Любимой жене Лерочке –
первому в природе вечному двигателю,
рядом с которым ни на секунду нельзя остановиться
и приходится все время развиваться.
В мире, который постоянно пытается
сделать тебя кем-то другим,
оставаться собой –
величайшее достижение

Оглавление

Предисловие от автора 7

Введение 8

Об авторе 8
Для кого предназначена эта книга 8
Структура книги 9
Сопроводительные материалы и исходные файлы 10

ЧАСТЬ I. ПРИНЦИПЫ ВИЗУАЛИЗАЦИИ И ОСНОВЫ БИБЛИОТЕКИ SEABORN 11

Глава 1. Основы визуализации данных и обзор библиотек 12

Преаттентивные атрибуты – ключевые аспекты визуализации 12
Типы данных 13
Библиотеки для визуализации данных в Python 13

Глава 2. Основы библиотеки Seaborn 16

Установка и импорт Seaborn 16
Проба пера в Seaborn 17
Простой и прозрачный API. Базовые примеры 20

Глава 3. Иерархия функций в Seaborn 33

Функции уровня Figure и уровня Axes 33
Вывод графиков в конкретных областях построения 36
Дополнительная настройка с помощью объекта FacetGrid 37
Указание размеров для графиков 38

Глава 4. Структуры данных: длинные и широкие таблицы 42

Длинные таблицы 42
Широкие таблицы 43
Визуализация длинных и широких таблиц в Seaborn 45
Другие примеры визуализации широких таблиц 49

ЧАСТЬ II. СМЕНА ЭПОХ В SEABORN – ОТ КЛАССИЧЕСКОГО ИНТЕРФЕЙСА К МОДЕЛИ SEABORN.OBJECTS 57

Глава 5. Революционный интерфейс seaborn.objects 58
Что представляет собой интерфейс seaborn.objects 58
Интерфейс seaborn.objects: первые шаги 59

Глава 6. Преобразование данных перед визуализацией – базовый класс Stat 78
Агрегация данных – класс Agg и аргумент estimator 79
Статистические интервалы – класс Est и аргумент errorbar 82
Подсчет значений – класс Count и функция countplot() 89
Распределение данных – класс Hist и функция histplot() 92
Плотности распределений – класс KDE, функция kdeplot() и аргумент kde 96
Полиномиальная регрессия – класс PolyFit и функция regplot() 102
Процентили распределений – класс Perc и функции stripplot() и pointplot() 108

Глава 7. Разнесение наблюдений – базовый класс Move 113
Разнесение маркеров – класс Dodge и аргумент dodge 114
Строим накопительные диаграммы – класс Stack и функция histplot() 122
Добавляем дрожание на диаграммы – класс Jitter и аргумент jitter 125
Фиксированный сдвиг маркеров – класс Shift 132

Глава 8. Все разнообразие маркеров – базовый класс Mark 134
Общие свойства маркеров 135
Наборы данных о качестве сна/жизни и об истории Формулы 1 144
Точки и линии – классы Dot (Dots) и Line (Lines) 146
Отрисовка маршрутов – классы Path и Paths 155
Столбик или столбики? Классы Bar и Bars 166
Чертим черными чернилами черточки – класс Dash 173
От минимума до максимума – класс Range 188
Рисуем площади – класс Area 200
Фоновая визуализация интервалов – класс Band 207
Текстовые метки на диаграммах – класс Text 214

Глава 9. Методы класса Plot 228
Добавляем маркеры – метод add() 228
Масштабируем данные – метод scale() 241
Размножаем графики – метод facet() 263
Распариваем графики – метод pair() 274
Отрисовываем графики – метод layout() 284
Подписываем графики – метод label() 286
Ограничиваем графики – метод limit() 287
Делимся осями – метод share() 289
Оформляем графики – метод theme() 291
Интегрируем графики с Matplotlib – метод on() 300

Готовим, сохраняем и показываем графики – методы plot(), save() и show() 301
Строим дашборды при помощи объектов SubFigure 302

ЧАСТЬ III. ДЕМОНСТРАЦИОННЫЕ ПРОЕКТЫ 309

Глава 10. Проект 1. Визуализация t-критерия Стьюдента для независимых выборок 310

Введение 310
Исходные данные и разведочный анализ 310
Разделение данных на группы 318
Проверка выборок на нормальность распределения 322
Проверка выборок на равенство дисперсий 325
Вычисление t-критерия Стьюдента и выводы 328

Глава 11. Проект 2. Построение и визуализация сверточной нейронной сети для классификации изображений 332

Импорт библиотек и загрузка данных 332
Разведочный анализ 333
Построение архитектуры нейронной сети 338
Обучение нейронной сети 339
Визуальный анализ характеристик нейросети 340
Предсказание при помощи нейронной сети 345

Заключение 350

Предметный указатель 351

Предисловие от автора

Лучше один раз увидеть, чем сто раз услышать о том, что лучше: увидеть или услышать. Пословица, лежащая в основе этого словесного экзерсиса, появилась тогда, когда о визуализации данных еще и слыхом не слыхивали. В то же время можно возразить, что люди начали визуализировать информацию еще 50–60 тысяч лет назад в виде наскальных рисунков. Вероятно, они довольно быстро смекнули, что описывать что-то словами бывает затруднительно, а рассказывать всем и каждому, какого именно мамонта ты увидел в пещере неподалеку, просто лень – легче один раз нарисовать. К тому же рисунок гарантирует, что все поймут описанное одинаково (тогда еще не появились художники-авангардисты – хотел бы я посмотреть на их мамонта!). Через несколько десятков тысяч лет, когда возникла первая письменность, рисунки, как это ни странно, не утратили актуальности, а даже укрепили свои позиции как отдельный вид искусства и в качестве визуального дополнения к тексту. Как бы хорошо вы ни владели словом, рисунки на полях никогда не будут лишними. Спросите у Пушкина.

Меня, далеко не столь именитого тезку великого поэта, всегда поражала способность людей изображать на бумаге не только то, что случайно получилось, а то, что они заранее запланировали. Для меня лично каждое прикосновение к карандашу – это настоящая лотерея, из которой я еще ни разу не вышел победителем. Но не будем о грустном.

Конечно, это книга не о наскальной живописи и даже не о рисунках поэтов на полях, а о визуализации данных в аналитике. По моим личным наблюдениям, очень многие недооценивают роль графиков и диаграмм при анализе данных, необоснованно полагая, что одних числовых расчетов будет вполне достаточно. Но если еще несколько лет назад это практически так и было, и даже большие руководители анализировали оперативное состояние своих компаний по табличкам, то сегодня, в век клипового мышления, никто эти строчки и колонки даже видеть не хочет – всем подавай яркие и понятные графики, не требующие длительного времени для анализа. Это не хорошо и не плохо – это просто веяние времени.

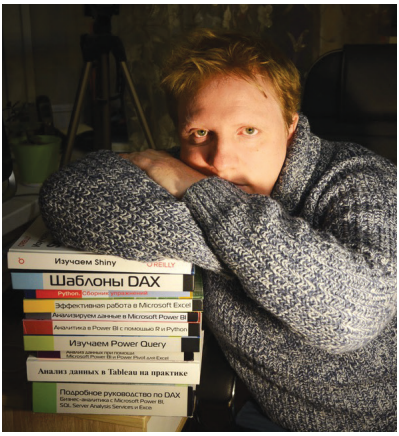
Как только я познакомился с библиотекой Matplotlib, а позже и с Seaborn, я время от времени делал попытки найти какие-то более или менее полные руководства или книги по визуализации данных в Python. Но все они не приводили ни к каким результатам. А недавно, когда вышел новый объектный интерфейс Seaborn под названием seaborn.objects, поразивший меня в самое сердечко, я решил написать такое руководство сам и доказать, что рожденный морем летать вполне способен!

Книга, которую вы держите в руках, – мой посильный вклад в обширную область знаний под названием визуализация данных. Надеюсь, вы проведете время с пользой.

Введение

Об авторе

Позвольте представиться. Меня зовут Александр Гинько, и я более семи лет занимаюсь переводом книг в сотрудничестве с издательством «ДМК Пресс».



За это время успел перевести уже более 30 книг и продолжаю переводить новые. Общий стаж переводов – более 15 лет. Также я написал две собственные книги, посвященные аналитической системе Yandex DataLens.

Среди переведенных мной книг – все бестселлеры известных итальянских специалистов и популяризаторов Power BI и DAX Марко Руссо и Альберто Феррари, а также книги по Tableau, Excel, Power Query, Python (+Dash/Plotly), R (+Shiny), SQL, машинному и глубокому обучению.

Тема визуализации данных в аналитике буквально преследует меня, но я не особенно от нее скрываюсь. Power BI, Yandex DataLens, Dash/Plotly, Shiny, Tableau – все это так или иначе связано с визуальным представлением информации в аналитике. За эти годы я накопил достаточно знаний и опыта в данной области, чтобы поделиться ими с вами.

Всю информацию, анонсы и промокоды со скидками на покупку моих книг вы можете найти на моем канале в Telegram, подписывайтесь и следите за новостями: https://t.me/alexanderginko_books. Также у меня есть собственноручно написанный бот, который поможет вам узнать подробнее обо всех моих книгах и наблюдать в онлайне за переводом новых книг: https://t.me/alexanderginko_books_bot.

В добрый путь!

Для кого предназначена эта книга

В первую очередь эта книга предназначается для тех, кто без предубеждений относится к визуализации данных и хочет освоить одну из мощнейших и хорошо структурированных библиотек для построения разнообразных графиков и диаграмм в Python. Но если вы, как и многие, недооцениваете значимость визуального представления данных, не закрывайте книгу сразу, дайте ей шанс, и я уверяю, что уже через несколько десятков страниц вы можете всерьез заинтересоваться этой темой.

Также эта книга будет чрезвычайно полезна тем, кто, как некогда и я, по роду деятельности часто сталкивается с необходимостью быстро проанализировать

визуально те или иные данные, но каждый раз боится подступиться ко всему многообразию библиотек. Моргните, когда начнете узнавать себя: этот кто-то сначала безуспешно пытается вспомнить скороговорку «matplotlib.pyplot as plt», затем делает пару попыток создать fig и ax, но в результате дело обычно заканчивается копированием из интернета фрагмента чужого кода с нужным графиком и мучительным изменением его под себя. Прочитав эту книгу, вы сможете не суетиться, а уверенно чувствовать себя при необходимости в том или ином виде визуализировать самые разные данные.

Конечно, нелишним при чтении этой книги будет опыт программирования на языке Python. Но ничего особенно сложного в плане обработки данных мы делать не будем, так что начальных знаний Python и библиотеки Pandas вам вполне должно хватить. А получить эти знания можно из любого начального курса, коих сегодня великое множество. С другой стороны, трудно представить себе читателя, который хочет что-то визуализировать в Python, не зная самого языка. Это как пытаться писать стихи, не зная правил орфографии и пунктуации.

Итак, если вы хотите погрузиться в мир полезных и привлекательных графиков и диаграмм в Python, вы пришли по адресу.

Структура книги

Книга поделена на три части.

В **первой части** мы поговорим об основных принципах, связанных с визуализацией данных в целом и в библиотеке Seaborn в частности:

- в главе 1 мы рассмотрим базовые концепции визуализации данных и пройдемся по различным библиотекам, с помощью которых можно строить графики и диаграммы в Python;
- глава 2 будет посвящена основам библиотеки Seaborn. Здесь мы научимся устанавливать нужные пакеты и попробуем свои силы в создании простейших графиков;
- в главе 3 мы подробно обсудим различия между двумя базовыми концепциями реализации функций в Seaborn: функциями уровня Figure и функциями уровня Axes;
- в главе 4 мы заглянем на территорию данных и детально поговорим про все варианты хранения информации с описанием их преимуществ и недостатков.

Во **второй части** книги мы во всех подробностях обсудим имеющиеся в арсенале библиотеки Seaborn инструменты для визуализации данных. Также мы везде будем проводить параллели с классическим интерфейсом Seaborn, и если вы с ним не были знакомы, то сможете познакомиться:

- в главе 5 мы узнаем о новой революционной концепции в виде объектной модели, получившей название Seaborn.objects;
- глава 6 будет посвящена классам, с помощью которых можно преобразовывать данные прямо во время их визуализации;
- в главе 7 мы узнаем, как можно визуально разносить наблюдения на графиках и диаграммах. Вы наверняка удивитесь легкости и элегантности используемого в Seaborn подхода;

- глава 8 будет целиком и полностью посвящена всем основным маркерам, представляющим собой классы для визуализации данных в объектной модели Seaborn;
- в главе 9 мы рассмотрим все методы базового класса `Plot` в объектной модели и покажем, как реализуемые с их помощью концепции можно воплощать в жизнь в классическом интерфейсе Seaborn.

В **третьей части** книги я решил реализовать два несложных практических проекта, на примере которых хотелось показать, как можно использовать библиотеку Seaborn с целью визуализации данных, с которыми вы работаете:

- проект из главы 10 будет посвящен расчету и визуализации t-критерия Стьюдента для независимых выборок;
- в главе 11 мы вместе реализуем несложную сверточную нейронную сеть, предназначенную для классификации изображений, а библиотекой Seaborn воспользуемся для анализа эффективности нашей сети.

Сопроводительные материалы и исходные файлы

Все исходные коды и файлы, которые будут использованы в книге, можно загрузить на странице книги на сайте издательства «ДМК Пресс».

Часть I

Принципы визуализации и основы библиотеки Seaborn

Глава 1

Основы визуализации данных и обзор библиотек

Как мы визуализируем данные? На что опираемся? Что берем за основу? Как строится мыслительный процесс, когда мы видим табличку и хотим представить буквы и числа из нее в графическом виде? Как сделать так, чтобы визуальное представление данных не усложнило их восприятие (бывает и такое!), а облегчило? На эти и другие вопросы мы постараемся вместе ответить в этой главе, прежде чем перейдем к обсуждению конкретных инструментов.

Допустим, в ваше распоряжение поступило огромное количество сырых необработанных данных. Представить их в виде таблицы можно, но ясности это не добавит, а только все запутает. Получается, что без их визуализации в сжатом и лаконичном виде просто не обойтись. А как сделать так, чтобы глазу было за что зацепиться на графике? Подобными визуальными «крючками» являются так называемые преаттентивные атрибуты, на которых базируется вся визуализация.

Преаттентивные атрибуты – ключевые аспекты визуализации

Из чего состоит любая визуализация? Представьте себе любой график или диаграмму: что приходит на ум в первую очередь? Цвет. Без цветового разделения данных на графике вы способны будете проанализировать только одну категорию. Если же вы хотите иметь перед глазами информацию сразу о нескольких категориях, без цветового акцента вам не обойтись.

Именно цвет является одним из базовых *преаттентивных атрибутов*. Так называются атрибуты, которые человеческий (да и не только) мозг считывает мгновенно – за какие-то миллисекунды, прежде чем начать анализировать все остальное. Представьте, что вам необходимо из великого множества объектов на большом экране выделить нужный. Гораздо проще будет справиться с этой задачей, если нужный вам объект будет выделен контрастным цветом по отношению к другим объектам, согласны?

Вторым важным преаттентивным атрибутом является размер. Если нужный вам объект сделать заметно бóльшим в сравнении с другими объектами, вы также сможете обнаружить его достаточно быстро. Какое-то количество миллисекунд вам потребуется на то, чтобы пережить разочарование по поводу от-

сутствия цветового выделения, но затем вы возьмете себя в руки и укажете пальцем на нужный объект.

Также к числу преаттентивных атрибутов относят длину, помогающую при сравнении категориальных переменных, позицию в пространстве, позволяющую визуально отделить группы наблюдений друг от друга, а также форму объектов и их ориентацию в пространстве.

С помощью преаттентивных атрибутов, перечисленных выше, мы кодируем данные на графиках. А с какими данными нам чаще всего приходится работать? Иными словами, какие существуют типы данных? От ответа на этот вопрос применительно к конкретной задаче может зависеть выбор наиболее подходящего вида визуализации.

Типы данных

Говоря о визуализации, под *типами данных* мы подразумеваем не те типы, которые обычно используются в программировании (целочисленный, логический, строковый и т. д.), а типы, характеризующие характер хранящихся в переменных данных.

К основным типам данных относятся *категориальный* и *числовой*. Также часто упоминается *порядковый тип данных*, но применительно к визуализации его обсуждением можно пренебречь, за исключением одного момента, который я упомяну ниже.

Категориальные, или *номинальные*, данные представляют собой некий вид перечисления и визуализируются при помощи отдельных маркеров, никак не соотносящихся друг с другом. К таким данным, например, относится пол человека, цвет глаз, профессия и т. д. Нельзя сказать, что одна профессия больше другой или наоборот, а упорядочить их можно только по алфавиту или с помощью вспомогательных критериев группировки.

Числовые, или *количественные*, данные представляют собой обычные числовые значения, которые можно сравнивать друг с другом, упорядочивать и разбивать на интервалы. К таким данным, к примеру, относятся рост человека, его вес, размер зарплаты, пенсии и т. д.

Также существует еще один важный тип данных, часто встречающийся в визуализациях. Этот тип представлен *датой и временем*, и его можно причислить к порядковому, поскольку дата и время практически всегда упорядочиваются по хронологии. Также они могут группироваться в определенные временные интервалы, такие как годы, кварталы, месяцы и т. д.

Библиотеки для визуализации данных в Python

Язык программирования Python и визуализация данных неразрывно связаны, ведь в том или ином виде отображать информацию в графическом виде нужно всем и всегда. Неудивительно, что недостатка в библиотеках, помогающих визуализировать данные, в Python не было никогда. В этом разделе мы рассмотрим основные библиотеки и обсудим их преимущества и недостатки, а также отдельно поговорим о сильных сторонах библиотеки Seaborn, которой посвящена данная книга.

Matplotlib

Библиотека Matplotlib по праву считается законодателем моды и традиций в области визуализации данных в Python, поскольку появилась в языке первой и сохраняет свое присутствие до сих пор. Несмотря на свою долгую историю, Matplotlib подвергается критике за излишнюю сложность реализации, многословность, недостаточную интерактивность и отсутствие стилистических изысков. В то же время многие более современные пакеты визуализации, и Seaborn не исключение, базируются именно на библиотеке Matplotlib.

Pandas

Простые визуализации вы можете строить, не выходя из pandas, при помощи встроенного метода `plot()`. Это максимально быстрый способ провести разведочный анализ данных прямо в датафрейме, но он сопряжен с определенными ограничениями, связанными с невозможностью создавать красочные цветовые схемы, дополнительно форматировать данные и добавлять им интерактивности.

Plotnine

Библиотека Plotnine является реализацией графической системы ggplot2, признанной в языке программирования R. В основе этой библиотеки заложены принципы и концепции грамматики графики (The Grammar of Graphics), впервые упомянутой в книге Лэланда Вилкинсона (Leland Wilkinson). Это достаточно мощная библиотека, основанная на слоях. К примеру, при построении графика вы можете начать с создания осей и далее последовательно наслаивать точки, линии, тренды и т. д. Также пакет Plotnine тесно интегрирован с pandas, что облегчает работу с датафреймами.

Bokeh

Подобно Plotnine, библиотека Bokeh базируется на принципах и концепциях грамматики графики, но при этом является родным пакетом для Python, а не портированным, как в случае с Plotnine. К преимуществам этой библиотеки можно причислить способность создавать интерактивные визуализации для веба с возможностью экспорта в JSON и HTML-объекты. Также поддерживает расширение при помощи языка JavaScript.

Plotly

Библиотека Plotly представляет богатые возможности для создания полностью интерактивных диаграмм и дашбордов. В то же время из минусов можно отметить высокую нагрузку на браузер при работе с объемными наборами данных, сложность тонкой настройки графиков и не самый простой синтаксис для создания сложных слоев.

Seaborn

Библиотека Seaborn построена на основе базового пакета Matplotlib, но, в отличие от своего прародителя, позволяет строить визуально привлекательные диаграммы и графики очень быстро – буквально в несколько строк кода, в чем вы не раз убедитесь, читая эту книгу. Также Seaborn бесшовно интегрирова-

на с pandas, что позволяет быстро и легко визуализировать данные на основе датафреймов. К преимуществам этой библиотеки можно отнести простой декларативный синтаксис, возможность создавать сложные макеты, состоящие из нескольких диаграмм, а также наличие встроенных механизмов статистических расчетов, позволяющих выводить на графиках доверительные интервалы и проводить регрессионный анализ. Кроме того, тесные родственные отношения с Matplotlib делают возможной тонкую настройку визуализаций на основе Seaborn посредством обращения к свойствам и методам Matplotlib. Еще к плюсам этой библиотеки можно отнести появившийся в 2022 году совершенно новый декларативный интерфейс `seaborn.objects`, позволяющий строить визуализации с использованием обновленной парадигмы. Далее в этой книге мы подробно поговорим о данном интерфейсе.

Книга, которую вы держите в руках, посвящена Seaborn, и к обсуждению именно этой библиотеки мы и переходим.

Глава 2

Основы библиотеки Seaborn

Что такое Seaborn? *Seaborn* представляет собой полноценную и мощную библиотеку для визуализации данных в Python на основе базового пакета *Matplotlib*. Эта библиотека предоставляет пользователю богатейший API для создания полезных и привлекательных графиков и диаграмм. Но чтобы пользоваться всем этим разнообразием, необходимо для начала установить библиотеку.

Установка и импорт Seaborn

Официальная страница библиотеки Seaborn на сайте PyPI располагается по адресу https://pypi.org/project/seaborn. Для установки библиотеки достаточно запустить в терминале следующую инструкцию:

```
pip install seaborn
```

Мы в этой книге будем работать в среде Jupyter Notebook, и для установки библиотеки Seaborn непосредственно из ячейки Jupyter вам понадобится поставить перед инструкцией `pip` восклицательный знак (`!`) или знак процента (`%`). В результате будет установлен как сам пакет Seaborn, так и все необходимые для его работы обязательные зависимости, в число которых входят пакеты numpy, pandas и matplotlib. Вы также можете установить опциональные зависимости, чтобы использовать особый функционал библиотеки, выполнив команду, подобную показанной ниже:

```
pip install seaborn[stats]
```

Воспользовавшись опциональными зависимостями, вы можете, к примеру, выполнять полноценный регрессионный анализ или быстрее осуществлять кластеризацию матриц.

После установки пакета Seaborn вы можете сразу приступать к работе, импортировав библиотеку следующим образом:

```
import seaborn as sns
```

Традиционно принято импортировать библиотеку Seaborn с коротким псевдонимом `sns`, но вы можете выбрать любое другое сокращение. Любопытный факт: название библиотеки произошло от имени персонажа американского политического телесериала «Западное крыло» **С**эмюэля **Н**ормана **С**иборна. Именно поэтому принято сокращать имя библиотеки до `sns` (инициалы героя).

Проба пера в Seaborn

Вам наверняка не терпится уже что-нибудь потрогать руками. Самый короткий путь – воспользоваться встроенной функцией `load_dataset()`, с помощью которой можно загрузить один из множества заранее подготовленных наборов данных и попробовать что-то визуализировать, не имея никаких знаний. Полный синтаксис этой функции выглядит следующим образом:

```
seaborn.load_dataset(name, cache=True, data_home=None, **kws)
```

Аргумент `name` служит для указания названия набора данных для загрузки из ограниченного перечня, ознакомиться с которым можно по адресу https://github.com/mwaskom/seaborn-data?tab=readme-ov-file#data-sources. С помощью аргумента `cache` вы можете попросить Seaborn загружать набор из кеша, если это возможно, а аргумент `data_home` служит для указания папки, в которой будут храниться кешированные данные. Дополнительным аргументом `kws` можно воспользоваться для сквозной передачи параметров функции `pandas.read_csv()` в формате ключей и значений. Возвращает функция *датафрейм* Pandas.

Давайте с места в карьер – загрузим небольшой, но очень любопытный датасет `geyser` и с помощью одного лишь графика, не прибегая к статистическим функциям, попытаемся проанализировать содержащиеся в нем данные. Для этого выполним следующие инструкции:

```
df_geyser = sns.load_dataset("geyser")
print(df_geyser)
```

Вывод:

```
     duration  waiting   kind
0       3.600       79   long
1       1.800       54  short
2       3.333       74   long
3       2.283       62  short
4       4.533       85   long
..        ...      ...    ...
267     4.117       81   long
268     2.150       46  short
269     4.417       90   long
270     1.817       46  short
271     4.467       74   long

[272 rows x 3 columns]
```

В этом наборе данных содержится 272 наблюдения за активностью одного из самых знаменитых гейзеров на Земле, расположенного в Йеллоустонском национальном парке в штате Вайоминг, США. Обнаружен он был в 1870 году экспедицией Уошберн–Лэнгфорд–Доан и тогда же получил свое любопытное название Старый Служака, или Олд-Фейтфул (Old Faithful).

В колонке `duration` содержится информация о длительности извержения гейзера в минутах, в колонке `waiting` – время ожидания в минутах до следующего извержения, а в колонке `kind` – тип извержения: короткое (`short`) или длинное (`long`). Это мы пока можем только предполагать.

Сейчас я вам докажу на простом примере, что можно, толком не разбираясь в самих числовых данных, с помощью одной лишь диаграммы рассказать, что происходит с вверенным нам для анализа гейзером. Вы словно окажетесь рядом с ним, никуда не перемещаясь со своего удобного дивана.

Для этого воспользуемся функцией *pairplot()*, предназначенной для визуализации попарных зависимостей между числовыми признаками. Сигнатура этой функции содержит большое количество аргументов, но основным является первый позиционный аргумент, принимающий датафрейм:

```
sns.pairplot(df_geyser, diag_kind="kde")
```

Здесь мы также явно указали, что распределения признаков (`diag_kind`) хотим вывести не в виде гистограмм, а в виде оценки плотности.

Если вы работаете в Jupyter Notebook или в терминале IPython с активированным режимом Matplotlib, вы сразу после запуска этой ячейки увидите диаграмму, показанную на рис. 2.1.

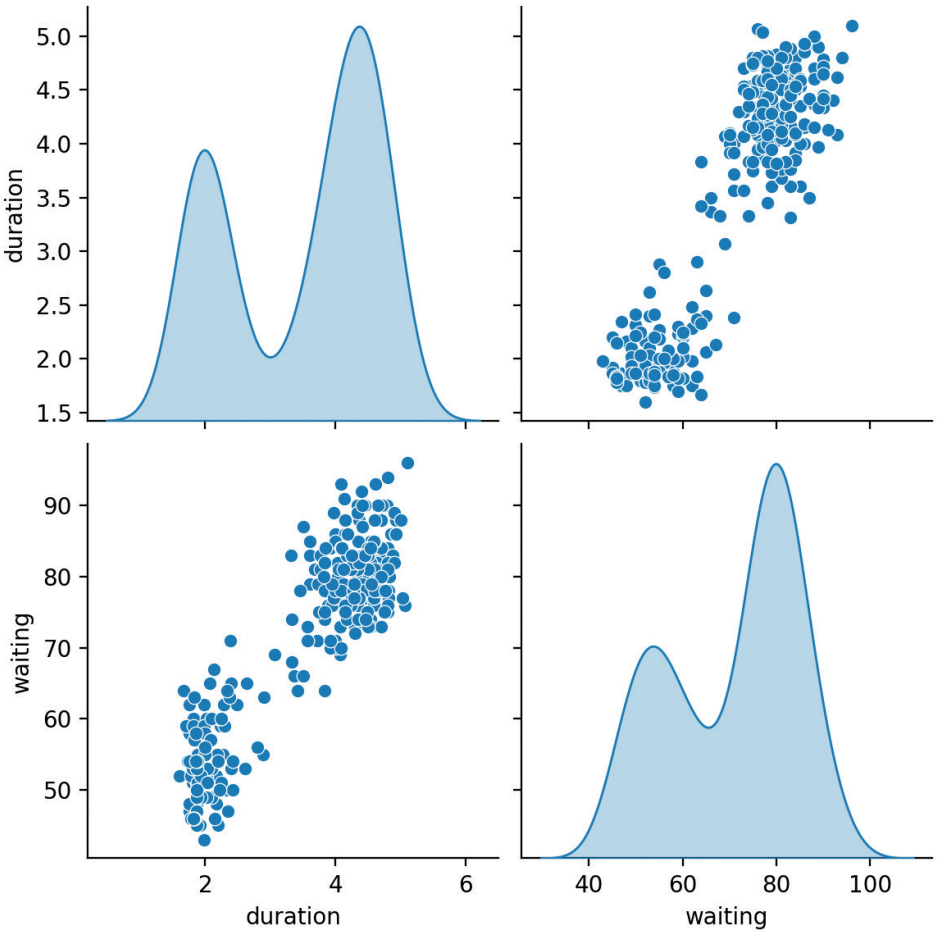


Рис. 2.1. Попарные зависимости признаков в наборе данных geyser

Конец ознакомительного фрагмента.
Приобрести книгу можно
в интернет-магазине
«Электронный универс»
e-Univers.ru