Руслан Назаров

Data Science для новичков

Руслан Назаров

Data Science для новичков

http://www.litres.ru/pages/biblio_book/?art=69435934
ISBN 9785006028869

Аннотация

Книга для тех, кто хочет разобраться в искусственном интеллекте, и даже заработать на этом. Основные сведения по статистике, программированию и нейронным сетям. И все это объяснется просто. Дополнительные сведения по языку Python позволят научиться программированию. Книга пригодится всем, кто учится программированию, Data Science, нейронным сетям.

Содержание

Data Science для новичков

Руслан Назаров

ISBN 978-5-0060-2886-9
Создано в интеллектуальной издательской системе Ridero

От автора

В этой книге я собрал все 1) важные и 2) базовые знания по Data Science.

В книге код дан так, как вы его увидите в Visual Studio Code.

По тексту много ссылок на другие книги, учебники, пособия в онлайне. Рекомендуется ознакомиться с этими материалами. Моя книга может служить только своеобразным путеводителем на начальном уровне для всех, кто желает стать специалистом по Data Science.

Принципы образования

Здесь я собрал несколько практических советов, принципов самообразования.

1. Прежде, чем начинать изучение новой науки, надо понять зачем вы это делаете. Карьера, деньги? Да, многие учатся ради этого. Но это слишком незначительно для человека. Деньги и карьера могут быть побочным продуктом от учебы, но не должны быть основными причинами. Ставьте себе великие цели. Если беретесь за математику, то ваша цель – решить одну из задач Гилберта. Если беретесь за DS, то ваша задача – изобрести искусственный интеллект. Не меньше этого. Пускай даже в итоге это у вас не получится, но идти надо по пути великих целей. Только так вы сможете убедить себя, что огромные усилия, которые вы будете прикладывать, стоят того. Только так вы сможет сделать учебу интересной для себя.

2. Ваш план образования должен иметь два направления: сложное и простое. Например, сложное – математика, простое – публицистические книги про ИИ, основы программирования. Почему? Чтобы мозг не скучал. И чтобы мозг понимал, что вы занимаетесь сложной задачей и поэтому мозг должен работать максимально эффективно. Кроме того, пока мозг не ощущает опасность, он не начинает работать

на максимальном уровне. Теория вероятностей вполне может быть такой опасностью. Делить учебу на сложные и простые занятия важно еще и потому, что так можно эффективно расходовать время. Если вы слишком заняты другими делами, вы болеете или устали, то можно взять легкие дела из плана учебы. Когда вы освободитесь, то можно будет заняться максимально сложными делами. Вы все время учитесь, а это очень важно. Помните, что нельзя тратить время.

3. Не пытайтесь все понять и запомнить с первого раза. Не надо себя насиловать. Пока вы учите новую науку, ваша задача не запомнить, а понять. Если вы поняли дифференциальное исчисление, а затем забыли половину теорем дифференциального исчисления, то ничего страшного. Понимание прежде всего!

4. Учитесь «слоями». Прочитайте книгу, например, по математике. Попытайтесь понять максимально много, но без насилия над собой. Не пытайтесь запомнить все теоремы, формулы. Ваша задача – понять, повторюсь. Теперь возьмите следующую книгу по математике. Попытайтесь понять максимально много. В этот раз вы поймете уже гораздо больше. Продолжайте, не останавливайтесь. Запомните, нельзя полностью выучить математику, Data Science или любую науку. Ваша цель – тренировать способность понимать. Чем больше будет «слоев» понимания, тем лучше.

5. Новый «слой» должен быть чуть сложнее предыдущего. Например, начните с учебника по математике для шко-

лы, затем возьмите учебник для университетов. Не останавливайтесь на этом «слое». Возьмите специальные книги, например подробное изложение линейной регрессии.

6. Не перечитывайте книги, которые вы уже прочитали. Если вы не понимаете какую-то тему, то лучше взять другую книгу по этой теме (или почитать пост в каком-то блоге). Однако у вас должна быть «опорная книга», например по математике, DS, программированию. Это такая книга, которая, на ваш взгляд и для вас, содержит самое простое и полное изложение темы. Это будет ваш справочник. Перечитывать такие книги можно.

7. Не бойтесь сложных книг. Даже если из всей книги вы поняли только 90%, польза огромная. Вы учите свой мозг чувствовать себя как дома в той теме, которой посвящена книга. Вы учите мозг не бояться этой темы. Вы учите термины, теоремы, формулы, алгоритмы.

8. Никогда себя не ругайте. Если что-то не получается сейчас, то это получится потом. Главное – это идти по правильному пути. Это не означает, что вы не должны заниматься самоанализом. Почему что-то не понятно? Что можно сделать, прочитать, чтобы лучше это понять?

9. Создавайте теории. Всегда создавайте теории, идеи. Не ждите, когда станете профессионалом. Ищите необычные решения, делайте безумные предположения, даже если только начали читать первые книги по математике. Главное помнить, что любая идея должна быть проверена. Пусть такой

проверкой будет следующая книга или ваш личный проект.

10. Не соглашайтесь с идеями типа «математика это сложно». Не позволяйте себя этим пугать. Для человека нет ничего сложного. Что такое «сложно»? Любое знание – это объекты и связи между ними. «Сложно» – это значит много объектов, много связей. Базовый принцип для приготовления бутерброда и решения математических задач один: возьмите элементы и свяжите их. Нет никакой абстрактной «сложности». Если все дело в количестве элементов и связей, то вам просто нужно время. Базовый принцип вы уже знаете и даже блестяще им владеете (бутерброды же хорошо у вас получаются!).

11. Если вы не понимаете книгу по математике или DS, то причина просто в том, что у вас нет навыка и вы не знаете всех предпосылок. Почему, например, книги по экономике не кажутся такими сложными? Потому что многие понятия из экономики знакомы нам: безработица, прибыль и т. п. Но элементы математики, особенно абстрактной, мы встречаем в жизни редко. Возьмем теорему математики. В том учебнике, который вы читаете, нельзя рассказать все причины, по которым возникла эта теорема. Поэтому вы можете не понять эту теорему. Это не ваша вина. Вы поймете эту теорему, пускай и в следующем «слое». И еще, помните, что формулы на самом деле очень упрощают жизнь! Не обращайте внимание на слова «это очевидно» и т. п. Возможно, это очевидно автору, но не вам. В этом нет проблемы!

12. Создавайте контекст. Что такое контекст? Вы читаете учебники по математике. Но вы должны также читать книги по истории математики, по философии математики и т. п. Чем больше разных знаний у вас будет, тем лучше. Так нашему мозгу легче запоминать и так мозг лучше понимает. Помните, наш мозг воспринимает информацию из контекста, потому что мир тоже состоит из связей.

13. Делайте упражнения. Но только, когда нет других более важных дел. Упражнения, как правило, это искусственные примеры, которые являются скучными и мало применяются в жизни. Например, в обучении языку давно уже поняли, что искусственные примеры – это плохой способ учить язык. Кроме того, упражнения из учебника часто пытаются вас запутать. Это плохо. Зачем? Вам должны объяснять. Опыт придет на конкретных примерах из жизни. Поэтому создавайте свои проекты. Пускай это будут примеры, которые основаны на понятных для вас проблемах. Может быть, вам интересно применить методы математики и DS для изучения экологических проблем. Займитесь этим!

14. Ищите хорошие книги. Как это сделать? Возьмите пять книг по одной теме наугад. В этих книгах будут ссылки на другие учебники, книги. Дальше собирайте библиотеку и читайте эти книги. И добавьте в эту библиотеку классические книги по теме. Это всегда хорошее решение!

15. Не смешивайте занятия. Если сейчас вы учите математику, то не пытайтесь вспомнить код из последнего проекта.

16. Старайтесь объяснять все своими словами. Вы учитесь, накапливаете знания. Расскажите о своих знаниях! Это может быть блог или подкаст. Если можете своими словами объяснить, значит хорошо все поняли.

17. Не бойтесь, что вы медленно учитесь. Главное учиться.

18. Найдите хобби. Пускай ваше хобби будет способом отвлечься от учебы. Это тоже необходимо. Умейте отдыхать! Возьмите пару недель или месяц, чтобы отдохнуть. Пока вы отдыхаете, мозг приводит ваши знания в порядок и находит неожиданные связи.

Глава 1. Подготовка данных

```
import pandas as pd  # для работы с датафреймом
import numpy as np  # для работы с матрицами, векторами
import matplotlib.pyplot as plt  # для графиков
import seaborn as sns  # для графиков
import re  # для регулярных выражений
import os
```

Data Science содержит три больших отдела:

1) получение и подготовка данных;

2) статистическая обработка данных;

3) машинное обучение.

Статистическая обработка нацелена на:

1) описание сгруппированных данных (медиана, среднее и т.п.);

2) описание взаимодействия между различными группами данных (корреляция и т.п.).

Другими словами, *статистическая обработка* требует понять данные, а значит и те реальные процессы, которые стоят за данными. Это важно учитывать. В конечном счете моя задача не просто получить корреляцию, а понять данные. Что это означает? Во-первых, я должен проверять как

корреляцию, так и другие статистики, на вменяемость, на соответствие действительности. Во-вторых, именно в действительности я должен искать подсказки, какие тесты применить, какие метрики получить. Так, например, понимание данных можно получить и из других источников, не только за счет применения статистических тестов. Можно сделать предположения о процессах, отраженных в данных, на основании опыта, а уже затем проверить предположения с помощью статистики. Важно помнить, что математика – это только язык, который используют, чтобы описать действительность. Не надо подменять математикой саму действительность.

Машинное обучение нацелено на создание алгоритма, который позволит предсказывать целевой признак на основании заданных признаков в автоматизированном режиме. Другими словами, статистическая обработка позволяет понять процессы, а машинное обучение – предсказать процессы.

Однако начинается все с предварительной подготовки данных. В самом деле, если не подготовить данные, не убрать пропуски, дубликаты и т.п., то это повлияет на качество как статистической обработки, так и машинного обучения (или даже не позволит их выполнить). В этом разделе я займусь именно подготовкой данных.

Подготовка данных включает, но не ограничивается, следующие элементы:

1) проверка правильности формирования индекса, наименования столбцов (признаков). Например, может быть обнаружено, что в наименовании столбцов есть лишние пробелы;

2) проверка типа данных. Например, численные данные могут быть отмечены как объекты или наоборот;

3) поиск дубликатов;

4) очистка строковых данных от лишних символов. Например, наличие слэша там, где это очевидно неуместно;

5) обработка значений, которые очевидно являются ошибочными. Например, в столбце с количеством страниц указан жанр книги и т.п.;

6) создание новых признаков. Например, по значениям двух уже имеющихся столбцов можно создать третий;

7) укрупнение категорий в категориальных признаках;

Предупреждение об источнике данных

Источник данных находится по адресу https://www.kaggle.com/jealousleopard/goodreadsbooks. Мне неизвестна процедура, которую применял автор для сбора данных. Поэтому всегда надо помнить, что особенности именно данного набора могут оказать влияние на выводы. Идеально было бы самостоятельно собрать данные или использовать дополнительно иные сборки данных, но пока в этой методичке такая задача не стоит. Кроме того, сайт Goodreads с конца 2020 ограничил использование API и получение данных.

Почему при таких ограничениях я выбрал именно данный набор? Как я указывал выше, прежде всего, надо осно-

вываться на действительности, чтобы понять данные. А значит я должен разбираться или хотя бы понимать те объекты, тот предмет, которого касаются данные. Так как я много читаю, полагаю, что неплохо понимаю, за что можно поставить книге ту или иную оценку, как на это влияет количество страниц и прочее. Поэтому я выбрал именно эти данные.

```
# скопирую путь к файлу из проводника Windows, но заменю \ на /
# error_bad_lines=False: если в строках есть ошибки, они будут пропущены
data = pd.read_csv(os.getcwd() + '\\books.csv', error_bad_lines=False)

# смотрю данные
data.head(3)
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9780439554893	eng	352	6333	244

Вижу, что данные можно разбить на две категории:
1) сведения о книге (название, автор, isbn, язык, количе-

ство страниц, дата публикации и издательство);

2) сведения о реакции читателей (средний рейтинг, количество отзывов, количество оценок).

Данные рассказывают не просто про книгу и не просто про реакцию на книгу, а про реакцию читателей, измеренную конкретными признаками, на книгу, также измеренную конкретными признаками. Если у меня нет специального задания по анализу, то я могу наметить стратегию исследования по имеющимся признакам. Так, например, может быть интересно, каким книгам чаще ставят положительные оценки? как зависит оценка книги от количества страниц в ней?

Сформулирую для себя общую цель – изучить от чего зависит оценка книги.

Таблица задает две оси: вертикальная – наблюдения, горизонтальная – признаки.

Метка для наблюдений – индекс, метка для признаков – название признаков (название столбцов). Поэтому, естественно, что подготовка данных должна начинаться с обследования меток. Однако индекс формируется автоматически в порядке возрастания от 0 до n (это поведение по умолчанию можно изменить). Названия столбцов были предоставлены вместе с данными, поэтому их-то отдельно и надо обследовать.

```python
# смотрю названия столбцов на наличе ошибок
data.columns

# хорошей практикой является создание копии данных
db = data.copy()

# хорошей практикой является создание копии данных
db = data.copy()

# уберу лишние пробелы в названии столбца
db = db.rename(columns = {'  num_pages' : 'num_pages'})

# проверю,
# что проблема устранена
db.columns
```

```
Index(['bookID', 'title', 'authors', 'average_rating', 'isbn', 'isbn13',
       'language_code', 'num_pages', 'ratings_count', 'text_reviews_count',
       'publication_date', 'publisher'],
      dtype='object')
```

```
1  # смотрю информацию о данных, чтобы определить тип объектов,
2  # отсутствие пропусков
3  db.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11123 entries, 0 to 11122
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   bookID              11123 non-null  int64  
 1   title               11123 non-null  object 
 2   authors             11123 non-null  object 
 3   average_rating      11123 non-null  float64
 4   isbn                11123 non-null  object 
 5   isbn13              11123 non-null  int64  
 6   language_code       11123 non-null  object 
 7   num_pages           11123 non-null  int64  
 8   ratings_count       11123 non-null  int64  
 9   text_reviews_count  11123 non-null  int64  
 10  publication_date    11123 non-null  object 
 11  publisher           11123 non-null  object 
dtypes: float64(1), int64(5), object(6)
memory usage: 1.0+ MB
```

Вижу, что есть 12 признаков (нумерация начинается с 0 и продолжается до 11) и 11123 наблюдений (строк). Пропусков нет (количество объектов по столбцам одинаковое). Индекс у нас это RangeIndex. По типам данных заметно две проблемы. **isbn** помечен как объект, а **isbn13** как int64. Кроме того, **publication_date** помечен как объект, хотя это очевидно дата. Изменю тип данных.

```python
# изменю тип данных
db['isbn13'] = db['isbn13'].astype('object')

# использую errors, иначе выводится ошибка
db['publication_date'] = pd.to_datetime(db['publication_date'], errors='coerce')

# смотрю результат изменений
db.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11123 entries, 0 to 11122
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype         
---  ------              --------------  -----         
 0   bookID              11123 non-null  int64         
 1   title               11123 non-null  object        
 2   authors             11123 non-null  object        
 3   average_rating      11123 non-null  float64       
 4   isbn                11123 non-null  object        
 5   isbn13              11123 non-null  object        
 6   language_code       11123 non-null  object        
 7   num_pages           11123 non-null  int64         
 8   ratings_count       11123 non-null  int64         
 9   text_reviews_count  11123 non-null  int64         
 10  publication_date    11121 non-null  datetime64[ns]
 11  publisher           11123 non-null  object        
dtypes: datetime64[ns](1), float64(1), int64(4), object(6)
memory usage: 1.0+ MB
```

Вижу, что в **publication_date** появилось два пропущенных значения. Так как подобных строк всего две, я могу их удалить.

```
1  # удалю данные с ошибочной датой
2  db = db.loc[~db['publication_date'].isin(['NaT'])]
```

Запись выше надо читать так. В таблице db выбрать только те строки, у которых в столбце «publication_date» нет значения NaT. Значок тильды ~ означает «не». Метод isin проверяет наличие указанных данных в ячейке.

Здесь также важно, что я могу взять изначальную таблицу, отфильтровать ее, как мне это необходимо, а затем заменить изначальную таблицу отфильтрованной. Другими словами, изначально у меня была таблица db, после изменений я получаю таблицу с тем же названием db, но уже отфильтрованную.

```
1  # смотрю результат изменений
2  db.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11121 entries, 0 to 11122
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   bookID             11121 non-null  int64
 1   title              11121 non-null  object
 2   authors            11121 non-null  object
 3   average_rating     11121 non-null  float64
 4   isbn               11121 non-null  object
 5   isbn13             11121 non-null  object
 6   language_code      11121 non-null  object
 7   num_pages          11121 non-null  int64
 8   ratings_count      11121 non-null  int64
 9   text_reviews_count 11121 non-null  int64
 10  publication_date   11121 non-null  datetime64[ns]
 11  publisher          11121 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(4), object(6)
memory usage: 1.1+ MB
```

Теперь я должен заняться дубликатами строк. Я могу искать либо полные дубликаты (данные в каждом столбце для строки полностью совпадают), либо искать дубликаты выборочно. Здесь надо обратить внимание, что **isbn** является уникальным идентификатором каждой изданной книги. Поэтому логично искать дубликаты только по этому признаку, так как книги вполне могут совпадать по иным признакам и это нормально.

```
# поиск дубликатов
db.duplicated(subset=['isbn13']).sum()
```

```
0
```

Дубликатов по **isbn13** нет. Но все-таки посмотрю дубликаты по названию и имени автора.

```
# строки, которые являются дубликатами
# первая строка, с которой сравниваются эти, не показана
db[db.duplicated(subset=['title', 'authors', 'publication_date', 'publisher'])].head(3)
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count
3147	11557	Swan Song	Robert R. McCammon	4.28	0671741039	9780671741037	eng	956	46244
3487	12683	The Shining	Stephen King/Campbell Scott	4.22	0743536991	9780743536998	eng	0	13
5298	19135	'Salem's Lot	Stephen King/Ron McLarty	4.02	0743536959	9780743536950	en-US	0	56

Такой подход позволяет понять, почему могут совпадать имя автора и название при различных **isbn**. Вижу, что, как правило, такие дубликаты – это аудиокниги. С этим придется разобраться отдельно. Для начала посмотрю, есть ли нулевое количество страниц у книг.

```
1  # количество книги с 0 страниц
2  db[db['num_pages'] == 0].count()
```

```
bookID               76
title                76
authors              76
average_rating       76
isbn                 76
isbn13               76
language_code        76
num_pages            76
ratings_count        76
text_reviews_count   76
publication_date     76
publisher            76
dtype: int64
```

Таких книг 76. Что их объединяет? Посмотрю издательства.

```
1  # количество книг по издательствам
2  # показаны только первые пять
3  db[db['num_pages'] == 0]['publisher'].value_counts().head(5)
```

```
Random House Audio              19
Simon & Schuster Audio           5
Tantor Media                     5
Random House Value Publishing    4
Macmillan Audio                  4
Name: publisher, dtype: int64
```

Вижу, что в основном это издательства, которые выпускают аудиокниги. Это логично. Если у книги нет страниц, то это просто аудиокнига. Но посмотрим количество страниц для тех книг, которые выпускали эти издательства.

Код выше весьма любопытен. Как его прочитать? Берем таблицу db. В этой таблице ищем такие строки, в которых столбец равен 0. Далее, в отфильтрованной таким образом таблице, берем столбец 'publisher'. После этого вызываем value_counts для подсчета количества и head для ограничения вывода результатов.

```
# перевожу все буквы в названии в нижний регистр
# затем ищу совпадение, показаны первые 5 результатов
db[db['publisher'].str.lower().str.contains('audio')]['num_pages'].sort_values(ascending=False).head(5)
```

```
7511    1162
5845     473
5849     447
2102     368
4841     356
Name: num_pages, dtype: int64
```

Вижу, что аудиоиздательства издают нечто, что имеет страницы, даже 1162 страницы! Посмотрю на это.

Заметка к коду

Код выше очень похож на предыдущий и может быть аналогично прочитан. Однако обращает внимание, что при первоначальной фильтрации таблицы я могу добавить дополни-

тельные методы, например str и т. п.

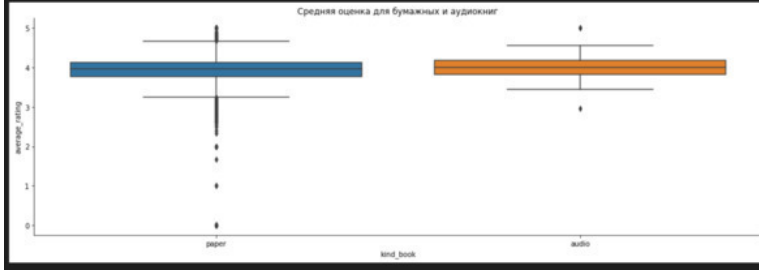
```
1 db[db['num_pages'] == 1162]
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count
7511	28845	The Story of Philosophy	Will Durant/Grover Gardner	4.12	1572704209	9781572704206	eng	1162	31

В интернете, например, на сайте Amazon, можно обнаружить эту книгу. И она оказывается аудиокнигой! Таким образом, количество «страниц» еще не говорит нам, что это бумажная книга. Это может быть, например, вес дисков. Более верный признак – это именно издательство. Как же поступить? Ведь сравнить книги аудио и бумажные по количеству страниц не получится. Следовательно, в одном признаке смешаны различные числа – количество страниц и вес дисков. Удалю все аудиокниги, но сначала сравню оценки по бумажным и аудиокнигам.

```
db['kind_book'] = np.where(db['publisher'].str.lower().str.contains('audio'), 'audio', 'paper')
sns.catplot(data=db, x='kind_book', y='average_rating', kind='box', aspect=3)
plt.title('Средняя оценка для бумажных и аудиокниг')
```

```
Text(0.5, 1.0, 'Средняя оценка для бумажных и аудиокниг')
```



Вижу, что медиана не отличается, хотя разброс оценок для бумажных книг больше, чем для аудиокниг. Удалю вспомогательный признак, а также все аудиокниги. Надо учитывать, что такой подход, когда сравниваются две категории книг по графикам, является довольно грубым. Здесь бы стоило применить, например, t-тест. Но у меня нет специальной задачи исследовать аудио- и бумажные книги, поэтому ограничусь графиками.

Заметка к коду

Как прочитать np. where? Здесь я беру исходные данные признака, нахожу один из них, например названия с «audio», и присваиваю значение «audio», а если это не выполняется, то присваиваю значение «paper».

```
db = db.drop('kind_book', axis=1)
db = db[~db['publisher'].str.lower().str.contains('audio')]
db.head(3)
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9780439554893	eng	352	6333	244

Еще раз посмотрю на таблицу, но выберу только количество страниц до 10. Посмотрю издательства.

```
db[db['num_pages'] < 10]['publisher'].value_counts().head(3)
```

```
Tantor Media                6
Caedmon                     6
Grand Central Publishing    5
Name: publisher, dtype: int64
```

Если изучить полный список, то можно заметить, что там есть издательства Listening Library и ряд других, которые очевидно выпускают аудиокниги. Прихожу к выводу, что книги с количеством страниц 10 – это аудиокниги. Удалю их.

```
1  db = db[db['num_pages'] > 10]
```

Добавлю два дополнительных признака в таблицу:

1) десятилетие, в котором вышла книга,

2) квартал, в котором вышла книга.

Это называется конструированием признаков, исходя из целей исследования. Специальных целей передо мной не ставили, я ищу их для себя сам. Меня будет интересовать, как распределяются книги по десятилетиям и в какой квартал их чаще выпускают. Почему именно эти признаки? Потому что десятилетия отражают развитие рынка книготорговли, изменение форматов и т. п. Кварталы же зависят от праздников, сезонности, что также может оказывать влияние на оценку книги.

Конструирование признаков возможно двумя путями:

1) самостоятельно определить новый признак,

2) признак создается автоматически, например простым возведением каждого числового признака в квадрат или перемножением каждой пары таких признаков.

Если затруднительно самостоятельно определить призна-

ки, которые было бы интересно изучить, то можно применить второй метод. В этом случае можно и не создавать признаки на этапе обработки данных, можно использовать специальные методы, например полиномиальную регрессию.

```
db.loc[ : , "decade"] = 10 * (db.loc[ : , "publication_date"].dt.year // 10)
db.loc[ : , 'quarter'] = db.loc[ : , 'publication_date'].dt.quarter
db.head(3)
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9780439554893	eng	352	6333	244

Теперь посмотрю на признак authors. Вижу, что здесь есть случаи, когда указано несколько имен через /. Сколько таких случаев?

Заметка к коду

Чтобы получить значение десятилетия, я использовал код 10 * (db. loc [:, «publication_date»].dt. year // 10). Почему? В этом коде я сначала делю год на 10, причем оставляю только целую часть. Например, если год 2001, то получаю 200.

А затем уже умножаю на 10 это «целое» число, что и дает декаду.

```
1 db[db['authors'].str.contains('/')].head(3)
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221
3	5	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9780439655484	eng	435	2339585	36325

Из примеров видно, что, как правило, через слеш указаны переводчики. Однако возможны и другие ситуации. Это могут быть соавторы или вариант написания имени. Без дополнительного исследования внешних источников это определить нельзя. В этой ситуации можно сделать следующее. Заменю слеш на запятую. Создам колонку tra_co (переводчик или соавтор) и присвою 1 тем случаям, где есть запятая, и 0 остальным.

```
db['authors'] = db['authors'].str.replace('/', ',')
db['tra_co'] = np.where(db['authors'].str.contains(','), 1, 0)
db.head(3)
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling,Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling,Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9780439554893	eng	352	6333	244

Теперь разберусь с книгами, у которых слишком большие значения количества страниц. Посмотрю на них повнимательнее.

```
db[db['num_pages'] > 1000].head(3)
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count
4	8	Harry Potter Boxed Set Books 1-5 (Harry Potte...	J.K. Rowling,Mary GrandPré	4.78	0439682584	9780439682589	eng	2690	41428	164
6	10	Harry Potter Collection (Harry Potter #1-6)	J.K. Rowling	4.73	0439827604	9780439827607	eng	3342	28242	808
21	30	J.R.R. Tolkien 4-Book Boxed Set: The Hobbit an...	J.R.R. Tolkien	4.59	0345538374	9780345538376	eng	1728	101233	1550

Как правило, книги с количеством страниц больше 1000 – это многотомные издания. Очевидно, что просто убрать такие книги, как я сделал с книгами, у которых было 0 страниц, нельзя. Что же тогда? Я должен найти все такие книги и пометить их. Для этого надо определить маркеры, которые позволят найти многотомные издания. Уже представленный выше список дает идеи:

1) книги с наличием знака #,

2) книги со словами Boxed Set.

Кроме того, под подозрением все книги со словами «books», «vol.», «volume», «series».

См. хорошее руководство по регулярным выражениям https://developers.google.com/edu/python/regular-expressions.

```
db['multivolume'] = np.where(
    db['title'].str.contains('#|boxed|books|volume|series|vol.', flags=re.IGNORECASE), 1, 0
    )

db.head(3)
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling,Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling,Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9780439554893	eng	352	6333	244

```
# статистическое описание оценок
db['average_rating'].describe()
```

```
count    10862.000000
mean         3.933448
std          0.348318
min          0.000000
25%          3.770000
50%          3.960000
75%          4.130000
max          5.000000
Name: average_rating, dtype: float64
```

Минимальная оценка это 0. Но на сайте нельзя поставить такую оценку. Поэтому 0 означает отсутствие оценки, то есть это категориальный признак, который «пробрался» в числовой. Посмотрю количество и удалю, так как такое смешение недопустимо. Однако, если оценка 0, но количество оценок не 0, то это просто ошибка. Проверю это.

```
db[(db['average_rating'] == 0) & (db['ratings_count'] > 0)].sum()
```

```
bookID               0.0
title                0.0
authors              0.0
average_rating       0.0
isbn                 0.0
isbn13               0.0
language_code        0.0
num_pages            0.0
ratings_count        0.0
text_reviews_count   0.0
publisher            0.0
decade               0.0
quarter              0.0
tra_co               0.0
multivolume          0.0
dtype: float64
```

```
db[db['average_rating'] == 0]['average_rating'].count()

db = db[db['average_rating'] != 0]

# проверю минимальное значение
db['average_rating'].min()

# удалю признаки, которые больше не пригодятся
db = db.drop(['bookID', 'isbn', 'isbn13', 'authors', 'publication_date', 'publisher'], axis=1)
```

Удалю редкие категории. Для этого можно применить следующий код к каждой категориальной переменной.

```
db['decade'].value_counts().head(3)

2000    7332
1990    2436
1980     610
Name: decade, dtype: int64
```

Здесь не привожу вывод по каждой категории. Однако общий вывод такой: редкие категории встречаются в ****decade****, поэтому объединю все года, у которых менее 20 значений в год 1940.

Почему необходимо укрупнений категорий? Потому что маленькие категории несут мало информации, в то же время увеличение размерности данных ведет к тому, что известно как «проклятие размерности».

```python
db['decade'] = db['decade'].replace([1900, 1930, 2020, 1920, 1910], 1940)

# укажу index=False, чтобы при сохранении не был создан
# дублирующий индекс
db.to_csv('gd_clean_data.csv', index=False)

db.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10838 entries, 0 to 11122
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   title               10838 non-null  object
 1   average_rating      10838 non-null  float64
 2   language_code       10838 non-null  object
 3   num_pages           10838 non-null  int64
 4   ratings_count       10838 non-null  int64
 5   text_reviews_count  10838 non-null  int64
 6   decade              10838 non-null  int64
 7   quarter             10838 non-null  int64
 8   tra_co              10838 non-null  int32
 9   multivolume         10838 non-null  int32
dtypes: float64(1), int32(2), int64(5), object(2)
memory usage: 846.7+ KB
```

Глава 2. Статистическое исследование

```
# ! pip install wordcloud
# ! pip install textblob
# ! pip install mlxtend
# ! pip install pingouin
# ! pip install matplotlib
# ! pip install papanda
```

```
import papanda as pp  # для расчета робастного среднего
import pandas as pd  # для работы с датафреймом
import numpy as np  # для вычислений
import matplotlib.pyplot as plt  # для графиков
import seaborn as sns  # для графиков
import os
import pingouin as pg  # базовые функции библиотеки
import re  # для регулярных выражений
import nltk  # для обработки естественного языка
from scipy.stats import norm  # проверка распределения
from scipy import stats  # статистический анализ
from mlxtend.evaluate import permutation_test  # перестановочный тест
from pingouin import welch_anova  # тест Уэлча
from nltk.tokenize import RegexpTokenizer  # библиотека для работы с естественным языком
from textblob import TextBlob  # для анализа эмоций
from wordcloud import WordCloud, STOPWORDS  # для облака слов

# убираю в выводе экспоненту
pd.set_option('display.float_format', lambda x: '%.3f' % x)

```

Введение

Статистическое исследование данных может быть осуществлено двумя основными способами – это либо классический статистический анализ, либо то, что известно как Exploratory Data Analysis (EDA).

Понять отличие можно по следующим схемам.

Классический анализ идет по схеме: Проблема => Данные => Модель => Анализ => Выводы. В свою очередь EDA строится чуть иначе: Проблема => Данные => Анализ => Модель => Выводы.

Отличие в том, что в классическом подходе сначала идет модель, а затем анализ, а в EDA сначала анализ данных, а затем уже модель. Другими словами, классический анализ как бы навязывает определенную модель данным, в то время как EDA пытается по данным определить, какая модель больше подходит.

Как итог, в EDA больше используют графики, например гистограммы, ящики с усами и т. п. Классический же подход больше использует тесты, проверку гипотез. Например, это ANOVA, t-tests, chi-squared tests, и F-tests.

В этой методичке я использую элементы каждого из подходов. Поэтому я провожу деление всех способов статистического анализа на:

1) количественные (тестирование гипотез, анализ распре-

деления и прочее);

2) графические (гистограммы, скаттерплоты и прочее).

Вот типичные вопросы, на которые старается ответить статистический анализ данных:

1) Какие значения являются типичными?

2) Каким распределением можно описать данные?

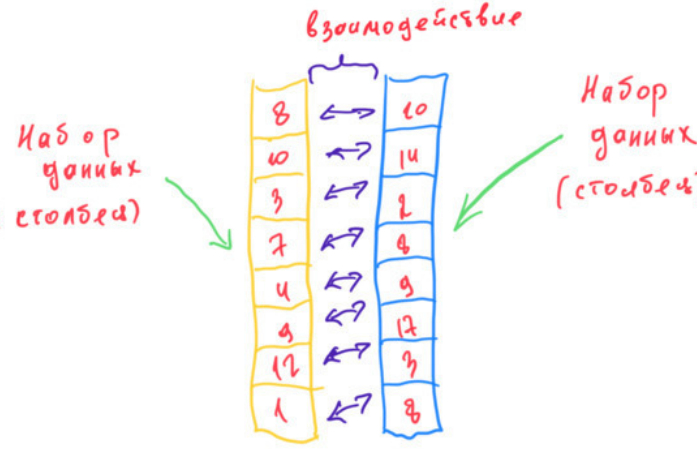
3) Как данный фактор влияет на целевой признак?

4) Какие факторы самые важные?

5) Есть ли в данных выбросы?

Важно всегда помнить, что большинство способов статанализа предполагает, что данные получены случайным образом. Если это предположение не выполняется, то результаты тестов, модели перестают быть достоверными.

Теперь еще раз, но уже больше с привязкой к данным. Статанализ (математическая статистика) работает с данными. Но что такое данные? Как правило, данные – это совокупность строк и столбцов. Пускай их будет только два. Вот такие например.



Что я могу с этим сделать? Как я могу «раскрутить», «покрутить» эти данные? Я могу, например, заинтересоваться только одним из столбцов. Какое среднее значение? А как отличаются от среднего фактические значения? Насколько вероятно появление одного из значений или нового значения? Но меня может заинтересовать и взаимодействие столбцов. Если растет значение в одном столбце, то растет ли значение в другом? Связаны ли эти столбцы? И если связаны, то насколько сильно? И вот еще что важно. Данные по Goodreads, которые я здесь использую, – это только небольшая выборка всех книг, изданных в мире. Поэтому те данные, которые видны выше на рисунке, – это тоже только выборка из генеральной совокупности. А раз так, то сто*и*т также задача

оценить по этим выборочным данным генеральную совокупность (или же наоборот, если известны характеристики генеральной совокупности).

Все это можно сделать со столбцами. И математическая статистика как раз пытается ответить на вопросы выше. Итак, математическая статистика:

1. дает математическое описание набора данных (столбца);

2. определяет вид распределения (для определения вероятности новых значений и не только);

3. дает описание того, как взаимодействуют два и более набора данных (столбцы).

Глядя на рисунок выше надо также учитывать, что, как правило, набор данных далеко не отражает всех данных. Например, в данных Goodreads приведена только небольшая выборка из всех книг. Это ставит перед математической статистикой дополнительные задачи.

В качестве учебника по математической статистике я рекомендую учебник Гмурмана «Теория вероятностей и математическая статистика» (далее – Гмурман). Вот как этот автор описывает, чем занимается матстат (стр. 187 Гмурман):

1. «оценка неизвестной вероятности события; оценка неизвестной функции распределения; оценка параметров распределения, вид которого неизвестен; оценка зависимости случайной величины от одной или нескольких случай-

ных величин и др.»;

2. «проверка статистических гипотез о виде неизвестного распределения или о величине параметров распределения, вид которого неизвестен».

Некоторые важные концепции математической статистики

> «Математическим ожиданием дискретной случайной величины называют сумму произведений всех ее возможных значений на их вероятности» (Гмурман, стр. 76).

Математическое ожидание примерно равно среднему значению. Причем «математическое ожидание приближенно равно (тем точнее, чем больше число испытаний) среднему арифметическому наблюдаемых значений случайной величины» (Гмурман, стр. 78). Поэтому – чем больше данных, тем лучше.

Понятие «центрированная величина» возникает из-за того, что такая величина получается как «разность между случайной величиной и ее математическим ожиданием» (Гмурман, стр. 87). Само же математическое ожидание принимается за центр распределения набора данных.

> «Дисперсией (рассеиянием) дискретной случайной величины называют математическое ожидание квадрата отклонения случайной величины от ее математического ожидания» (Гмурман, стр. 88).

Вот формула:

$$D(X) = M\left[X - M(X)\right]^2$$

В этой записи надо учитывать, что прописная X означает весь набор данных, например 3, 8, 19 и т. д. То есть формулу надо читать так, что из каждого из единичных значений X производится вычитание. Например, вычитаем матожидание из 3, из 8, из 19 и т. д.

Подробнее про компоненты дисперсии можно посмотреть в учебнике для инженеров [7.4.4. What are variance components?] (https://www.itl.nist.gov/div898/handbook/prc/section4/)

Совет

> «В тех случаях, когда желательно, чтобы оценка рассеяния имела размерность случайной величины, вычисляют среднее квадратическое отклонение, а не дисперсию. Например, если X выражается

в линейных метрах, то среднее квадратическое отклонение будет также выражаться в линейных метрах, а дисперсия – в квадратных метрах» (Гмурман, стр. 94).

Теперь разберу концепцию начальных и центральных моментов, очень важную для математической статистики. Для этого возьму произвольный набор данных, в котором для каждого значения известна вероятность.

X	1	2	5	100
p	0,6	0,2	0,19	0,01

Вот как считается математическое ожидание:

$$M(X) = 1 \cdot 0{,}6 + 2 \cdot 0{,}2 + 5 \cdot 0{,}19 + 100 \cdot 0{,}01$$

Еще раз, важно запомнить, что в записи M (X) вот это X

означает случайную величину, скажем измерения линейкой. Отдельное значение из этой случайной величины (верхняя строка в таблице выше) обозначается как x. Когда же есть запись с X, то имеются ввиду все значения x.

Итак, теперь возведу в квадрат случайную величину.

X^2	1	4	25	10000
p	0,6	0,2	0,19	0,01

Вероятность не изменилась. Это можно понять так. Возведением в квадрат изменяется масштаб, но не вероятность. Каким будет математическое ожидание?

$$M(x^2) = 1 \cdot 0,6 + 4 \cdot 0,2 + 25 \cdot 0,19 + 10000 \cdot 0,01 = 106,15$$

Какой вывод я могу сделать? Второе математическое ожидание гораздо больше первого. Почему? Потому что в пер-

вом случае я умножал вероятность 0,01 на 100, а во втором ту же вероятность 0,01 я умножил уже на 10000. Это позволило «лучше учесть влияние на математическое ожидание того возможного значения, которое велико и имеет малую вероятность» (Гмурман, 98). В зависимости от количества подобных величин, того, насколько они «маленькие», может потребоваться возведение не только в квадрат, но и в более высокие степени.

Начальным моментом порядка k называют математическое ожидание случайной величины, возведенной в степень (k, это может быть и степень k=1). Центральным моментом порядка k называют математическое ожидание степени разности между случайной величиной и математическим ожиданием случайной величины.

[Не так строго понять это можно следующим образом. Сначала я нахожу среднее значение набора данных (это будет математическим ожиданием). Затем я вычитаю из каждого значения набора данных это среднее значение. У меня получится новый набор данных. Теперь я могу найти среднее этого нового набора данных (это также будет математическим ожиданием, но для нового набора данных).]

Применение закона больших чисел разъясняется в главе 9 Гмурмана. Здесь я не буду останавливаться на этом подробнее.

Выборкой «называют совокупность случайно отобранных объектов» (Гмурман, стр. 188). Выборка осуществляется

по специальным правилам. Подробнее об этом можно узнать здесь (https://www.itl.nist.gov/div898/handbook/ppc/section3/ppc333.htm), а также в [7.2.4.2. Sample sizes required] (https://www.itl.nist.gov/div898/handbook/prc/section2/prc242.htm).

Генеральной совокупностью «называют совокупность объектов, из которых производится выборка» (там же).

В теории вероятностей

> «под распределением понимают соответствие между возможными значениями случайной величины и их вероятностями, а в математической статистике – соответствие между наблюдаемыми вариантами и их частотами или относительными частотами» (Гмурман, стр. 192).

В случае, который разбираю я на данных Goodreads, имеющиеся у меня данные – это выборка, по которой я хочу оценить генеральную совокупность – все книги на сайте Goodreads.

Вот как это работает.

> «Пусть требуется изучить количественный признак генеральной совокупности. Допустим, что из теоретических соображений удалось установить, какое именно распределение имеет признак. Естественно возникает задача оценки параметров, которыми определяется это распределение. Например, если наперед известно, что изучаемый признак распределен в генеральной совокупности нормально, то необходимо оценить (приближенно найти)

математическое ожидание и среднее квадратическое отклонение, так как эти два параметра полностью определяют нормальное распределение; если же есть основания считать, что признак имеет, например, распределение Пуассона, то необходимо оценить параметр лямбда, которым это распределение определяется» (Гмурман, стр. 197).

Например, генеральная совокупность – все книги на Goodreads. Параметр, который нас интересует, – это количество страниц. Количество страниц в каждой книге Goodreads – это и есть количественный признак генеральной совокупности.

Итак, есть оцениваемые параметры, а есть статистические оценки таких параметров. Такие оценки должны соответствовать определенным требованиям. Буду делать выборки из генеральной совокупности книг. Для каждой выборки оценю параметр, например среднее значение страниц в книге. Каждая такая выборка даст свое значение, совокупность таких значений и будет набором данных, у которого также может быть математическое ожидание (среднее). Отсюда появляется понятие несмещенной оценки.

«Несмещенной называют статистическую оценку, математическое ожидание которой равно оцениваемому параметру при любом объеме выборки» (Гмурман стр. 198).

Соответственно, если оценка не соответствует указанным

свойствам, то она является смещенной.

Вот еще пояснение из учебника «Теория и методы эконометрики», авторов Дэвидсона, Мак-Кинона (далее – Дэвидсон), который я также рекомендую.

> «На интуитивном уровне это означает, что если мы станем использовать метод оценивания, дающий несмещенные оценки для расчета оценок по очень большому числу выборок, то среднее значение получаемых с его помощью оценок будет приближаться к оцениваемой величине. При прочих равных статистических свойствах двух методов оценивания несмещенный метод всегда предпочтительнее смещенного».

Однако, даже если оценка является несмещенной, все-таки дисперсия в наборе данных, на основе которых посчитана оценка, может быть большой. Поэтому еще одним требованием к оценке является эффективность.

> «Эффективной называют статистическую оценку, которая (при заданном объеме выборки n) имеет наименьшую возможную дисперсию» (Гмурман, стр. 199).

Кроме того, если количество объектов в выборке стремится к бесконечности, то устанавливают требование о состоятельности.

> «Состоятельной называют статистическую оценку, которая при [стремлении количества объектов

к бесконечности] стремится по вероятности к оцениваемому параметру».

Про доверительные интервалы см. параграфы 14—16 гл. 16 Гмурмана.

Отдельные важные концепции математической статистики можно изучить по следующим ссылкам:

1. Про виды распределений – гл. 4 Дэвидсона. Хорошая галерея графиков с видами распределений находится здесь (https://www.itl.nist.gov/div898/handbook/eda/section3/eda366.htm), там же можно найти компактное описание распределений. Еще одно описание можно найти в том же учебнике [8.1.6. What are the basic lifetime distribution models used for non-repairable populations?] (https://www.itl.nist.gov/div898/handbook/apr/section1/apr16.htm). Почему важно правильно определить вид распределения? Потому что от этого зависит как применение тестов, так и проверка гипотез. Распределения также применяются для определения доверительных интервалов.

Подробнее остановлюсь на нормальном распределении.

Нормальное распределение определяется двумя параметрами: математическим ожиданием (**a**) и средним квадратическим отклонением. Про график нормального распределения, который выглядит как колокол и близок к приведенному выше, нужно помнить три правила:

1) «Изменение величины параметра a (математического

ожидания) не изменяет форму нормальной кривой, а приводит лишь к ее сдвигу вдоль оси Ох: вправо, если **а** возрастает, и влево, если **а** убывает» (Гмурман, стр. 131).

2) «С возрастанием параметра средего квадратического отклонения максимальная ордината нормальной кривой убывает, а сама кривая становится более пологой, то есть сжимается к оси Ох; при убывании – нормальная кривая становится более „островершинной“ и растягивается в положительном направлении оси Оу» (там же).

3) Площадь под кривой всегда остается равной 1.

[Математическое ожидание показывает среднее значение в наборе. Поэтому, если такое среднее значение «двигается», то двигается и график, который построен ведь вокруг этого среднего значения. Среднее отклонение показывает разброс отдельных значений данных вокруг среднего. Если такой разброс увеличивается или уменьшается, то соответственно изменяется и график.]

В связи с нормальным распределением есть центральная предельная теорема (теорема Ляпунова). Вот ее формулировка:

> «Если случайная величина X представляет собой сумму очень большого числа взаимно независимых случайных величин, влияние каждой из которых на всю сумму ничтожно мало, то X имеет распределение, близкое к нормальному» (Гмурман, стр. 135).

Таким образом, все распределения оцениваются приме-

нительно к нормальному. Поэтому нужны инструменты, которые бы показывали, что данное распределение отличается и насколько отличается от нормального. Для этого используют показатели эксцесса и ассиметрии. Для нормального распределения ассиметрия и эксцесс равны нулю. Если у данного набора эти значения сильно больше 0, то его распределение тем сильнее отличается от нормального, и наоборот. Ниже я покажу также иные способы определения нормальности распределения.

2. Проклятие размерности – гл. 2 «Основы статистического обучения» Тревор Хасти, Роберт Тибширани, Джером Фридман. Важность этой проблемы можно понять из следующей цитаты тех же авторов:

> «С увеличением размерности сложность функций многих переменных может расти экспоненциально, и если мы хотим иметь возможность оценивать такие функции с той же точностью, что в пространствах малой размерности, то нам необходимо, чтобы размер нашего обучающего множества также рос экспоненциально» (стр. 24 английского издания).

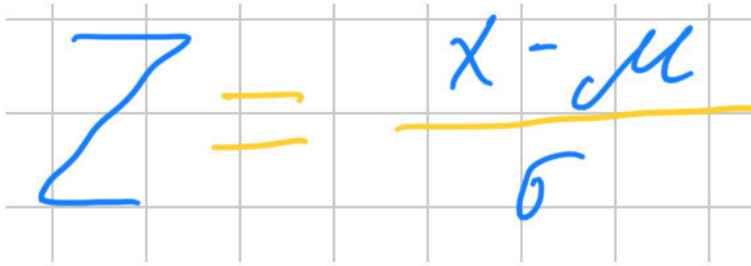
Здесь же объясняется разложение среднеквадратической ошибки (MSE) на дисперсию и смещение. Привожу только вывод формулы для примера:

$$\begin{aligned}\mathrm{MSE}(x_0) &= \mathrm{E}_{\mathcal{T}}[f(x_0) - \hat{y}_0]^2 \\ &= \mathrm{E}_{\mathcal{T}}[\hat{y}_0 - \mathrm{E}_{\mathcal{T}}(\hat{y}_0)]^2 + [\mathrm{E}_{\mathcal{T}}(\hat{y}_0) - f(x_0)]^2 \\ &= \mathrm{Var}_{\mathcal{T}}(\hat{y}_0) + \mathrm{Bias}^2(\hat{y}_0).\end{aligned}$$

В учебнике Машинное обучение указывается следующее.

> «В контексте моделей МО [машинного обучения] дисперсия измеряет постоянство (либо изменчивость) прогноза модели для классификации отдельного образца при многократном обучении модели, например, на разных подмножествах обучающего набора данных. Мы можем сказать, что модель чувствительна к случайности обучающих данных. Напротив, смещение измеряет, насколько далеко прогнозы находятся от корректных значений в целом при многократном обучении модели на разных обучающих наборах данных; смещение представляет собой меру систематической ошибки, которая не является результатом случайности».

3. Z-оценка часто используется, например для определения выбросов. Вот формула для расчета:



В этой формуле: x – это единичное значение из набора данных; мю – среднее набора данных; сигма – стандартное отклонение.

4. Доверительные интервалы, см. подробнее в [1.3.5.2. Confidence Limits for the Mean] (https://www.itl.nist.gov/div898/handbook/eda/section3/eda352.htm)

5. Дисперсия, ковариация, корреляция. Разница между дисперсией, ковариацией и корреляцией:

1) дисперсия – это мера изменчивости конкретного значения от среднего значения по всему набору данных;

2) ковариация – это мера взаимосвязи между изменчивостью двух переменных. Ковариация зависит от масштаба, поскольку она не стандартизирована;

3) корреляция – это связь между изменчивостью двух переменных. Корреляция стандартизирована, что делает ее не зависящей от масштаба.

Справочное руководство [Engineering statistics handbook]

(https://www.itl.nist.gov/div898/handbook/eda/eda.htm) содержит пример схемы анализа данных:

1. Посчитать базовые статистики:

a) среднее;

b) стандартное отклонение. При этом надо помнить следующие эмпирические правила. Приблизительно 60—78% данных находятся в пределах одного стандартного отклонения от среднего. Приблизительно 90—98% данных находятся в пределах двух стандартных отклонений от среднего. Более 99% данных находятся в пределах трех стандартных отклонений от среднего;

c) коэффициент автокорреляции для проверки данных на случайность;

d) коэффициенты корреляции, коэффициенты, показывающие, что распределение является нормальным, например Wilk-Shapiro test.

2. Построить график для нормального распределения.

3. Линейная аппроксимация данных в зависимости от времени для оценки дрейфа (тест на фиксированное положение).

4. Тест Барлетта для дисперсии.

5. Критерий Anderson-Darling для нормального распределения.

6. Теста Граббса для определения выбросов.

Можно ознакомиться с примером анализа по указанной схеме [1.4.2.1.3. Quantitative Output and

Interpretation] (https://www.itl.nist.gov/div898/handbook/eda/section4/eda4213.htm)

Загрузка и описание данных

```
# скопирую путь к файлу из проводника Windows, но заменю \ на /
# дополнительно исправлю тип данных для отдельных признаков
data = pd.read_csv(os.getcwd() + '\\gd_clean_data.csv', dtype={'decade':'object','quarter':'object', 'tra_co':'object'
})

# сделаю копию данных
db = data.copy()
db.head(3)
```

	title	average_rating	language_code	num_pages	ratings_count	text_reviews_count	decade
0	Harry Potter and the Half-Blood Prince (Harry ...	4.570	eng	652	2095690	27591	2000
1	Harry Potter and the Order of the Phoenix (Har...	4.490	eng	870	2153167	29221	2000
2	Harry Potter and the Chamber of Secrets (Harry...	4.420	eng	352	6333	244	2000

```
# общее описание данных
db.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10838 entries, 0 to 10837
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   title              10838 non-null  object
 1   average_rating     10838 non-null  float64
 2   language_code      10838 non-null  object
 3   num_pages          10838 non-null  int64
 4   ratings_count      10838 non-null  int64
 5   text_reviews_count 10838 non-null  int64
 6   decade             10838 non-null  object
 7   quarter            10838 non-null  object
 8   tra_co             10838 non-null  object
 9   multivolume        10838 non-null  object
dtypes: float64(1), int64(3), object(6)
memory usage: 846.8+ KB
```

Теперь мне надо определить, что я хочу узнать из данных. Специальных целей передо мной никто не ставил, поэтому

определю их самостоятельно. Что интересного могут рассказать данные? Здесь же я сразу укажу, какими методами буду решать эти задачи. Надо помнить, что не всегда можно заранее знать, какой метод подойдет. Например, мне нужно сначала проверить распределение на нормальность, чтобы применить корреляцию. Поэтому в этот список можно вносить изменения по ходу анализа.

Как указывалось ранее, я могу разделить статистическое обследование на изучение набора данных и изучение отношений между наборами данных. С учетом этого и разделю задачи.

Изучение каждой группы данных отдельно:

* Характеристики центрального положения для количественных признаков. Метод describe;

* Характеристики категориальных данных. Тот же describe;

* Какой тип распределения у средних оценок для книг, для количества страниц в книгах? (здесь не рассматривается).

Изучение отношений между группами данных:

* Как распределены книги по десятилетиям? Использую график;

* Какие книги получили высокие оценки? Использую график;

* Как распределены книги по кварталам? Использую гра-

фик;

* Какие книги чаще всего издавались в рамках набора данных? Использую график;

* Если у книги есть соавтор или переводчик, как это влияет на оценку? Использую дисперсионный анализ;

* От каких признаков зависит оценка книги? Использую корреляцию, дисперсионный анализ;

* Какие слова чаще всего используются в названии книги? Использую NLTK.

Начну с характеристик центрального положения.

Я буду для простоты писать «статистика» вместо «математическая статистика», «статистический анализ», хотя строго говоря это не одно и то же.

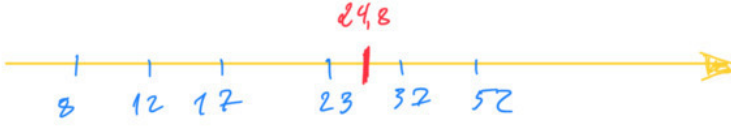
```
1 db.describe()
```

	average_rating	num_pages	ratings_count	text_reviews_count
count	10838.000	10838.000	10838.000	10838.000
mean	3.942	344.175	18347.883	553.075
std	0.295	238.556	113866.010	2599.330
min	1.000	11.000	0.000	0.000
25%	3.770	203.000	117.000	10.000
50%	3.960	304.000	803.500	49.000
75%	4.130	417.000	5226.750	248.000
max	5.000	6576.000	4597666.000	94265.000

Уже в этой таблице можно видеть важнейшие концепции статистики. Выше я писал, что статистика должна описывать наборы данных и их взамиодействие. Здесь мы видим опи-

сание именно наборов данных, взаимодействие будет позже.

В таблице дается описание для четерых численных признаков: average_rating, num_pages, ratings_count, text_reviews_count. Остальные признаки являются категориальными и в эту таблицу не попали, но ниже я также рассмотрю и эти признаки. Пока продолжу изучать таблицу. Для каждого признака, например для text_reviews_count, приведен ряд характеристик (метрик): count, mean и т. д. Об этих характеристиках можно говорить как о характеристиках центрального положения. Откуда пошло такое выражение? Это показано на рисунке ниже



Среднее значение 24,8 есть характеристика центрального положения, так как фактические данные (8,12…52) расположены вокруг этого среднего. Отсюда же видно, например, что можно посчитать расстояние от центра до каждого значения, что приводит к дисперсии и стандартному отклонению.

В чем смысл таких характеристик? У меня есть набор данных. Я хочу его как-то охарактеризовать. Зачем? Во-пер-

вых, чтобы лучше понять объект, который описывается этими данными. Например, про среднюю оценку я теперь знаю, что она у книг составляет 3.9. Во-вторых, чтобы уметь предсказывать будущие события. Например, я хочу знать, а какую оценку поставят новой книге. При прочих равных можно считать, что эта оценка будет близка к среднему значению. Но так как точно сказать этого нельзя, то меня интересует, в каком диапазоне может быть эта оценка, здесь помогает std. Минимум и максимум определяют, в каких границах расположены оценки. Благодаря этому я достоверно знаю, что оценка не может быть меньше 1 и не может быть больше 5. А к чему же все эти проценты: 25%, 50%, 75%? Эти проценты показывают следующее: 25% оценок ниже чем 3.77, 50% оценок ниже чем 3.96 и т. д. Это условно можно представить как вероятность: вероятность того, что оценка книги будет 3.77 составляет 25%.

Так я изучаю характеристики каждого набора данных. Замечаю, что в num_pages, ratings_count, text_reviews_count есть странности. Так, например, среднее в num_pages составляет 344, но максимальное значение 6576. Говоря иначе, в среднем в одной книге 344 страницы, но есть книга, у которой 6576 страниц. Это может свидетельствовать о выбросах в данных. Непосредственно о выбросах я расскажу позже, но уже сейчас надо это учитывать. Если я предполагаю, что в моем наборе данных есть выбросы, то я могу использовать робастные, то есть устойчивые к выбросам методы оценки

среднего. Я могу найти такую оценку с помощью библиотеки papanda.

```
# робастная оценка среднего
from papanda import trimmed_mean
av = np.array(db['average_rating'])
trimmed_mean.trim_mean(av)
```

```
   per_outliers  trim_mean
0         0.050      3.943
1         0.100      3.946
2         0.150      3.946
3         0.180      3.947
4         0.200      3.946

Mean for whole data: 3.942158147259642

Median for whole data: 3.96
```

```
# еще одна робастная оценка, медианное абсолютное отклонение
db['average_rating'].mad()
```

```
0.224434581719725
```

```
# книга с самым большим количеством отзывов
db[db['ratings_count'] == 4597666]
```

	title	average_rating	language_code	num_pages	ratings_count	text_reviews_count	decade	quarter	tra_co	multivolume
10077	Twilight (Twilight #1)	3.590	eng	501	4597666	94265	2000	3	0	1

```
# опишу категориальные признаки
db[['title', 'language_code', 'decade', 'quarter', 'tra_co', 'multivolume']].describe()
```

	title	language_code	decade	quarter	tra_co	multivolume
count	10838	10838	10838	10838	10838	10838
unique	10149	26	8	4	2	2
top	The Odyssey	eng	2000	3	0	0
freq	8	8669	7332	2851	6492	8147

Вижу, что в данных 10149 уникальных названий книг

из 10838.

Чаще всего встречается The Iliad, 8 раз. Однако надо учитывать, что есть еще несколько книг, которые в изданы 8 раз. Например, Анна Каренина. Поэтому The Iliad можно считать случайным.

26 различных языков, самый частый eng, 8669.

Декад всего 13, самая частая – это 2000, на которую приходится 7332.

Кварталов 4, самый частый 3, на него приходится 2851 книга.

Категорий «с соавтором, переводчиком» и без две: либо переводчик или соавтор есть, либо их нет. Чаще всего их нет, таких случаев 6492.

Аналогично, либо книга является частью многотомного издания и тогда в колонке **multivolume** стоит 1, либо не является частью такого издания и тогда получается 0. Вижу, что в наборе, как правило, не многотомные издания (их 8147).

Опять-таки, а зачем мне эти знания? Во-первых, чтобы лучше понимать природу того объекта, который исследуется. В каком году книге чаще выходили? А в каком квартале? С каким названием? Все это может помочь понять не только, *что* выпускают издательства, но и вкусы читателей. Во-вторых, категориальные признаки позволяют разбить объекты на группы, скажем можно разбить книги по языку, и уже внутри каждой группы посмотреть оценки, количество стра-

ниц. Эта возможность ставит дополнительные задачи. Например, категории не должны быть слишком маленькими, так как невозможно будет доверять результатам. Например, если у меня по какому-то языку только две книги, то определить среднюю я смогу, но эта средняя может быть слишком далека от истинного значения. Надо помнить, что в статистике всегда руководствуются законом больших чисел.

Ниже я приведу несколько способов агрегирования (группировки) данных.

```
1  # самые частые по названию книги
2  db['title'].value_counts()[0:5]
```

```
The Odyssey                  8
The Iliad                    8
Anna Karenina                8
The Picture of Dorian Gray   7
The Brothers Karamazov       7
Name: title, dtype: int64
```

```python
# книги с самым высоким рейтингом
db.groupby('title')['average_rating'].mean().sort_values(ascending=False)[0:2]
```

```
title
The New Big Book of America                                5.000
Bulgakov's the Master and Margarita: The Text as a Cipher  5.000
Name: average_rating, dtype: float64
```

```python
# какие книги получают больше отзывов
db.groupby('title')['text_reviews_count'].mean().sort_values(ascending=False)[0:2]
```

```
title
Twilight (Twilight  #1)   94265.000
The Book Thief            86881.000
Name: text_reviews_count, dtype: float64
```

Визуальный анализ

Визуальный анализ используют для того, чтобы «схватить» какие-то гипотезы на данных. Визуально человеку проще это сделать. Однако надо помнить, что размещение слишком большого количества данных на графике уменьшает этот эффект простоты. Например, на одном графике можно точками отразить распределение по двум признакам, сюда же добавить третий признак цветом точек, четвертый – размером точек, пятый – их формой. Но такой график будет крайне сложно прочитать, а значит и смысл в визуализации теряется.

Здесь я хочу установить следующее:

1) вид распределения целевого признака – средняя оценка;

2) наличие выбросов по количественным признакам;

3) наличие взаимосвязи между признаками;

4) соотношение категориальных данных там, где это уместно.

Ниже рассмотрю графики распределений.

В учебнике для инженеров выделяется четыре главных типа графиков:

1. график последовательного выполнения;

2. график задержки;

3. гистограмма;

4. график нормального распределения.

Подробнее прочитать о каждом типе графиков можно здесь (https://www.itl.nist.gov/div898/handbook/eda/section3/4plot.htm). Как выбрать необходимый тип графика можно понять из рисунка:

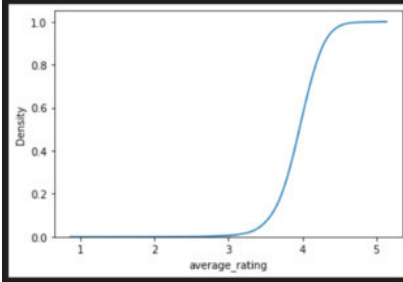
```
x_y_train['text_reviews_count_groups'] = pd.qcut(train_enc['len_title'], 5).astype('object')

biv = pd.crosstab(x_y_train['text_reviews_count_groups'], x_y_train['rating_groups'])
print(biv)

feature,target = 'text_reviews_count_groups','rating_groups'
train_woe_iv = (pd.crosstab(x_y_train[feature],x_y_train[target],
                            normalize='columns')
                .assign(woe=lambda dfx: np.log(dfx[1] / dfx[0]))
                .assign(iv=lambda dfx: np.sum(dfx['woe']*
                                              (dfx[1]-dfx[0]))))
print(train_woe_iv)

print(train_woe_iv['iv'].sum())
```

```
rating_groups               0    1
text_reviews_count_groups
(1.999, 13.0]             912  608
(13.0, 19.0]              953  732
(19.0, 27.0]              740  708
(27.0, 45.0]              639  808
(45.0, 209.0]             629  857
rating_groups                0         1       woe        iv
text_reviews_count_groups
(1.999, 13.0]             0.235476  0.163749 -0.363276  0.075513
(13.0, 19.0]              0.246062  0.197145 -0.221645  0.075513
(19.0, 27.0]              0.191066  0.190681 -0.002017  0.075513
(27.0, 45.0]              0.164988  0.217614  0.276847  0.075513
(45.0, 209.0]             0.162406  0.230811  0.351496  0.075513
0.37756443346937774
```

Ниже я рассмотрю только важные для моих задач графики.

```
# график функции распределения
# cumulative distribution function
sns.kdeplot(data=db, x='average_rating', cumulative=True)
```

```
<AxesSubplot:xlabel='average_rating', ylabel='Density'>
```



Сейчас я разберу, что такое функция распределения. Это имеет важное значение для понимания видов распределения и их смысла. Все начинается с функции распределения. Вот формальное определение:

> «Функцией распределения называют функцию F (x), определяющую вероятность того, что случайная величина X в результате испытания примет значение, меньшее x» (Гмурман, ст. 111).

А вот геометрический образ функции распределения:

> «F (x) есть вероятность того, что случайная величина примет значение, которое изображается на числовой оси точкой, лежащей левее точки x» (Гмурман).

Выше показан график функции распределения. Как его

правильно читать? Проведу из произвольной точки на оси x перпендикуляр. Точка пересечения перпендикуляра с графиком даст значение y – вероятность того, что моя произвольная точка примет значение равное или меньшее x. Например, беру оценку 4. Вижу, что вероятность получить такую оценку между 0,5 и 0,6, примерно 55%.

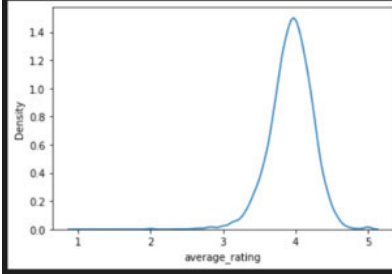
Раз у меня есть непрерывный график функции, я могу посчитать производные. Поэтому случайную величину можно представить и через т. н. плотность распределения (плотность вероятности).

> «Плотностью распределения вероятностей непрерывной случайной величины X называют функцию f (x) – первую производную от функции распределения F (x): f (x) = F» (x)» (Гмурман, стр. 116).

Пример такой функции ниже.

```
1  sns.kdeplot(data=db, x='average_rating')
```

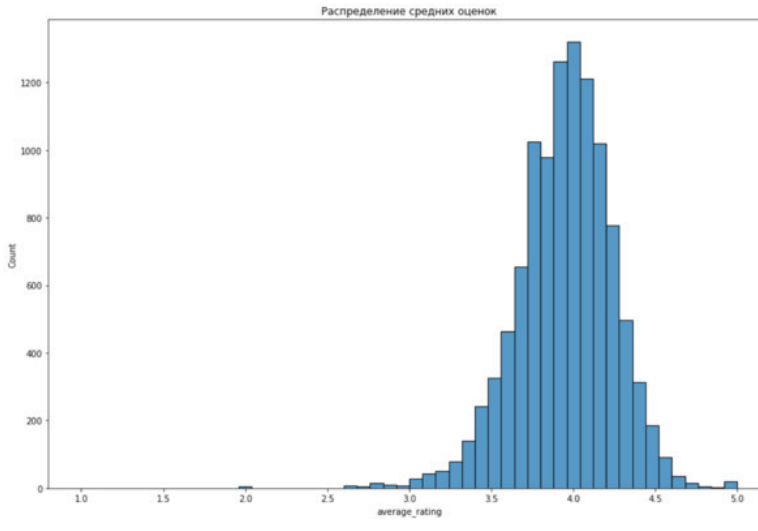
```
<AxesSubplot:xlabel='average_rating', ylabel='Density'>
```



Законами распределения называют различные виды плотности распределения. Например, это может быть равномерное, нормальное, показательное распределение. Чаще всего используется нормальное распределение.

Про графики PDF, CDF, PPF подробнее здесь [1.3.6.2. Related Distributions] (https://www.itl.nist.gov/div898/handbook/eda/section3/eda362.htm)

```
fig, ax = plt.subplots(figsize=(15, 10))
sns.histplot(db['average_rating'], bins=50)
plt.title('Распределение средних оценок')
```



Гистограмма позволяет сделать предположение о виде распределения данных. Знать вид распределения данных необходимо по нескольким причинам. Во-первых, это позволяет делать предсказания о вероятности того или иного события. Во-вторых, для проведения статистических тестов, определения некоторых метрик требуется распределение определенного вида. Как правило, распределение должно быть нормальным. Если распределение нормальным не является, то данные можно привести к нормальному распределению или можно использовать специальные тесты, метрики. Поэтому важно ответить на вопрос: распределены ли данные нормально? Если нет, то нужно установить вид рас-

пределения.

Гистограмма показывает, что средние оценки распределены практически нормально. Интересно, что в интервалах 2.5—3.0 и 4.8—5.0 видны небольшие подъемы линии. При нормальном распределении этого быть не должно. Это означает, что оценки в указанных интервалах имеют вероятность большую, чем это предсказывает нормальное распределение. Дополнительная проверка на нормальность распределения с помощью статистических методов будет показана ниже.

Гистограмму можно построить разными способами. В случае выше ширина столбика показывает частичный интервал, а высота – количество значений в этом интервале. Возможно построить гистограмму, где высота столбика будет показывать плотность. Подробнее см. в официальной документации функции (https://seaborn.pydata.org/generated/seaborn.histplot.html).

Про интерпретацию гистограммы можно также прочитать в [учебнике для инженеров] (https://www.itl.nist.gov/div898/handbook/eda/section3/histogra.htm). Там же можно обнаружить различные типы гистограмм (как нормальную, так и, например, бимодальную), а также дополнительные статистические методы для определения типа распределения в зависимости от типа гистограммы. Гистограмма показывает:

1) центральную характеристику данных;

2) масштаб данных;

3) скошенность;

4) наличие выбросов;

5) наличие нескольких мод в данных.

Трансформация данных к нормальному распределению объясняется в 6.5.2. What to do when data are non-normal (https://www.itl.nist.gov/div898/handbook/pmc/section5/pmc52.htm)

Выбросы

В учебнике для инженеров дано следующее определение выбросов:

> «Выбросы – это точки данных, которые получены не из того же распределения, из которого получена основная масса данных».

То есть выброс – это такое значение, которое пришло не из того распределения, из которого пришли основные данные. В этом смысл того, чтобы определить распределение для большинства данных, а затем уже выброс. Редкие данные возможны и в границах распределения для основных данных, но вот выброс выходит вообще за границы распределения, то есть например за пределы колокола в нормальном распределении. В этом смысл того, что сначала надо найти отличающиеся от других данные, а затем проверить их на влиятельность.

Вот рекомендации по обработке выбросов из учебника для инженеров:

1. К каждому выбросу необходимо относиться серьезно. Не рекомендуется автоматически удалять выбросы. Наличие выбросов может быть не просто ошибкой в данных, выбросы могут сообщать важную информацию о данных. Поэтому надо постараться объяснить, чем вызваны выбросы в данных.

2. Если гистограмма показывает наличие выбросов, то ре-

комендуется следующее:

1) применить ящик с усами, который лучше гистограммы показывает наличие и количество выбросов;

2) применить Grubbs' Test или иные тесты для обнаружения выбросов.

Рекомендуемые тесты на выбросы:

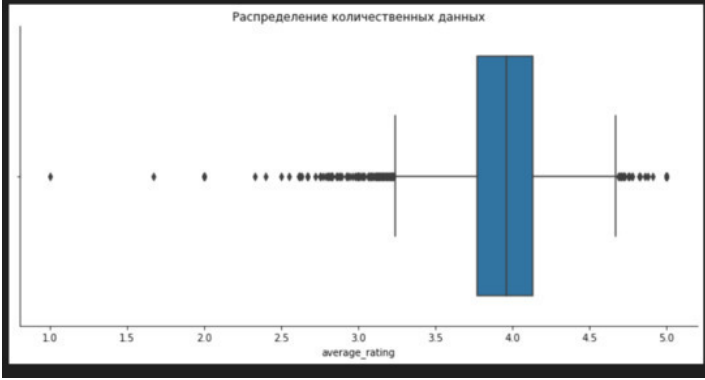
1) Grubbs' Test – если тест на единичный выброс;

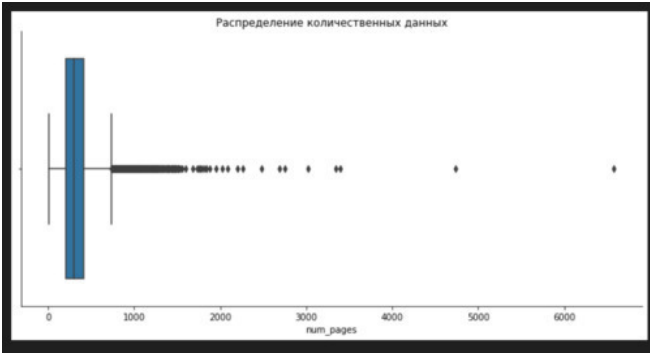
2) Tietjen-Moore Test – в случае, если в данных предполагается более одного выброса. Необходимо заранее знать точное количество выбросов.

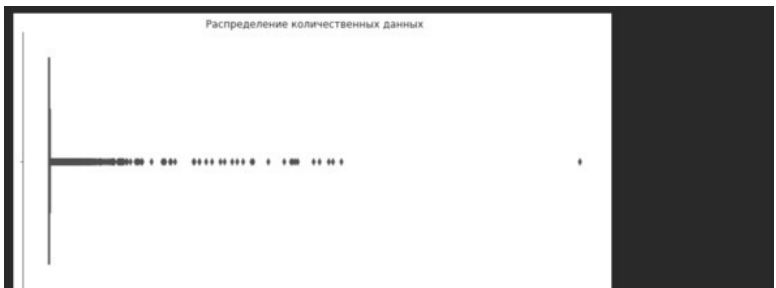
3) Generalized Extreme Studentized Deviate (ESD) Test – также, если в данных более одного выброса. Необходимо знать только верхнюю границу ожидаемого числа выбросов. Рекомендуется, когда точное количество выбросов неизвестно.

Как правило, при обнаружении выбросов исходят из того, что данные распределены нормально. Если это не так, то можно привести данные к нормальному распределению.

```python
# стандартная проверка на выбросы
for i in ['average_rating', 'num_pages', 'ratings_count', 'text_reviews_count']:
    sns.catplot(kind='box', data=db, aspect=2, x=i)
    plt.title('Распределение количественных данных')
```







Это – ящики с усами. Их придумал отец-основатель анализа данных Тьюки. В середине прямоугольников показана медиана. Левый край прямоугольника – это 25%-квартиль, правый край – 75%. Усы – это межквартильный размах. За пределами усов – выбросы.

Интересный вопрос про ящик – это почему он то сжима-

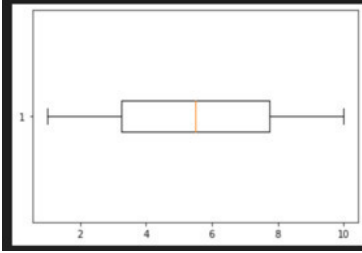
ется, то растягивается. Почему медиана скачет от левой стороны к правой? Ведь это медиана, она должна быть посередине. Все верно, почти. Разберу это на примере.

Важное терминологическое замечание .

> «Квантиль 0,5 называют медианой. Для a = 0,25, 0,5 и 0,75 соответствующие квантили называются квартилями, a = 0,2, 0,4, 0,6, 0,8 они называются квинтилями».

```
1  # создаю вектор данных
2  example = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
3
4  # и строю боксплот
5  plt.boxplot(example, vert=False)
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x2c55761cfa0>,
  <matplotlib.lines.Line2D at 0x2c557762370>],
 'caps': [<matplotlib.lines.Line2D at 0x2c557762700>,
  <matplotlib.lines.Line2D at 0x2c557762a90>],
 'boxes': [<matplotlib.lines.Line2D at 0x2c55761cc10>],
 'medians': [<matplotlib.lines.Line2D at 0x2c557762e20>],
 'fliers': [<matplotlib.lines.Line2D at 0x2c5577701f0>],
 'means': []}
```

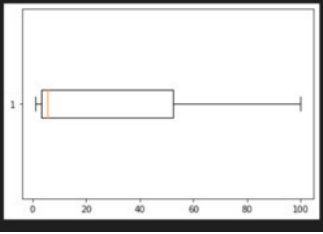


```
# найду медиану
np.median(example)
```

```
5.5
```

```
# создадю еще один вектор данных
example1 = np.array([1, 2, 3, 4, 5, 6, 51, 53, 54, 100])

# и опять строю боксплот
plt.boxplot(example1, vert=False)
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x2c557d572e0>,
  <matplotlib.lines.Line2D at 0x2c557d57670>],
 'caps': [<matplotlib.lines.Line2D at 0x2c557d57a00>,
  <matplotlib.lines.Line2D at 0x2c557d57d90>],
 'boxes': [<matplotlib.lines.Line2D at 0x2c557d4af70>],
 'medians': [<matplotlib.lines.Line2D at 0x2c557d64160>],
 'fliers': [<matplotlib.lines.Line2D at 0x2c557d644f0>],
 'means': []}
```



```
# найду медиану на новых данных
np.median(example1)
```

```
5.5
```

```
# ниже подробно разберу, как образуются усы,
# а заодно и как писать функции
q1, q3 = np.percentile(example1, [25, 75])

# q1 левый (нижний) квартиль, q3 правый (верхний) квартиль
iqr = q3 - q1
iqr

# левый (нижний) ус
lower_bound = q1 - (iqr*1.5)
lower_bound

# правый (верхний) ус
upper_bound = q3 + (iqr*1.5)
upper_bound
```

```
126.375
```

Что здесь произошло? Я создал второй вектор, где заменил 7, 8, 9, 10 на 51, 53, 54, 100. Количество значений не изменилось. Не изменилась и сама медиана – она осталась 5.5. Однако линия медианы «прижалась» к левой стороне ящика. Почему?

Конец ознакомительного фрагмента.