numpy
>>> numpy.version.full_version
1.8.1
```

Поскольку мы не хотим загрязнять пространство имен, то, конечно же, не станем писать такой код:

```
>>> from numpy import *
```

Так как, к примеру, пакет `numpy.array` мог бы замаскировать пакет `array` из стандартной библиотеки Python. Вместо этого мы будем пользоваться следующим удобным сокращением:

```
>>> import numpy as np
>>> a = np.array([0,1,2,3,4,5])
>>> a
array([0, 1, 2, 3, 4, 5])
>>> a.ndim
1
>>> a.shape
(6,)
```

Только что мы создали массив – точно так же, как список в Python. Однако в массивах NumPy хранится дополнительная информация о форме. В данном случае мы имеем одномерный массив с шестью элементами. Пока никаких сюрпризов.

Теперь можно преобразовать этот массив в двумерную матрицу:

```
>>> b = a.reshape((3,2))
>>> b
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> b.ndim
2
>>> b.shape
(3, 2)
```

Интересные вещи начинаются с осознанием того, как сильно оптимизирован пакет NumPy. Например, в следующих операциях копирование по возможности не производится:

```
>>> b[1][0] = 77
>>> b
array([[ 0,  1],
       [77,  3],
       [ 4,  5]])
>>> a
array([ 0,  1, 77,  3,  4,  5])
```

В данном случае мы изменили один элемент `b` с `2` на `77` и, как видим, это изменение отразилось и на массиве `a`. Но если копирование таки необходимо, то это можно организовать:

```
>>> c = a.reshape((3,2)).copy()
>>> c
array([[ 0,  1],
       [77,  3],
       [ 4,  5]])
>>> c[0][0] = -99
>>> a
array([ 0,  1, 77,  3,  4,  5])
>>> c
array([[ -99,  1],
       [ 77,  3],
       [  4,  5]])
```

Теперь `c` и `a` – совершенно независимые копии.

Еще одно преимущество массивов NumPy – распространение операций на отдельные элементы. Например, умножение массива NumPy на число порождает массив такого же размера, в котором все элементы умножены на это число:

```
>>> d = np.array([1,2,3,4,5])
>>> d*2
array([ 2,  4,  6,  8, 10])
```

То же справедливо и для других операций:

```
>>> d**2
array([ 1,  4,  9, 16, 25])
```

Сравните с обычным списком Python:

```
>>> [1,2,3,4,5]*2
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
>>> [1,2,3,4,5]**2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
```

Разумеется, прибегая к массивам NumPy, мы приносим в жертву гибкость списков Python. Такие простые операции, как добавление или удаление элементов, для массивов NumPy реализуются сложнее. Но, к счастью, в нашем распоряжении есть и то, и другое, так что можно выбрать подходящий инструмент по ситуации.

## Индексирование

Своей эффективностью библиотека NumPy отчасти обязана разнообразием способов доступа к массивам. Помимо знакомого по спискам индексирования, можно использовать в качестве индексов сами массивы:

```
>>> a[np.array([2,3,4])]
array([77, 3, 4])
```

Учитывая, что логические условия также распространяются на отдельные элементы, мы получаем очень удобный способ доступа к данным:

```
>>> a>4
array([False, False, True, False, False, True], dtype=bool)
>>> a[a>4]
array([77, 5])
```

Следующая команда позволяет ограничить выбросы:

```
>>> a[a>4] = 4
>>> a
array([0, 1, 4, 3, 4, 4])
```

Поскольку эта бывает необходимо очень часто, существует специальная функция `clip`, которая позволяет задать обе границы с помощью одного вызова:

```
>>> a.clip(0,4)
array([0, 1, 4, 3, 4, 4])
```

## Обработка отсутствующих значений

Средства индексирования NumPy оказываются особенно полезны, когда нужно произвести предварительную обработку данных, прочитанных из текстового файла. Чаще всего данные содержат некорректные значения, которые мы можем пометить как не-числа с помощью константы `numpy.NaN`:

```
>>> c = np.array([1, 2, np.NaN, 3, 4]) # допустим, что мы прочли
                                     # это из текстового файла
>>> c
array([ 1.,  2., nan,  3.,  4.])
>>> np.isnan(c)
array([False, False,  True, False, False], dtype=bool)
>>> c[~np.isnan(c)]
array([ 1.,  2.,  3.,  4.])
>>> np.mean(c[~np.isnan(c)])
2.5
```

## Сравнение времени работы

Давайте сравним производительность NumPy и обычных списков Python. Следующая программа вычисляет сумму квадратов чисел от 1 до 1000 и замеряет время работы. Для большей точности мы прогоним ее 10 000 раз и напечатаем общее время.

```
import timeit
normal_py_sec = timeit.timeit('sum(x*x for x in range(1000))',
                              number=10000)
naive_np_sec = timeit.timeit(
    'sum(na*na)',
    setup="import numpy as np; na=np.arange(1000)",
    number=10000)
good_np_sec = timeit.timeit(
    'na.dot(na)',
    setup="import numpy as np; na=np.arange(1000)",
    number=10000)

print("Normal Python: %f sec" % normal_py_sec)
print("Naive NumPy: %f sec" % naive_np_sec)
print("Good NumPy: %f sec" % good_np_sec)

Normal Python: 1.050749 sec
Naive NumPy: 3.962259 sec
Good NumPy: 0.040481 sec
```

Сделаем два любопытных наблюдения. Во-первых, если использовать NumPy только для хранения данных (Naive NumPy), то программа работает в 3,5 раза дольше. Это странно, потому что мы ожидали, что написанное на C расширение будет работать гораздо быстрее. Одна из причин заключается в том, что доступ к отдельным элементам массива из Python обходится довольно дорого. Ускорение мы получим лишь в том случае, если сумеем применить алгоритм, не выходя за пределы оптимизированного кода расширения. Другое наблюдение поражает: использование входящей в NumPy функции `dot()`, которая делает в точности то же самое, приводит к 25-кратному ускорению. Вывод – реализуя любой алгоритм, нужно стремиться к тому, чтобы вместо цикла по отдельным элементам массива на Python воспользоваться какой-нибудь оптимизированной функцией, входящей в состав NumPy или SciPy.

Однако за быстродействие приходится расплачиваться. Работая с массивами NumPy, мы утрачиваем невероятную гибкость списков Python, в которых можно хранить практически всё. Все элементы массива NumPy должны иметь одинаковый тип.

```
>>> a = np.array([1,2,3])
>>> a.dtype
dtype('int64')
```

Если мы попытаемся использовать элементы разных типов, как в следующем примере, то NumPy постарается привести их к одному и тому же, наиболее разумному в данных обстоятельствах, типу:

```
>>> np.array([1, "stringy"])
array(['1', 'stringy'], dtype='<U7')
>>> np.array([1, "stringy", set([1,2,3])])
array([1, stringy, {1, 2, 3}], dtype=object)
```

## Изучаем SciPy

Поверх эффективных структур данных NumPy библиотека SciPy надстраивает многочисленные алгоритмы, работающие с массивами. Какой бы численный алгоритм из описываемых в современных учебниках ни взять, с большой вероятностью он будет так или иначе поддержан в SciPy. Это относится к операциям над матрицами, линейной алгебре, оптимизации, кластеризации, пространственным операциям и даже быстрому преобразованию Фурье. Поэтому прежде чем приступить к самостоятельной реализации численного алгоритма, обязательно посмотрите, нет ли его в модуле `scipy`.

Для удобства полное пространство имен NumPy доступно также через SciPy. Поэтому, начиная с этого места, мы будем обращаться ко всем средствам NumPy с помощью пространства имен SciPy. В корректности этого подхода легко убедиться, сравнив ссылки на любую функцию:

```
>>> import scipy, numpy
>>> scipy.version.full_version
0.14.0
>>> scipy.dot is numpy.dot
True
```

Алгоритмы разбиты на следующие группы.

SciPy packages	Functionalities
<code>cluster</code>	<ul style="list-style-type: none"> <li>Иерархическая кластеризация (<code>cluster.hierarchy</code>)</li> <li>Векторное квантование / метод <math>k</math>-средних (<code>cluster.vq</code>)</li> </ul>
<code>constants</code>	<ul style="list-style-type: none"> <li>Физические и математические константы</li> <li>Методы преобразования</li> </ul>
<code>fftpack</code>	Алгоритм дискретного преобразования Фурье
<code>integrate</code>	Процедуры интегрирования
<code>interpolate</code>	Интерполяция (линейная, кубическая и т. п.)
<code>io</code>	Ввод-вывод данных
<code>linalg</code>	Процедуры линейной алгебры на основе оптимизированных библиотек BLAS и LAPACK
<code>ndimage</code>	Обработка многомерных изображений

SciPy packages	Functionalities
odr	Ортогональная регрессия
optimize	Оптимизация (нахождение минимумов и корней)
signal	Обработка сигналов
sparse	Разреженные матрицы
spatial	Пространственные структуры данных и алгоритмы
special	Специальные математические функции, например, функция Бесселя и якобиан
stats	Математическая статистика

Для нас наибольший интерес представляют пакеты `scipy.stats`, `scipy.interpolate`, `scipy.cluster` и `scipy.signal`. Сейчас мы вкратце рассмотрим возможности пакета `stats`, а изучение остальных отложим до тех глав, где они понадобятся.

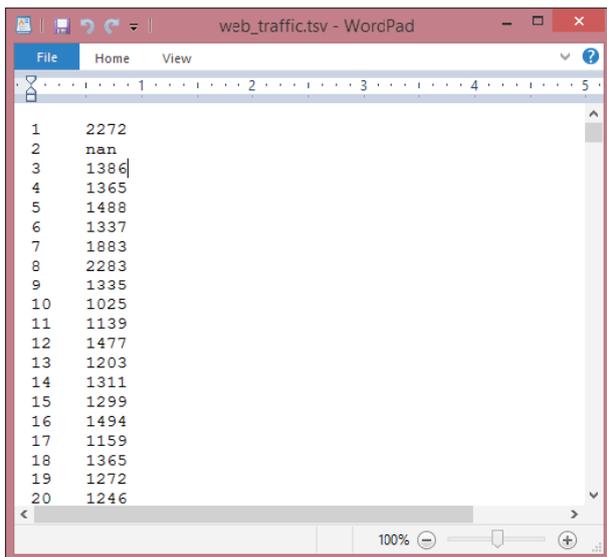
## Наше первое (простенькое) приложение машинного обучения

Хватит слов, возьмем для примера гипотетическую недавно образованную компанию MLaaS, которая предоставляет платные услуги машинного обучения через Интернет. Наша компания растет, и ей понадобилось улучшить инфраструктуру для обслуживания поступающих запросов. Мы не хотим выделять слишком много ресурсов, потому что это будет дорого стоить. С другой стороны, если не резервировать достаточно ресурсов, то мы потеряем деньги из-за невозможности обслужить все запросы. Вопрос заключается в том, когда мы достигнем предельной пропускной способности инфраструктуры, которую оценили в 100 000 запросов в час. Хотелось бы заранее знать, когда запросить дополнительные серверы в облаке, чтобы обслужить все входящие запросы, но не платить за неиспользованные ресурсы.

### Чтение данных

Мы собрали и агрегировали статистику веб за последний месяц, эти данные находятся в файле `ch01/data/web_traffic.tsv` (расширение `tsv` означает, что значения разделены знаками табуляции). Данные представляют собой количество запросов в час. В каждой строке указан час (по порядку) и количество запросов за этот час.

Ниже показаны первые несколько строк:



1	2272
2	nan
3	1386
4	1365
5	1488
6	1337
7	1883
8	2283
9	1335
10	1025
11	1139
12	1477
13	1203
14	1311
15	1299
16	1494
17	1159
18	1365
19	1272
20	1246

Метод `genfromtxt()` из библиотеки `SciPy` позволяет легко прочитать эти данные:

```
>>> import scipy as sp
>>> data = sp.genfromtxt("web_traffic.tsv", delimiter="\t")
```

Чтобы столбцы правильно распознавались, необходимо указать, что разделителем служит знак табуляции.

Простая проверка показывает, что данные прочитаны верно:

```
>>> print(data[:10])
[[ 1.00000000e+00  2.27200000e+03]
 [ 2.00000000e+00  nan]
 [ 3.00000000e+00  1.38600000e+03]
 [ 4.00000000e+00  1.36500000e+03]
 [ 5.00000000e+00  1.48800000e+03]
 [ 6.00000000e+00  1.33700000e+03]
 [ 7.00000000e+00  1.88300000e+03]
 [ 8.00000000e+00  2.28300000e+03]
 [ 9.00000000e+00  1.33500000e+03]
 [ 1.00000000e+01  1.02500000e+03]]
>>> print(data.shape)
(743, 2)
```

Как видим, создан двумерный массив, содержащий 743 результата измерений.