O'REILLY®

Data Science
Наука о данных с нуля

bhv®

Джоэл Грас

Data Science from Scratch

Joel Grus

Beijing • Cambridge • Farnham • Sebastopol • Tokyo

O'REILLY®

Джоэл Грас

Data Science
Наука о данных с нуля

Санкт-Петербург
«БХВ-Петербург»
2017

УДК 004.6
ББК 32.81
Г77

Грас Дж.

Г77 Data Science. Наука о данных с нуля: Пер. с англ. — СПб.: БХВ-Петербург, 2017. — 336 с.: ил.

ISBN 978-5-9775-3758-2

Книга позволяет изучить науку о данных (Data Science) и применить полученные знания на практике. Она написана так, что способствует погружению в Data Science аналитика, фактически не обладающего глубокими знаниями в этой прикладной дисциплине.

В объемах, достаточных для начала работы в области Data Science, книга содержит интенсивный курс языка Python, элементы линейной алгебры, математической статистики, теории вероятностей, методов сбора, очистки, нормализации и обработки данных. Даны основы машинного обучения. Описаны различные математические модели и их реализация по методу k ближайших соседей, наивной байесовской классификации, линейной и логистической регрессии, а также модели на основе деревьев принятия решений, нейронных сетей и кластеризации. Рассказано о работе с рекомендательными системами, описаны приемы обработки естественного языка, методы анализа социальных сетей, основы баз данных, SQL и MapReduce.

Для аналитиков данных

УДК 004.6
ББК 32.81

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Перевод с английского	*Андрея Логунова*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 09.02.17.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 27,09.
Тираж 1000 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-1-491-90142-7 (англ.)
ISBN 978-5-9775-3758-2 (рус.)

Оглавление

Предисловие

Наука о данных

Аналитиков данных (data scientists) называют "самой сексуальной профессией XXI века". Очевидно тот, кто так выразился, никогда не бывал в пожарной части. Тем не менее, наука о данных (data science) — это действительно передовая и быстроразвивающаяся отрасль знаний, а чтобы отыскать обозревателей рыночных тенденций, которые возбужденно предвещают, что через 10 лет нам потребуются на миллиарды и миллиарды больше аналитиков данных, чем мы имеем на текущий момент, не придется долго рыскать по Интернету.

Но что же это такое — наука о данных? В конце концов нельзя же выпускать специалистов в этой области, если не знаешь, что она собой представляет. Согласно диаграмме Венна[1], которая довольно известна в этой отрасли, наука о данных находится на пересечении:

- навыков алгоритмизации и программирования;
- знаний математики и статистики;
- профессионального опыта в предметной области.

Собираясь с самого начала написать книгу, охватывающую все три направления, я быстро пришел к выводу, что всестороннее обсуждение профессионального опыта в предметной области потребовало бы десятки тысяч страниц. Тогда я решил сосредоточиться на первых двух. Задача — помочь желающим развить свои навыки алгоритмизации и программирования, которые потребуются для того, чтобы приступить к решению задач в области науки о данных, а также почувствовать себя комфортно с математикой и статистикой, которые находятся в центре этой междисциплинарной практической сферы.

Довольно трудновыполнимая задача для книги. Развивать свои навыки алгоритмизации и программирования лучше всего решая прикладные задачи. Прочтя эту книгу, читатель получит хорошее представление о моих способах алгоритмизации прикладных задач, некоторых инструментах, которыми я пользуюсь в работе, и моем подходе к решению задач, связанных с анализом данных. Но это вовсе не означает, что мои способы, инструменты и подход являются единственно возможными. Надеюсь, что мой опыт вдохновит попробовать пойти собственным путем. Все ис-

---

[1] Джон Венн (1834–1923) — английский логик, предложивший схематичное изображение всех возможных пересечений нескольких (часто — трех) множеств (см. **https://ru.wikipedia.org/wiki/Венн,_Джон**). — *Прим. пер.*

ходные коды и данные из книги доступны на GitHub (**https://github.com/joelgrus/data-science-from-scratch**); они помогут приступить к работе.

То же самое касается и математики. Ею лучше всего заниматься, решая математические задачи. Данная книга, разумеется, не является руководством по математике, и мы не собираемся, по большей части, заниматься ею. Однако действительность такова, что заниматься анализом данных не получится без *некоторых* представлений о теории вероятностей, математической статистике и линейной алгебре. Это значит, что по мере надобности мы будем уходить с головой в математические равенства, интуицию, аксиомы и мультяшные версии глобальных математических идей. Надеюсь, читатель не побоится погрузиться в них вместе со мной.

По ходу изложения я также надеюсь передать ощущение, что импровизировать с данными — это увлекательное занятие, потому что, как бы это сказать, импровизировать с ними на самом деле увлекательно! В особенности, если сравнивать с такими альтернативами, как подготовка налоговой декларации или добыча угля.

С чистого листа

Для работы в области науки о данных разработана масса программных библиотек, платформ, модулей и инструментариев, которые эффективно реализуют наиболее (нередко, и наименее) общие алгоритмы и приемы, применяемые в науке о данных. Тот, кто станет аналитиком данных, несомненно, будет досконально знать библиотеку для научных вычислений NumPy, библиотеку для машинного обучения scikit-learn, библиотеку для анализа данных pandas и множество других. Они прекрасно подходят для решения задач, связанных с наукой о данных. Но они также способствуют тому, чтобы начать решать задачи в области науки о данных, фактически не понимая ее.

В этой книге мы начнем двигаться в сторону науки о данных, стартовав с нулевой отметки, а именно займемся разработкой инструментов и реализацией алгоритмов вручную с тем, чтобы лучше понять их. Я вложил немало своего умственного труда в создание ясных, хорошо задокументированных и читаемых реализаций алгоритмов и примеров. В большинстве случаев инструменты, которые мы станем конструировать, будут иметь не практический, а разъясняющий характер. Они хорошо работают с малыми, почти игрушечными, наборами данных, но не справляются с данными "веб-масштаба".

По ходу изложения я буду отсылать читателя к библиотекам, которые подойдут для применения этих методов на более крупных наборах данных. Но мы их не будем тут использовать.

По поводу того, какой язык программирования лучше всего подходит для обучения науке о данных, развернулась здоровая полемика. Многие настаивают на языке статистического программирования R (мы называем таких людей неправильными). Некоторые предлагают Java или Scala. Однако, по моему мнению, Python — идеальный вариант.

Он обладает несколькими особенностями, которые делают его особенно пригодным для изучения и решения задач в области науки о данных:

- он бесплатный;
- он относительно прост в написании кода (и в особенности в понимании);
- он располагает сотнями прикладных библиотек, предназначенных для работы в области науки о данных.

Не решусь назвать Python моим предпочтительным языком программирования. Есть другие языки, которые я нахожу более удобными, продуманными либо просто интересными для программирования. И все же практически всякий раз, когда я начинаю новый проект в области науки о данных, либо, когда мне нужно быстро набросать рабочий прототип, либо продемонстрировать концепции этой практической дисциплины ясным и легким для понимания способом, я всякий раз в итоге использую Python. И поэтому в этой книге используется Python.

Эта книга не предназначена для того, чтобы научить программировать на Python (хотя, я почти уверен, что, прочтя ее, этому можно немного научиться). Тем не менее, я проведу интенсивный курс программирования на Python (ему будет посвящена целая глава), где будут высвечены характерные черты, которые в данном случае приобретают особую важность. Однако если знания, как программировать на Python (или о программировании вообще), отсутствуют, то читателю остается самому дополнить эту книгу чем-то вроде руководства по Python для начинающих.

В последующих частях нашего введения в науку о данных будет принят такой же подход — углубляться в детали там, где они оказываются решающими или показательными. В других ситуациях на читателя возлагается задача домысливать детали самому или заглядывать в Википедию.

За годы работы в отрасли я подготовил некоторое количество специалистов в области науки о данных. Хотя не все из них стали меняющими мир супер-мега-рок-звездами в области анализа данных, тем не менее, я их всех выпустил более подготовленными специалистами, чем они были до этого. И я все больше убеждаюсь в том, что любой, у кого есть некоторые математические способности вкупе с определенным набором навыков в программировании, располагает всем необходимым для решения задач в области науки о данных. Все, чего она требует, — это лишь пытливый ум, готовность усердно трудиться и наличие данной книги. Отсюда и эта книга.

Условные обозначения, принятые в книге

В книге используются следующие типографические условные обозначения:

- *курсив* указывает на новые термины;
- моноширинный шрифт используется для листингов программ, а также внутри абзацев для ссылки на элементы программ, такие как переменные или имена функций, базы данных, типы данных, переменные окружающей среды, операторы и ключевые слова;

- **`жирный моноширинный шрифт`** показывает команды либо другой текст, который должен быть напечатан пользователем, а также ключевые слова в коде;
- *`моноширинный шрифт курсивом`* показывает текст, который должен быть заменен значениями пользователя либо значениями, определяемыми по контексту.

Данный элемент обозначает подсказку или совет.

Данный элемент обозначает общее замечание.

Данный элемент обозначает предупреждение или предостережение.

Использование примеров кода

Дополнительный материал (примеры кода, упражнения и пр.) доступен для скачивания по адресу **https://github.com/joelgrus/data-science-from-scratch**.

Эта книга предназначена для того, чтобы помочь вам решить ваши задачи. В целом, если код примеров предлагается вместе с книгой, то вы можете использовать его в своих программах и документации. Вам не нужно связываться с нами с просьбой о разрешении, если вы не воспроизводите значительную часть кода. Например, написание программы, которая использует несколько фрагментов кода из данной книги, официального разрешения не требует.

Адаптированный вариант примеров в виде электронного архива вы можете скачать по ссылке **ftp://ftp.bhv.ru/9785977537582.zip**. Эта ссылка доступна также со страницы книги на сайте **www.bhv.ru**.

После распаковки архива у вас получится несколько папок:

- папки code-python2-original и code-python3-original содержат исходные примеры автора соответственно для Python 2 и Python 3;
- папка code-python3-ru — адаптированные исходные примеры программ;
- папка "Записные книжки Jupyter" — соответственно записные книжки Jupyter и html-версии книжек;
- папка Прочее — файлы корпуса текстов SpamAssassin и whl-файл библиотеки BeautifulSoup, которая понадобится в *главе 10*, как образец whl-пакета, и которую впрочем можно скачать и установить самостоятельно *(см. разд. "Комментарии переводчика")*.

Благодарности

Прежде всего, хотел бы поблагодарить Майка Лукидеса (Mike Loukides) за то, что принял мое предложение по поводу этой книги, и за то, что настоял на том, чтобы я довел ее до разумного объема. Он мог бы легко сказать: "Кто этот человек, вообще, который шлет мне образцы глав, и как заставить его прекратить, наконец?" И я признателен, что он этого не сделал. Хотел бы поблагодарить моего редактора, Мари Богуро (Marie Beaugureau), которая консультировала меня в течение всей процедуры публикации и привела книгу в гораздо более удобоваримый вид, чем если б этим я занимался сам.

Я бы не написал эту книгу, если бы не изучил науку о данных, и, возможно, я бы не научился ей, если бы не влияние Дэйва Хсу (Dave Hsu), Игоря Татаринова (Igor Tatarinov), Джона Раузера (John Rauser) и остальной группы единомышленников из Farecast (это было так давно, что в то время даже не было понятия науки о данных). Хорошие парни в Coursera также заслуживают много добрых слов.

Я так же благодарен моим бета-читателям и рецензентам. Джэй Фандлинг (Jay Fundling) обнаружил тонны ошибок и указал на многие нечеткие объяснения, и в итоге книга оказалась намного лучше и корректней, благодаря ему. Дэбаши Гош (Debashis Ghosh) геройски провела санитарную проверку моих статистических выкладок. Эндрю Массельман (Andrew Musselman) предложил смягчить первоначальный аспект книги, касающийся утверждения, что "люди, предпочитающие язык R вместо Python, есть моральные извращенцы", что, думаю, в итоге оказалось неплохим советом. Трэй Кози (Trey Causey), Райан Мэттью Балфанц (Ryan Matthew Balfanz), Лорис Муларони (Loris Mularoni), Нуриа Пуджол (Núria Pujol), Роб Джефферсон (Rob Jefferson), Мэри Пэт Кэмрбелл (Mary Pat Campbell), Зак Джири (Zach Geary) и Венди Грас (Wendy Grus) также высказали неоценимые замечания. За любые оставшиеся ошибки, конечно же, отвечаю только я.

Я многим обязан сообществу с хештегом **#datascience** в Twitter за то, что продемонстрировали мне массу новых концепций, познакомили меня со множеством замечательных людей и заставили почувствовать себя двоечником, да так, что я ушел и написал книгу в качестве противовеса. Особые благодарности снова Трэю Кози (Trey Causey) за то, что нечаянно упомянул, чтобы я включил главу про линейную алгебру, и Шин Дж. Тэйлор (Sean J. Taylor) за то, что неумышленно указала на пару больших несоответствий в главе "Работа с данными".

Но больше всего я задолжал огромную благодарность Ганге (Ganga) и Мэдлин (Madeline). Единственная вещь, которая труднее написания книги, — это жить рядом с тем, кто пишет книгу, и я бы не смог ее закончить без их поддержки.

Комментарий переводчика

Приведенные в книге примеры были протестированы в инструментальной среде для научных вычислений для языка Python — Spyder (Scientific PYthon Development EnviRonment) — версии 3.0.0 на основе Python 3.5.2 для Windows. Spyder — это простая, легковесная и, главное, бесплатная, интерактивная среда разработки на Python, которая предлагает функции, похожие на среду разработки в MATLAB. Она также предоставляет написанные полностью на Python и готовые к использованию виджеты PyQt4 и PySide, редактор исходного кода с подсветкой синтаксиса и функциями самоанализа/анализа кода, редактор массивов данных NumPy, редактор словарей, консоли Python и IPython и многое другое. Большинство графиков и диаграмм были воссозданы именно в этой инструментальной среде, хотя стоит признать, есть и другие, например PyCharm и Eclipse PyDev.

В связи с тем, что используемые в примерах имена переменных и функций являются логическим продолжением излагаемого материала, они переведены вместе с авторскими комментариями и дополнены кратким описанием. Кроме того, в силу различий в наименовании терминов как в отечественной, так и зарубежной литературе, для некоторых из них приведены соответствующие эквивалентные варианты наименований или пояснения. Некоторые термины, которые в тексте остались без объяснения, кратко определены в сносках.

В целом в книге принят сбалансированный подход к теории и программированию. Разумеется, можно найти более подробное изложение аспектов науки о данных и на основе других языков программирования (Clojure, Julia, Haskell, MATLAB, R или Scala). Однако если необходимо познакомиться с этой тематикой на основе языка Python, то, скорее всего, начинать следует именно с этой книги.

Книга может быть интересной широкому кругу специалистов, в том числе в области машинного обучения, начинающим аналитикам данных, преподавателям, студентам, а также всем, кто интересуется программированием.

Python 2 и Python 3

Сегодня используются две главные разновидности Python: Python версии 2.*x* существует уже в течение многих лет и все еще широко применяется, в то время как Python версии 3.*x*, которая не является обратно несовместимой с Python 2, становится все более популярной, т. к. эта версия Python взята за основу для дальнейшего развития.

Один из главных факторов, сдерживающих повсеместное принятие Python 3, — недостаточная поддержка сторонних библиотек, в частности связанных с разработкой

геоприложений или визуальных интерфейсов. Но мир не стоит на месте, а вместе с ним и Python 3, и все крупнейшие библиотеки, которые указаны в этой книге, теперь в равной степени можно выполнять, используя Python 3. Все примеры программного кода в этой книге преобразованы таким образом, чтобы использовать синтаксис Python 3. К счастью, различия в синтаксисе между Python 2 и Python 3 предельно простые и потребуют незначительных изменений. В простых примерах часто единственное отличие состоит в том, что в версии Python 3 после оператора `print` требуются круглые скобки; все остальные отличия задокументированы в сносках.

Установка и удаление дистрибутива Anaconda

Anaconda — это полностью свободный дистрибутив Python (предназначенный в том числе для коммерческого использования и повторного распространения). Он содержит более 400 самых популярных библиотек Python для вычислений в области естественных наук, математики, инженерии и анализа данных.

Установка дистрибутива в Windows выполняется стандартным образом после загрузки с сайта бинарного файла дистрибутива. Папка с установленными приложениями находится в разделе приложений меню **Пуск**. Откройте папку меню **Пуск**, выберите **Все приложения** и нажмите на папке **Anaconda3** (**Пуск | Все приложения | Anaconda3**). Папка содержит ряд приложений, таких как навигатор Anaconda Navigator, инструментальная среда Spyder и сервер записных книжек Jupyter. Исполнимые файлы находятся в папке C:\Users*[имя_пользователя]*\Anaconda3\.

В Linux сначала необходимо скачать установщик — скриптовый файл с расширением .sh для оболочки Bash (**https://www.continuum.io/downloads#_unix**). На примере дистрибутива Anaconda версии 3 (для Python 3.5.2) команда установки выглядит так:

```
bash Anaconda3-4.1.1-Linux-x86_64.sh
```

Отметим, что инсталляция вступит в силу после того, как вы закроете и заново откроете окно терминала.

Для обновления дистрибутива Anaconda следует набрать в терминале следующую команду:

```
conda update conda
```

Удаление дистрибутива Anaconda:

```
rm -rf ~/anaconda
```

Более подробная информация по деинсталляции Anaconda содержится по указанной выше ссылке.

Чтобы удостовериться, что дистрибутив Anaconda установлен успешно, воспользуйтесь следующей командой:

```
conda --version
```

В результате будет выведен номер установленной версии.

Настройка дистрибутива Anaconda

В случае если вы в основном работаете в среде Python 3, но иногда возникает необходимость переключиться в среду Python 2 и использовать ее для работы с библиотеками, которые предназначены только для Python 2, то Anaconda предлагает функционал создания и активации новой среды, куда можно установить нужную версию языка. В табл. 1 приведено несколько команд, которые можно выполнить прямо в Spyder во встроенном окне командной строки (**Инструменты | Открыть командную строку**).

Таблица 1

Команда	Описание
`conda create -n py27 python=2.7.9 anaconda`	Установить другую версию Python (2.7.9) в новую среду с именем py27 (имя может быть любым)
`conda info --envs`	Проверить, что среда с именем py27 установлена (команада выводит список всех сред, при этом активная среда выделяется знаком *)
`source activate py27` (Linux, OS X), `activate py27` (Windows)	Переключиться в среду py27 с другой версией Python (команда `activate` добавляет в начало строки путь к среде py27)
`deactivate`	Деактивировать текущую среду и вернуться к среде по умолчанию
`python --version`	Проверить, что среда использует нужную версию Python
`conda remove -n py27 --all`	Удалить среду py27 (после выполнения команды выполнить `conda clean --lock`)
`conda install --name py27 scypy`	Установить библиотеку scypy в среду py27
`conda remove --name py27 scypy`	Удалить библиотеку scypy в среде py27
`conda clean --lock`	Очистить блокировку, если произошел сбой при установке среды (в Windows иногда сперва требуется удалить conda в диспетчере задач)

Установка инструментальной среды Spyder

Для пользователей Windows хорошая новость заключается в том, что инструментальная среда программирования Spyder уже включена в состав дистрибутива Anaconda, и исполнимый файл находится в папке C:\Users*[имя_пользователя]*\Anaconda3\Scripts\spyder.exe.

Чтобы установить среду Spyder в Ubuntu Linux, используя официальный менеджер пакетов, нужна всего одна команда (тройка соответствует Python версии 3):

```
sudo apt-get install spyder
```

Чтобы установить с использованием менеджера пакетов pip:

```
sudo apt-get install python3-qt4 python3-sphinx
sudo pip3 install spyder
```

И чтобы обновить:

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

Для выполнения команд менеджера пакетов `pip` следует открыть консольное окно (**Инструменты | Открыть командную строку**) и в открывшемся в правой части интерфейса пользователя окне набрать нужную команду, например, `pip3 install jupyter`.

Настройка инструментальной среды Spyder

При инсталляции дистрибутива Anaconda3 по умолчанию базовым является интерпретатор Python 3 (в данном случае версии 3.5.2). В случае если нужно на время переключиться в режим работы в интерпретаторе Python версии 2 (в данном случае версии 2.9.7), существует два варианта: дополнительно инсталлировать дистрибутив Anaconda для Python 2 либо выполнить небольшую настройку инструментальной среды Spyder, по умолчанию работающей на основе Python 3.

Настройка среды Spyder с Python 3 для работы с Python 2

Для такой настройки нужно в основном меню выбрать **Инструменты** и затем **Параметры**. В открывшемся окне **Параметры** выбрать **Интерпретатор Python**. В разделе **Интерпретатор Python** с двумя переключателями установить переключатель **Использовать следующий интерпретатор Python** и затем нажать кнопку напротив текстового поля, чтобы выбрать путь к интерпретатору Python 2.7.9, который находится в папке C:\Users*[имя_пользователя]*\Anaconda3\envs\py27, где py27 — это имя среды, созданной вами для Python 2.7.9 *(см. разд. "Настройка дистрибутива Anaconda" ранее)*. Если открыть новую консоль (**Консоли | Открыть консоль Python**), то в строке приветствия будет указана нужная версия Python.

Теперь в этой консоли можно набирать команды и вставлять целые программы, однако запускать программы из рабочего окна редактора Spyder не получится. Для сценария, работающего с Python 2, требуется еще одна и последняя настройка. В основном меню нужно выбрать **Запуск**, затем **Настроить** и в разделе **Консоль** выбрать второй переключатель **Выполнить во внешнем системном терминале** (т. е. в новой специально выделенной консоли). Теперь этот конкретный спенарий будет выполняться в консоли с Python 2.

Закончив работу с интерпретатором Python 2.7.9, не забудьте вернуться к исходному интерпретатору Python 3.5.2. Для этого нужно в разделе **Интерпретатор Python** просто установить переключатель **По умолчанию** (тот же, что и для Spyder).

Напомним, что в Windows месторасположение базового интерпретатора следующее: C:\Users*[имя_пользователя]*\Anaconda3\python.exe. Разумеется, такой режим работы вносит некоторые неудобства, но по крайней мере он не такой громоздкий, как установка еще одной версии Anaconda 4.1.1, но уже с Python 2.7.9 в качестве базового интерпретатора.

Факультативно

Запуск сервера записных книжек Jupyter

В Windows и Ubuntu Linux локальный сервер записных книжек запускается из командной строки двумя способами. Первый: откройте окно командной оболочки (`cmd`) в Windows или окно терминала в Ubuntu Linux и просто наберите:

```
jupyter notebook
```

Второй вариант:

```
ipython notebook
```

Теперь интерактивная вычислительная среда IPython — это составная часть интегрированной среды Jupyter, и поэтому второй вариант запуска не рекомендован к применению и будет в будущем удален, к тому же в нем используется формат записной книжки предыдущей версии 3.0+ (текущая 5.0+).

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 обретка 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 pip3 (тройка соответствует Python версии 3). В отличие от обычной установки командой `pip3 install <имя_пакета>`, вместо имени пакета указывается путь к whl-файлу: `pip3 install <путь_к_whl_файлу>`. Например,

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

Либо откройте окно командной строки и при помощи команды `cd` перейдите в каталог, где размещен ваш whl-файл. В Anaconda по умолчанию подобные файлы нахо-

дятся в каталоге Scripts. Просто скопируйте туда ваш whl-файл. В этих случаях полный путь указывать не понадобится. Например,

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

При выборе пакета важно, чтобы разрядность устанавливаемой библиотеки и разрядность интерпретатора совпадали. Пользователи Windows могут брать whl-файлы с сайта **http://www.lfd.uci.edu/~gohlke/pythonlibs/**. Библиотеки там постоянно обновляются, и в архиве содержатся все, какие только могут понадобиться.

Подготовка среды Python 3 в ОС Ubuntu Linux

Таблица 2

Действие	**Команда**
Обновление ОС Ubuntu/Debian	`sudo apt-get update && sudo apt-get upgrade && sudo apt-get dist-upgrade && sudo apt-get autoremove`
Интерпретатор языка Python 3 и менеджер пакетов	`sudo apt-get install python3 python3-pip`
Основные научные библиотеки	`sudo apt-get install python3-numpy python3-matplotlib python3-scipy python3-pandas python3-simpy`
Записные книжки Jupyter	`sudo pip3 install jupyter`
OpenGL	`sudo apt-get install python3-opengl`
Разработка графического интерфейса пользователя	`sudo apt-get install python3-pyqt5 python3-pyqt5.qtopengl python3-pyqt5.qtquick`
Хранение данных	`sudo apt-get install python3-h5py`
Компьютерное зрение	`sudo apt-get install python3-skimage` `sudo apt-get install libatlas-dev libatlas3gf-base && sudo pip3 install scikit-learn`
Интегрированная среда программирования (IDE) Spyder3	`sudo apt-get install spyder3`
Автозавершение кода в IDE	`sudo pip3 install --user rope_py3k`

Управление пакетами .deb в Ubuntu Linux

Установка пакетов .deb в терминале (пакет должен находиться под корневым каталогом home)

```
sudo dpkg -i [пакет].deb
```

или

```
sudo dpkg --install [пакет].deb
```

Удаление пакетов .deb

```
sudo dpkg -r [пакет].deb
```

Удаление вместе с конфигурационными файлами:

```
sudo dpkg -P [пакет].deb
```

Об авторе

Джоэл Грас работает инженером-программистом в компании Google. До этого он занимался аналитической работой в нескольких стартапах. Он живет в Сиэтле, где регулярно посещает неформальные встречи специалистов в области науки о данных. Он редко пишет в свой блог **joelgrus.com** и всегда доступен в Twitter по хештегу **@joelgrus**.

ГЛАВА 1

Введение

— Данные! Где данные? — раздраженно восклицал он. — Когда под рукой нет глины, из чего лепить кирпичи?

Артур Конан Дойль[1]

Господство данных

Мы живем в мире, страдающем от переизбытка данных. Веб-сайты отслеживают любое нажатие любого пользователя. Смартфоны накапливают сведения о вашем местоположении и скорости в ежедневном и ежесекундном режиме. "Оцифрованные" селферы носят шагомеры на стероидах, которые не переставая записывают их сердечные ритмы, особенности движения, схемы питания и сна. Умные авто собирают сведения о манерах вождения своих владельцев, умные дома — об образе жизни своих обитателей, а умные маркетологи — о наших покупательских привычках. Сам Интернет представляет собой огромный граф знаний, который, среди всего прочего, содержит обширную гипертекстовую энциклопедию, специализированные базы данных о фильмах, музыке, спортивных результатах, игровых автоматах, мемах[2] и коктейлях... и слишком много статистических отчетов (причем некоторые почти соответствуют действительности!) от слишком большого числа государственных исполнительных органов, и все это для того, чтобы вы объяли необъятное.

В этих данных кроятся ответы на бесчисленные вопросы, которые никто даже не думает задавать. Эта книга научит вас, как их находить.

Что такое наука о данных?

Существует шутка, что аналитик данных — это тот, кто знает статистику лучше, чем специалист в области информатики, а информатику — лучше, чем специалист в области статистики. Не утверждаю, что это хорошая шутка, но на самом деле, некоторые аналитики данных действительно являются специалистами в области математической статистики, в то время как другие почти неотличимы от инженеров программного обеспечения. Некоторые являются экспертами в области машинного

---

[1] Реплика Шерлока Холмса из рассказа Артура Конан Дойля (1859–1930) "Медные буки". — *Прим. пер.*

[2] Мем (англ. *meme*) — единица культурной информации: любая идея, символ, манера или образ действия, осознанно или неосознанно передаваемые от человека к человеку посредством речи, письма, видео, ритуалов, жестов и т. д. (см. **https://ru.wikipedia.org/wiki/Мем**). — *Ред.*

обучения, в то время как другие не смогли бы машинно обучиться, чтобы найти выход из детского сада. Некоторые имеют ученые степени доктора наук с впечатляющей историей публикаций, в то время как другие никогда не читали академических статей (хотя, им должно быть стыдно). Короче говоря, в значительной мере неважно, как определять понятие науки о данных, потому что всегда можно найти практикующих аналитиков данных, для которых это определение будет всецело и абсолютно неверным[3].

Тем не менее, этот факт не остановит нас от попыток. Мы скажем, что аналитик данных — это тот, кто извлекает ценные наблюдения из запутанных данных. В наши дни мир переполнен людьми, которые пытаются превратить данные в ценные наблюдения.

Например, сайт знакомств OkCupid просит своих членов ответить на тысячи вопросов, чтобы отыскать наиболее подходящего для них партнера. Но он также анализирует эти результаты, чтобы вычислить виды безобидных вопросов, с которыми вы можете обратиться, чтобы узнать, насколько высока вероятность близости после первого же свидания.

Компания Facebook просит вас указывать свой родной город и нынешнее местоположение, якобы чтобы облегчить вашим друзьям находить вас и связываться с вами. Но она также анализирует эти местоположения, чтобы определить схемы глобальной миграции и места проживания фанатов различных футбольных команд.

Крупный оператор розничной торговли Target отслеживает покупки и взаимодействия онлайн и в магазине. Он использует данные, чтобы строить прогнозные модели в отношении того, какие клиентки беременны, чтобы лучше продавать им товары, предназначенные для младенцев.

В 2012 г. избирательный штаб Барака Обамы нанял десятки аналитиков данных, которые вовсю копали и экспериментировали, чтобы определить избирателей, которым требовалось дополнительное внимание, при этом подбирая оптимальные обращения и программы по привлечению финансовых ресурсов, которые направлялись в адрес конкретных получателей, и сосредотачивая усилия по выводу соперника из предвыборной гонки там, где эти усилия могли быть наиболее успешными. Существует общее мнение, что эти усилия сыграли важную роль в переизбрании президента, вследствие чего совершенно очевидно, что будущие политические кампании будут все более и более управляемыми данными, ведя к бесконечному наращиванию усилий в области науки о данных и методов сбора данных.

И прежде чем вы почувствуете пресыщение, скажем еще пару слов: некоторые аналитики данных время от времени используют свои навыки во благо, чтобы сделать

---

[3] *Наука о данных* — это практическая дисциплина, которая занимается изучением методов обобщаемого извлечения знаний из данных. Она состоит из различных составляющих и основывается на методах и теориях из многих областей знаний, включая обработку сигналов, математику, вероятностные модели, машинное и статистическое обучение, программирование, технологии данных, распознавание образов, теорию обучения, визуальный анализ, моделирование неопределенности, организацию хранилищ данных, а также высокоэффективные вычисления с целью извлечения смысла из данных и создания продуктов обработки данных. — *Прим. пер.*

правительство более эффективным, помочь бездомным и усовершенствовать здравоохранение. И конечно же вы не нанесете вреда своей карьере, если вам нравится заниматься поисками наилучшего способа, как заставить людей щелкать на рекламных баннерах.

Оправдание для выдумки: DataSciencester

Поздравляем! Вас только что приняли на работу, чтобы вы возглавили усилия в области науки о данных в DataSciencester, *уникальной* социальной сети для аналитиков данных.

Предназначенная исключительно для аналитиков данных, тем не менее, DataSciencester еще ни разу не вкладывалась в развитие собственной практической деятельности в этой области (справедливости ради, она ни разу по-настоящему не вкладывалась даже в развитие своего продукта). Теперь это будет вашей работой! На протяжении всей книги мы будем изучать понятия этой практической дисциплины путем решения задач, с которыми вы будете сталкиваться на работе. Мы будем обращаться к данным, иногда поступающим прямо от пользователей, иногда полученным в результате их взаимодействий с сайтом социальной сети, а иногда даже взятыми из экспериментов, которые будем планировать сами.

И поскольку в DataSciencester царит дух новизны, то мы займемся конструированием собственных инструментов с чистого листа. В результате у вас будет достаточно твердое понимание основ науки о данных, и вы будете готовы применить свои навыки в любой компании или же в любых других задачах, которые окажутся для вас интересными.

Добро пожаловать на борт и удачи! (Вам разрешено носить джинсы по пятницам, и туалет — по коридору направо.)

Поиск ключевых звеньев

Итак, настал ваш первый рабочий день в DataSciencester, и директор по развитию сети полон вопросов о ее пользователях. До сего дня ему не к кому было обратиться, поэтому он очень воодушевлен вашим прибытием.

В частности, он хочет, чтобы вы установили, кто среди специалистов является "ключевым звеном". Для этих целей он передает вам "снимок" всей социальной сети. (В реальной жизни обычно никто не торопится вручать вам требующиеся данные. *Глава 9* посвящена способам сбора данных.)

Как выглядит этот "снимок" данных? Он состоит из списка пользователей `users`, где каждый пользователь представлен ассоциативным списком, или словарем, `dict`, состоящим из идентификатора `id` (т. е. числа) пользователя и его или ее имени `name` (которое по одному из необычайных космических совпадений рифмуется с идентификатором `id`):

```
users = [
    { "id": 0, "name": "Hero" },
    { "id": 1, "name": "Dunn" },
```

```
    { "id": 2, "name": "Sue" },
    { "id": 3, "name": "Chi" },
    { "id": 4, "name": "Thor" },
    { "id": 5, "name": "Clive" },
    { "id": 6, "name": "Hicks" },
    { "id": 7, "name": "Devin" },
    { "id": 8, "name": "Kate" },
    { "id": 9, "name": "Klein" }
]
```

Кроме того, он передает вам данные о "дружеских отношениях" `friendships` в виде списка кортежей, состоящих из пар идентификаторов ID пользователей:

```
friendships = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 4),
               (4, 5), (5, 6), (5, 7), (6, 8), (7, 8), (8, 9)]
```

Например, кортеж `(0, 1)` означает, что аналитик с `id 0` (Hero) и аналитик с `id 1` (Dunn) являются друзьями. Сеть представлена на рис. 1.1.

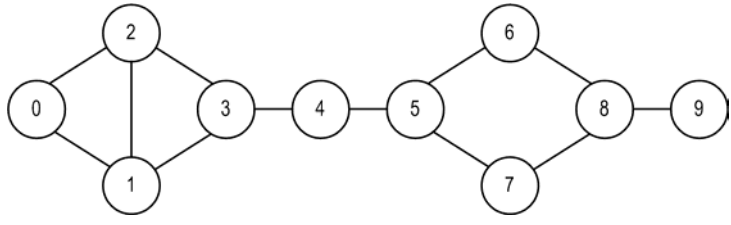

Рис. 1.1. Сеть DataSciencester

Поскольку пользователи представлены словарями `dict`, то пополнение информации о них становится простым делом.

Не следует прямо сейчас погружаться в подробности реализации кода. В *главе 2* будет предложен интенсивный курс языка Python. Пока же следует всего лишь попытаться получить общее представление о том, что мы делаем.

Например, может понадобиться добавить каждому пользователю список друзей. Для этого сначала назначим новому свойству `friends` каждого пользователя пустой список:

```
# свойство friends содержит друзей для пользователя user
for user in users:
    user["friends"] = []
```

А затем заполним эти списки данными из списка кортежей `friendships`:

```
for i, j in friendships:
    # это работает, потому что users[i] - это пользователь,
    # чей id равен i
    users[i]["friends"].append(users[j])  # добавить j как друга для i
    users[j]["friends"].append(users[i])  # добавить i как друга для j
```

После того, как в словаре `dict` каждого пользователя размещен список его друзей, получившийся граф можно легко опросить, например, по поводу среднего числа связей.

Сперва находим суммарное число связей, сложив длины всех списков друзей `friends`:

```
# число друзей
def number_of_friends(user):
    """сколько друзей есть у пользователя user?"""
    return len(user["friends"])    # длина списка id друзей

  total_connections = sum(number_of_friends(user)  # общее число связей
                          for user in users)       # 24
```

А затем просто делим сумму на число пользователей:

```
from __future__ import division # в Python 2 целочисленное деление хромает
num_users = len(users)                    # длина списка пользователей
avg_connections = total_connections / num_users  # среднее число связей =
                                                 # 2.4
```

Также легко находим наиболее связанных людей, т. е. лиц, имеющих наибольшее число друзей.

Поскольку пользователей не слишком много, их можно упорядочить по убыванию числа друзей:

```
# число друзей для каждого id пользователя
# создать список в формате (id пользователя, число друзей)
num_friends_by_id = [(user["id"], number_of_friends(user))
                     for user in users]

sorted(num_friends_by_id,                  # упорядочить его по полю
       key=lambda (user_id, num_friends): num_friends, # num_friends
       reverse=True)                       # в убывающем порядке

# каждая пара состоит из id и числа друзей (user_id, num_friends)
# [(1, 3), (2, 3), (3, 3), (5, 3), (8, 3),
#  (0, 2), (4, 2), (6, 2), (7, 2), (9, 1)]
```

Всю проделанную работу можно рассматривать, как способ определить лиц, которые так или иначе занимают в сети центральное место. На самом деле, то, что мы вычислили, представляет собой метрику связи в социальном графе под названием *центральность по степени* узлов (degree centrality) (рис. 1.2).

Достоинство этой метрики заключается в простоте вычислений, однако она не всегда дает нужные или ожидаемые результаты. Например, в DataSciencester пользователь Thor (`id 4`) имеет всего две связи, тогда как Dunn (`id 1`) — три. Несмотря на это, схема сети показывает, что Thor находится ближе к центру. В *главе 21* мы займемся

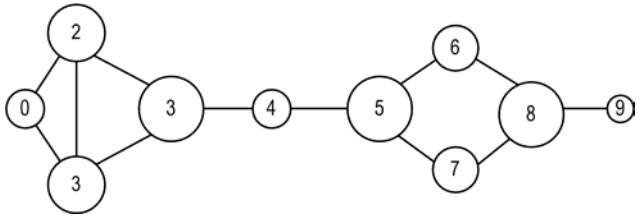

Рис. 1.2. Сеть DataSciencester, измеренная метрикой степени узлов

изучением социальных сетей более подробно и рассмотрим более сложные представления о центральности, которые могут лучше или хуже соответствовать интуиции.

Аналитики, которых вы должны знать

Вы еще не успели заполнить все страницы формуляра найма новых сотрудников, как директор по побратимским связям подходит к вашему столу. Она хочет простимулировать рост связей среди членов сети и просит вас разработать систему рекомендации новых друзей "Аналитики, которых Вы должны знать".

Ваша первая реакция — предположить, что пользователь может знать друзей своих друзей. Их легко вычислить: надо лишь просмотреть друзей каждого пользователя и собрать результаты:

```
# список id друзей пользователя user (плохой вариант)
def friends_of_friend_ids_bad(user):
    # "foaf" означает "друг конкретного друга"
    return [foaf["id"]
            for friend in user["friends"]  # для каждого из друзей
                                           # пользователя
            for foaf in friend["friends"]] # получить всех ЕГО друзей
```

Если эту функцию вызвать с аргументом `users[0]` (Hero), она покажет:

```
[0, 2, 3, 0, 1, 3]
```

Список содержит пользователя 0 (два раза), т. к. пользователь Hero на самом деле дружит с обоими своими друзьями; содержит пользователей 1 и 2, хотя оба эти пользователя уже дружат с Hero; и дважды содержит пользователя 3, поскольку Chi достижима через двух разных друзей:

```
print([friend["id"] for friend in users[0]["friends"]])  # [1, 2]
print([friend["id"] for friend in users[1]["friends"]])  # [0, 2, 3]
print([friend["id"] for friend in users[2]["friends"]])  # [0, 1, 3]
```

Информация о том, что некто может стать другом вашего друга несколькими путями, заслуживает внимания, поэтому, напротив, стоит добавить счетчик взаимных друзей, а для этих целей точно потребуется вспомогательная функция, которая будет исключать тех лиц, уже известных пользователю:

```
from collections import Counter  # словарь Counter не загружается по умолчанию

# не тот же самый
def not_the_same(user, other_user):
    """два пользователя не одинаковые, если их ключи имеют разные id"""
    return user["id"] != other_user["id"]

# не друзья
def not_friends(user, other_user):
    """other_user - не друг, если он не принадлежит user["friends"], т. е.
    если он not_the_same (не тот же что и все люди в user["friends"])"""
    return all(not_the_same(friend, other_user)
               for friend in user["friends"])

# список id друзей пользователя user
def friends_of_friend_ids(user):
    return Counter(foaf["id"]
                   for friend in user["friends"] # для каждого моего друга
                   for foaf in friend["friends"] # подсчитать ИХ друзей,
                   if not_the_same(user, foaf)   # которые не являются мной
                   and not_friends(user, foaf))  # и не мои друзья

print(friends_of_friend_ids(users[3]))     # Counter({0: 2, 5: 1})
```

Такая реализация безошибочно сообщает Chi (`id 3`), что у нее с Hero (`id 0`) есть двое взаимных друзей, а с Clive (`id 5`) — всего один такой друг.

Помимо этого, вы понимаете, что, как аналитику данных, вам было бы интересно встречаться с пользователями, которые имеют одинаковую сферу интересов (кстати, это хороший пример аспекта "профессионального опыта в предметной области" науки о данных). После того, как вы поспрашивали вокруг, вам удалось получить на руки данные о сферах интересов в виде списка пар `(user_id, interest)`, состоящих из `id` пользователя и интересующей его темы:

```
# интересующие темы
interests = [
    (0,"Hadoop"), (0,"Big Data"), (0,"HBase"), (0,"Java"),
    (0,"Spark"), (0,"Storm"), (0,"Cassandra"),
    (1,"NoSQL"), (1,"MongoDB"), (1,"Cassandra"), (1,"HBase"),
    (1,"Postgres"), (2,"Python"), (2,"scikit-learn"), (2,"scipy"),
    (2,"numpy"), (2,"statsmodels"), (2,"pandas"), (3,"R"), (3,"Python"),
    (3,"statistics"), (3,"regression"), (3,"probability"),
    (4,"machine learning"), (4,"regression"), (4,"decision trees"),
    (4,"libsvm"), (5,"Python"), (5,"R"), (5,"Java"), (5,"C++"),
    (5,"Haskell"), (5,"programming languages"), (6,"statistics"),
    (6,"probability"), (6,"mathematics"), (6,"theory"),
    (7,"machine learning"), (7,"scikit-learn"), (7,"Mahout"),
    (7,"neural networks"), (8,"neural networks"), (8,"deep learning"),
```

```
    (8,"Big Data"), (8,"artificial intelligence"), (9,"Hadoop"),
    (9,"Java"), (9,"MapReduce"), (9,"Big Data")
]
```

Например, у Thor (`id 4`) нет общих друзей с Devin (`id 7`), но они оба заинтересованы в машинном обучении.

Функция, которая находит пользователей, интересующихся определенной темой, элементарна:

```
# аналитики, которым нравится целевая тема target_interest
def data_scientists_who_like(target_interest):
    return [user_id
            for user_id, user_interest in interests
            if user_interest == target_interest]
```

Все работает, однако функция в такой реализации просматривает весь список тем при каждом запросе. Если пользователей и тем много (либо планируется выполнять много запросов), то лучше создать индексный список пользователей, сгруппированный по теме:

```
from collections import defaultdict

# id пользователей по значению темы
# ключи - это интересующие темы,
# значения - это списки из id пользователей, интересующихся этой темой
user_ids_by_interest = defaultdict(list)

for user_id, interest in interests:
    user_ids_by_interest[interest].append(user_id)
```

И еще индексный список тем, сгруппированный по пользователям:

```
# идентификаторы тем по идентификатору пользователя
# ключи - это id пользователей, значения - списки тем для конкретного id
interests_by_user_id = defaultdict(list)
for user_id, interest in interests:
    interests_by_user_id[user_id].append(interest)
```

Теперь легко найти лицо, у кого с конкретным пользователем больше всего общих интересов. Для этого нужно:

1. Выполнить обход интересующих пользователя тем.
2. По каждой теме осуществить обход других пользователей, интересующихся той же самой темой.
3. Подсчитать, сколько раз встретятся другие пользователи.

```
# наиболее общие интересующие темы с пользователем user
def most_common_interests_with(user):
    return Counter(interested_user_id
        for interest in interests_by_user_id[user["id"]]
```

```
        for interested_user_id in user_ids_by_interest[interest]
        if interested_user_id != user["id"])
```

Эту функцию можно было бы затем применить в усовершенствованной версии системы рекомендации новых друзей "Аналитики, которых Вы должны знать", основанной на сочетании взаимных друзей и общих тем. Эти виды приложений мы исследуем в *главе 22*.

Зарплаты и опыт работы

В тот самый момент, когда вы собираетесь пойти пообедать, директор по связям с общественностью обращается к вам с вопросом, можете ли вы предоставить для сайта какие-нибудь примечательные факты об уровне зарплат аналитиков данных. Разумеется, данные о зарплатах имеют конфиденциальный характер, тем не менее, ему удалось снабдить вас анонимным набором данных, содержащим зарплату каждого пользователя (в долларах) и его стаж работы в качестве аналитика данных (в годах):

```
# зарплаты и стаж
salaries_and_tenures = [(83000, 8.7), (88000, 8.1),
                        (48000, 0.7), (76000, 6),
                        (69000, 6.5), (76000, 7.5),
                        (60000, 2.5), (83000, 10),
                        (48000, 1.9), (63000, 4.2)]
```

Первым делом вы выводите данные на диаграмму (в *главе 3* мы рассмотрим, как это делать). Результаты можно увидеть на рис. 1.3.

Совершенно очевидно, что лица с более продолжительным опытом, как правило, зарабатывают больше. Но как это превратить в примечательный факт? Ваша первая попытка — взять среднюю арифметическую зарплату по каждому стажу:

```
# зарплата в зависимости от стажа
# ключи - это годы, значения - это списки зарплат для каждого стажа
salary_by_tenure = defaultdict(list)

for salary, tenure in salaries_and_tenures:
    salary_by_tenure[tenure].append(salary)

# средняя зарплата в зависимости от стажа
# ключи - это годы, каждое значение - это средняя зарплата по этому стажу
average_salary_by_tenure = {
    tenure : sum(salaries) / len(salaries)
    for tenure, salaries in salary_by_tenure.items()
}
```

Но, как оказалось, это не несет какой-то практической значимости, т. к. у всех пользователей разный стаж, и, значит, мы просто сообщаем о зарплатах отдельных пользователей: