ДИСЦИПЛИНА	**Системы синхронизированного планирования ресурсов предприятия**
	(полное наименование дисциплины без сокращений)
ИНСТИТУТ	**информационных технологий**
КАФЕДРА	**корпоративных информационных систем**
	полное наименование кафедры
ВИД УЧЕБНОГО МАТЕРИАЛА	**Учебно-методическое пособие**
	(в соответствии с пп.1-11)
ПРЕПОДАВАТЕЛЬ	**Демидова Лилия Анатольевна**
	(фамилия, имя, отчество)
СЕМЕСТР	**2 семестр (весенний), 2024 – 2025 учебный год**
	(семестр обучения, учебный год)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
МИРЭА – РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ. PYTHON ЧАСТЬ 1

Москва – 2022

УДК 004
ББК 32.97
Д 30
Демидова Л.А. Разведочный анализ данных. Python. Часть 1 [Электронный ресурс]: Учебное-методическое пособие / Демидова Л.А. — М.: МИРЭА – Российский технологический университет, 2022. — 1 электрон. опт. диск (CD-ROM)

Аннотация учебного-методического пособия.

В учебно-методическом пособии рассматриваются аспекты выполнения разведочного анализа количественных данных средствами языка Python. Предлагается перечень заданий, при выполнении которых применяются различные инструменты разведочного анализа данных. Предназначено для магистрантов, изучающих дисциплину «Интеллектуальные системы и технологии» по направлению 09.04.04 Программная инженерия.

Учебно-методическое пособие издается в авторской редакции.

Автор: Демидова Лилия Анатольевна

Рецензенты:

Головин С.А., д.т.н., профессор, заведующий кафедрой МОСИТ ИИТ РТУ МИРЭА.

Бочаров Н.А.., начальник отдела 3.4.5 «Защита информации» ПАО «ИНЭУМ им. И.С. Брука», к.т.н.

Минимальные системные требования:
Наличие операционной системы Windows, поддерживаемой производителем.
Наличие свободного места в оперативной памяти не менее 128 Мб.
Наличие свободного места в памяти хранения (на жестком диске) не менее 30 Мб.
Наличие интерфейса ввода информации.
Дополнительные программные средства: программа для чтения pdf-файлов (Adobe Reader).
Подписано к использованию по решению Редакционно-издательского совета
МИРЭА – Российского технологического университета от ___ _____ 2022 г.
Объем ___ Мб
Тираж 10

Оглавление

ВВЕДЕНИЕ

Учебно-методическое пособие излагает принципы разведочного анализа данных, уделяя особое внимание анализу количественных данных.

Рассматриваются:

– метрики центральной тенденции и вариабельности количественных данных;

– аспекты анализа распределения количественных данных;

– инструментарий анализа корреляции количественных данных.

Особое внимание уделяется применению инструментов визуализации, позволяющих выявить скрытую в количественных данных информацию.

Приводятся примеры, демонстрирующие принципы применения инструментария разведочного анализа к количественным данным.

Читателю предлагается перечень заданий: каждое из них содержит 2 варианта набора данных из репозитория данных для машинного обучения, для которых необходимо осуществить разведочный анализ с использованием предлагаемых инструментов разведочного анализа, реализованных в программных библиотеках языка Python. При этом в одном из вариантов известны истинные метки классов для объектов, а в другом – нет.

Читатель может получить дополнительные сведения по теоретическим вопросам, а также по аспектам программной реализации для рассматриваемых инструментов разведочного анализа данных в источниках, указанных в списке литературы.

РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ

1. Основные понятия

Разведочный анализ данных (исследовательский анализ данных, exploratory data analysis, EDA) – подход к анализу данных для обобщения их основных характеристик, зачастую – с использованием статистической графики и других методов визуализации данных[1, 2].

Основоположником разведочного анализа данных принято считать Джона Уайлдера Тьюки (John Wilder Tukey), который в 70-е годы 20 века предложил подразделить статистический анализ на два этапа: разведочный (exploratory data analysis) и подтверждающий (confirmatory data analysis). Разведочный этап должен был охватывать преобразование наборов данных и способы их наглядного представления, позволяющие выявить внутренние закономерности, проявляющиеся в данных, а подтверждающий – традиционные статистические методы оценки параметров и проверки гипотез в контексте анализируемого набора данных. При этом цель разведочного анализа заключалась в побуждении статистиков к исследованию данных и, возможно, к формулировке гипотез, которые могут привести к сбору новых данных и экспериментам.

Следует отметить, что многие идеи, предложенные Джоном Тьюки, позволили сформировать основы науки о данных, представляющей собой сплав многих дисциплин, включая статистику, информатику, информационные технологии с конкретикой предметных областей.

Разведочный анализ данных отличается от традиционного анализа исходных данных (initial data analysis IDA), который более узко фокусируется на проверке предположений, необходимых для подбора модели и проверки гипотез, а также на обработке пропущенных значений и выполнении преобразований переменных по мере необходимости. EDA включает в себя IDA.

В 1961 году Джон Тьюки определил анализ данных как «процедуры анализа данных, методы интерпретации результатов таких процедур, способы планирования сбора данных, чтобы сделать их анализ более простым, точным или более точным, а также все механизмы и результаты математической статистики, которые применяются для анализа данных» [3].

В 1977 году Джон Тьюки написал книгу, посвященную аспектам разведочного анализа данных [4], отметив, что в статистике много внимания уделяется только проверке статистических гипотез, то есть подтверждающему анализу данных, и указав, что необходимо уделять больше внимания вопросам использования данных для выдвижения гипотез для проверки.

Целесообразно указать следующие цели разведочного анализа данных:

– максимальное «проникновение» в данные, подразумевающее поиск и извлечение знаний, скрытых в данных;

– выдвижение гипотез о причинах наблюдаемых явлений;

– оценка допущений, на которых будут основаны статистические выводы;

– поддержка выбора подходящих статистических инструментов и методов;

– обоснование плана для дальнейшего сбора данных с помощью опросов или экспериментов [5].

Разведочный анализ данных подразумевает выполнение анализа основных свойств данных, выявление общих закономерностей, основных структур, наиболее важных переменных, распределений, отклонений и аномалий в данных. Кроме того, при проведении разведочного анализа могут осуществляться проверка основных гипотез и построение начальных моделей на основе имеющихся данных.

Разведочный анализ данных предполагает, что будут осуществляться:

– систематический сбор данных, в ходе которого выполнять проверка гипотез и оценка результатов;

– очистка данных, в ходе которой осуществляется проверка правильности и достоверности данных, в частности, выявление ошибок в данных или отсутствие данных, а также исправление, удаление ошибок и т.п.;

– предварительная обработка данных, в ходе которой осуществляется преобразование необработанных данных в некоторый формат с применением инструментов масштабирования, извлечения, выбора признаков и т.п. с последующим получением итогового набора данных, который может быть использован для решения различных прикладных задач;

– визуализация данных, в ходе которой реализуется графическое представление информации и данных с использованием статистических информационных графиков, а также других инструментов, обеспечивающих четкую и эффективную передачу информации.

При проведении разведочного анализа принято осуществлять изучение вероятностных распределений переменных, выполнять построение и анализ корреляционных матриц, а также – применять многомерное шкалирование, дискриминантный анализ и факторный анализ.

Типичными графическими инструментами, используемыми в разведочном анализе, являются [6]:

– диаграмма размаха (box plot, box-and-whiskers diagram, «ящик с усами», «ящичковая диаграмма»), разработанная Джоном Тьюки в 1970-х годах;

– гистограмма (histogram, bar chart);

– многовариантная диаграмма (multi-vari chart);

– диаграмма Парето (Pareto chart);

– диаграмма рассеяния в 2D/3D (scatter plot);

– стебле-листовой график (stem-and-leaf plot);

– параллельные координаты (parallel coordinates);

– отношение шансов (odds ratio);

– целенаправленное преследование проекций (targeted projection pursuit);

– тепловая карта (heat map);

– столбчатая диаграмма (bar chart);

– график горизонта (horizon graph);

– визуализация на основе глифов (glyph-based visualization methods), например, с использованием PhenoPlot [7] и лиц Чернова;

– проецирование, такое как обзорная экскурсия (grand tour), экскурсия с гидом (guided tour) и экскурсия вручную (manual tour).

Кроме того, могут применяться интерактивные версии графических инструментов.

Типичными инструментами снижения размерности, используемыми в разведочном анализе, являются:

– многомерное шкалирование (multidimensional scaling), представляющее собой метод анализа и визуализации данных с помощью расположения точек, соответствующих изучаемым (шкалируемым) объектам, в пространстве меньшей размерности, чем пространство признаков объектов;

– методы и алгоритмы линейного снижения размерности (linear dimensionality reduction);

– методы и алгоритмы нелинейного снижения размерности (nonlinear dimensionality reduction);

– иконография корреляций (iconography of correlations), заключающаяся в замене корреляционной матрицы диаграммой, где «замечательные» (remarkable) корреляции представлены сплошной линией, в случае положительной корреляции и пунктирной линией, в случае отрицательной корреляции.

В качестве наиболее известного инструмента линейного снижения размерности можно назвать метод главных компонент (Principal Component Analysis, PCA).

В качестве наиболее популярных в последние годы инструментов нелинейного снижения размерности можно назвать t-SNE-алгоритм (t-distributed

stochastic neighbor embedding, t-распределенное стохастическое вложение соседей) и UMAP-алгоритм (Uniform Manifold Approximation and Projection, равномерная аппроксимация и проекция топологического многообразия).

Эффективными количественными инструментами, применяемыми в разведочном анализе данных, являются:

– медианная полировка (median polish), предложенная Джоном Тьюки и позволяющая итеративно оценить влияние строк и столбцов на данные с применением медиан, вычисленных на основе строк и столбцов набора данных;

– тримин-мера (trimean), предложенная Джоном Тьюки и позволяющая оценить расположение распределения вероятностей на основе средневзвешенного для медианы распределения и его двух квартилей;

– ординационный или градиентный анализ (ordination or gradient analysis), дополняющий кластеризацию данных и позволяющий упорядочивать многомерные объекты, которые характеризуются значениями нескольких переменных таким образом, что похожие объекты находятся рядом друг с другом, а непохожие объекты находятся дальше друг от друга.

В настоящее время многие методы и алгоритмы разведочного анализа данных адаптированы для интеллектуального анализа данных.

1.1. Типы данных

Данные, к которым применяется разведочный анализ, поступают из различных источников. В качестве данных могут выступать показания датчиков, изображения, текстовая, графическая, аудио- и видеоинформация и т.п. Значительная часть этих данных не структурирована. Такие данные принято называть «сырыми».

Очевидно, что «сырые» данные должны быть переработаны в данные, полезные на практике, посредством обработки и представления их в структурированном виде (например, как в реляционных базах данных) или в виде, приемлемом для выполнения статистического исследования.

Основными типами структурированных данных являются *числовой* (*количественный*) и *категориальный* (*качественный*) типы.

Числовые (*количественные*) *данные* – данные, измеряемые с помощью чисел, имеющих содержательный смысл.

Числовые (*количественные*) *данные* могут *непрерывными* (continuous data) и *дискретными* (discrete data).

Числовые (*количественные*) *данные* являются *непрерывными*, если множество их возможных значений представляет собой некоторый конечный или бес-

конечный промежуток числовой оси, т.е. такие данные принимают свои значения на непрерывной шкале значений.

Числовые (*количественные*) *данные* являются *дискретными,* если множество их возможных значений конечно или счётно, т.е. такие данные принимают свои значения из ограниченного набора значений, обычно представленных целыми числами.

К *непрерывным числовым данным*, например, относятся температура тела, показатель гемоглобина в крови, рост и вес человека, а к *дискретным числовым данным* – число детей в семье, число рабочих дней в месяце и число дней в больничном листе.

Над *непрерывными данными* можно производить арифметические операции сложения, вычитания, умножения, деления, и они имеют смысл. По отношению к *дискретным данным* применение арифметических операций также возможно, но необходимо внимательно следить за сохранением типа данных и смыслом выполняемых операций.

Непрерывные данные могут быть преобразованы в дискретные с помощью операции квантования, т.е. посредством замены значений непрерывных данных отрезками, каждый из которых представляет некоторый диапазон.

Категориальные (*качественные*) *данные* – данные, которые могут принимать значения только из некоторого фиксированного набора значений, т.е. это данные с ограниченным числом уникальных значений или категорий.

Категориальные (*качественные*) *данные* – данные, описывающие качество и не имеющие количественного выражения. В *категориальных* (*качественных*) *данных* каждая единица наблюдения назначается определенной группе или номинальной категории на основе некоторого качественного свойства.

Категориальные (*качественные*) данные могут быть *номинальными* (nominal data) и *порядковыми* (ordered data), которые отражают соответственно условные коды неизмеримых категорий или условную степень выраженности признака.

К *качественным номинальным данным*, например, относится название применяемого для лечения препарата, название заболевания пациента.

Значения *категориальных* (*качественных*) *порядковых* данных могут быть ранжированы по какому-либо принципу, но интервал между значениями таких данных не может быть выражен количественно. Обычно *категориальные* (*качественные*) *порядковые данные* качественно отражают условную степень выраженности какого-либо признака, например, тяжесть состояния больного при

поступлении в стационар (тяжёлое, средней тяжести,…), степень ожога (1, 2, 3 или 4), группа инвалидности (первая, вторая или третья).

При работе с *категориальными* (*качественными*) *данными* применяются только операции сравнения (такие, как «равно» и «не равно») и производится упорядочение данных, например, по алфавиту. Применение арифметических операций к категориальным данным некорректно, даже если они представлены числами.

Особый случай категориальных данных – *бинарные данные*, которые принимают только одно из двух допустимых альтернативных значений: 0 или 1 («да» или «нет»; «истина» или «ложь»; «здоров» или «болен»).

В зависимости от того, каков тип данных – числовой или категориальный – определяется тип применяемого в дальнейшем инструмента разведочного анализа: тип визуального отображения, тип анализа данных, тип статистической модели и т.п.

Знание о том, каков тип данных, используется как для улучшения вычислительный производительности в процессе анализа данных, так и для определения того, какие операции (вычисления) допустимы для анализируемых данных.

1.2. Представление данных

При выполнении разведочного анализа данных *обычно* принято работать с так называемыми «прямоугольными» данными, т.е. данными, которые можно представить в виде таблицы, например, в виде электронной таблицы в MS Excel или таблицы базы данных. Очень часто для представления табличных данных используют текстовый формат *CSV* (*Comma-Separated Values* – значения, разделённые запятыми). Файлы с расширением csv (csv-файлы) удобно использовать для представления как исходных, так и итоговых наборов данных при выполнении разведочного анализа данных на языке Python.

«Прямоугольные» данные, по сути, представляют собой двумерную матрицу. Строки матрицы являются записями (признаковыми описаниями объектов, т.е. векторами, которые составлены из значений, соответствующих некоторому набору признаков, описывающих объект). Столбцы матрицы соответствуют признакам (при этом значения признаков могут быть разного типа). В этом случае говорят, что «прямоугольные» данные – это матрица «объекты × признаки». Такое представление данных является стандартным и наиболее распространённым в задачах разработки статистических моделей и моделей машинного обучения (например, в задачах кластеризации, классификации и регрессионного анализа).

Любые неструктурированные данные (например, текст) необходимо обработать так, чтобы их можно было представить как набор признаков в матрице «объекты × признаки».

При выполнении разведочного анализа данных на языке Python целесообразно использовать программную библиотеку `pandas` (https://pandas.pydata.org/) [8], в которой основной прямоугольной структурой данных является объект DataFrame, содержащий таблицу данных. По умолчанию для DataFrame создается автоматический целочисленный индекс, который основывается на порядке следования строк. Для повышения эффективности некоторых операций в программной библиотеке `pandas` можно задавать многоуровневые/иерархические индексы.

В процессе статистического исследования рассматривают набор данных, полученных в результате измерения одного или нескольких признаков у тех или иных объектов.

Реально наблюдаемая совокупность объектов, статистически представленная рядом наблюдений $x_1, x_2, ..., x_n$ случайной величины X, является выборкой, а гипотетически существующая (домысливаемая) совокупность объектов – генеральной совокупностью.

Генеральная совокупность может быть конечной или бесконечной, а выборка из генеральной совокупности всегда является результатом ограниченного ряда n наблюдений.

Следует отметить существование некоторых различий в терминологии, имеющие место в статистике и науке о данных (data science), при описании одних и тех же сущностей. Так, в статистике принято говорить о предикторных переменных (предикторах), которые используются для предсказания зависимой переменной (отклика), а в науке о данных принято говорить о *признаках*, которые используются для предсказания *целевой* переменной. В статистике для значений, вычисляемых на основе имеющихся данных, обычно применяется термин «оценка». Использование этого термина позволяет отличать вычисленные значения от теоретически истинных и учитывать возможную неопределенность данных. В науке о данных, интеллектуальном анализе данных и бизнес-аналитике вместо термина «оценка» обычно используют термин «метрика» («метрический показатель»), т.к. в центре внимания находятся конкретные данные, анализ которых выполняется с конкретными деловыми или организационными целями. Таким образом, в статистике принято *оценивать*, а в аналитике – *измерять*.

2. Обработка пропусков данных

Зачастую наборы данных содержат пропуски, т.е. для некоторых объектов, соответствующих строкам набора данных, имеет место отсутствие значений по некоторым признакам, соответствующим столбцам набора данных.

Очевидно, что такие наборы данных должны быть каким-либо образом преобразованы, для того чтобы к ним можно было применить те или иные средства разведочного анализа данных, а также – чтобы преобразованные наборы данных можно было бы использовать при решении задач кластеризации, классификации, прогнозирования и т.п.

Самый простой подход к решению проблемы пропусков данных – это удаление из набора данных информации об объектах с пропусками значений по некоторым признакам, т.е. удаление строк набора данных, соответствующих этим объектам. Однако применение этого подхода может привести к тому, что набор данных станет не репрезентативным, поскольку будет удалено очень много строк.

Альтернативой подходу, реализующему удаление из набора данных информации об объектах с пропусками значений по некоторым признакам, является подход, предполагающий заполнение пропусков данных некоторыми значениями. Этот подход применим как в случае работы с количественными признаками, так и с качественными. Однако при применении этого подхода целесообразно внимательно изучить саму природу набора данных, а также – особенности (специфику) признаков, для которых будет осуществляться заполнение пропусков значений.

Заполнение пропусков в случае количественных признаков может быть осуществлено, например, на основе метрик центрального значения, описанных в п. 3.1 ниже. Например, могут быть использованы такие метрики, как *среднее* и *медиана*. Заполнение пропусков в случае качественных признаков может быть осуществлено, например, на основе наиболее часто встречающегося значения признака.

Возможно, что при заполнении пропусков значений одного признака следует предварительно обратить внимание на значения других признаков. Например, в случае пропусков в наборе данных значений по признаку *возраст*, следует сначала проанализировать значения по признаку *пол* (если он есть в наборе данных), и осуществить заполнение пропусков по признаку *возраст* с учетом гендерной принадлежности, т.е. выполнить вычисление *среднего* или *медианы* для женщин и мужчин по отдельности с целью заполнения пропусков значений с учетом пола.

Отметим, что заполнение пропусков данных обычно кажется более предпочтительным решением. Однако это не всегда так. Неудачный выбор подхода к заполнению пропусков данных может не только не улучшить, но и сильно ухудшить результаты (например, при использовании таких наборов данных решении задач кластеризации, классификации, прогнозирования и т.п.).

Естественно, что после применения различных подходов (например, таких метрик, как *среднее* и *медиана*) к заполнению пропусков данных итоговые значения метрик будут отличаться друг от друга.

3. Метрики для количественных признаков

Количественные признаки могут принимать большое число различных значений.

При выполнении обобщения признака необходимо получить его «типичное значение», т.е. понять, где расположено большинство его значений (какова их центральная тенденция), а также – измерить вариабельность признака [2].

3.1. Метрики центральной тенденции для количественных признаков

Самый простой подход к обобщению количественного признака предполагает вычисление *среднего,* т.е. среднего арифметического значения на основе всех известных значений анализируемого признака. Хотя *среднее* вычисляется очень просто и его удобно использовать, оно не всегда является лучшей мерой центральной тенденции. В связи с этим на практике используют несколько альтернативных метрик, характеризующих центральную тенденцию анализируемого признака, в частности, *среднее*, *среднее усеченное*, *среднее взвешенное*, *медиану, медиану взвешенную.*

Пусть $X = \{x_1, x_2, ..., x_n\}$ – множество значений признака.

Среднее(*mean, среднее арифметическое*) – сумма всех значений признака, деленная на число этих значений:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \qquad (1)$$

где x_i – i-е значение признака ($i = \overline{1,n}$); n – число значений признака.

Пусть имеется набор значений признака, характеризующего частоту сердечных сокращений (число ударов в минуту, пульс): {60, 80, 80}.

Тогда $\bar{x} = \frac{60+80+80}{3} = \frac{220}{3} \approx 73.333$.

Среднее усеченное (*trimmed mean*) является разновидностью *среднего*.

Среднее усеченное вычисляется посредством отбрасывания фиксированного числа значений с каждого конца сортированных последовательности и последующего взятия среднего арифметического оставшихся значений.

Пусть значения анализируемого признака отсортированы как $x_{(1)}, x_{(2)}, ..., x_{(n)}$, где $x_{(1)}$ – самое маленькое значение, а $x_{(n)}$ – самое большое. Тогда *среднее усеченное* при удалении p самых малых и p самых больших значений анализируемого признака вычисляется как:

$$\bar{x} = \frac{\sum_{i=1+p}^{n-p} x_i}{n - 2 \cdot p}. \qquad (2)$$

Среднее усеченное позволяет устранить влияние предельных значений. Во многих случаях *среднее усеченное* более предпочтительно, чем обычное *среднее*.

Среднее взвешенное (*weighted mean, среднее арифметическое взвешенное*) вычисляется как

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i \cdot x_i}{\sum_{i=1}^{n} w_i}, \qquad (3)$$

где x_i – i-е значение признака; w_i – вес i-го значения признака; $i = \overline{1,n}$; $\bar{x}$ – среднее; n – число значений признака.

Среднее взвешенное целесообразно применять, например, если разные значения признака имеют разную точность (тогда менее точным значениям следует придать меньший вес) или значения признака не одинаково представляют разные оцениваемые (измеряемые) группы (тогда более высокий вес следует придать значениям признака из тех групп, которые были представлены недостаточно). Однако при применении *среднего взвешенного* возникает проблема адекватности используемых весов, т.к. процедура назначения весов имеет субъективный характер.

Медиана (*median*) – значение признака, расположенное в сортированном списке значений признаков ровно посередине. Если число значений признаков четное, средним значением является то, которое не находится в наборе данных фактически, а является средним арифметическим двух значений, которые делят сортированные значения признаков на верхнюю и нижнюю половины.

Медиану также называют 50-м процентилем (о процентилях будет сказано ниже).

Если набор значений признака, характеризующего частоту сердечных сокращений, имеет вид {60, 80, 80}, то медиана m вычисляется как m=80.

Если набор значений признака, характеризующего частоту сердечных сокращений, имеет вид {60, 76, 80, 80}, то медиана m вычисляется как m=78.

По сравнению со *средним*, в котором используются абсолютно все значения признака, *медиана* зависит только от значений в центре сортированных значений признака. *Медиана* намного менее чувствительна к значениям признака, чем *среднее* (что можно считать недостатком *медианы*), однако в ряде случаев *медиана* является лучшей метрикой центральной тенденции.

Медиана взвешенная (*weighted median*) – значение признака, для которого суммы весов для нижней и верхней половин сортированного списка значений признака равны.

Медиана и *медиана взвешенная* устойчивы к выбросам.

Медиану называют *робастной* (*robust*) метрикой центральной тенденции, т.к. она не находится под влиянием выбросов, которые могут исказить результаты. Медиана не чувствительна к выбросам, т.е. устойчива к ним.

Выброс (*outlier*) – любое значение признака, которое сильно удалено (сильно отличается) от других значений признака.

Выброс не делает значение признака недопустимым или ошибочным. Тем не менее, выбросы часто являются результатом ошибок в данных, таких как смешивание данных с разными единицами измерения (например, килограммы и граммы) или плохие показания датчика.

Если выбросы являются результатом ошибочных данных, среднее значение будет плохой метрикой центральной тенденции, в то время как медиана будет по-прежнему допустимой. Следует отметить, что выбросы всегда должны быть идентифицированы и исследованы.

Среднее усеченное принято считать компромиссом между *медианой* и *средним*: *среднее усеченное* устойчиво к выбросам, но использует больше данных для вычисления метрики центральной тенденции.

Следует отметить, что *среднее усеченное* также можно считать робастной метрикой центральной тенденции, т.к. оно позволяет предотвратить влияние выбросов. Например, усечение нижних и верхних 10% данных (что общепринято) обеспечивает защиту от выбросов во всех, кроме самых малых, наборах данных.

В настоящее время известны различные инструменты оценки [так называемые оценщики (эстиматоры, estimators)] центральной тенденции, которые имеют хорошую робастность (по сравнению со *средним*) и более эффективны,

т.е. способны лучше обнаруживать небольшие различия в центральной тенденции между наборами данных.

Таким образом, можно сделать следующие выводы:

– основной метрикой центральной тенденции является среднее, но оно может быть чувствительным к выбросам;

– такие метрики центральной тенденции, как медиана, медиана взвешенная и среднее усеченное, более робастны.

3.2. Метрики вариабельности для количественных признаков

Центральная тенденция – одна из характеристик в обобщении признака. Еще одна характеристика – вариабельность (дисперсность) – показывает, сгруппированы ли значения признака плотно, или же они разбросаны.

Для измерения вариабельности признака, как и в случае центрального значения, можно использовать различные метрики.

3.2.1. Метрики вариабельности на основе отклонений

Метрики вариабельности признака могут быть основаны на *отклонениях* (*ошибках, остатках, deviations*) между метрикой центральной тенденции и наблюдаемыми значениями признака.

Пусть имеется набор значений признака, характеризующего частоту сердечных сокращений (число ударов в минуту, пульс): {60, 80, 80}.

Среднее для этого набора равно 73.333, а отклонения от среднего имеют вид: −13.333, 6.667, 6.667. Эти отклонения показывают, насколько значения признака разбросаны вокруг центральной тенденции. Очевидно, что следует каким-либо образом оценить типичное значение этих отклонений. *Среднее* (среднее арифметическое) отклонений может оказаться равным нулю (или очень близким к нулю), если сумма отклонений равна нулю (близка к нулю), например, как в рассматриваемом примере: −13.333+6.667+6.667 ≈0), хотя по факту отклонения ненулевые. Как вариант, в качестве метрики вариабельности можно использовать *среднее абсолютное отклонение от среднего* (mean absolute deviation, *meanAD*), вычисляемое как:

$$meanAD = \frac{\sum_{i=1}^{n} \left| x_i - \bar{x} \right|}{n}, \qquad (4)$$

где x_i – i-е значение признака ($i = \overline{1,n}$); $\bar{x}$ – среднее; n – число значений признака.

Для рассматриваемого набора данных вида {60, 80, 80} *meanAD* вычисляется как $meanAD = (13.333 + 6.667 + 6.667)/3 = 8.889$.

Наиболее часто в качестве метрик вариабельности используют *дисперсию* и *стандартное отклонение*, основанные на квадратических отклонениях.

Дисперсия (*variance*) – сумма квадратических отклонений от среднего, деленная на $n-1$, где n – число значений признака:

$$variance = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}, \qquad (5)$$

где x_i – i-е значение признака ($i = \overline{1,n}$); $\bar{x}$ – среднее; n – число значений признака.

По формуле (5) вычисляется несмещённая оценки дисперсии, т.к. в знаменателе стоит $n-1$. Если заменить $n-1$ на n, то оценка будет смещенной. Следует отметить, что при очень больших n не имеет большого значения, будет ли деление выполняться на n или $n-1$. Тем не менее, использование величины $n-1$ связано с понятием число степеней свободы. Очевидно, что анализируемые значения признака принадлежат некоторой выборке из генеральной совокупности. Если в формуле (5) для дисперсии использовать интуитивно понятный знаменатель n, то истинное значение дисперсии и стандартного отклонения в генеральной совокупности будет недооценено, что приведет к смещенной оценке. Однако если использовать в знаменателе на $n-1$ вместо n, то оценка будет несмещенной.

Почему использование n приводит к смещенной оценке? Оказывается, необходимо учитывать, чему равно число степеней свободы, принимая во внимание число ограничений, которые необходимо учитывать при вычислении оценки. В случае метрики для дисперсии существуют $n-1$ степеней свободы, т.к. имеется только одно ограничение: дисперсия зависит от вычисления среднего в выборке.

Дисперсия представляет собой среднее квадратических отклонений.

Стандартное отклонение (*среднеквадратическое отклонение*, standard deviation, *STD*, среднеквадратическое отклонение) – квадратный корень из *дисперсии*:

$$\sigma = \sqrt{variance}. \qquad (6)$$

Стандартное отклонение интерпретируется проще, чем дисперсия, т.к. оно находится на той же шкале измерения, что и значения признака. Хотя формула (6), определяющая *стандартное отклонение*, более сложна и интуитивно менее понятна, чем формула (4), определяющая среднее абсолютное отклонение,

именно стандартному отклонению отдают большее предпочтение при использовании его в качестве метрики вариабельности. Это объясняется тем, что с точки зрения математики работа с квадратическими значениями более удобна, чем с абсолютными, в особенности – в случае оперирования статистическими моделями.

Т.к. стандартное отклонение использует квадраты разностей, оно всегда будет больше или равно среднему абсолютному отклонению. Различие между стандартным отклонением и средним абсолютным отклонением будет особенно велико, если в наборе данных есть большие выбросы.

Если набор данных имеет нормальное распределение, т.е. в нем не очень много выбросов, стандартное отклонение является более предпочтительной метрикой вариабельности. Если в наборе данных есть большие выбросы, стандартное отклонение дает более высокие значения разброса (отклонения от центра), чем среднее абсолютное отклонение.

Для рассматриваемого набора данных вида {60, 80, 80} $variance = (13.333^2 + 6.667^2 + 6.667^2)/2 = 133.333$, $STD = \sqrt{133.333} \approx 11.547$.

Среднее абсолютное отклонение, *дисперсия* и *стандартное отклонение* не устойчивы к выбросам. При этом дисперсия и стандартное отклонение чувствительны к выбросам сильнее, чем среднее абсолютное отклонение, т.к. они основаны на квадратических отклонениях.

Робастной метрикой вариабельности является медианное абсолютное отклонение от медианы (median absolute deviation, $medianAD$):

$$medianAD = медиана\left(|x_1 - m|, |x_2 - m|, ..., |x_n - m|\right), \qquad (7)$$

где m – медиана; x_i – i-е значение признака ($i = \overline{1,n}$); n – число значений признака.

Для рассматриваемого набора данных $medianAD$ вычисляется как $medianAD = медиана(0,0,20) = 0$.

Как и в случае с *медианой*, $medianAD$ не находится под влиянием выбросов.

Следует отметить, что можно вычислить *усеченное стандартное отклонение* по аналогии со *средним усеченным. Усеченное стандартное отклонение* можно считать робастной метрикой вариабельности, т.к. оно позволяет предотвратить влияние выбросов.

Дисперсия, *стандартное отклонение*, *среднее абсолютное отклонение* и *медианное абсолютное отклонение* не являются эквивалентными метриками, даже если признак имеет нормальный закон распределения. *Стандартное отклонение* всегда больше *среднего абсолютного отклонения*, которое всегда больше *медианного абсолютного отклонения*.

3.2.2. Метрики вариабельности на основе процентилей

Метрики вариабельности признака могут быть основаны на процентилях.

В этом случае рассматривают разброс (*спред*, *spread*) сортированных данных (значений признака).

Порядковые статистики – статистические показатели на основе сортированных (ранжированных) данных.

Самая элементарная метрика, оценивающая разброс,– *размах*, т.е. разница между самым большим и самым малым значениями признака.

Размах очень чувствителен к выбросам и не очень полезен в качестве метрики вариабельности данных, хотя минимальные и максимальные значения признака полезно знать, т.к. они помогают идентифицировать выбросы.

Чтобы снизить чувствительность к выбросам, целесообразно рассмотреть размах данных (размах значений признака) после отбрасывания части значений с каждого конца сортированного набора значений признака. Метрики такого типа формально основаны на разницах между процентилями.

В наборе значений признака *P*-й процентиль – такое число z_P, что значения *P*-й части набора меньше или равны z_P.

Например, 25-й процентиль – такое число z_P, что 25% значений признака меньше или равны z_P. Медиана соответствует 50-й процентили.

Процентиль аналогичен квантилю, при этом квантили индексируются долями (так, например, квантиль 0,25 – это то же самое, что и 25-й процентиль).

Обычно в качестве еще одной метрики вариабельности используют разницу между 25-м и 75-м процентилями. Эта метрика называется *межквартильным размахом* (*interquartile range, IQR*).

Пусть имеется набор значений признака, характеризующего частоту сердечных сокращений (число ударов в минуту, пульс): {60, 55, 80, 75, 95, 90, 80, 70}.

Сначала числа в этом наборе необходимо отсортировать по возрастанию:

55, 60, 70, 75, 80, 80, 90, 95.

25-й процентиль для рассматриваемого набора равен $z_{25} = 65$, 75-й процентиль для рассматриваемого набора равен $z_{75} = 90$, поэтому межквартильный размах будет $90 - 65 = 25$.

Для больших наборов данных (с большим числом значений признака) расчет точных процентилей может быть вычислительно затратным ввиду необходимости выполнения процедуры сортировки всех значений признака. В этом случае могут быть использованы специальные алгоритмы, которые позволяют

получить очень быстро приблизительный процентиль, гарантируя обеспечение требуемой точности.

Необходимо отметить следующее: если имеется четное число значений признаков (т.е. n – чётное), то определение процентиля не является однозначным. Можно взять любое значение между порядковыми статистиками $x_{(j)}$ и $x_{(j+1)}$, где j удовлетворяет условию:

$$100 \cdot \frac{j}{n} \le P \le 100 \cdot \frac{j+1}{n}. \quad (8)$$

Формально процентиль – средневзвешенное значение:

$$P = (w-1) \cdot x_{(j)} + w \cdot x_{(j+1)}, \quad (9)$$

где w – весовой коэффициент; $w \in (0, 1)$.

При этом выбор значения весового коэффициента w может быть осуществлен, например, программно.

Таким образом, можно сделать следующие выводы:

– основными метриками вариабельности являются дисперсия и стандартное отклонение, но они чувствительны к выбросам;

– такие метрики вариабельности, как медианное абсолютное отклонение от медианы и процентили (квантили), более робастны.

4. Анализ распределения данных для количественных признаков

Все рассмотренные в п. 3 метрики обобщают значения признака в одном числе с целью описания центральной тенденции или вариабельности признака. Однако существенный интерес представляет анализ характера распределения значений признака в целом.

Для выполнения исследования характера распределения значений признака в целом могут быть использованы такие инструменты, как *коробчатая диаграмма* (boxplot), *частотная таблица* (frequency table), *гистограмма* (histogram), *график плотности* (density plot) [2].

Коробчатую диаграмму («ящик с усами», box-and-whiskers diagram) удобно использовать в качестве быстрого способа визуализации распределения данных. *Частотная таблица* представляет собой сводку количеств числовых значений, которые разбиты на серию частотных интервалов (корзин, бинов). *Гистограмма* является графическим изображением частотной таблицы, где частотные интервалы откладываются на оси *x*, а количества (или доли) – на оси *y*. *График плотности* представляет собой сглаженную версию гистограммы (зачастую – на основе ядерной оценки плотности).

Процентили могут использоваться как для измерения разброса данных, так и для обобщения всего распределения в целом. Обычно рассматривают квартили (то есть 25-й, 50-й и 75-й перцентили) и децили (10-й, 20-й, ..., 90-й процентили). Процентили целесообразно использовать для обобщения хвостов (внешнего размаха) распределения. Так, например, в литературе можно встретить термин «однопроцентовики», который относится к людям в верхнем 99-м процентиле богатства.

Коробчатые диаграммы основаны на процентилях и обеспечивают быстрый способ визуализации распределения данных (рисунок 1).При вертикальном изображении коробчатой диаграммы верх и низ коробки (границы ящика) соответствуют 75-му и 25-му процентилям, т.е. первому и третьему квартилю. Кроме того, посередине коробки коробчатой диаграммы горизонтальной линией изображается медиана (50-й процентиль).

Линии, называемые усами и выходящие из верха и низа коробки, говорят о размахе основной части данных (значений признака). Концы усов представляют собой края статистически значимой выборки (без выбросов).

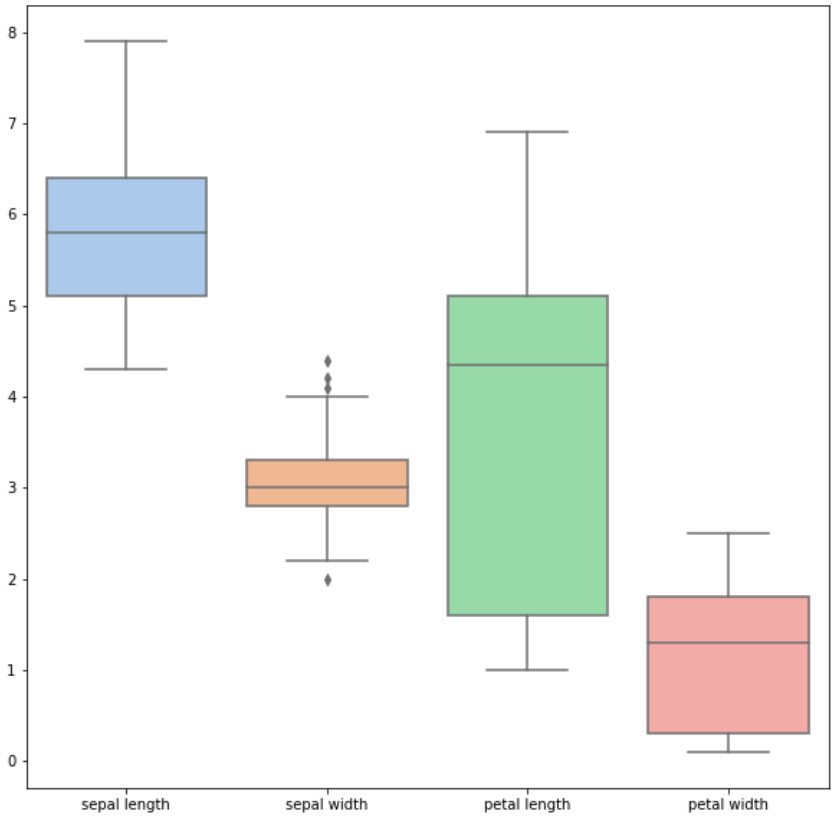

Рисунок 1. Визуализация распределения данных на примере набора «Ирисы Фишера» (The Iris Dataset) с применением коробчатых диаграмм

Концы усов могут определяться несколькими способами. Наиболее часто в качестве значений, определяющих длину «усов», используются:

– минимальное и максимальное наблюдаемые значения данных по выборке (в таком случае считается, что выбросы отсутствуют);

– среднее минус одно стандартное отклонение и среднее одно стандартное отклонение;

– 9-й и 91-й процентили;

– 2-й и 98-й процентили;

– разность первого квартиля и k межквартильных расстояний *IQR* (InterQuartile Range) и сумма третьего квартиля и k межквартильных расстояний *IQR* (в таком случае выбросы присутствуют).

В последнем случае границы усов могут быть вычислены в соответствии с формулами:

$$X_1 = Q_1 - k \cdot (Q_3 - Q_1), \quad (10)$$

$$X_2 = Q_3 + k \cdot (Q_3 - Q_1), \quad (11)$$

где X_1 – граница нижнего уса; X_2 – граница верхнего уса; Q_1 – первый квартиль; Q_3 – третий квартиль; k – коэффициент.

Наиболее часто используется значение коэффициента k, равное 1,5 (в этом случае формируются так называемые внутренние границы (inner fences) усов на коробчатой диаграмме). Кроме того, часто используют значение коэффициента k, равное 3 (в этом случае формируются так называемые внешние границы (outer fences) усов на коробчатой диаграмме). Выбор значения коэффициента k влияет на то, какое число объектов будут рассматриваться в качестве выбросов.

Длину верхнего уса ограничивают максимальным значением по выборке, попадающим в верхнюю границу уса; длину нижнего уса ограничивают минимальным значением по выборке, попадающим в нижнюю границу уса. Длины верхнего и нижнего усов могут не совпадать. Данные, выходящие за границы усов, – выбросы – отображаются на коробчатой диаграмме в виде точек, маленьких кружков или звёздочек. Кроме того, на коробчатой диаграмме может быть отмечено среднее и его доверительный интервал (в виде клина-выреза («зарубки») на коробке коробчатой диаграммы). Иногда «зарубками» отмечают доверительный интервал для медианы.

Скрипичная диаграмма (violin plot) представляет собой дополнение к коробчатой диаграмме (рис. 2). *Скрипичная диаграмма* графически изображает оценки плотности, при этом значения плотности расположены на оси `y`. Плотность зеркально отражена и перевернута, а получившаяся фигура – заполнена. Полученное изображение напоминает скрипку. Достоинство скрипичной диаграммы заключается в том, что она может показывать особенности в распреде-

лении данных, которые не заметны на коробчатой диаграмме. Однако следует отметить, что выбросы в данных более ясно показывает коробчатая диаграмма.

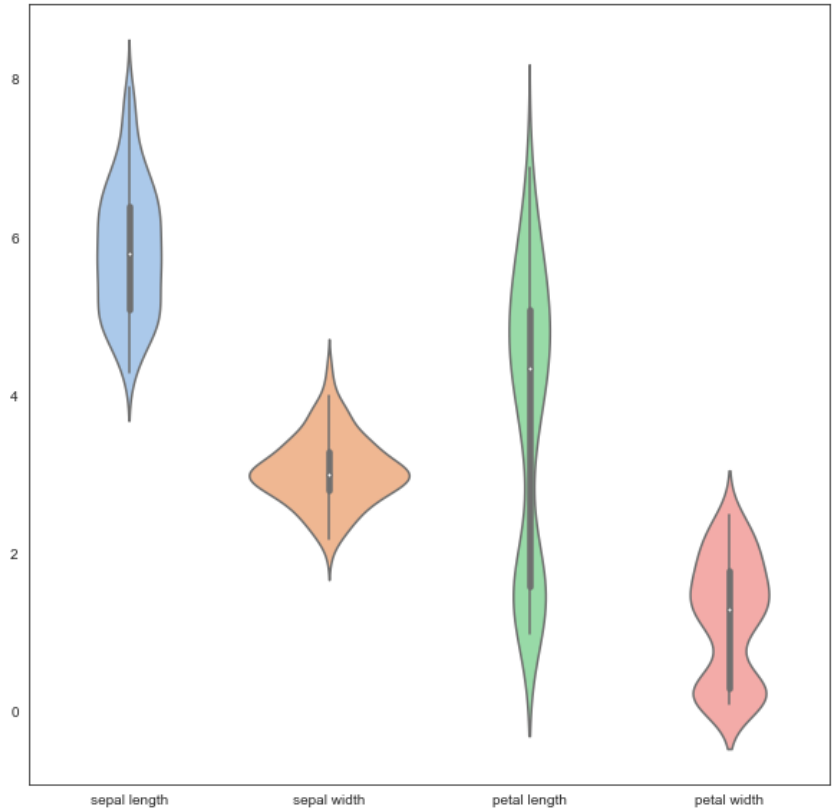

Рисунок 2. Визуализация распределения данных на примере набора «Ирисы Фишера» (The Iris Dataset) с применением скрипичных диаграмм

Частотная таблица делит диапазон значений признака на равноотстоящие интервалы и показывает, сколько значений признака попадает в каждый интервал. При этом необходимо учитывать пустые интервалы: обнаружение отсутствия значений признака в некоторых интервалах является полезной информацией.

Полезно экспериментировать с разными размерами интервалов. Если размеры интервалов будут очень большими, то полученные результаты окажутся слишком общими, а важные особенности распределения могут быть утеряны (сглажены). Если размеры интервалов будут очень маленькими, то полученные результаты будут очень детальными, и будет невозможно наблюдать общую картину. Процентили и частотные таблицы обобщают данные за счет создания частотных интервалов. Следует отметить, что квартили и децили будут иметь одинаковое число значений признака в каждом интервале, но размеры интервалов будут различаться. Частотная таблица, в отличие от них, будет иметь разное число признаков в интервалах одинакового размера.

Гистограмма (histogram) реализует визуализацию частотной таблицы таким образом, что частотные интервалы откладываются на оси *x*, а число значений признака – на оси *y* (рис. 3). Гистограмма представляет собой диаграмму, которая обычно является столбиковой. Высота каждого столбика гистограммы отражает число значений признака для интервала, на который он опирается.

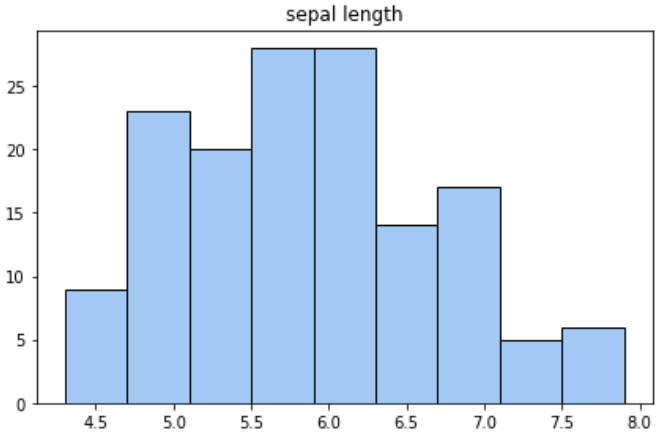

Рисунок 3. Гистограмма распределения для признака sepallength набора данных «Ирисы Фишера» (The Iris Dataset)

На гистограмме частот отображают число значений признака, попавших в каждый интервал. Для получения гистограммы для оценок вероятностей необходимо разделить число значений признака в каждом интервале на общее число значений признака. В этом случае сумма высот всех столбцов гистограммы будет равна 1 (как площадь под кривой закона распределения в теории вероятностей).

Гистограммы строятся так, что:

– пустые интервалы включаются в график;
– интервалы имеют равную ширину;
– число интервалов задается пользователем;
– столбцы гистограммы непрерывны (пробелы между ними отсутствуют, если нет пустого интервала).

По результатам анализа формы гистограммы может предположить, какому статистическому закону распределения подчиняется анализируемый признак. Так, если все столбцы гистограммы имеют примерно одинаковую высоту, то можно предположить соответствие данных равномерному закону распределения, если все столбцы гистограммы образуют симметричный «холм», то предположить соответствие данных нормальному закону распределения.

График плотности (density plot) распределения связан с гистограммой: он показывает распределение значений данных в виде сплошной линии (рис. 4).

График плотности можно рассматривать как сглаженную гистограмму (хотя он обычно вычисляется непосредственно из данных с помощью ядерной оценки плотности).

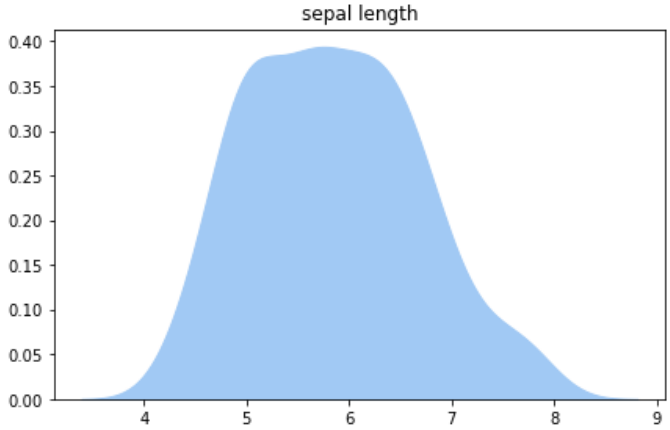

Рисунок 4. График плотности распределения, для признака sepallength набора данных «Ирисы Фишера» (The Iris Dataset)

В статистической теории центральная тенденция значение и вариабельность позиционируются как моменты первого и второго порядка. При этом асимметрия и эксцесс позиционируются как моменты третьего и четвертого порядка. Асимметрия и эксцесс характеризуют геометрическую форму распределения.

Асимметрия (skewness) характеризует смещение данных к бóльшим или меньшим значениям и является мерой скошенности распределения данных вправо или влево относительно центральной тенденции.

Эксцесс характеризует склонность данных к предельным значениям и является мерой его высоты.

Выявить асимметрию и эксцесс можно визуально, например, на основе коробчатой диаграммы (рис. 1) и гистограммы (рис. 2).

Коэффициент асимметрии вычисляется как:

$$A = \frac{m_3}{\sigma^3}, \qquad (12)$$

где σ – стандартное отклонение; m_3 – центральный момент третьего порядка; $m_3 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^3}{n}$; x_i – i-е значение признака ($i = \overline{1,n}$); n – число значений признака.

Если $A > 0$, то распределение скошено вправо, если $A < 0$ – то влево. Используется следующая условная градация: если $|A| < 0.1$, то асимметрия практически отсутствует; если $0.1 \le |A| < 0.25$, то асимметрия мала; если $0.25 \le |A| < 1$, то асимметрия заметная; если $|A| \ge 1$, то асимметрия существенная.

Так, у нормального распределения $A = 0$.

Коэффициент эксцесса вычисляется как:

$$E = \frac{m_4}{\sigma^4} - 3, \qquad (13)$$

где σ – стандартное отклонение; m_4 – центральный момент четвертого порядка; $m_4 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{n}$; x_i – i-е значение признака ($i = \overline{1,n}$); n – число значений признака.

Если $E > 0$, то распределение является более высоким («островершинным») относительно «эталонного» нормального распределения $N(\bar{x}, \sigma)$. Если $E < 0$, то распределение является более низким и пологим. Чем больше E по модулю, тем «аномальнее» высота в ту или иную сторону.

Так, у нормального распределения $E = 0$.

Таким образом, можно сделать следующие выводы:

– частотная гистограмма показывает частоты на оси *y* и значения переменных – на оси *x*, давая визуальное представление о распределении данных;

– частотная таблица является табличной версией частот, которые отображены на гистограмме;

– коробчатая диаграмма реализует представление данных с применением так называемого «ящика с усами», в котором верх и низ находятся соответственно в 75-м и 25-м процентилях, дает визуальное представление о распределении данных и зачастую используется на парных графиках с целью сравнения распределений данных;

– график плотности распределения представляет собой сглаженную версию гистограммы.

5. Корреляция признаков

Существенный интерес в разведочном анализе данных представляет изучение корреляции между признаками, в том числе – корреляция признаков, описывающих объекты, с целевым признаком [2].

Признаки *X* и *Y* коррелируют *положительно*, если большим значениям признака *X* соответствуют большие значения признака *Y*, а малым значениям признака *X* соответствуют малые значения признака *Y*.

Признаки *X* и *Y* коррелируют *отрицательно*, если большим значениям признака *X* соответствуют малые значения признака *Y*, а малым значениям признака *X* соответствуют большие значения признака *Y*.

Для оценки корреляции между двумя признаками, находящимися на одинаковой шкале измерения, используют *коэффициент корреляции* (correlation coefficient) Пирсона:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\cdot(y_i - \bar{y})}{(n-1)\cdot\sigma_x\cdot\sigma_y}, \qquad (14)$$

где x_i, y_i – i-е значения признаков X и Y соответственно ($i = \overline{1,n}$); n – число значений признака; $\bar{x}$, $\bar{y}$ – средние для признаков X и Y соответственно; σ_x, σ_y – стандартные отклонения для признаков X и Y соответственно.

Коэффициент корреляции всегда находится между числом «−1», соответствующим идеальной отрицательной корреляции, и числом «+1», соответствующим идеальной положительной корреляции. Значение коэффициента корреляции, равное числу «0», свидетельствует об отсутствии корреляции. Коэффициент корреляции, как среднее и стандартное отклонение, чувствителен к выбросам данных. Следует отметить, что признаки могут иметь нелинейную связь. В этом случае коэффициент корреляции может оказаться бесполезным.

На основе значений коэффициента корреляции для различных пар признаков может быть создана *корреляционная матрица* (correlation matrix) (рис. 5). В этой матрице строки и столбцы соответствуют признакам в наборе данных, а значения на пересечении строк и столбцов представляют собой значения коэффициента корреляции для соответствующей пары признаков. Можно отметить, что на диагонали корреляционной матрицы стоят единицы (т.к. корреляция любого признака с самим собой равна 1), при этом матрица симметрична относительно главной диагонали.

Для визуализации корреляционной матрицы может быть использована *тепловая карта* (heatmap), позволяющая разными цветами различной насыщенности отразить величину значений коэффициента корреляции для разных пар признаков (рис. 5).

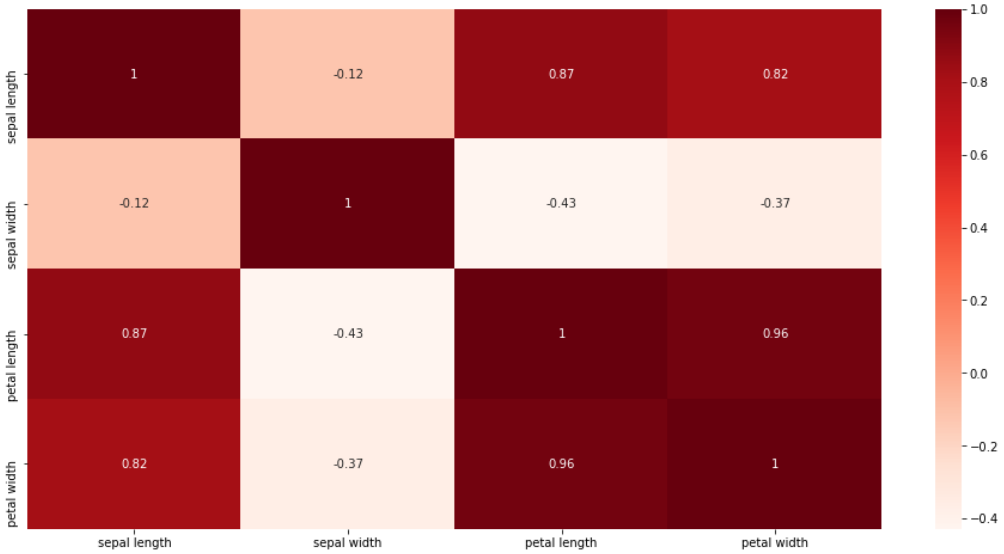

Рисунок 5. Тепловая карта на основе корреляционной матрицы для набора данных «Ирисы Фишера» (The Iris Dataset)

В статистике известны также коэффициент ранговой корреляции Спирмена ρ и коэффициент ранговой корреляции Кендалла τ. Эти коэффициенты корреляции основаны на ранге данных, т.е. на номерах объектов в наборе. Т.к. эти коэффициенты корреляции работают с рангами, а не со значениями, то они устойчивы к выбросам и могут справляться с определенными типами нелинейности. Тем не менее, при выполнении разведочного анализа обычно применяют именно коэффициента корреляции Пирсона и его робастные альтернативы. Ранговые оценки чаще всего используют для небольших наборов данных, а также при определенных проверках статистических гипотез.

Диаграмма рассеяния (scatter plot) является стандартным инструментом визуализации связи между двумя количественными признаками (рис. 6). На такой диаграмме ось *x* представляет один признак, ось *y* – другой. Для визуализации связи двух признаков могут быть использованы *контурные графики* (contour plots), в которых контуры накладываются на диаграмму рассеяния (рис. 7). Контуры подобны линиям уровня: линия каждого контура представляет соответствующую плотность объектов, которая увеличивается по мере приближается к «пику».

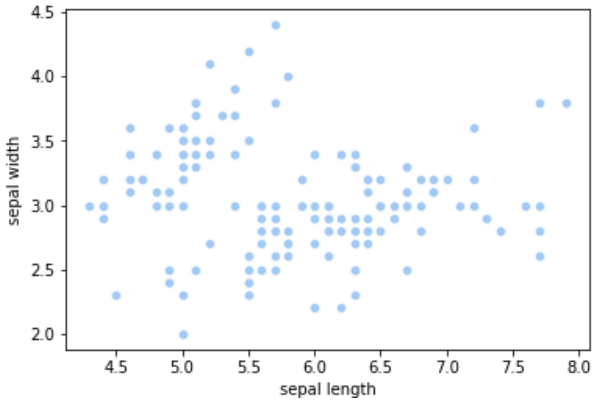

Рисунок 6. Диаграмма рассеяния для пары признаков sepallength и sepalwidth набора данных «Ирисы Фишера» (The Iris Dataset)

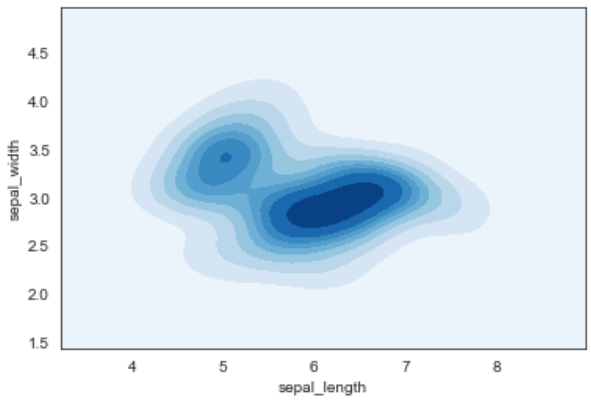

Рисунок 8. Контурный график для пары признаков sepallength и sepalwidth набора данных «Ирисы Фишера» (The Iris Dataset)

Диаграммы рассеяния удобно использовать, если в наборе данных имеется небольшое число объектов. Для наборов данных с сотнями тысяч и миллионов объектов диаграмма рассеяния будет очень плотной, в этом случае удобнее использовать *график с шестиугольной сеткой* (hexagonal binning) для отображения связи между двумя значениями признаков (рис. 9). При работе с шестиугольной сеткой данные группируют в шестиугольные сегменты, показывая цветом число записей в конкретном сегменте.

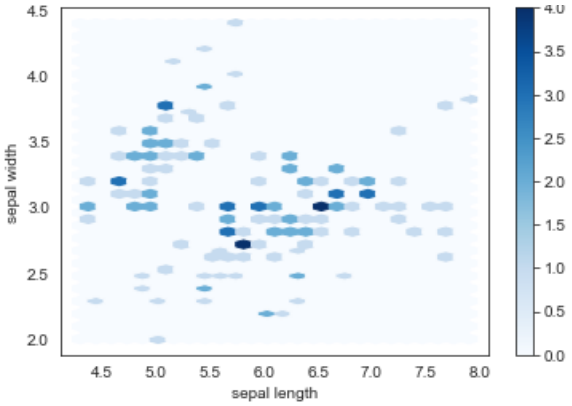

Рисунок 9. Визуализация связи для пары признаков sepallength и sepalwidth набора данных «Ирисы Фишера» (The Iris Dataset) с использованием графика с шестиугольной сеткой

Тепловые карты, графики с шестиугольной сеткой и контурные графики являются естественными аналогами гистограмм и графиков плотности.

6. Разведочный анализ для наборов данных с количественными признаками на языке Python

Для демонстрации принципов работы библиотек и модулей, реализующих инструменты разведочного анализа данных, будем использовать Jupyter Notebook.

Для работы с метриками центральной тенденции и вариативности для количественных признаков можно воспользоваться различными программными библиотеками и модулями. В частности, можно использовать программные инструменты, предложенные по ссылкам:

https://pypi.org/project/statistics/,
https://numpy.org/ (https://pypi.org/project/numpy/),
https://scipy.org/ (https://pypi.org/project/scipy/).

Во многих рассматриваемых ниже функциях, предназначенных для решения задач в области статистики, в качестве входного параметра присутствует параметр `data`, определяющий набор данных, к которому применяется функция. Набор данных `data` может содержать значения одного или нескольких

признаков. Искомая статистика, определяемая функцией, вычисляется для каждого признака.

6.1. Разведочный анализ с использованием модуля statistics

Для установки программного модуля `statistics`[9] необходимо выполнить команду:

```
pip install statistics
```

Подробная документация по всем функциям, реализованным в программном модуле statistics, приведена по ссылке:

https://docs.python.org/3/library/statistics.html.

Модуль `statistics` предоставляет функции для вычисления математической статистики числовых данных (таблицы 1 – 3), он нацелен на уровень графических и научных калькуляторов и не конкурирует с такими библиотеками, как `NumPy`, `SciPy`, или с проприетарными полнофункциональными статистическими пакетами, предназначенными для профессиональных статистиков, такими как `Minitab`, `SAS` и `Matlab`.

Таблица 1. Функции для вычисления метрик центральной тенденции

Функция и её аргументы (входные параметры)	Описание
`mean (data)`	Возвращает *среднее арифметическое*. **Входной параметр:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков. **Выходной параметр:** среднее арифметическое. Вычисляется для каждого признака.
`fmean (data)`	Возвращает *быстрое среднее арифметическое с плавающей запятой*, вычисленное на основе значений данных, предварительно преобразованных в числа с плавающей запятой и представляющее собой число с плавающей запятой. **Входной параметр:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков. **Выходной параметр:** быстрое среднее арифметическое с плавающей запятой. Вычисляется для каждого признака. `fmean()` работает быстрее, чем `mean()`.

Продолжение таблицы 1

`geometric_mean (data)`	Возвращает *среднее геометрическое,* вычисленное на основе значений данных, предварительно преобразованных в числа с плавающей запятой и представляющее собой число с плавающей запятой. Среднее геометрическое указывает центральную тенденцию (типичное значение) данных, используя произведение значений. **Входной параметр:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков. **Выходной параметр:** среднее геометрическое. Вычисляется для каждого признака.
`harmonic_mean (data, weights=None)`	Возвращает *среднее гармоническое*. **Входные параметры:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков; `weights` – весовые коэффициенты для значений признаков; если параметр `weights` не определен, предполагается равное взвешивание. **Выходной параметр:** среднее гармоническое. Вычисляется для каждого признака.
`median (data)`	Возвращает *медиану (среднее значение)* числовых данных, используя обычный метод «среднее значение двух». Если число значений данных нечетное, возвращается среднее значений данных. Если число значений данных четное, медиана интерполируется путем получения среднего значения двух средних значений. **Входной параметр:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков. **Выходной параметр:** медиана. Вычисляется для каждого признака.
`median_low (data)`	Возвращает *нижнюю медиану* числовых данных. Нижняя медиана всегда является членом набора данных. Если число значений данных нечетное, возвращается среднее значение. Если число значений данных четное, возвращается меньшее из двух средних значений. Целесообразно использовать нижнюю медиану, если данные дискретны и требуется, чтобы медиана была фактическим значением данных, а не интерполяцией.

Продолжение таблицы 1

	Входной параметр: `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков. **Выходной параметр:** нижняя медиана. Вычисляется для каждого признака.
`median_high (data)`	Возвращает *верхнюю медиану* числовых данных. Верхняя медиана всегда является членом набора данных. Если число значений данных нечетное, возвращается среднее значение. Если число значений данных четное, возвращается большее из двух средних значений. Целесообразно использовать верхнюю медиану, если данные дискретны и требуется, чтобы медиана была фактическим значением данных, а не интерполяцией. **Входной параметр:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков. **Выходной параметр:** верхняя медиана. Вычисляется для каждого признака.
`median_grouped (data, interval=1)`	Возвращает *медиану сгруппированных непрерывных данных, рассчитанную как 50-й процентиль*, используя интерполяцию. **Входные параметры:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков; `interval`– интервал, по умолчанию равный 1. Изменение интервала естественным образом изменяет интерполяцию. **Выходной параметр:** медиана сгруппированных непрерывных данных, рассчитанная как 50-й процентиль. Вычисляется для каждого признака.
`mode (data)`	Возвращает *одиночную моду (одно наиболее частое значение из дискретных или номинальных данных*). Мода (если она существует) является наиболее типичной величиной и служит мерой центральной тенденции. Если имеется несколько мод с одинаковой частотой, функция возвращает моду, первой обнаруженную в данных. **Входной параметр:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков. **Выходной параметр:** Одиночная мода. Вычисляется для каждого признака.

Продолжение таблицы 1

`multimode (data)`	Возвращает *список мод (наиболее часто встречающихся значений) в порядке их первого появления* в данных. Возвращает более одного результата, если есть несколько мод. **Входной параметр:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков. **Выходной параметр:** список мод. Вычисляется для каждого признака.
`quantiles (data, *, n=4, method='exclusive')`	Возвращает *квантили (разбиение данных на интервалы с равной вероятностью).* Реализует разбиение данных на `n` непрерывных интервалов с равной вероятностью. Возвращает список из `n-1` точек разбиения, разделяющих интервалы. **Входные параметры:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков; `n` – число разбиений данных на интервалы (устанавливают `n` равным 4 для квартилей (это значение установлено по умолчанию); устанавливают `n` равным 10 для децилей; устанавливают `n` равным 100 для процентилей, что дает 99 точек отсечения, разделяющих данные на 100 групп одинакового размера; `method` – метод вычисления квантилей; может принимать значения `'exclusive'` и `'inclusive'`; при этом значения разбиения линейно интерполируются из двух ближайших значений данных. Метод вычисления квантилей может варьироваться в зависимости от того, содержит или не содержит набор данных `data` самые маленькие и самые большие возможные значения из совокупности (например, 0% и 100% в случае работы с процентами). Метод `'exclusive'` – это метод, используемый по умолчанию, он применяется для выборки данных из совокупности, которая может иметь более экстремальные значения, чем найденные в выборке. Доля совокупности, попадающая ниже `i`-го из `m` отсортированных значений данных, вычисляется как `i/(m+1)`. Метод `'inclusive'` используется для описания данных совокупности или выборки, для которой, известно, что она включает самые экстремальные значения из совокупности.

Окончание таблицы 1

	Минимальное значение в наборе данных `data` рассматривается как 0-й процентиль, а максимальное значение – как 100-й процентиль. Доля совокупности, попадающая ниже `i`-го из `m` отсортированных значений данных, вычисляется как `(i-1)/(m-1)`. **Выходной параметр:** квантили. Вычисляются для каждого признака.

Таблица 2. Функции для вычисления метрик вариабельности

Функция и её аргументы (входные параметры)	Описание
`pvariance (data, mu=None)`	Возвращает *дисперсию для генеральной совокупности данных*. Большая дисперсия указывает на то, что данные разбросаны; небольшая дисперсия указывает на то, что данные сгруппированы близко к среднему значению. **Входные параметры:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков; `mu` – среднее значение данных; может использоваться для вычисления дисперсии относительно значения данных, которое не является средним; если этот аргумент отсутствует, среднее арифметическое значений данных вычисляется автоматически. **Выходной параметр:** дисперсия для генеральной совокупности данных. Функцию `pvariance()` целесообразно использовать, чтобы вычислить дисперсию для генеральной совокупности данных, для вычисления дисперсии по выборке лучше использовать функцию `variance()`. Вычисляется для каждого признака.
`pstdev (data, mu=None)`	Возвращает *стандартное отклонение для генеральной совокупности данных (квадратный корень из дисперсии генеральной совокупности).* **Входные параметры:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков; `mu` – среднее значение данных; может использоваться для вычисления дисперсии относительно значения данных, которое не является средним; если этот аргумент отсутствует, среднее арифметическое значений данных вычисляется автоматически.

Окончание таблицы 2

	Выходной параметр: стандартное отклонение для генеральной совокупности данных. Функцию `pstdev()` целесообразно использовать, чтобы вычислить дисперсию для генеральной совокупности данных, для вычисления дисперсии по выборке лучше использовать функцию `stdev()`. Вычисляется для каждого признака.
`variance (data, xbar=None)`	Возвращает *выборочную дисперсию*. Большая дисперсия указывает на то, что данные разбросаны; небольшая дисперсия указывает на то, что данные сгруппированы близко к среднему значению. **Входные параметры:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков; `xbar` – среднее значение данных; может использоваться для вычисления дисперсии относительно значения данных, которое не является средним; если этот аргумент отсутствует, среднее арифметическое значений данных вычисляется автоматически. **Выходной параметр:** выборочная дисперсия. Функцию `variance()` используют для вычисления дисперсии для выборки из генеральной совокупности данных. Вычисляется для каждого признака.
`stdev (data, xbar=None)`	Возвращает *выборочное стандартное отклонение (квадратный корень из выборочной дисперсии).* **Входные параметры:** `data` – последовательность или итерируемый объект; определяет анализируемый набор данных; может представлять значения одного или нескольких признаков; `xbar` – среднее значение данных; может использоваться для вычисления дисперсии относительно значения данных, которое не является средним; если этот аргумент отсутствует, среднее арифметическое значений данных вычисляется автоматически. **Выходной параметр:** выборочное стандартное отклонение. Вычисляется для каждого признака.

Таблица 3. Функции для вычисления статистик взаимосвязи пары признаков

Функция и её аргументы (входные параметры)	Описание
`covariance (x, y, /)`	Возвращает *выборочную ковариацию двух признаков* (входных параметров) `x` и `y`. Ковариация – мера совместной изменчивости пары двух признаков (входных параметров) `x` и `y`.

Окончание таблицы 3

	Входные параметры: x и y – признаки, для которых рассчитывается ковариация. **Выходной параметр:** выборочная ковариация.
`correlation (x, y, /)`	Возвращает *коэффициент корреляции Пирсона для двух признаков* (входных параметров) x и y. Коэффициент корреляции Пирсона принимает значения от «–1» до «+1». Он измеряет силу и направление линейной зависимости: «+1» означает очень сильную, положительную линейную зависимость, «–1» очень сильную, отрицательную линейную зависимость, а «0» означает отсутствие линейной зависимости. **Входные параметры:** x и y – признаки, для которых рассчитывается ковариация. **Выходной параметр:** коэффициент корреляции Пирсона.
`linear_regression (x, y, /)`	Возвращает *коэффициент наклона* (k) *и смещение* (bias) *для простой линейной регрессии*, вычисленные с применением метода наименьших квадратов. Простая линейная регрессия описывает отношение между независимым признаком x и зависимым признаком y с точки зрения линейной функции: y = k * x + bias + шум, где k и bias – оцениваемые параметры регрессии, а шум – изменчивость данных, которая не была объяснена линейной регрессией (он равен разнице между прогнозируемым и фактическим значениями зависимого признака). **Входные параметры:** x и y – признаки, для которых рассчитывается ковариация. **Выходные параметры:** коэффициент наклона и смещение для простой линейной регрессии.

6.2. Разведочный анализ с использованием библиотеки NumPy

Для установки программной библиотеки NumPy [10] необходимо выполнить команду:

```
pip install numpy
```

NumPy (https://numpy.org/) – программная библиотека с открытым исходным, которая предоставляет объект многомерного массива, различные производные объекты (такие как маскированные массивы и матрицы) и набор подпрограмм для быстрых операций с массивами, включая математические и логические операции, манипуляции с фигурами, сортировку, выбор (поиск), ввод-

вывод, дискретные преобразования Фурье, основы линейной алгебры, основные статистические операции, случайное моделирование и многое другое.

Модуль `Statistics` из программной библиотеки `NumPy` содержит функции, позволяющие вычислить порядковые статистики, метрики среднего положения и вариативности, оценить корреляцию, выполнить расчеты для построения гистограмм.

Подробная документация по функциям, предназначенным для решения задач в области статистики, приведена по ссылке:

https://numpy.org/devdocs/reference/routines.statistics.html.

Наиболее часто используются представленные в таблицах 4 – 6 функции.

Таблица 4. Функции для расчета метрик центральной тенденции и вариабельности

Функция и её аргументы (входные параметры)	Описание
`median (data[, axis, out, overwrite_input, keepdims])`	Возвращает *медиану* вдоль выбранного измерения массива. **Входные параметры:** `data` (тип: `array_like`) – массив или объект, который можно преобразовать в массив; `axis` (тип: `int`, последовательность из `int` или `None`) – измерение (или измерения), по которому (по которым) рассчитывается статистика; по умолчанию (если `axis` отсутствует) вычисления проводятся по всему массиву`data`; последовательность измерений поддерживается с версии 1.9.0; `out` (тип: `ndarray`) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости; `overwrite_input` (тип: `bool`) – параметр, который принимает 2 значения: `True` или `False`; если `overwrite_input` имеет значение `True`, то разрешено использование памяти массива `data` для вычислений:массив будет изменен вызовом функции,что позволит сэкономить память, когда не нужно сохранять содержимое входного массива; по умолчанию `overwrite_input`имеет значение `False`; `keepdims` (тип: `bool`) – параметр, который принимает 2 значения: `True` или `False`; если параметр `keepdims` имеет значение `True`, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве.

Продолжение таблицы 4

	Выходной параметр: median (тип: ndarray) – массив, содержащий результат; если входные данные содержат целые числа или числа с плавающей запятой меньшие, чем float64, то тип выходных данных – np.float64,в противном случае тип выходных данных совпадает с типом входных данных; если параметр out специфицирован, то возвращается ссылка на выходной массив.
average (data[, axis, weights, returned, keepdims])	Возвращает *взвешенное среднее* вдоль выбранной оси массива. **Входные параметры:** data (тип: array_like) – массив или объект, который можно преобразовать в массив; axis (тип: int, последовательность из int или None) – измерение (или измерения), по которому (по которым) рассчитывается статистика; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data; последовательность измерений поддерживается с версии 1.9.0; weights (тип: array_like) – массив весов, связанных со значениями в массиве data:каждое значение в массиве data вносит свой вклад в среднее значение в соответствии с соответствующим весом;массив весов может быть одномерным (в этом случае его длина должна быть равна размеру массива data по заданному измерению axis) или иметь ту же форму, что и массив data;если weights=None, то предполагается, что все данные в массиве data имеют вес, равный единице; взвешенное среднее вычисляется как avg=sum(a*weights)/sum(weights), при этом sum(weights) не должна быть равна нулю; returned (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр returned имеет значение True, функция возвращается кортеж (average, sum_of_weights), в противном случае возвращается только среднее значение; по умолчанию параметр returnedимеет значение False.Если weights=None, величина sum_of_weights эквивалентна числу элементов, по которым берется среднее значение; keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве.

Продолжение таблицы 4

	Выходной параметр: retval, [sum_of_weights] (тип: array_type или double) – среднее значение по выбранной оси.Если returned равно True, функция возвращает кортеж со средним значением в качестве первого элемента и суммой весов в качестве второго элемента; параметр sum_of_weights имеет тот же тип, что и retval.
mean (data[, axis, dtype, out, keepdims, where])	Возвращает *среднее арифметическое* вдоль выбранной оси массива. **Входные параметры:** data (тип: array_like) – массив или объект, который можно преобразовать в массив; axis (тип: int, последовательность из int или None) – измерение (или измерения), по которому (по которым) рассчитывается статистика; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data; последовательность измерений поддерживается с версии 1.9.0; dtype (тип: data-type) – тип результата при вычислении среднего значения:для целочисленных входных данных используется тип float64;для входных данных с плавающей запятой используется тот же тип, что и тип у массива data; out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости; keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве; where (тип: array_like,при этом тип элементов –bool) – массив, показывающий, какие элементы массива data следует использовать при вычислении среднего. **Выходной параметр:** m (тип: ndarray) – массив, содержащий средние значения, если out=None; в противном случае – ссылка на выходной массив.
std (data[, axis, dtype, out, ddof, keepdims, where])	Возвращает *стандартное отклонение* вдоль выбранной оси массива. **Входные параметры:** Data (тип: array_like) – массив или объект, который можно преобразовать в массив;

Продолжение таблицы 4

	axis (тип: int, последовательность из int или None) – измерение (или измерения), по которому (по которым) рассчитывается статистика; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data; последовательность измерений поддерживается с версии 1.9.0; dtype (тип: data-type) – тип результата при вычислении среднего значения:для целочисленных входных данных используется тип float64;для входных данных с плавающей запятой используется тот же тип, что и тип у массива data; out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости; ddof (тип: int) – число степеней свободы;в расчетах используется делитель n- ddof, где n – число элементов;по умолчанию параметр ddof равен нулю; keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве; where (тип: array_like,при этом тип элементов – bool) – массив, показывающий, какие элементы массива data следует использовать при вычислении стандартного отклонения. **Выходной параметр:** standard_deviation (тип: ndarray) – массив, содержащий стандартные отклонения, если out=None; в противном случае – ссылка на выходной массив.
var (data[, axis, dtype, out, ddof, keepdims, where])	Возвращает *дисперсию* вдоль выбранной оси массива. **Входные параметры:** data (тип: array_like) – массив или объект, который можно преобразовать в массив; axis (тип: int, последовательность из int или None) – измерение (или измерения), по которому (по которым) рассчитывается статистика; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data; последовательность измерений поддерживается с версии 1.9.0; dtype (тип: data-type) – тип результата при вычислении среднего значения:для целочисленных входных данных используется тип float64;для входных данных с плавающей запятой используется тот же тип, что и тип у массива data;

Окончание таблицы 4

	out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости; ddof (тип: int) – число степеней свободы;в расчетах используется делитель n- ddof, где n – число элементов;по умолчанию параметр ddof равен нулю; keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве; where (тип: array_like,при этом тип элементов –bool) – массив, показывающий, какие элементы массива data следует использовать при вычислении дисперсии. **Выходной параметр:** variance (тип: ndarray) – массив, содержащий значения дисперсии, если out=None; в противном случае – ссылка на выходной массив.

Таблица 5. Функции для расчета порядковых статистик

Функция и её аргументы (входные параметры)	Описание
percentile (data, q[, axis, out, overwrite_input, method, keepdims, *, interpolation])	Возвращает *q-процентиль* вдоль выбранного измерения массива. **Входные параметры:** data (тип: array_like) – массив или объект, который можно преобразовать в массив; q (тип: array_like типаfloat) – процентиль или последовательность процентилей, которые должны быть в диапазоне от 0 до 100 включительно; axis (тип: int, кортеж из int или None) – измерение, по которому ищутся пики; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data; out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости; overwrite_input (тип: bool) – параметр, который принимает 2 значения: True или False; если overwrite_input имеет значение True, то разрешено использование памяти массива data для вычислений:массив будет изменен вызовом функции,что позволит сэкономить память, когда не нужно сохранять содержимое входного массива; по умолчанию overwrite_input имеет значение False;

Продолжение таблицы 5

	Method (тип:str) – метод, используемый для оценки процентиля:'inverted_cdf','averaged_inverted_cdf','closest_observation','interpolated_inverted_cdf','hazen','weibull','linear' (по умолчанию), 'median_unbiased','normal_unbiased',кроме того, имеются 3 вариации метода'linear':'lower', 'higher','midpoint' и 'nearest'; keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве; interpolation (тип:str) – устаревшее имя для параметра method. **Выходной параметр:** percentile (тип: scalar или ndarray) – массив, содержащий результат; если q – одиночный процентиль и axis=None, то результат – скаляр; если дано несколько процентилей, то первое измерение результата соответствует процентилям, а другие измерения – это измерения, которые остались после выполнения действий над массивом data; если входные данные содержат целые числа или числа с плавающей запятой меньшие, чем float64, то тип выходных данных – np.float64,в противном случае тип выходных данных совпадает с типом входных данных; если параметр out специфицирован, то возвращается ссылка на выходной массив.
quantile (data, q[, axis, out, overwrite_input, method, keepdims, *, interpolation])	Возвращает *q-квантиль* вдоль выбранного измерения массива. **Входные параметры:** data (тип: array_like) – массив или объект, который можно преобразовать в массив; q (тип: array_like типаfloat) – квантиль или последовательность квантилей, которые должны быть в диапазоне от 0 до 1 включительно; axis (тип: int, кортеж из int или None) – измерение, по которому ищутся пики; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data; out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости;

Продолжение таблицы 5

	overwrite_input (тип: bool) – параметр, который принимает 2 значения: True или False; если overwrite_inputимеет значение True, то разрешено использование памяти массива data для вычислений: массив будет изменен вызовом функции, что позволит сэкономить память, когда не нужно сохранять содержимое входного массива; по умолчанию overwrite_inputимеет значение False; method(тип:str) – метод, используемый для оценки процентиля: 'inverted_cdf','averaged_inverted_cdf','closest_observation','interpolated_inverted_cdf','hazen','weibull','linear' (по-умолчанию), 'median_unbiased','normal_unbiased', кроме того, имеются 3 вариации метода 'linear':'lower','higher','midpoint'и'nearest'; keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве; interpolation (тип:str) – устаревшее имя для параметра method. **Выходной параметр:** quantile (тип: scalar или ndarray) – массив, содержащий результат; если q – одиночный квантиль и axis=None, то результат – скаляр; если дано несколько квантилей, то первое измерение результата соответствует квантилям, а другие измерения – это измерения, которые остались после выполнения действий над массивом data; если входные данные содержат целые числа или числа с плавающей запятой меньшие, чем float64, то тип выходных данных – np.float64,в противном случае тип выходных данных совпадает с типом входных данных; если параметр out специфицирован, то возвращается ссылка на выходной массив.
ptp(a[, axis, out, keepdims])	Возвращает *диапазон значений (максимум – минимум)* вдоль выбранного измерения массива. **Входные параметры:** data (тип: array_like) – массив или объект, который можно преобразовать в массив; axis (тип: int, кортеж из int или None) – измерение, по которому ищутся пики; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data;

Окончание таблицы 5

	out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости; keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве. **Выходной параметр:** ptp (тип: ndarray) – массив, содержащий результат; если параметр out специфицирован, то возвращается ссылка на выходной массив.

Таблица 6. Функции для оценки корреляции

Функция и её аргументы (входные параметры)	Описание
corrcoef (x[, y, rowvar, dtype])	Возвращает *коэффициенты корреляции Пирсона*. **Входные параметры:** x (тип: array_like) – одномерный или двумерный массив, содержащий несколько переменных и наблюдений: каждая строка массива x представляет собой переменную, а каждый столбец – одно наблюдение всех этих переменных; y (тип: array_like) – дополнительный массив, содержащий несколько переменных и наблюдений: каждая строка массива x представляет собой переменную, а каждый столбец – одно наблюдение всех этих переменных; массив y имеет те же размеры, что и массив x; rowvar (тип: bool) – параметр, который принимает 2 значения: True или False; если rowvar имеет значение True (по умолчанию), то каждая строка представляет собой переменную с наблюдениями в столбцах; в противном случае массив транспонируется: каждый столбец представляет собой переменную, а строки содержат наблюдения; dtype (тип: data-type)– тип данных результата; по умолчанию возвращаемый тип данных будет иметь точность не менее numpy.float64. **Выходной параметр:** R (тип: ndarray) – массив коэффициентов корреляции переменных.

Продолжение таблицы 6

correlate (a, v[, mode])	Возвращает *взаимную корреляцию двух одномерных последовательностей*. **Входные параметры:** a, v(тип: array_like) – входные последовательности; mode ({'valid','same','full'}) – параметр, определяющий тип свертки данных; значение 'full' используется по умолчанию. **Выходной параметр:** out (тип: ndarray) – массив, характеризующий дискретную взаимную корреляцию массивов a и v.
cov (m[, y, rowvar, bias, ddof, fweights, aweights, dtype])	Возвращает *ковариационную матрицу*. **Входные параметры:** m (тип: array_like) – одномерный или двумерный массив, содержащий несколько переменных и наблюдений: каждая строка массива x представляет собой переменную, а каждый столбец – одно наблюдение всех этих переменных; y (тип: array_like) – дополнительный массив, содержащий несколько переменных и наблюдений: каждая строка массива x представляет собой переменную, а каждый столбец – одно наблюдение всех этих переменных; массив y имеет те же размеры, что и массив x; rowvar (тип: bool) – параметр, который принимает 2 значения: True или False; если rowvar имеет значение True (по умолчанию), то каждая строка представляет собой переменную с наблюдениями в столбцах; в противном случае массив транспонируется: каждый столбец представляет собой переменную, а строки содержат наблюдения; dtype (тип: data-type) – тип данных результата; по умолчанию возвращаемый тип данных будет иметь точность не менее numpy.float64. bias (тип: bool) – параметр, который принимает 2 значения: True или False; если bias=False (по умолчанию), нормализация осуществляется с (n−1), где n – число приведенных наблюдений (непредвзятая оценка);если bias=True, то нормализация осуществляется с n; эти значения можно переопределить с помощью параметра ddof в версиях numpy >= 1.5; ddof (тип: int) – параметр, принимающий по умолчанию значение None; если ddof=1, то функция возвращает несмещенную оценку, даже если указаны как fweights, так и aweights; если ddof=0, то функция возвращает простое среднее;

Окончание таблицы 6

	fweights (тип: array_like, int) – одномерный массив целочисленных частотных весов, определяющий число повторений каждого вектора наблюдения; aweights (тип: int) – одномерный массив весов векторов наблюдения;эти относительные веса обычно велики для наблюдений, считающихся «важными», и малы для наблюдений, считающихся менее «важными»;если ddof=0, массив весов можно использовать для присвоения вероятностей векторам наблюдения; dtype (тип: int) – тип данных результата; по умолчанию возвращаемый тип данных будет иметь точность не меньше, чем numpy.float64. **Выходной параметр:** out (тип: ndarray) – ковариационная матрица переменных.

Таблица 7. Функции для расчета гистограмм

Функция и её аргументы (входные параметры)	Описание
histogram (a[, bins, range, density, weights])	Возвращает *гистограмму*. **Входные параметры:** data (тип: array_like) – массив или объект, который можно преобразовать в массив; гистограмма вычисляется по всему массиву; bins (тип: int или последовательность скаляров или str) – параметр, определяющий бины (интервалы); если bins имеет значение int, то параметр определяет число бинов равной ширины в заданном диапазоне (по умолчанию 10);если bins –последовательность, то параметр определяет монотонно увеличивающийся массив ребер бинов, включая крайнее правое ребро, что допускает неравномерную ширину бинов; если bins – строка, то параметр определяет метод, используемый для вычисления оптимальной ширины бина; range (тип: (float, float)) – нижняя и верхняя граница бинов (значения вне диапазона игнорируются); если параметр не специфицирован, то диапазон определяется как (a.min(), a.max()); density (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр density равен False, результат будет содержать число значений в каждом бине; если параметр densityравен True, результат будет равен значению функции плотности вероятности в бине, нормализованной таким образом, что интеграл по диапазону равен 1;

Окончание таблицы 7

	weights (тип: array_like) – массив весов того же размера, что и массив data;если для параметра density установлено значение True, веса нормализуются, так что интеграл плотности по диапазону остается равным 1; **Выходной параметр:** hist (тип: array) – массив, содержащий значения гистограммы; bin_edges (тип: array of dtype float) – массив ребер бинов.
bincount (x, /[, weights, minlength])	Возвращает *число вхождений каждого значения в массиве неотрицательных целых чисел*. x (тип: array_like) – массив или объект, который можно преобразовать в массив; weights (тип: array_like) – массив весов тех же размеров, что и массив x; minlength (тип: int) – минимальное число бинов для выходного массива. out (тип: ndarrayofints) – массив с результатами бинирования массива x;длина массива out равна np.amax(x)+1.

Кроме функций, приведенных в таблице 4, в модуле Statistics есть функции для расчета метрик центральной тенденции и вариабельности, игнорирующие NaN значения: nanmedian, nanmean, nanstd и nanvar.

Кроме функций, приведенных в таблице 5, в модуле Statistics есть функции для расчета порядковых статистик, игнорирующие NaN значения: nanpercentile иnanquantile.

Кроме функций, приведенных в таблице 7, в модуле Statistics есть функции для расчета двумерных и многомерных гистограмм – histogram2d и histogramdd, а также функция для вычисления только ребер бинов, используемых функцией гистограммы, – histogram_bin_edges и функция для вычисления индексов бинов, которым принадлежит значения данных во входном массиве, – digitize.

6.3. Разведочный анализ с использованием библиотеки SciPy

Для установки программной библиотеки SciPy[11] необходимо выполнить команду:

```
pip install scipy
```

SciPy (https://scipy.org/) – программная библиотека с открытым исходным кодом для решения различных задач в сферах математики, науки и техники.

SciPy предоставляет программные реализации алгоритмов для задач оптимизации, интегрирования, интерполяции, статистики, поиска решений алгебраических и дифференциальных уравнений и многих других классов задач.

Модуль stats из программной библиотеки SciPy содержит большое число функций, например, функции для моделирования вероятностных распределений, функции для вычисления сводной и частотной статистики, корреляционные функции и функции для выполнения статистических тестов, функции маскированной статистики, функции оценки плотности ядра, а также подмодуль квази-Монте-Карло и многое другое.

Подробная документация по функциям, предназначенным для решения задач в области статистики, приведена по ссылке:

https://docs.scipy.org/doc/scipy/reference/stats.html.

Следует отметить, что есть темы в области статистики, которые выходят за рамки SciPy и охватываются другими пакетами. Так, пакет statsmodels (https://www.statsmodels.org/stable/index.html) предоставляет классы и функции для оценки множества различных статистических моделей, а также для проведения статистических тестов и исследования статистических данных (в частности, имеются инструменты для анализа временных рядов и для работы с регрессионными моделями); пакет pandas (https://pandas.pydata.org/)предоставляет инструменты для работы с табличными данными, функциональностью временных рядов, а также интерфейсы к другим статистическим языкам; пакет PyMC (https://www.pymc.io/projects/docs/en/stable/learn.html) предоставляет инструменты для байесовского статистического моделирования и вероятностного машинного обучения; пакет scikit-learn (https://scikit-learn.org/stable/) предоставляет инструменты для решения задач классификации, регрессии и выбора модели; пакет seaborn (https://seaborn.pydata.org/) позволяет осуществить визуализацию статистических данных.

В таблице 8 приведена информация по функциям, позволяющим вычислить некоторые наиболее часто используемые статистические метрики.

Таблица 8. Функции для вычисления статистических метрик

Функция и её аргументы (входные параметры)	Описание
describe (data [, axis, ddof, bias, nan_policy])	Возвращает *несколько описательных статистик*. **Входные параметры:** data (тип: array_like) – набор данных; axis (тип: int или None) – ось, по которой рассчитывается статистика (по умолчанию: 0); если None, то вычисления проводятся по всему набору данных data;

Продолжение таблицы 8

	ddof (тип: int) – число степеней свободы (используется только для дисперсии);по умолчанию ddof=1; bias (тип: bool) – параметр, который принимает 2 значения: True или False; если bias принимает значение False, то расчеты асимметрии и эксцесса корректируются на статистическую погрешность; nan_policy ({'propagate', 'raise', 'omit'}) параметр, который определяет, что делать, если входные данные содержат значения nan; при этом возможны следующие значения аргумента nan_policy: 'propagate' (используется по умолчанию): функция возвращает значения NaN; 'raise': функция генерирует ошибку; 'omit': функция выполняет вычисления, игнорируя значения NaN. **Выходные параметры:** nobs (тип: int или ndarray из данных типа int) – число наблюдений (длина данных по оси axis); если в качестве nan_policy выбрано 'omit', длина вдоль каждого среза оси считается отдельно; minmax (тип: tupleиз данных типа ndarray или float) – минимальное и максимальное значения набора данных data вдоль заданной оси; mean (тип: ndarray или float) – среднее арифметическое набора данных data вдоль заданной оси; variance (тип: ndarray или float) – несмещенная дисперсия набора данных data по заданной оси; в знаменателе – число наблюдений минус один; skewness (тип: ndarray или float) – асимметрия вдоль заданной оси, основанная на расчете моментов со знаменателем, равным числу наблюдений, т.е. без коррекции степеней свободы. Kurtosis (тип: ndarray или float) – эксцесс (по Фишеру) набора данных data вдоль заданной оси; эксцесс нормализуется так, что для нормального распределения он равен нулю; степени свободы не используются. **Обращение к выходным параметрам**: по индексу. Например, при указании индекса 2, будет выведено mean.
gmean (data [, axis, dtype, weights, nan_policy, keepdims])	Возвращает *взвешенное среднее геометрическое* вдоль указанной оси axis. **Входные параметры:** data (тип: array_like) – набор данных; axis (тип: int или None) – ось, вдоль которой рассчитывается статистика (по умолчанию: 0); если None, входные данные будут обработаны перед вычислением статистики (статистика каждой оси из входных данных отображается в соответствующем элементе выходных данных);

Продолжение таблицы 8

	dtype (тип: dtype) – тип возвращаемого массива (аккумулятора), в котором суммируются элементы; если dtype не указан, по умолчанию используется dtype для набора данных data (с учетом особенностей целого типа данных используемой платформы, если тип набора данных data целый); weights (тип: array_like) – массив весов, который может быть либо одномерным (в этом случае его длина должна быть равна размеру набора данных data по заданной оси), либо иметь те же размеры, что и набор данных data; по умолчанию установлено значение None, при этом каждому весу присвоено значение, равное 1.0; nan_policy (propagate, omit, raise) – параметр, определяющий порядок действий при обнаружении NaN во входных данных: propagate: при возникновении NaN вдоль оси, по которой вычисляется статистика, соответствующее выходное значение будет NaN; omit: все NaN будут пропущены при осуществлении вычислений; если вдоль оси, по которой вычисляется статистика, данных будет недостаточно, соответствующее выходное значение будет NaN; raise: при возникновении NaN будет выведена ошибка ValueError; keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в массиве исходного набора данных data. **Выходной параметр:** gmean (тип: ndarray) – массив (аккумулятор) типа dtype.
hmean (data [, axis, dtype, weights, nan_policy, keepdims])	Возвращает *взвешенное среднее гармоническое* вдоль указанной оси axis**.** **Входные параметры:** data (тип: array_like) – набор данных; axis (тип: int или None) – ось, вдоль которой рассчитывается статистика (по умолчанию: 0); если None, входные данные будут обработаны перед вычислением статистики (статистика каждой оси из входных данных отображается в соответствующем элементе выходных данных);

Продолжение таблицы 8

	dtype (тип: dtype) – тип возвращаемого массива (аккумулятора), в котором суммируются элементы; если dtype не указан, по умолчанию используется dtype для набора данных data (с учетом особенностей целого типа данных используемой платформы, если тип значений в наборе данных data целый); weights (тип: array_like) – массив весов, который может быть либо одномерным (в этом случае его длина должна быть равна размеру набора данных data по заданной оси), либо иметь те же размеры, что и набор данных data; по умолчанию установлено значение None, при этом каждому весу присвоено значение, равное 1.0; nan_policy (propagate, omit, raise) – параметр, определяющий порядок действий при обнаружении NaN во входных данных: propagate: при возникновении NaN вдоль оси, по которой вычисляется статистика, соответствующее выходное значение будет NaN; omit: все NaN будут пропущены при осуществлении вычислений; если вдоль оси, по которой вычисляется статистика, данных будет недостаточно, соответствующее выходное значение будет NaN; raise: при возникновении NaN будет выведена ошибка ValueError; keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в массиве исходного набора данных data. **Выходной параметр:** hmean (тип: ndarray) – массив (аккумулятор) типа dtype.
kurtosis(data [, axis, fisher, bias, nan_policy, keepdims])	Возвращает *эксцесс* (Фишера или Пирсона) набора данных. **Входные параметры:** data (тип: array_like) – набор данных; axis (тип: int или None) – ось, вдоль которой рассчитывается статистика (по умолчанию: 0); если None, входные данные будут обработаны перед вычислением статистики (статистика каждой оси из входных данных отображается в соответствующем элементе выходных данных);

Продолжение таблицы 8

	fisher (тип:bool) – параметр, который принимает 2 значения: True или False; если fisher принимает значение True, то вычисляется эксцесс Фишера, если fisher принимает значение False, то вычисляется эксцесс Пирсона; bias (тип: bool) – параметр, который принимает 2 значения: True или False; если bias принимает значение False, то расчеты эксцесса корректируются на статистическую погрешность; nan_policy (propagate, omit, raise) – параметр, определяющий порядок действий при обнаружении NaN во входных данных: propagate: при возникновении NaN вдоль оси, по которой вычисляется статистика, соответствующее выходное значение будет NaN; omit: все NaN будут пропущены при осуществлении вычислений; если вдоль оси, по которой вычисляется статистика, данных будет недостаточно, соответствующее выходное значение будет NaN; raise: при возникновении NaN будет выведена ошибка ValueError; keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что измерений в результирующем массиве, что и в массиве исходного набора данных data. **Выходной параметр:** kurtosis (тип: array) – эксцесс значений по оси, равный NaN, где все значения одинаковые.
skew(data [, axis, bias, nan_policy, keepdims])	Возвращает *асимметрию* набора данных. **Входные параметры:** data (тип: array_like) – набор данных; axis (тип: int или None) – ось, вдоль которой рассчитывается статистика (по умолчанию: 0); если None, входные данные будут обработаны перед вычислением статистики (статистика каждой оси из входных данных отображается в соответствующем элементе выходных данных); bias (тип: bool) – параметр, который принимает 2 значения: True или False; если bias принимает значение False, то расчеты эксцесса корректируются на статистическую погрешность; nan_policy (propagate, omit, raise) – параметр, определяющий порядок действий при обнаружении NaN во входных данных:

Продолжение таблицы 8

	propagate: при возникновении NaN вдоль оси, по которой вычисляется статистика, соответствующее выходное значение будет NaN; omit: все NaN будут пропущены при осуществлении вычислений; если вдоль оси, по которой вычисляется статистика, данных будет недостаточно, соответствующее выходное значение будет NaN; raise: при возникновении NaN будет выведена ошибка ValueError; keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в массиве исходного набора данных data. **Выходной параметр:** skewness (тип: array) – асимметрия значений по оси, равная NaN, где все значения одинаковые.
tmean(data [, limits, inclusive, axis])	Возвращает *усеченное среднее*. **Входные параметры:** data (тип: array_like) – набор данных; limits **(**None или(lower limit, upper limit)) – параметр, определяющий границы диапазона такие, что значения во входном массиве, меньшие нижней границы и большие верхней границы игнорируются; если установлено значение None (по умолчанию), используются все значения; любое из предельных значений в кортеже, задающем диапазон может иметь значение None (тогда будет осуществляться работа с полуоткрытым интервалом); inclusive (bool, bool) – параметр, определяющий кортеж (lower flag, upper flag), задающего нижний и верхний флаги; флаги определяют, будут ли включены в расчеты значения, точно равные нижней или верхней границе диапазона (по умолчанию: (True, True)); axis (тип: int или None) – ось, вдоль которой рассчитывается статистика (по умолчанию: None). **Выходной параметр:** tmean (тип: ndarray) – усеченное среднее.
tvar(data [, limits, inclusive, axis, ddof])	Возвращает *усеченную дисперсию*. **Входные параметры:** data (тип: array_like) – набор данных;

Продолжение таблицы 8

	limits **(**None или (lower limit, upper limit)) – параметр, определяющий границы диапазона такие, что значения во входном массиве, меньшие нижней границы и большие верхней границы игнорируются; если установлено значение None (по умолчанию), используются все значения; любое из предельных значений в кортеже, задающем диапазон может иметь значение None (тогда будет осуществляться работа с полуоткрытым интервалом); inclusive (bool, bool) – параметр, определяющий кортеж вида (lower flag, upper flag), задающего нижний и верхний флаги; флаги определяют, будут ли включены в расчеты значения, точно равные нижней или верхней границе диапазона (по умолчанию: (True, True)); axis (тип: int или None) – ось, вдоль которой рассчитывается статистика (по умолчанию: 0); если None, то вычисления проводятся по всему набору данных data; ddof (тип: int) – число степеней свободы (по умолчанию: 1). **Выходной параметр:** tvar (тип: float) – усеченная дисперсия.
tstd(data [, limits, inclusive, axis, ddof])	Возвращает *усеченное стандартное отклонение*. **Входные параметры:** data (тип: array_like) – набор данных; limits **(**Noneили (lower limit, upper limit)) – параметр, определяющий границы диапазона такие, что значения во входном массиве, меньшие нижней границы и большие верхней границы игнорируются; если установлено значение None (по умолчанию), используются все значения; любое из предельных значений в кортеже, задающем диапазон может иметь значение None (тогда будет осуществляться работа с полуоткрытым интервалом); inclusive (bool, bool) – параметр, определяющий кортеж (lower flag, upper flag), задающего нижний и верхний флаги; флаги определяют, будут ли включены в расчеты значения, точно равные нижней или верхней границе диапазона (по умолчанию: (True, True)); axis (тип: int или None) – ось, вдоль которой рассчитывается статистика (по умолчанию: 0); если None, то вычисления проводятся по всему набору данных data; ddof (тип: int) – число степеней свободы (по умолчанию: 1). **Выходной параметр:** tstd (тип: float) – усеченное стандартное отклонение.

Продолжение таблицы 8

variation(data [, axis, nan_policy, ddof, keepdims])	Возвращает *коэффициент вариации*. **Входные параметры:** data(тип: array_like) – набор данных; axis (тип: int или None) – ось, по которой рассчитывается статистика (по умолчанию: 0); если None, то вычисления проводятся по всему набору данных data; nan_policy (propagate, omit, raise) – параметр, определяющий порядок действий при обнаружении NaN во входных данных: propagate: при возникновении NaN вдоль оси, по которой вычисляется статистика, соответствующее выходное значение будет NaN; используется по умолчанию; omit: все NaN будут пропущены при осуществлении вычислений; если вдоль оси, по которой вычисляется статистика, данных будет недостаточно, соответствующее выходное значение будет NaN; raise: при возникновении NaN будет создано исключение; ddof (тип: int) – число степеней свободы (по умолчанию: 0); многие определения коэффициента вариации используют квадратный корень несмещенной выборочной дисперсии для выборочного стандартного отклонения, что соответствует ddof=1. keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что измерений в результирующем массиве, что и в массиве исходного набора данных data. **Выходной параметр:** variation (тип: ndarray) – коэффициент вариации.
trim_mean(data, proportiontocut [, axis])	Возвращает *среднее значение после отсечения данных с обоих концов*. **Входные параметры:** data (тип: array_like) – набор данных; proportiontocut (тип: float) – параметр, определяющий долю для отсечения данных с обоих концов; если proportiontocut = 0.1, то отсекаются слева и справа по 10% данных, предварительно упорядоченных по возрастанию; axis (тип: int или None) – ось, по которой рассчитывается статистика (по умолчанию: 0); если None, то вычисления проводятся по всему набору данных data. **Выходной параметр:** trim_mean (тип: ndarray) – среднее значение усеченного набора данныхdata.

Окончание таблицы 8

iqr(data[, axis, rng, scale, nan_policy, interpolation, keepdims])	Возвращает *межквартильный размах* вдоль указанной оси axis. **Входные параметры:** data (тип: array_like) – набор данных; axis (тип: int или None) – ось, по которой рассчитывается статистика (по умолчанию: 0); если None, то вычисления проводятся по всему набору данных data; rng (двухэлементная последовательность, содержащая значения с плавающей точкой в диапазоне [0,100]) – параметр, определяющий процентили, по которым можно вычислить диапазон, при этом каждый из процентелей должен быть в диапазоне от 0 до 100 включительно (по умолчанию используется пара значений (25, 75); nan_policy(propagate, omit, raise) – параметр, определяющий порядок действий при обнаружении NaN во входных данных: propagate: при возникновении NaN соответствующее выходное значение будет равно NaN; используется по умолчанию; omit: при возникновении NaN вычисления будут выполнены с игнорированием значений NaN; raise: при возникновении NaN будет возвращена ошибка; interpolation (тип: str) – параметр, определяющий метод интерполяции, который следует использовать, когда границы процентилей лежат между двумя точками данных i и j; возможно использование таких методов, как: 'linear':i+(j-i)*fraction, гдеfraction– вещественное число из [0, 1] (используется по умолчанию); 'lower':i; 'higher':j; 'nearest':i илиj (в зависимости от того, что ближе); 'midpoint':(i+j)/2; keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что измерений в результирующем массиве, что и в массиве исходного набора данных data. **Выходной параметр:** iqr (тип: scalar или ndarray) – межквартильный размах; если axis=None, возвращается скаляр; если входные данные содержат целые числа или числа с плавающей точкой меньшей точности, чем np.float64, то тип выходных данных –np.float64,в противном случае тип выходных данных будет таким же, как и у входных.

Кроме того, модуль stats предоставляет возможность работы с функциями для расчета среднего степенного взвешенного (pmean()), модальных (наиболее часто встречающихся) значений (mode()), n-х моментов относительно среднего значения (moment()), n-йk-статистики (kstat(),1<=n<=4), несмещенной оценкидисперсии k-статистики (kstatvar()), усеченного минимума (tmin()), усеченного максимума (tmax()),усеченной стандартной ошибки среднего (tsem()), повторений и их количества (find_repeats()), геометрического стандартногоотклонения (gstd()), стандартной ошибки среднего (sem()), байесовских доверительных интервалов для среднего, дисперсии и стандартного отклонения (bayes_mvs()), «замороженных» распределений для среднего значения, дисперсии и стандартного отклонения (mvsdist()), энтропии распределения для заданных значений вероятности (entropy()), дифференциальной энтропии (differential_entropy()), медианного абсолютного отклонения данных вдоль выбранной оси (median_abs_deviation()).

В таблице 9 представлена информация по некоторым наиболее часто используемым корреляционным функциям.

Таблица 9. Корреляционные функции

Функция и её аргументы (входные параметры)	Описание
pearsonr(x, y, *[, alternative])	Возвращает *коэффициент корреляции Пирсона и значение p-value*. **Входные параметры:** x (тип: array_like) – входной массив; y (тип: array_like) – входной массив; alternative ('two-sided','greater','less')– параметр, определяющий альтернативные гипотезы; возможны следующие варианты: 'two-sided': корреляция отлична от нуля; 'greater': корреляция отрицательна (меньше нуля); 'less': корреляция положительна (больше нуля). **Выходной параметр:** Result – объект с атрибутами: statistic (тип: float) – коэффициент корреляции Пирсона; pvalue (тип: float) – значение p-value, связанное с выбранной альтернативной гипотезой. Объектимеетметод: trust_interval (confidence_level = 0,95). Этот метод вычисляет доверительный интервал статистики коэффициента корреляции для заданного уровня достоверности. Доверительный интервал возвращается в именованном кортеже с полями low и high.

Окончание таблицы 9

spearmanr(a [, b, axis, nan_policy, alternative])	Возвращает *коэффициент корреляции Спирмена и значение p-value*. **Входные параметры:** a, b (тип:1D или2D array_like) – массив, содержащий несколько переменных и наблюдений; если массив одномерный, то представляет собой вектор наблюдений одной переменной. Поведение в двумерном случае определяется с учетом параметра axis;оба массива должны иметь одинаковую длину по выбранной в axis оси; axis (тип: int или None) – параметр, определяющей ось, по которой проводятся вычисления; если axis=0 (по умолчанию), то каждый столбец представляет переменную с наблюдениями в строках; если axis=1, отношение транспонируется: каждая строка представляет переменную, а столбцы содержат наблюдения; если axis=None, то оба массива будут рассмотрены целиком; nan_policy (propagate, omit, raise) – параметр, определяющий порядок действий при обнаружении NaN во входных данных: propagate: при возникновении NaN соответствующее выходное значение будет равно NaN; используется по умолчанию; omit: при возникновении NaN вычисления будут выполнены с игнорированием значений NaN; raise: при возникновении NaN будет возвращена ошибка; alternative ('two-sided','greater','less') – параметр, определяющий альтернативные гипотезы; возможны следующие варианты: 'two-sided': корреляция отлична от нуля; 'greater': корреляция отрицательна (меньше нуля); 'less': корреляция положительна (больше нуля). **Выходные параметры:** correlation (тип: float или ndarray (2-Dsquare)) – матрица корреляции Спирмена или коэффициент корреляции (если в качестве параметров заданы только 2 переменные); матрица корреляции представляет собой квадрат с длиной, равной общему числу переменных (столбцов или строк) в a и b вместе взятых; pvalue (тип: float) – значение p-value для теста, нулевая гипотеза которого состоит в том, что два набора данных не коррелированы.

Кроме того, модуль stats предоставляет возможность работы с функциями для выполнения однофакторного дисперсионного анализа [f_oneway(), one-

wayANOVA (ANalysis Of VAriance)], теста Говерна (`alexandergovern()`, Alexander Govern test), а также для расчета точечного бисериального коэффициента корреляции и значения его `p-value` (`pointbiserialr()`), корреляции для порядковых данных с применением тау Кендалла (`kendalltau()`, Kendall'stau), взвешенная версии тау Кендалла (`weightedtau()`), оценки Зигеля для набора точек `(x, y)` (`siegelslopes()`, Siegelestimator), оценки Тейла-Сена для набора точек `(x, y)` (`theilslopes()`, Theil-Senestimator) и некоторых других статистик.

6.4. Функции для визуализации результатов разведочного анализа данных

Для визуализации результатов разведочного анализа данных могут быть использованы различные функции, названия и краткие описания которых приведены ниже.

Кроме того, для каждой функции приведена ссылка на Интернет-страницу с подробной документацией.

6.4.1. Визуализация с применением библиотеки seaborn

Большинство рассматриваемых функций входят в состав библиотеки `seaborn` [12], доступной по ссылке:

https://seaborn.pydata.org/.

Библиотека `seaborn` – библиотека визуализации данных на языке программирования Python, основанная на библиотеке `matplotlib`. Эта библиотека предоставляет высокоуровневый интерфейс для построения привлекательных и информативных статистических графиков. При этом приводятся примеры, демонстрирующие различные варианты использования инструментов визуализации.

Информация по некоторым функциям, применяемым для визуализации, приведена ниже.

1. Функция **`seaborn.histplot`.**

Функция реализует построение одномерных или двумерных гистограмм для отображения распределения наборов данных. Гистограмма – классический инструмент визуализации, который представляет распределение одного или нескольких признаков посредством подсчета числа наблюдений, попадающих в дискретные интервалы. Качество представления данных зависит от выбора хороших параметров сглаживания. Эта функция может нормализовать статистику, вычисленную в каждом бине, для оценки частоты, плотности или вероят-

ностной массы, а также может добавить гладкую кривую, полученную с использованием оценки плотности ядра (аналогично функции `kdeplot()`).

Синтаксис:

`seaborn.histplot(`*`data=None, *, x=None, y=None, hue=None, weights=None, stat='count', bins='auto', binwidth=None, binrange=None, discrete=None, cumulative=False, common_bins=True, common_norm=True, multiple='layer', element='bars', fill=True, shrink=1, kde=False, kde_kws=None, line_kws=None, thresh=0, pthresh=None, pmax=None, cbar=False, cbar_ax=None, cbar_kws=None, palette=None, hue_order=None, hue_norm=None, color=None, log_scale=None, legend=True, ax=None, **kwargs`*`)`

Подробная информация по функции приведена по ссылке: https://seaborn.pydata.org/generated/seaborn.histplot.html.

2. Функция **`seaborn.kdeplot`**.

Функция реализует построение одномерных или двумерных распределений, используя оценку плотности ядра. График оценки плотности ядра (kernel density estimate, KDE) – метод визуализации распределения наблюдений в наборе данных, аналогичный гистограмме. KDE представляет данные, используя непрерывную кривую плотности вероятности в одном или нескольких измерениях. По сравнению с гистограммой, KDE может создать менее загроможденный и более интерпретируемый график, особенно при рисовании нескольких распределений. Но это может привести к искажениям, если базовое распределение ограничено или не является гладким. Как и в гистограмме, качество представления данных зависит от выбора хороших параметров сглаживания.

Синтаксис:

`seaborn.kdeplot(`*`data=None, *, x=None, y=None, hue=None, weights=None, palette=None, hue_order=None, hue_norm=None, color=None, fill=None, multiple='layer', common_norm=True, common_grid=False, cumulative=False, bw_method='scott', bw_adjust=1, warn_singular=True, log_scale=None, levels=10, thresh=0.05, gridsize=200, cut=3, clip=None, legend=True, cbar=False, cbar_ax=None, cbar_kws=None, ax=None, **kwargs`*`)`

Подробная информация по функции приведена по ссылке: https://seaborn.pydata.org/generated/seaborn.kdeplot.html.

3. Функция **seaborn.boxplot**.

Функция реализует построение коробчатой диаграммы (диаграммы «ящик с усами»). Коробчатая диаграмма показывает распределение количественных данных: в коробке показаны квартили набора данных, усы показывают остальную часть распределения, за исключением данных, которые определены как «выбросы» с использованием метода, который является функцией межквартильного диапазона.

Синтаксис:

```
seaborn.boxplot(data=None, *, x=None, y=None, hue=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, width=0.8, dodge=True, fliersize=5, linewidth=None, whis=1.5, ax=None, **kwargs)
```

Подробная информация по функции приведена по ссылке: https://seaborn.pydata.org/generated/seaborn.boxplot.html.

4. Функция **seaborn.violinplot**.

Функция реализует построение скрипичного диаграммы, представляющей собой комбинацию коробчатой диаграммы с оценкой плотности ядра. Скрипичная диаграмма, как и коробчатая диаграмма, показывает распределение количественных данных. В отличие от коробчатой диаграммы, в которой все компоненты диаграммы соответствуют фактическим точкам данных, в скрипичной диаграмме используется оценка плотности ядра базового распределения данных. Скрипичная диаграмма позволяет эффективно и привлекательно показать сразу несколько распределений данных одновременно. При этом следует иметь в виду, что на процедуру оценки влияет размер выборки, поэтому скрипичные диаграммы для относительно небольших выборок могут выглядеть обманчиво гладкими.

Синтаксис:

```
seaborn.violinplot(data=None, *, x=None, y=None, hue=None, order=None, hue_order=None, bw='scott', cut=2, scale='area', scale_hue=True, gridsize=100, width=0.8, inner='box', split=False, dodge=True, orient=None, linewidth=None, color=None, palette=None, saturation=0.75, ax=None, **kwargs)
```

Подробная информация по функции приведена по ссылке: https://seaborn.pydata.org/generated/seaborn.violinplot.html.

5. Функция **seaborn.barplot**.

Функция реализует построение столбчатой диаграммы с отображением точечных оценок и ошибок в виде прямоугольников. Столбчатая диаграмма отоб-

ражает сравнение нескольких дискретных категорий. Столбчатая диаграмма представляет собой оценку центральной тенденции для количественной переменной с высотами прямоугольников, пропорциональными величинам, которые они отображают. Столбчатая диаграмма дает некоторое представление о неопределенности вокруг оценки центральной тенденции с использованием планок погрешностей. Столбчатая диаграмма включает 0 в диапазоне количественной оси. Она являются хорошим выбором, когда 0 является значимым значением для количественной переменной, и требуется провести сравнение с ним. Важно иметь в виду, что столбчатая диаграмма показывает только среднее (или другое оценочное) значение. Иногда более подходящими могут оказаться другие подходы, такие, например, как коробчатая или скрипичная диаграмма.

Синтаксис:

`seaborn.barplot(`*`data=None, *, x=None, y=None, hue=None, order=None, hue_order=None, estimator='mean', errorbar=('ci',95), n_boot=1000, units=None, seed=None, orient=None, color=None, palette=None, saturation=0.75, width=0.8, errcolor='.26', errwidth=None, capsize=None, dodge=True, ci='deprecated', ax=None, **kwargs`*`)`

Подробная информация по функции приведена по ссылке:
https://seaborn.pydata.org/generated/seaborn.barplot.html.

6. Функция `seaborn.heatmap`.

Функция реализует построение тепловой карты, являющейся графическим представлением данных, в котором индивидуальные значения в таблице (матрице) отображаются (кодируются) при помощи цвета.

Синтаксис:

`seaborn.heatmap(`*`data, *, vmin=None, vmax=None, cmap=None, center=None, robust=False, annot=None, fmt='.2g', annot_kws=None, linewidths=0, linecolor='white', cbar=True, cbar_kws=None, cbar_ax=None, square=False, xticklabels='auto', yticklabels='auto', mask=None, ax=None, **kwargs`*`)`

Подробная информация по функции приведена по ссылке:
https://seaborn.pydata.org/generated/seaborn.heatmap.html.

7. Функция **`seaborn.scatterplot`**.

Функция реализует построение диаграммы рассеяния (скатерограммы, точечной диаграммы) с возможностью нескольких семантических группировок. При этом связь между признаками `x` и `y` может быть показана для различных

подмножеств данных с использованием параметров оттенка, размера и стиля (`hue`, `size` и `style`). Эти параметры определяют, какая визуальная семантика используется для идентификации различных подмножеств. Можно независимо отобразить до трех измерений, используя все три семантических типа, но такое представление информации может быть трудно интерпретировать. Использование избыточной семантики (то есть оттенка `hue` и стиля `style` для одного и того же признака) может помочь сделать графику более доступной. Обработка по умолчанию семантики оттенка `hue` (и, в меньшей степени, размера `size`), если она присутствует, зависит от того, предполагается ли, что признак представляет «числовые» или «категориальные» данные. В частности, числовые признаки по умолчанию представлены последовательной цветовой палитрой, а записи легенды показывают обычные «галочки» со значениями, которые могут существовать или не существовать в данных.

Синтаксис:

```
seaborn.scatterplot(data=None, *, x=None, y=None,
hue=None, size=None, style=None, palette=None,
hue_order=None, hue_norm=None, sizes=None,
size_order=None, size_norm=None, markers=True,
style_order=None, legend='auto', ax=None, **kwargs)
```

Подробная информация по функции приведена по ссылке:
https://seaborn.pydata.org/generated/seaborn.scatterplot.html.

Пример с построением матрицы рассеяния доступен по ссылке:
https://seaborn.pydata.org/examples/scatterplot_matrix.html.

6.4.2. Визуализация с применением библиотеки matplotlib

Широкие, но менее привлекательные возможности для визуализации результатов расчетов представляет также библиотека `matplotlib` [13]. Библиотека `matplotlib` – библиотека на языке программирования Python для визуализации данных с применением двумерной графики.

Эта библиотека доступна по ссылке:
https://matplotlib.org/.

Многие средства визуализации, предлагаемые библиотекой `seaborn`, реализованы и в библиотеке `matplotlib`. В частности, графические средства, применяемые в статистике и разведочном анализе данных, доступны по ссылке:
https://matplotlib.org/stable/plot_types/stats/index.html.

В библиотеке `matplotlib` приводятся примеры, демонстрирующие различные варианты использования инструментов визуализации.

Информация по некоторым функциям, применяемым для визуализации, приведена ниже.

1. Функция **`matplotlib.axes.Axes.bar`**.

Функция реализует построение столбчатой диаграммы. Прямоугольники располагаются по `x` с заданным выравниванием. Размеры прямоугольников указываются по высоте и ширине. Вертикальная базовая линия – нижняя (по умолчанию `0`). Многие параметры могут принимать либо одно значение, применимое ко всем столбцам, либо последовательность значений, по одному для каждого столбца.

Синтаксис:

```
Axes.bar(x, height, width=0.8, bottom=None, *,
align='center', data=None, **kwargs)
```

Подробная информация по функции приведена по ссылке:
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.bar.html.

2. Функция **`matplotlib.axes.Axes.hist`**.

Функция реализует вычисление и построение гистограммы. Функция использует `numpy.histogram` для группировки данных по x и подсчета числа значений в каждом бине, а затем строит распределение в виде `BarContainer` или `Polygon`. Параметры бинов, диапазона, плотности и веса передаются в `numpy.histogram`.

Синтаксис:

```
Axes.hist(x, bins=None, range=None, density=False,
weights=None, cumulative=False, bottom=None,
histtype='bar', align='mid', orientation='vertical',
rwidth=None, log=False, color=None, label=None,
stacked=False, *, data=None, **kwargs)
```

Подробная информация по функции приведена по ссылке:
https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.hist.html#matplotlib.axes.Axes.hist.

3. Фукция **`matplotlib.axes.Axes.boxplot`**.

Функция реализует построение коробчатой диаграммы (диаграммы «ящик с усами»). Коробка простирается от первого квартиля Q_1 до третьего квартиля Q_3 данных с линией в медиане. Усы выходят за пределы коробки на 1,5-кратный межквартильный диапазон *IQR*. Точки-выбросы – это данные, которые находятся за концами усов.

Синтаксис:

`Axes.boxplot(x, notch=None, sym=None, vert=None, whis=None, positions=None, widths=None, patch_artist=None, bootstrap=None, usermedians=None, conf_intervals=None, meanline=None, showmeans=None, showcaps=None, showbox=None, showfliers=None, boxprops=None, labels=None, flierprops=None, medianprops=None, meanprops=None, capprops=None, whiskerprops=None, manage_ticks=True, autorange=False, zorder=None, capwidths=None, *, data=None)`

Подробная информация по функции приведена по ссылке:

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.boxplot.html#matplotlib.axes.Axes.boxplot.

4. Функция **`matplotlib.axes.Axes.violinplot`**.

Функция реализует построение скрипичной диаграммы для каждого столбца набора данных или каждого вектора в наборе данных последовательности. Каждая заполненная область расширяется, чтобы представить весь диапазон данных, с необязательными линиями для среднего значения, медианы, минимума, максимума и заданных пользователем квантилей.

Синтаксис:

`Axes.violinplot(dataset, positions=None, vert=True, widths=0.5, showmeans=False, showextrema=True, showmedians=False, quantiles=None, points=100, bw_method=None, *, data=None)`

Подробная информация по функции приведена по ссылке:

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.violinplot.html#matplotlib.axes.Axes.violinplot.

5. Функция **`matplotlib.axes.Axes.hexbin`**.

Функция реализует построение двухмерного шестиугольного графика бинирования точек `x`, `y`. Если значение параметра `C` равно `None`, значение шестиугольника определяется числом точек в шестиугольнике. Если значение параметра `C` указано, задаются значения в координатах `(x[i], y[i])`.Для каждого шестиугольника эти значения уменьшаются с помощью функции `reduce_C_function()`.

Синтаксис:

`Axes.hexbin(x, y, C=None, gridsize=100, bins=None, xscale='linear', yscale='linear', extent=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, edgecolors='face', re-`

*duce_C_function=<functionmean>, mincnt=None, margin-als=False, *, data=None, **kwargs*)

Подробная информация по функции приведена по ссылке: https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.hexbin.html#matplotlib.axes.Axes.hexbin.

6. Функция **matplotlib.axes.Axes.contour**.

Функция реализует построение контурного графика (контурных линий).

Синтаксис:

Axes.hexbin(*x, y, C=None, gridsize=100, bins=None, xscale='linear', yscale='linear', extent=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, lin-ewidths=None, edgecolors='face', re-duce_C_function=<functionmean>, mincnt=None, margin-als=False, *, data=None, **kwargs*)

Подробная информация по функции приведена по ссылке: https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.contour.html#matplotlib.axes.Axes.contour.

7. Функция **matplotlib.axes.Axes.pie**.

Функция реализует построение круговой диаграммы массива `x`. Дробная площадь каждого клина определяется как `x/sum(x)`. Клинья строятся против часовой стрелки, по умолчанию начиная с оси `x`.

Синтаксис:

Axes.pie(*x, explode=None, labels=None, colors=None, au-topct=None, pctdistance=0.6, shadow=False, labeldis-tance=1.1, startangle=0, radius=1, counterclock=True, wedgeprops=None, textprops=None, center=(0,0), frame=False, rotatelabels=False, *, normalize=True, da-ta=None*)

Подробная информация по функции приведена по ссылке: https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.pie.html#matplotlib.axes.Axes.pie.

8. Функция `matplotlib.pyplot.scatter`.

Функция реализует построение диаграммы рассеяния для признаков `x` и `y` с разным размером и/или цветом маркера.

Синтаксис:

matplotlib.pyplot.scatter(*x, y, s=None, c=None, mark-er=None, cmap=None, norm=None, vmin=None, vmax=None, al-*

*pha=None, linewidths=None, *, edgecolors=None, plotnonfinite=False, data=None, **kwargs*)

Подробная информация по функции приведена по ссылке:
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html.

6.4.3. Визуализация с применением библиотеки pandas

Ряд инструментов для визуализации имеется в библиотеке `pandas` [8], доступной по ссылке:

https://pandas.pydata.org/.

Библиотека `pandas` –библиотека на языке программирования Python для обработки и анализа данных. Работа библиотеки `pandas` с данными строится поверх библиотеки `NumPy`, являющейся инструментом более низкого уровня. Библиотека `pandas` предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами.

Инструменты визуализации доступны по ссылке:
https://pandas.pydata.org/docs/reference/plotting.html.

В библиотеке `pandas` приводятся примеры, демонстрирующие различные варианты использования инструментов визуализации.

Информация по некоторым функциям, применяемым для визуализации, приведена ниже.

1. Функция **`pandas.DataFrame.plot.bar`**.

Функция реализует построение столбчатой диаграммы.

Синтаксис:

`DataFrame.plot.bar(`*`x=None, y=None, **kwargs`*`)`

Подробная информация по функции приведена по ссылке:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.bar.html.

2. Функция **`pandas.DataFrame.plot.hist`**.

Функция реализует построение гистограммы на основе столбцов `DataFrame`. Функция группирует значения всех заданных серий `Series` в `DataFrame` в бины и отображает их в одном `matplotlib.axes.Axes`. Это полезно, когда серии `Series`в `DataFrame` имеют одинаковый масштаб.

Синтаксис:

`DataFrame.plot.hist(`*`by=None, bins=10, **kwargs`*`)`

Подробная информация по функции приведена по ссылке:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.hist.html.

3. Функция **`pandas.DataFrame.plot.scatter`**.

Функция реализует построение диаграмму рассеяния с различным размером и цветом точки маркера. Координаты каждой точки определяются двумя

столбцами датафрейма. Для представления каждой точки используются круглые маркеры с заливкой цветом.

Синтаксис:

`DataFrame.plot.scatter(x, y, s=None, c=None, **kwargs)`

Подробная информация по функции приведена по ссылке:

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.scatter.html.

4. Функция **`pandas.plotting.scatter_matrix`**.

Функция реализует построение матрицы рассеяния.

Синтаксис:

`pandas.plotting.scatter_matrix(frame, alpha=0.5, figsize=None, ax=None, grid=False, diagonal='hist', marker='.', density_kwds=None, hist_kwds=None, range_padding=0.05, **kwargs)`

Подробная информация по функции приведена по ссылке:

https://pandas.pydata.org/docs/reference/api/pandas.plotting.scatter_matrix.html.

5. Функция **`pandas.plotting.boxplot`**.

Функция реализует построение коробчатой диаграмму (диаграммы «ящик с усами») из столбцов DataFrame при необходимости сгруппированных по некоторым другим столбцам. «Ящик» простирается от квартильных значений данных Q_1 до Q_3 с линией в медиане Q_2. Усы отходят от краев «ящика», показывая диапазон данных. По умолчанию усы простираются не более чем на 1,5**IQR* (*IQR*=Q_3–Q_1) от краев «ящика», заканчиваясь в самой дальней точке данных в этом интервале. Выбросы наносятся отдельными точками.

Синтаксис:

`pandas.plotting.boxplot(data, column=None, by=None, ax=None, fontsize=None, rot=0, grid=True, figsize=None, layout=None, return_type=None, **kwargs)`

Подробная информация по функции приведена по ссылке:

https://pandas.pydata.org/docs/reference/api/pandas.plotting.boxplot.html.

6. Функция **`pandas.DataFrame.plot.hexbin`**.

Функция реализует построение шестиугольного графика бинирования точек `x`, `y`. Если значение параметра `C` равно `None` (по умолчанию), то строится гистограмма числа вхождений наблюдений в `(x[i], y[i])`. Если значение параметра `C` указано, задаются значения в координатах `(x[i], y[i])`. Эти значения накапливаются для каждого шестиугольного бина, а затем уменьшаются в соответствии с функцией `reduce_C_function`, представляющую собой по умолчанию функцию среднего значения `NumPy` (`numpy.mean()`).

Синтаксис:

```
DataFrame.plot.hexbin(x, y, C=None, reduce_C_function=None, gridsize=None, **kwargs)
```

Подробная информация по функции приведена по ссылке:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.hexbin.html.

7. Примеры выполнения разведочного анализа на языке Python

Ниже приведены 2 примера выполнения разведочного анализа данных, в одном из которых подробно представлено использование различных инструментов, а в другом – продемонстрирована только обработка пропусков данных и сводные результаты её применения.

7.1. Пример выполнения разведочного анализа для набора данных «Ирисы Фишера» (The Iris Dataset)

Рассмотрим аспекты применения инструментов разведочного анализа на примере набора данных «Ирисы Фишера» (The Iris Dataset) [14], который представлен, например, по ссылке:
http://archive.ics.uci.edu/ml/datasets/Iris.

Кроме того, для работы с этим набором данных можно выполнить импортирование наборов данных `datasets` из библиотеки `sklearn`.

«Ирисы Фишера» – набор данных для задачи многоклассовой классификации. Он содержит информацию о 150 экземплярах ириса трёх видов, сопоставленных трем классам (рис. 9). Это такие виды ирисов, как:

– ирис щетинистый (*Iris setosa*);
– ирис виргинский (*Iris virginica*);
– ирис разноцветный (*Iris versicolor*).

Набор данных содержит информацию о 50 экземплярах ирисов каждого класса. Класс ириса можно рассматривать как целевой признак в наборе данных.

Iris setosa

Iris virginica

Iris versicolor

Рисунок 9. Ирисы

Каждый экземпляр ириса в наборе данных описывается с использованием таких признаков, как:

– длина наружной доли околоцветника (*sepal length*);
– ширина наружной доли околоцветника (*sepal width*);
– длина внутренней доли околоцветника (*petal length*);
– ширина внутренней доли околоцветника (*petal width*).

Значения измерений по признакам представлены в сантиметрах (рис. 10).

Ирисы Фишера

Длина чашелистика	Ширина чашелистика	Длина лепестка	Ширина лепестка	Вид ириса
5.1	3.5	1.4	0.2	*setosa*
4.9	3.0	1.4	0.2	*setosa*
4.7	3.2	1.3	0.2	*setosa*
4.6	3.1	1.5	0.2	*setosa*
5.0	3.6	1.4	0.2	*setosa*
5.4	3.9	1.7	0.4	*setosa*
4.6	3.4	1.4	0.3	*setosa*
5.0	3.4	1.5	0.2	*setosa*
4.4	2.9	1.4	0.2	*setosa*
4.9	3.1	1.5	0.1	*setosa*
5.4	3.7	1.5	0.2	*setosa*
4.8	3.4	1.6	0.2	*setosa*
4.8	3.0	1.4	0.1	*setosa*

Рисунок 10. Фрагмент набора данных

На рис. 11 представлен фрагмент программного кода, реализующего импорт библиотек и вывод сводной информации о наборе данных, загруженном в датафрейм.

```
# импорт библиотек
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
df = sns.load_dataset('iris')
```

```
# Вывод информации о наборе данных
df.info()
```

Рисунок 11. Фрагмент программного кода, реализующего импорт библиотек и вывод сводной информации о наборе данных, загруженном в датафрейм

На рис. 12 представлен скриншот, отражающий вывод сводной информации о наборе данных, загруженном в датафрейм.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Рисунок 12. Скриншот, отражающий вывод сводной информации о наборе данных

На рис. 13 представлен фрагмент программного кода, реализующего вывод информации о первых 5 строках набора данных, загруженного в датафрейм. На рис. 14 представлен скриншот с информацией о первых 5 строках набора данных, загруженного в датафрейм.

```
1 df.head()
```

Рисунок 13. Фрагмент программного кода, реализующего вывод информации о первых 5 строках набора данных, загруженного в датафрейм

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Рисунок 14. Скриншот с информацией о первых 5 строках набора данных, загруженного в датафрейм

На рис. 15 представлен фрагмент программного кода, реализующего поиск и вывод уникальных значений для каждого признака. На рис. 16 представлен

скриншот с частью результатов поиска уникальных значений для каждого признака.

```
# Поиск уникальных значений для каждого признака
# (для каждого столбца)
for col in df:
    uniq = df[col].value_counts()
    print(uniq, '\n==============\n')
```

Рисунок 15. Фрагмент программного кода, реализующего поиск и вывод уникальных значений для каждого признака

```
6.6    2
4.7    2
7.1    1
7.4    1
7.6    1
7.9    1
4.5    1
7.0    1
5.3    1
7.3    1
4.3    1
Name: sepal_length, dtype: int64
==============

3.0    26
2.8    14
3.2    13
3.4    12
3.1    11
2.9    10
```

Рисунок 16. Скриншот с частью результатов поиска уникальных значений для каждого признака

На рис. 17 представлен фрагмент программного кода, реализующего выявление числовых признаков в наборе данных. На рис. 18 представлен скриншот с выводом результатов поиска числовых признаков в наборе данных.

```
# Определение числовых признаков
num_cols = list(df.select_dtypes(['int64','float64']))
num_cols
```

Рисунок 17. Фрагмент программного кода, реализующего выявление числовых признаков в наборе данных

```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
```

Рисунок 18. Скриншот с выводом результатов поиска числовых признаков в наборе данных

На рис. 19 представлен фрагмент программного кода, реализующего вывод информации по пропускам данных по каждому признаку. На рис. 20 представлен скриншот с выводом информации по пропускам данных по каждому признаку. На рис. 21 представлен фрагмент программного кода, реализующего вывод круговой диаграммы, позволяющей выполнить анализ распределения меток классов для целевого признака. При этом также осуществляется вывод названий меток классов целевого признака (рис. 22).

```
# Вывод информации по пропускам данных по каждому признаку,
# Наличие пропусков также можно увидеть
# по результатам работы команды data.info()
df.isnull().sum()
```

Рисунок 19. Фрагмент программного кода, реализующего вывод информации по пропускам данных по каждому признаку

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

Рисунок 20. Скриншот с выводом информации по пропускам данных по каждому признаку

```
# Анализ распределения меток классов для целевого признака

# Вывод уникальных меток классов для целевого признака
print(df.species.unique())
# Выбор палитры для круговой диаграммы
sns.set_palette('pastel')
# Подготовка данных
attr_count = df.species.value_counts()
attr_labels=df.species.unique()
# Создание рисунка
fig,ax = plt.subplots(figsize=(8,8))
# Круговая диаграмма
ax.pie(attr_count, labels=attr_labels, autopct='%.2f%%', startangle=90)
# Визуализация
plt.show()
```

Рисунок 21. Фрагмент программного кода, реализующего вывод круговой диаграммы, позволяющей выполнить анализ распределения меток классов для целевого признака

```
['setosa' 'versicolor' 'virginica']
```

Рисунок 22. Вывод названий меток классов целевого признака

На рис. 23 представлен скриншот с выводом круговой диаграммы, отражающей распределение меток классов для целевого признака.

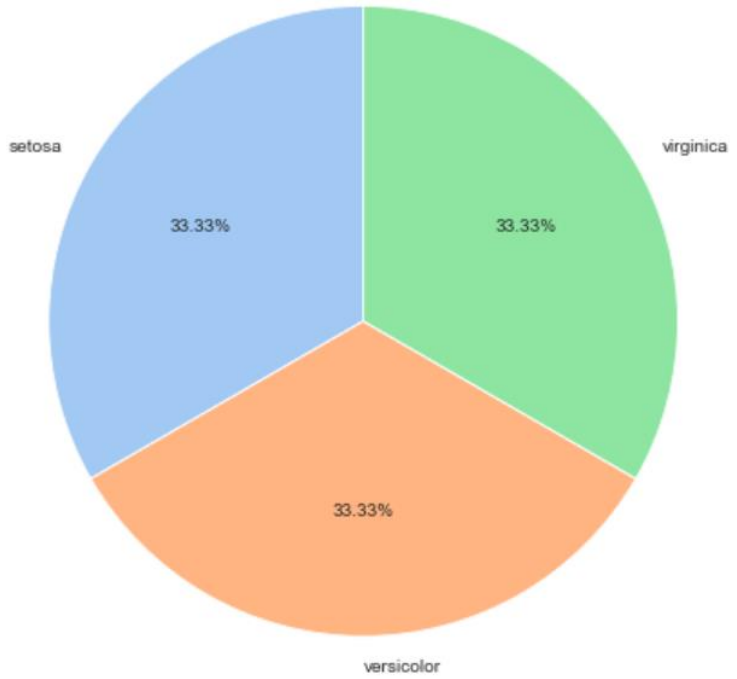

Рисунок 23. Скриншот с выводом круговой диаграммы, отражающей распределение меток классов для целевого признака

На рис. 24 представлен фрагмент программного кода, реализующего вывод описательной статистики числовых признаков. На рис. 25 представлен скриншот с выводом описательной статистики числовых признаков.

```
1  # Описание статистики числовых признаков
2  num_stat = df[num_cols].describe().transpose().reset_index()
3  num_stat.rename(columns={'index': 'Признак'}, inplace=True)
4  num_stat
```

Рисунок 24. Фрагмент программного кода, реализующего вывод описательной статистики числовых признаков

	Признак	count	mean	std	min	25%	50%	75%	max
0	sepal_length	150.0	5.843333	0.828066	4.3	5.1	5.80	6.4	7.9
1	sepal_width	150.0	3.057333	0.435866	2.0	2.8	3.00	3.3	4.4
2	petal_length	150.0	3.758000	1.765298	1.0	1.6	4.35	5.1	6.9
3	petal_width	150.0	1.199333	0.762238	0.1	0.3	1.30	1.8	2.5

Рисунок 25. Скриншот с выводом описательной статистики числовых признаков

На рис. 26 представлен фрагмент программного кода, реализующего построение графиков для плотностей распределения признаков. На рис. 27 представлен скриншот с выводом графиков для плотностей распределения.

```
# Построение графиков для плотностей распределения
fig, ax = plt.subplots(ncols=1, nrows=4, figsize=(7,20))
i=0
for col in num_cols:
    sns.kdeplot(x=df[col], fill=True, alpha=1, ax=ax[i])
    ax[i].set_xlabel(' ')
    ax[i].set_ylabel(' ')
    ax[i].set_title(col, fontsize=12)
    i=i+1
plt.show()
```

Рисунок 26. Фрагмент программного кода, реализующего построение графиков для плотностей распределения признаков

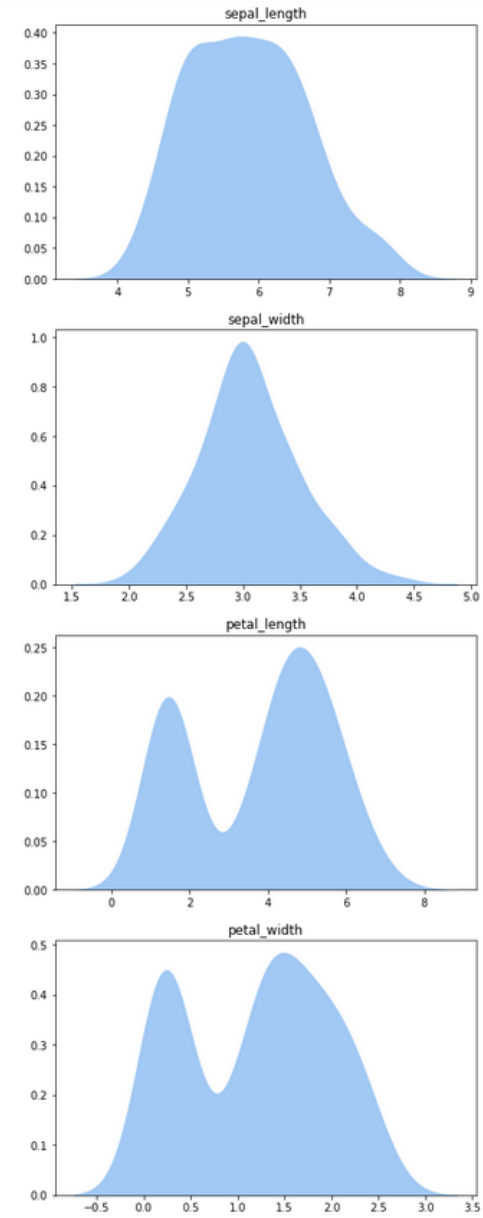

Рисунок 27. Скриншот с выводом графиков для плотностей распределения

На рис. 28 представлен фрагмент программного кода, реализующего построение гистограмм для признаков. На рис. 29 представлен скриншот с выводом гистограмм для признаков.

```
# Построение гистограмм
fig, ax = plt.subplots(ncols=1, nrows=4, figsize=(7,20))
i=0
for col in num_cols:
    sns.histplot(x=df[col], fill=True, alpha=1, ax=ax[i])
    ax[i].set_xlabel(' ')
    ax[i].set_ylabel(' ')
    ax[i].set_title(col, fontsize=12)
    i=i+1
plt.show()
```

Рисунок 28. Фрагмент программного кода, реализующего построение гистограмм для признаков

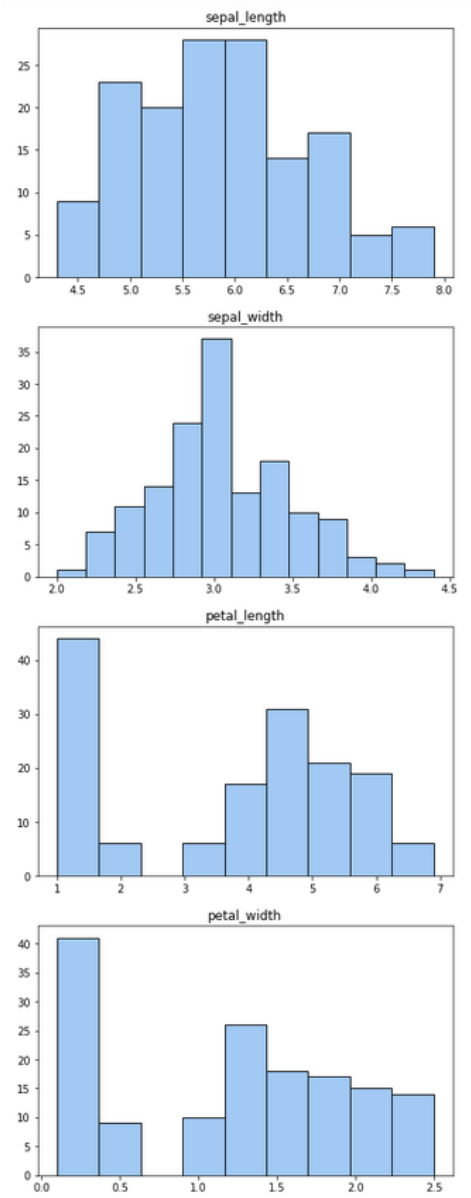

Рисунок 29. Скриншот с выводом гистограмм для признаков

На рис. 30 представлен фрагмент программного кода, реализующего расчет асимметрии распределения признаков. На рис. 31 представлен скриншот с выводом результатов расчета асимметрии распределения признаков.

```
# Проверка асимметрии распределения
skew = []
for col in num_cols:
    skew.append(round(df[col].skew(),3))
num_skew = pd.DataFrame({'Признак':num_cols, 'skewness':skew})
num_skew
```

Рисунок 30. Фрагмент программного кода, реализующего проверку асимметрии распределения признаков

	Признак	skewness
0	sepal_length	0.315
1	sepal_width	0.319
2	petal_length	-0.275
3	petal_width	-0.103

Рисунок 31. Скриншот с выводом результатов проверки асимметрии распределения признаков

На рис. 32 представлен фрагмент программного кода, реализующего формирование списка признаков с нормальным распределением. На рис. 33 представлен скриншот с выводом списка признаков с нормальным распределением. Предполагается, что признаки с асимметрией от «–0,05» до «0,05» имеют нормальное (гауссово) распределение. Как видно из рис. 33, признаки с нормальным распределением в рассматриваемом наборе данных отсутствуют (т.к. сформированный список – пустой).

```
gauss_feat = list(num_skew.query('skewness < 0.05 & skewness > -0.05')
                  ['Признак'])
gauss_feat
```

Рисунок 32. Фрагмент программного кода, реализующего формирование списка признаков с нормальным распределением

```
[]
```

Рисунок 33. Скриншот с пустым списком

На рис. 34 представлен фрагмент программного кода, реализующего создание тепловой карты на основе корреляционной матрицы. На рис. 35 представлен скриншот с выводом тепловой карты на основе корреляционной матрицы.

```python
# Создание рисунка для тепловой карты на основе корреляционной матрицы
corr = df[num_cols].corr()
plt.figure(figsize=(16,8))
sns.heatmap(corr, cmap="Reds",annot=True)
plt.show()
```

Рисунок 34. Фрагмент программного кода, реализующего создание тепловой карты на основе корреляционной матрицы

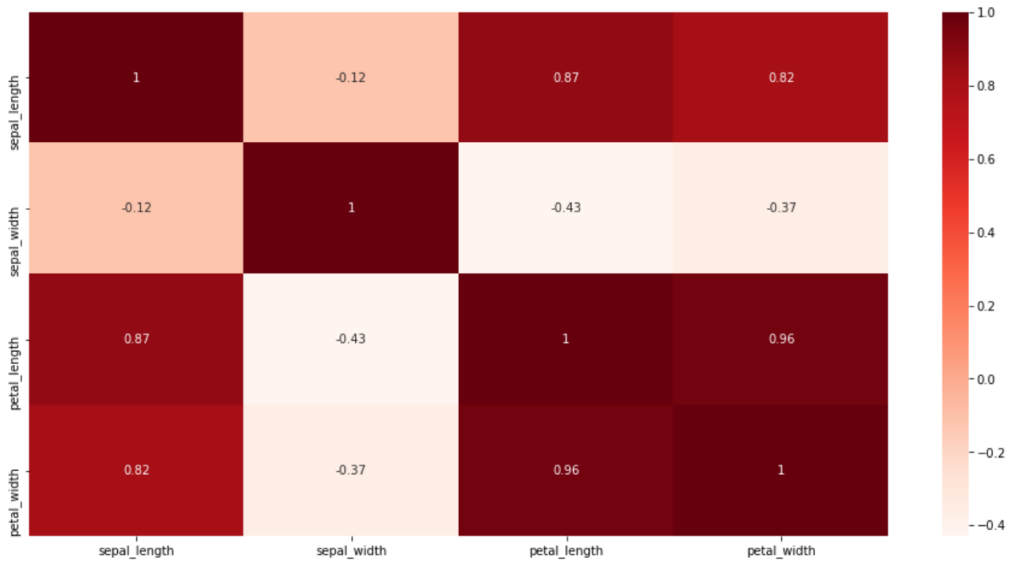

Рисунок 35. Скриншот с выводом тепловой карты на основе корреляционной матрицы

Представление тепловой карты, приведенное на рис. 35, может показаться избыточным (т.к. все значения корреляционной матрицы, за исключением равных единице значений на главной диагонали, дублируются ввиду их симметричного расположения относительно главной диагонали).

На рис. 36 представлен фрагмент программного кода, реализующего создание тепловой карты на основе корреляционной матрицы с исключением избыточной информации. На рис. 37 представлен скриншот с выводом тепловой карты на основе корреляционной матрицы с исключением избыточной информации.

```python
# Создание рисунка для тепловой карты на основе корреляционной матрицы
plt.figure(figsize=(16,8))
plt.title('Корреляция между всеми числовыми признаками', size=15)

# Создание маски
mask = np.triu(np.ones_like(df[num_cols].corr()))
# Создание цветовой карты
colormap = sns.color_palette("Blues")
# Создание тепловой карты (heatmap)
sns.heatmap(df[num_cols].corr(), annot=True, cmap=colormap, mask=mask)

plt.show()
```

Рисунок 36. Фрагмент программного кода, реализующего создание тепловой карты на основе корреляционной матрицы с исключением избыточной информации

Рисунок 37. Скриншот с выводом тепловой карты на основе корреляционной матрицы с исключением избыточной информации

На рис. 38 представлен фрагмент программного кода, реализующего построение матрицы рассеяния. На рис. 39 представлен скриншот с выводом матрицы рассеяния.

```
# Построение матрицы рассеяния
pd.plotting.scatter_matrix(df[num_cols], figsize=(10,10), alpha=1)
```

Рисунок 38. Фрагмент программного кода, реализующего построение матрицы рассеяния

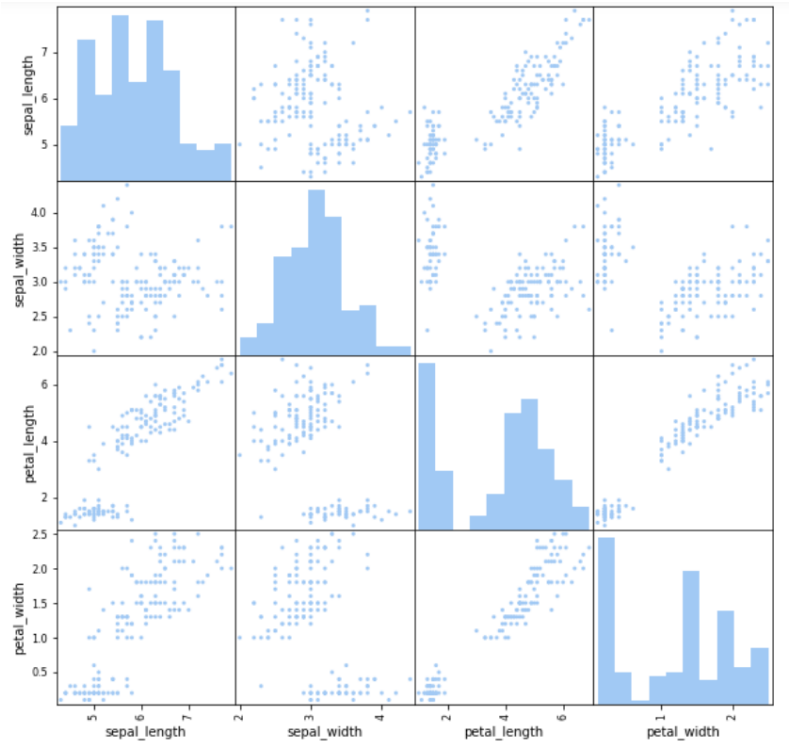

Рисунок 39. Скриншот с выводом матрицы рассеяния

На рис. 40 представлен фрагмент программного кода, реализующего построение 12 диаграмм рассеяния, которые можно сопоставить недиагональным изображениям в матрице рассеяния. На рис. 41 представлен скриншот с выводом диаграммы рассеяния, аналог которой можно увидеть в позиции (2, 1) матрицы рассеяния на рис. 39.

```
# Построение диаграмм рассеяния
for col1 in num_cols:
    for col2 in num_cols:
        if (col1 != col2):
            sns.scatterplot(data=df[num_cols], x=col1, y=col2)
            plt.show()
```

Рисунок 40. Фрагмент программного кода, реализующего построение 12 диаграмм рассеяния, которые можно сопоставить недиагональным изображениям в матрице рассеяния

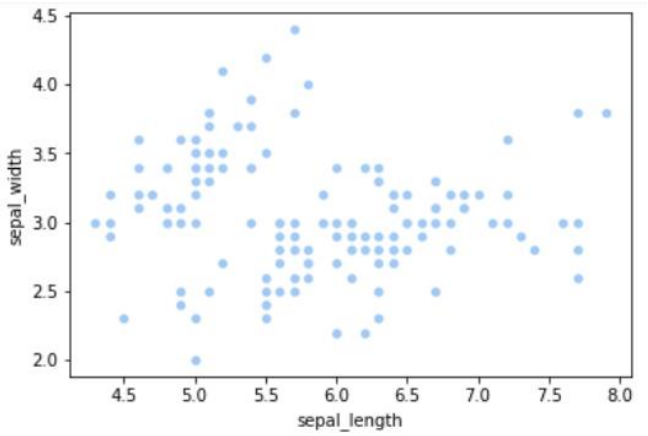

Рисунок 41. Скриншот с выводом диаграммы рассеяния, аналог которой можно увидеть в позиции (2, 1) матрицы рассеяния

На рис. 42 представлен фрагмент программного кода, реализующего построение 12 графиков с шестиугольной сеткой, которые можно сопоставить недиагональным изображениям в матрице рассеяния. На рис. 43 представлен скриншот с выводом графика с шестиугольной сеткой, аналог которого можно увидеть в позиции (2, 1) матрицы рассеяния на рис. 39.

```
# Построение графиков с шестиугольной сеткой
for col1 in num_cols:
    for col2 in num_cols:
        if (col1 != col2):
            plt.hexbin(data=df[num_cols], x=col1, y=col2,
                       gridsize=(25,25), cmap=plt.cm.Blues)
            plt.colorbar()
            plt.xlabel(col1)
            plt.ylabel(col2)
            plt.show()
```

Рисунок 42. Фрагмент программного кода, реализующего построение 12 графиков с шестиугольной сеткой, которые можно сопоставить недиагональным изображениям в матрице рассеяния

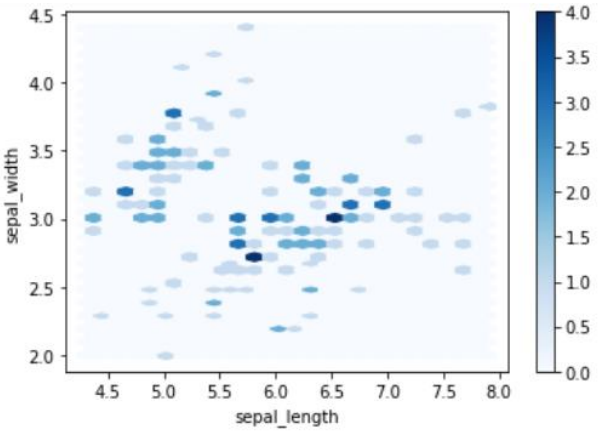

Рисунок 43. Скриншот с выводом графика с шестиугольной сеткой, аналог которого можно увидеть в позиции (2, 1) матрицы рассеяния

На рис. 44 представлен фрагмент программного кода, реализующего построение трёх вариантов контурных графиков для признаков **sepal_length** и **sepal_width**. На рис. 45 представлен скриншот с выводом 3 вариантов контурных графиков.

```
# Построение контурных графиков
# Установка seaborn-стиля
sns.set_style("white")

# Базовый 2D-график плотности
sns.kdeplot(x=df.sepal_length, y=df.sepal_width)
plt.show()

# Custom the color, add shade and bandwidth
sns.kdeplot(x=df.sepal_length, y=df.sepal_width,
            cmap="Reds", shade=True, bw_adjust=.5)
plt.show()

# Add thresh parameter
sns.kdeplot(x=df.sepal_length, y=df.sepal_width,
            cmap="Blues", shade=True, thresh=0)
plt.show()
```

Рисунок 44. Фрагмент программного кода, реализующего построение трёх вариантов контурных графиков для признаков sepal_length и sepal_width

На рис. 46 представлен фрагмент программного кода, реализующего вычисление корреляции между числовыми признаками и целевым признаком.

На рис. 47 представлен скриншот с выводом результатов вычисления корреляции между числовыми признаками и целевым признаком.

На рис. 48 представлен фрагмент программного кода, реализующего визуализацию результатов вычисления корреляции между числовыми признаками и целевым признаком. На рис. 49 представлен скриншот с визуализацией результатов вычисления корреляции между числовыми признаками и целевым признаком.

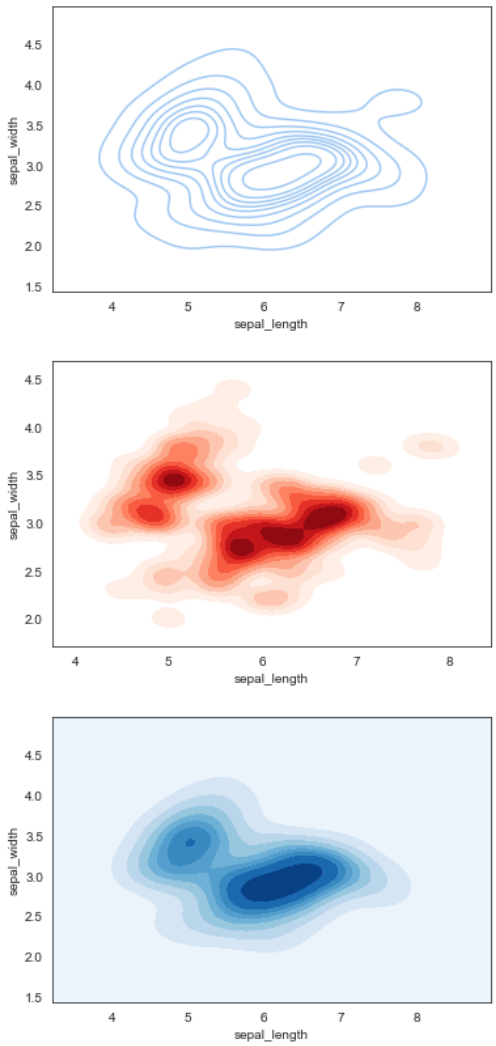

Рисунок 45. Скриншот с выводом 3 вариантов контурных графиков

```python
# Вычисление корреляции между числовыми признаками и целевым признаком,
# определяющим метку класса
species_codes = {"species":
                 {"setosa": 0,
                  "versicolor": 1,
                  "virginica": 2}}
df_codes = df.replace(species_codes)
# Инициализация пустого списка
tg_num_corr = []
for col in num_cols:
    tg_num_corr.append(df_codes[col].corr(df_codes.species))
# Создание датафрейма DataFrame
tg_num_df = pd.DataFrame({'Числовые признаки': num_cols,
                          'Корреляция с целевым признаком': tg_num_corr})
# Сортировка DataFrame по абсолютной величине коэффициента корреляции
# (по убыванию)
tg_num_df = tg_num_df.sort_values(by='Корреляция с целевым признаком',
                                  key=abs, ascending=False).reset_index(drop=True)
# Вывод DataFrame
tg_num_df
```

Рисунок 46. Фрагмент программного кода, реализующего вычисление корреляции между числовыми признаками и целевым признаком

	Числовые признаки	Корреляция с целевым признаком
0	petal_width	0.956547
1	petal_length	0.949035
2	sepal_length	0.782561
3	sepal_width	-0.426658

Рисунок 47. Скриншот с выводом результатов вычисления корреляции между числовыми признаками и целевым признаком

```
# Визуализация
plt.figure(figsize=(7,5))
sns.barplot(x=tg_num_df['Корреляция с целевым признаком'],
            y=tg_num_df['Числовые признаки'], color='#a2c9f4')
plt.ylabel('')
plt.xlabel('Коэффициент корелляции')
plt.title('Числовой признак - Целевой признак', fontsize=12)
plt.show()
```

Рисунок 48. Фрагмент программного кода, реализующего визуализацию результатов вычисления корреляции между числовыми признаками и целевым признаком

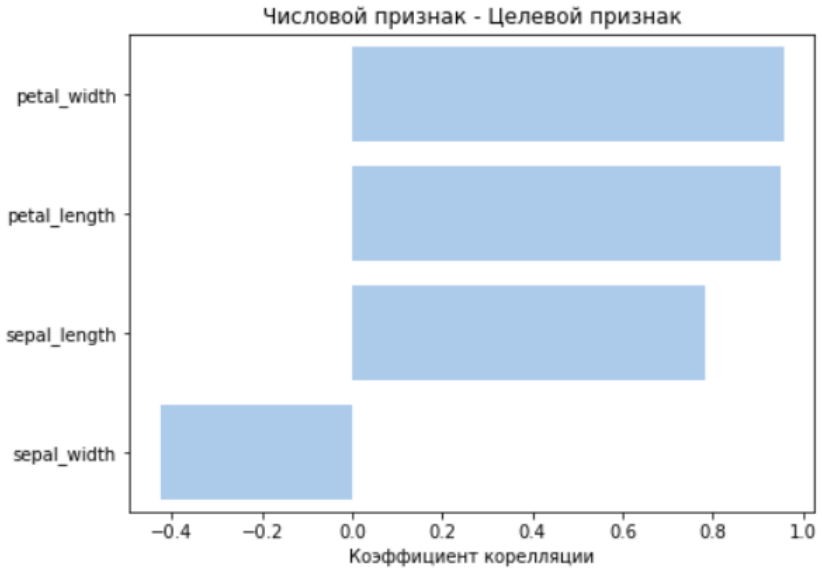

Рисунок 49. Скриншот с визуализацией результатов вычисления корреляции между числовыми признаками и целевым признаком

На рис. 50 представлен фрагмент программного кода, реализующего построение коробчатых диаграмм для признаков.

На рис. 51 представлен скриншот с выводом коробчатых диаграмм для признаков. Анализ рис. 51 позволяет сделать вывод о наличии выбросов по второму признаку, т.е. по признаку `sepal_width`.

```python
# Построение коробчатых диаграмм
fig, ax = plt.subplots(ncols=1, nrows=4, figsize=(7,20))
i=0
for col in num_cols:
    sns.boxplot(data=df[num_cols], x=col, ax=ax[i], palette='pastel')
    ax[i].set_xlabel(' ')
    ax[i].set_ylabel(' ')
    ax[i].set_title(col, fontsize=12)
    i=i+1
plt.show()
```

Рисунок 50. Фрагмент программного кода, реализующего построение коробчатых диаграмм для признаков

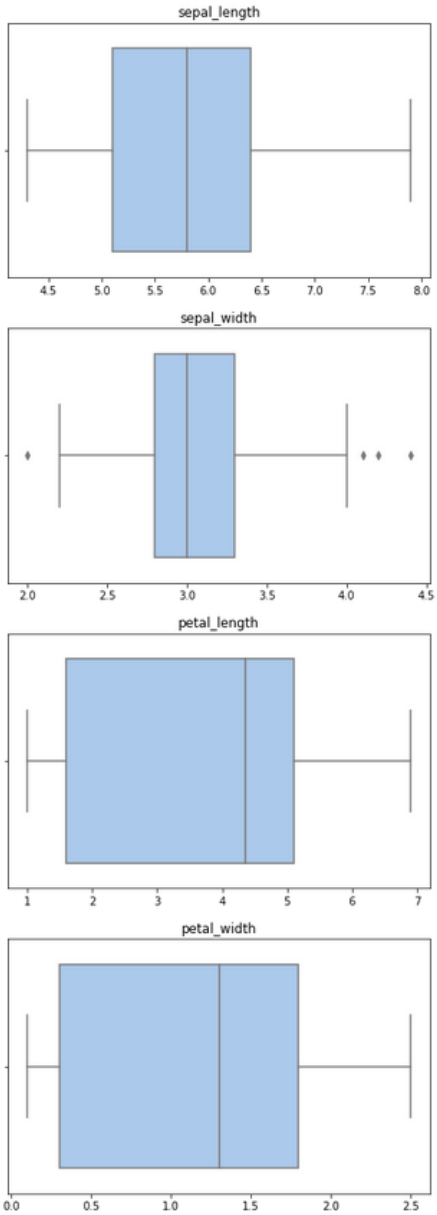

Рисунок 51. Скриншот с выводом коробчатых диаграмм для признаков

На рис. 52 представлен фрагмент программного кода, реализующего построение скрипичных диаграмм для признаков. На рис. 53 представлен скриншот с выводом скрипичных диаграмм для признаков.

```python
# Построение скрипичных диаграмм
fig, ax = plt.subplots(ncols=1, nrows=4, figsize=(7,20))
i=0
for col in num_cols:
    sns.violinplot(data=df[num_cols], x=col, ax=ax[i], palette='pastel')
    ax[i].set_xlabel(' ')
    ax[i].set_ylabel(' ')
    ax[i].set_title(col, fontsize=12)
    i=i+1
plt.show()
```

Рисунок 52. Фрагмент программного кода, реализующего построение скрипичных диаграмм для признаков

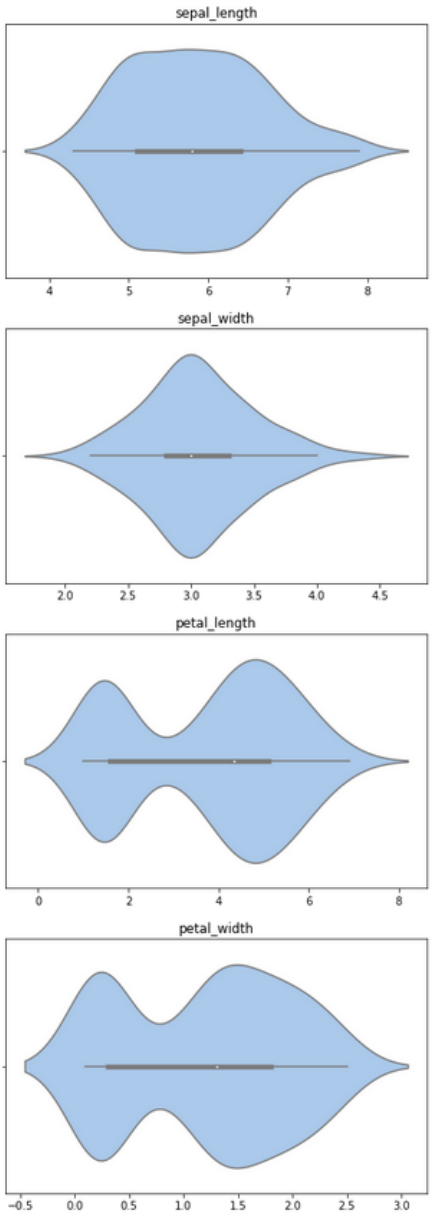

Рисунок 53. Скриншот с выводом скрипичных диаграмм для признаков

На рис. 54 представлен фрагмент программного кода, реализующего поиск выбросов для каждого признака с использованием Inner Fence. На рис. 55 представлен фрагмент программного кода, реализующего обзор выбросов для каж-

дого признака с использованием Inner Fence. На рис. 56 представлен скриншот с выводом выбросов для каждого признака с использованием Inner Fence. На рис. 57 представлен фрагмент программного кода, реализующего обзор уникальных выбросов для каждого признака с использованием Inner Fence. На рис. 58 представлен скриншот с выводом уникальных выбросов для каждого признака с использованием Inner Fence.

```
# Поиск выбросов для каждого признака с использованием Inner Fence
# Создание пустого словаря и списка для цикла
outlier_dict_if = {}          # Словарь признаков и всех выбросов
outlier_dict_unique_if = {}   # Словарь признаков и уникальных выбросов
# Цикл по всем признакам
for col in num_cols:
    # Поиск верхней и нижней границы по каждому признаку с использованием Inner Fence
    stats = df[col].describe()
    q1, q3 = stats['25%'], stats['75%']
    iqr = q3-q1
    lower_bound_if = q1 - (1.5 * iqr)
    upper_bound_if = q3 + (1.5 * iqr)

    # Добавление данных вне нижней и верхней границ в словарь
    # Инициализация пустого списка
    outlier_list_if = []
    outlier_list_unique_if = []
    for x in df[col]:
        if ((x > upper_bound_if) or (x < lower_bound_if)):
            outlier_list_if.append(x)
            # Если значение еще не в списке
            if x not in outlier_list_unique_if:
                outlier_list_unique_if.append(x)
        # Добавление списка к словарям
        outlier_dict_if['{}'.format(col)] = outlier_list_if
        outlier_dict_unique_if['{}'.format(col)] = outlier_list_unique_if
```

Рисунок 54. Фрагмент программного кода, реализующего поиск выбросов для каждого признака с использованием Inner Fence с последующим их обзором

```
# Обзор выбросов для каждого признака
for k,v in outlier_dict_if.items():
    print(k, 'Выбросы (Inner Fence):')
    print(v)
    print('Число:',len(v))
    print('====\n')
```

Рисунок 55. Фрагмент программного кода, реализующего обзор выбросов для каждого признака с использованием Inner Fence

На рис. 59 представлен фрагмент программного кода, реализующего формирование датафрейма с информацией о выбросах для каждого признака с использованием Inner Fence. На рис. 60 представлен скриншот с датафреймом, содержащим информацию о выбросах для каждого признака с использованием Inner Fence.

```
sepal_length Выбросы (Inner Fence):
[]
Число: 0
====

sepal_width Выбросы (Inner Fence):
[4.4, 4.1, 4.2, 2.0]
Число: 4
====

petal_length Выбросы (Inner Fence):
[]
Число: 0
====

petal_width Выбросы (Inner Fence):
[]
Число: 0
====
```

Рисунок 56. Скриншот с выводом выбросов для каждого признака с использованием Inner Fence

```
# Обзор уникальных выбросов для каждого признака
for k,v in outlier_dict_unique_if.items():
    print(k, 'Уникальный выброс (inner fence):')
    print('Число:',len(v))
    print('====\n')
```

Рисунок 57. Фрагмент программного кода, реализующего обзор уникальных выбросов для каждого признака с использованием Inner Fence

```
sepal_length Уникальный выброс (inner fence):
Число: 0
====

sepal_width Уникальный выброс (inner fence):
Число: 4
====

petal_length Уникальный выброс (inner fence):
Число: 0
====

petal_width Уникальный выброс (inner fence):
Число: 0
====
```

Рисунок 58. Скриншот с выводом уникальных выбросов для каждого признака с использованием Inner Fence

На рис. 61 представлен фрагмент программного кода, реализующего удаление выбросов для каждого признака с использованием Inner Fence. На рис. 62 представлен скриншот с информацией о проценте удаленных из набора данных строк с использованием Inner Fence.

```python
# Создание Dataframe
# Инициализация пустых списков
outlier_list_count_if = []
outlier_unique_count_if = []

# Цикл
for k, v in outlier_dict_if.items():
    outlier_list_count_if.append(len(v))
for k, v in outlier_dict_unique_if.items():
    outlier_unique_count_if.append(len(v))

# Создание Dataframe
outlier_df_if = pd.DataFrame({'Признак': num_cols,
                              'outlier_count_if': outlier_list_count_if,
                              'unique_outlier_count_if': outlier_unique_count_if})
outlier_df_if
```

Рисунок 59. Фрагмент программного кода, реализующего формирование датафрейма с информацией о выбросах для каждого признака с использованием Inner Fence

	Признак	outlier_count_if	unique_outlier_count_if
0	sepal_length	0	0
1	sepal_width	4	4
2	petal_length	0	0
3	petal_width	0	0

Рисунок 60. Скриншот с датафреймом, содержащим информацию о выбросах

```python
# Удаление выбросов на основе IQR с использованием внутренней границы
after_removal_if = df[num_cols]

for col in num_cols:
    stats = df[col].describe()
    q1, q3 = stats['25%'], stats['75%']
    iqr = q3-q1
    lower_bound_if = q1 - (1.5 * iqr)
    upper_bound_if = q3 + (1.5 * iqr)
    after_removal_if = after_removal_if[(after_removal_if[col] >= lower_bound_if)
                                        & (after_removal_if[col] <= upper_bound_if)]
# Процент удаленных записей
removed_if = 100*(df[num_cols].shape[0] -
                  after_removal_if.shape[0])/df[num_cols].shape[0]
print('Процент удаленных строк: {}%'.format(round(removed_if,2)))
```

Рисунок 61. Фрагмент программного кода, реализующего удаление выбросов для каждого признака с использованием Inner Fence

```
Процент удаленных строк: 2.67%
```

Рисунок 62. Скриншот с информацией о проценте удаленных из набора данных строк с использованием Inner Fence

Аналогичным образом может быть выполнен анализ и удаление выбросов при использовании Outer Fence (в этом случае коэффициент 1.5 заменяется на 3.5). В рассматриваемом примере при использовании Outer Fence выбросы обнаружить не удалось.

7.2. Пример выполнения обработки пропусков данных и сводных результатов её применения для набора данных, содержащего информацию о держателях кредитных карт

Рассмотрим аспекты обработки пропусков данных и сводные результаты её применения на примере набора данных **`CCGENERAL.csv`**, который представлен по ссылке:

https://www.kaggle.com/arjunbhasin2013/ccdata.

Этот набор данных содержит информацию об 8950 активных держателях кредитных карт, каждый из которых описывается с помощью 18 признаков, таких как:

CUST_ID – признак, определяющий идентификатор держателя кредитной карты;

BALANCE – признак, определяющий сумму баланса, имеющегося на кредитной карте, для совершения покупок;

BALANCE_FREQUENCY – признак, определяющий, как часто обновляется BALANCE, и принимающий значения от 0 до 1 (1 – часто обновляется, 0 – не часто обновляется);

PURCHASES – признак, определяющий число покупок, совершенных со счета;

ONEOFF_PURCHASES – признак, определяющий максимальную сумму покупки за один раз;

INSTALLMENTS_PURCHASES – признак, определяющий сумму покупки, совершенной в рассрочку;

CASH_ADVANCE – признак, определяющий предоплату, предоставленную держателем кредитной карты;

PURCHASES_FREQUENCY – признак, определяющий, как часто совершаются покупки, и принимающий значения от 0 до 1 (1 – часто покупают, 0 – не часто покупают);

ONEOFF_PURCHASES_FREQUENCY – признак, определяющий частоту покупок за один раз (1 – часто покупают, 0 – не часто покупают);

PURCHASES_INSTALLMENTS_FREQUENCY – признак, определяющий частоту совершения покупки в рассрочку (1 – часто, 0 – не часто);

CASH_ADVANCE_FREQUENCY– признак, определяющий частоту предоплаты наличными;

CASH_ADVANCE_TRX – признак, определяющий число транзакций, совершенных с помощью «Cash in Advanced»;

PURCHASES_TRX – признак, определяющий число совершенных транзакций при покупках;

CREDIT_LIMIT – признак, определяющий лимит кредитной карты;

PAYMENTS – признак, определяющий сумму платежа, произведенного держателем кредитной карты;

MINIMUM_PAYMENTS – признак, определяющий минимальную сумму платежей, совершаемых держателем кредитной карты;

PRC_FULL_PAYMENT – признак, определяющий процент полной оплаты держателем кредитной карты;

TENURE – признак, определяющий срок действия кредитной карты для держателя кредитной карты.

Набор данных **`CCGENERAL.csv`** обычно используется для демонстрации результатов решения задачи кластеризации с применением того или иного алгоритма кластеризации. Следует отметить, что при решении задачи кластеризации признак CUST_ID необходимо исключить из набора данных, т.к. значения этого признака являются уникальными.

В результате набор данных будет содержать17 числовых признаков (все признаки, кроме признака CUST_ID), 3 из которых имеют тип `int64`, а 14 – тип `float64`.

На рис. 63 представлен фрагмент программного кода, реализующего импорт библиотеки и вывод сводной информации о наборе данных, загруженном в датафрейм.

```
# Импорт библиотеки
import os
```

```
# Считывание набора данных
directory='./Data/' # папка с файлами данных
f = os.path.join(directory, 'CC GENERAL.csv')
data=pd.read_csv(f, sep=',')
```

```
# Определение числа строк и столбцов в наборе данных
[n1,n2]=data.shape
```

```
# Число объектов равно числу строк
# (первая строка, содержащая названия признаков, не учитывается)
n1
```

```
8950
```

```
# Число признаков
n2
```

```
18
```

```
# Вывод информации по набору данных
data.info()
```

Рисунок 63. Фрагмент программного кода, реализующего импорт библиотеки и вывод сводной информации о наборе данных, загруженном в датафрейм

На рис. 64 представлен скриншот, отражающий вывод сводной информации о наборе данных, загруженном в датафрейм.

```
# Вывод информации по набору данных
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUST_ID                           8950 non-null   object
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64
 12  PURCHASES_TRX                     8950 non-null   int64
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

Рисунок 64. Скриншот, отражающий вывод сводной информации о наборе данных

Анализ сводной информации по набору данных (рис. 64) позволяет сделать вывод о том, что в наборе данных есть пропуски по признакам CREDIT_LIMIT и MINIMUM_PAYMENTS в количестве 1 и 313 значений соответственно.

На рис. 65 представлен фрагмент программного кода, реализующего вывод информации о первых 5 строках набора данных, загруженного в датафрейм. На рис. 66 представлен скриншот с фрагментом информации о первых 5 строках набора данных, загруженного в датафрейм. При этом на рис. 66 видно, что одно из 5 отображенных значений признака MINIMUM_PAYMENTS равно `NaN`.

```
# Вывод первых 5 строк в DataFrame с набором данных
data.head()
```

Рисунок 65. Фрагмент программного кода, реализующего вывод информации о первых 5 строках набора данных, загруженного в датафрейм

CASH_ADVANCE_TRX	PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
0	2	1000.0	201.802084	139.509787	0.000000	12
4	0	7000.0	4103.032597	1072.340217	0.222222	12
0	12	7500.0	622.066742	627.284787	0.000000	12
1	1	7500.0	0.000000	NaN	0.000000	12
0	1	1200.0	678.334763	244.791237	0.000000	12

Рисунок 66. Скриншот с фрагментом информации о первых 5 строках набора данных, загруженного в датафрейм

На рис. 67 представлен фрагмент программного кода, реализующего вывод информации об общем числе пропусков данных. Кроме того, на рис. 67 отображено вычисленное значение для общего числа пропусков данных.

```
# Общее число пропусков
data.isnull().sum().sum()
```

```
314
```

Рисунок 67. Фрагмент программного кода, реализующего вывод информации об общем числе пропусков данных, а также вывод информации об общем числе пропусков данных

Для определения строк (т.е. объектов, а именно – держателей кредитных карт) со значениями `NaN` (т.е. с пропусками) и проверки, есть ли у какого-либо объекта пропуски по нескольким признакам, можно использовать фрагмент программного кода, приведенный на рис. 68, или фрагмент программного кода, приведенный на рис. 69. Фрагмент программного кода на рис. 68 реализует первый способ определения числа строк со значениями `NaN` и проверки, есть ли в какой-либо строке пропуски по нескольким признакам. Фрагмент программного кода на рис. 69 реализует первый способ определения числа строк со значениями `NaN` и проверки, есть ли в какой-либо строке пропуски по нескольким признакам (этот способ идентичен первому по выводимым результатам поиска).

```
# Определение строк (т.е. объектов) с NaN (т.е. с пропусками)
# и проверка: есть ли у какого-либо объекта пропуски по нескольким
# (в нашем случае по двум) признакам)

# (первый способ)
nan_rows = data[data.isnull().T.any()]
nan_rows
```

Рисунок 68. Фрагмент программного кода, реализующий первый способ определения числа строк со значениями NaN и проверки, есть ли в какой-либо строке пропуски по нескольким признакам

```
# Определение строк (т.е. объектов) с NaN (т.е. с пропусками)
# и проверка: есть ли у какого-либо объекта пропуски по нескольким
# (в нашем случае по двум) признакам)

# (второй способ, равносильный первому)
nan_rows = data[data.isnull().any(1)]
nan_rows
```

Рисунок 68. Фрагмент программного кода, реализующий второй способ определения числа строк со значениями NaN и проверки, есть ли в какой-либо строке пропуски по нескольким признакам

На рис. 70 приведен скриншот с выводом строк с пропусками данных (в том числе, выводится общее число строк с пропусками).

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0
45	C10047	2242.311686	1.000000	437.00	97.00	340.0
47	C10049	3910.111237	1.000000	0.00	0.00	0.0
54	C10056	6.660517	0.636364	310.00	0.00	310.0
55	C10057	1311.995984	1.000000	1283.90	1283.90	0.0
...	...	...	...	...	...	...
8919	C19160	14.524779	0.333333	152.00	152.00	0.0
8929	C19170	371.527312	0.333333	0.00	0.00	0.0
8935	C19176	183.817004	1.000000	465.90	0.00	465.9
8944	C19185	193.571722	0.833333	1012.73	1012.73	0.0
8946	C19187	19.183215	1.000000	300.00	0.00	300.0

314 rows × 18 columns

Рисунок 70. Скриншот с выводом строк с пропусками данных

Следует отметить, что пропущенные значения для тех или иных признаков могут быть помечены разными символами, соответственно, необходимо внимательно выявлять эти символы, чтобы адекватным образом обрабатывать пропуски данных.

В случае, когда для обозначения пропущенных значений используются `NA/NaN` значения, применяется функция `fillna()`, информация по которой доступна по ссылке:

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html.

Эта функция заполняет пропуски, используя тот или иной выбранный (специфицированный) метод.

Синтаксис функции:

`DataFrame.fillna(`*`value=None, method=None, axis=None, inplace=False, limit=None, downcast=None`*`)`

Ниже рассмотрены 2 варианта заполнения пропусков данных: средним значением по столбцу с использованием `mean()` и медианным значением по столбцу с использованием `median()`.

На рис. 71 представлен фрагмент программного кода, реализующего заполнение пропусков данных средним значением с применением `mean()` и вывод информации о результатах заполнения, свидетельствующих о том, что пропуски заполнены.

```
# Заполнение пропусков данных
data_fill=data.fillna(data.mean())
```

```
# Вывод информации по пропускам данных по каждому признаку
# Теперь пропусков нет!!!
data_fill.isnull().sum()
```

Рисунок 71. Фрагмент программного кода, реализующего заполнение пропусков данных средним значением с применением mean() и вывод информации о результатах заполнения, свидетельствующих о том, что пропуски заполнены

На рис. 72 представлен фрагмент программного кода, реализующего заполнение пропусков данных средним значением с применением `median()` и вывод информации о результатах заполнения, свидетельствующих о том, что пропуски заполнены.

```
# Заполнение пропусков данных
data_fill=data.fillna(data.median())
```

```
# Вывод информации по пропускам данных по каждому признаку
# Теперь пропусков нет!!!
data_fill.isnull().sum()
```

Рисунок 72. Фрагмент программного кода, реализующего заполнение пропусков данных средним значением с применением median() и вывод информации о результатах заполнения, свидетельствующих о том, что пропуски заполнены

На рис. 73 представлен скриншот с выводом информации о результатах заполнения, свидетельствующих о том, что пропуски данных заполнены.

Для выполнения сравнительного анализа результатов заполнения пропусков с применением 2 подходов предварительно было выполнено удаление из набора данных нечислового признака `CUST_ID`, определяющего идентификатор держателя кредитной карты.

На рис. 74 представлен фрагмент программного кода, реализующего удаление нечислового признака `CUST_ID`. На рис. 75 представлен скриншот с результатами удаления нечислового признака `CUST_ID`.

```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

Рисунок 73. Скриншот с выводом информации о результатах заполнения, свидетельствующих о том, что пропуски данных заполнены

На рис. 74 представлен фрагмент программного кода, реализующего вывод описательной статистики числовых признаков исходного набора данных. На рис. 75 представлен скриншот с выводом описательной статистики числовых признаков исходного набора данных.

```
# Описание статистики числовых признаков
num_stat = data[num_cols].describe().transpose().reset_index()
num_stat.rename(columns={'index': 'Признак'}, inplace=True)
num_stat
```

Рисунок 74. Фрагмент программного кода, реализующего вывод описательной статистики числовых признаков исходного набора данных

	Признак	count	mean	std	min	25%	50%	75%	max
0	BALANCE	8950.0	1564.474828	2081.531879	0.000000	128.281915	873.385231	2054.140036	19043.13856
1	BALANCE_FREQUENCY	8950.0	0.877271	0.236904	0.000000	0.888889	1.000000	1.000000	1.00000
2	PURCHASES	8950.0	1003.204834	2136.634782	0.000000	39.635000	361.280000	1110.130000	49039.57000
3	ONEOFF_PURCHASES	8950.0	592.437371	1659.887917	0.000000	0.000000	38.000000	577.405000	40761.25000
4	INSTALLMENTS_PURCHASES	8950.0	411.067645	904.338115	0.000000	0.000000	89.000000	468.637500	22500.00000
5	CASH_ADVANCE	8950.0	978.871112	2097.163877	0.000000	0.000000	0.000000	1113.821139	47137.21176
6	PURCHASES_FREQUENCY	8950.0	0.490351	0.401371	0.000000	0.083333	0.500000	0.916667	1.00000
7	ONEOFF_PURCHASES_FREQUENCY	8950.0	0.202458	0.298336	0.000000	0.000000	0.083333	0.300000	1.00000
8	PURCHASES_INSTALLMENTS_FREQUENCY	8950.0	0.364437	0.397448	0.000000	0.000000	0.166667	0.750000	1.00000
9	CASH_ADVANCE_FREQUENCY	8950.0	0.135144	0.200121	0.000000	0.000000	0.000000	0.222222	1.50000
10	CASH_ADVANCE_TRX	8950.0	3.248827	6.824647	0.000000	0.000000	0.000000	4.000000	123.00000
11	PURCHASES_TRX	8950.0	14.709832	24.857649	0.000000	1.000000	7.000000	17.000000	358.00000
12	CREDIT_LIMIT	8949.0	4494.449450	3638.815725	50.000000	1600.000000	3000.000000	6500.000000	30000.00000
13	PAYMENTS	8950.0	1733.143852	2895.063757	0.000000	383.276166	856.901546	1901.134317	50721.48336
14	MINIMUM_PAYMENTS	8637.0	864.206542	2372.446607	0.019163	169.123707	312.343947	825.485459	76406.20752
15	PRC_FULL_PAYMENT	8950.0	0.153715	0.292499	0.000000	0.000000	0.000000	0.142857	1.00000
16	TENURE	8950.0	11.517318	1.338331	6.000000	12.000000	12.000000	12.000000	12.00000

Рисунок 75. Скриншот с выводом описательной статистики числовых признаков исходного набора данных

На рис. 76 представлен фрагмент программного кода, реализующего вывод описательной статистики числовых признаков набора данных, полученного после заполнения пропусков данных (с использованием `mean()` или `median()`).

```
# Описание статистики числовых признаков
num_stat = data_fill[num_cols].describe().transpose().reset_index()
num_stat.rename(columns={'index': 'Признак'}, inplace=True)
num_stat
```

Рисунок 76. Фрагмент программного кода, реализующего вывод описательной статистики числовых признаков набора данных, полученного после заполнения пропусков данных

На рис. 77 представлен скриншот с выводом описательной статистики числовых признаков набора данных, полученного после заполнения пропусков данных с использованием `mean()`.

	Признак	count	mean	std	min	25%	50%	75%	max
0	BALANCE	8950.0	1564.474828	2081.531879	0.000000	128.281915	873.385231	2054.140036	19043.13856
1	BALANCE_FREQUENCY	8950.0	0.877271	0.236904	0.000000	0.888889	1.000000	1.000000	1.00000
2	PURCHASES	8950.0	1003.204834	2136.634782	0.000000	39.635000	361.280000	1110.130000	49039.57000
3	ONEOFF_PURCHASES	8950.0	592.437371	1659.887917	0.000000	0.000000	38.000000	577.405000	40761.25000
4	INSTALLMENTS_PURCHASES	8950.0	411.067645	904.338115	0.000000	0.000000	89.000000	468.637500	22500.00000
5	CASH_ADVANCE	8950.0	978.871112	2097.163877	0.000000	0.000000	0.000000	1113.821139	47137.21176
6	PURCHASES_FREQUENCY	8950.0	0.490351	0.401371	0.000000	0.083333	0.500000	0.916667	1.00000
7	ONEOFF_PURCHASES_FREQUENCY	8950.0	0.202458	0.298336	0.000000	0.000000	0.083333	0.300000	1.00000
8	PURCHASES_INSTALLMENTS_FREQUENCY	8950.0	0.364437	0.397448	0.000000	0.000000	0.166667	0.750000	1.00000
9	CASH_ADVANCE_FREQUENCY	8950.0	0.135144	0.200121	0.000000	0.000000	0.000000	0.222222	1.50000
10	CASH_ADVANCE_TRX	8950.0	3.248827	6.824647	0.000000	0.000000	0.000000	4.000000	123.00000
11	PURCHASES_TRX	8950.0	14.709832	24.857649	0.000000	1.000000	7.000000	17.000000	358.00000
12	CREDIT_LIMIT	8949.0	4494.449450	3638.815725	50.000000	1600.000000	3000.000000	6500.000000	30000.00000
13	PAYMENTS	8950.0	1733.143852	2895.063757	0.000000	383.276166	856.901546	1901.134317	50721.48336
14	MINIMUM_PAYMENTS	8637.0	864.206542	2372.446607	0.019163	169.123707	312.343947	825.485459	76406.20752
15	PRC_FULL_PAYMENT	8950.0	0.153715	0.292499	0.000000	0.000000	0.000000	0.142857	1.00000
16	TENURE	8950.0	11.517318	1.338331	6.000000	12.000000	12.000000	12.000000	12.00000

Рисунок 77. Скриншот с выводом описательной статистики числовых признаков набора данных, полученного после заполнения пропусков данных с использованием mean()

На рис. 78 представлен скриншот с выводом описательной статистики числовых признаков набора данных, полученного после заполнения пропусков данных с использованием `median()`.

При этом имеет смысл сравнить описательные статистики на рис. 75, 77 и 78 для признаков CREDIT_LIMIT и MINIMUM_PAYMENTS, сопоставленных соответственно строкам с номерами 12 и 14, чтобы оценить влияние разных способов заполнения пропусков данных на вычисленные значения статистик.

При этом, в первую очередь, следует обратить внимание на статистики `mean` и `std`, значения которых существенно различаются, особенно для признака MINIMUM_PAYMENTS, имеющего большое число пропусков.

	Признак	count	mean	std	min	25%	50%	75%	max
0	BALANCE	8950.0	1564.474828	2081.531879	0.000000	128.281915	873.385231	2054.140036	19043.13856
1	BALANCE_FREQUENCY	8950.0	0.877271	0.236904	0.000000	0.888889	1.000000	1.000000	1.00000
2	PURCHASES	8950.0	1003.204834	2136.634782	0.000000	39.635000	361.280000	1110.130000	49039.57000
3	ONEOFF_PURCHASES	8950.0	592.437371	1659.887917	0.000000	0.000000	38.000000	577.405000	40761.25000
4	INSTALLMENTS_PURCHASES	8950.0	411.067645	904.338115	0.000000	0.000000	89.000000	468.637500	22500.00000
5	CASH_ADVANCE	8950.0	978.871112	2097.163877	0.000000	0.000000	0.000000	1113.821139	47137.21176
6	PURCHASES_FREQUENCY	8950.0	0.490351	0.401371	0.000000	0.083333	0.500000	0.916667	1.00000
7	ONEOFF_PURCHASES_FREQUENCY	8950.0	0.202458	0.298336	0.000000	0.000000	0.083333	0.300000	1.00000
8	PURCHASES_INSTALLMENTS_FREQUENCY	8950.0	0.364437	0.397448	0.000000	0.000000	0.166667	0.750000	1.00000
9	CASH_ADVANCE_FREQUENCY	8950.0	0.135144	0.200121	0.000000	0.000000	0.000000	0.222222	1.50000
10	CASH_ADVANCE_TRX	8950.0	3.248827	6.824647	0.000000	0.000000	0.000000	4.000000	123.00000
11	PURCHASES_TRX	8950.0	14.709832	24.857649	0.000000	1.000000	7.000000	17.000000	358.00000
12	CREDIT_LIMIT	8950.0	4494.282473	3638.646702	50.000000	1600.000000	3000.000000	6500.000000	30000.00000
13	PAYMENTS	8950.0	1733.143852	2895.063757	0.000000	383.276166	856.901546	1901.134317	50721.48336
14	MINIMUM_PAYMENTS	8950.0	844.906767	2332.792322	0.019163	170.857654	312.343947	788.713501	76406.20752
15	PRC_FULL_PAYMENT	8950.0	0.153715	0.292499	0.000000	0.000000	0.000000	0.142857	1.00000
16	TENURE	8950.0	11.517318	1.338331	6.000000	12.000000	12.000000	12.000000	12.00000

,

Рисунок 78. Скриншот с выводом описательной статистики числовых признаков набора данных, полученного после заполнения пропусков данных с использованием median()

Все остальные инструменты разведочного анализа применяются аналогично тому, как они применялись в п. 7.1 для набора данных «»Ирисы Фишера». Результаты разведочного анализа будет несколько различаться ввиду использования разных способов заполнения пропусков данных.

ВАРИАНТЫ И ЗАДАНИЯ

Необходимо выполнить задания по разведочному анализу предлагаемых наборов данных в соответствии с предложенным вариантом.

Выполнить разведочный анализ данных наборов данных, приведенных в табл. 10 и 11, считая, что метки кластеров неизвестны, используя программные реализации инструментов разведочного анализа, приведенные в п. 7. Если метки классов известны, то необходимо выполнить анализ корреляционной связи числовых признаков объектов и целевого признака, определяющего метки классов объектов.

Информацию по работе с языком Python можно получить в [15, 16].

Варианты

1. Наборы данных с *известными* метками классов объектов.

Таблица 10. Варианты наборов данных

Вариант	Источник набора данных
1	http://archive.ics.uci.edu/ml/datasets/Post-Operative+Patient
2	http://archive.ics.uci.edu/ml/datasets/Primary+Tumor
3	http://archive.ics.uci.edu/ml/datasets/Soybean+%28Large%29
4	http://archive.ics.uci.edu/ml/datasets/Low+Resolution+Spectrometer
5	http://archive.ics.uci.edu/ml/datasets/Ionosphere
6	http://archive.ics.uci.edu/ml/datasets/Horse+Colic
7	http://archive.ics.uci.edu/ml/datasets/Hepatitis
8	http://archive.ics.uci.edu/ml/datasets/Heart+Disease
9	http://archive.ics.uci.edu/ml/datasets/Hayes-Roth
10	http://archive.ics.uci.edu/ml/datasets/Haberman%27s+Survival
11	http://archive.ics.uci.edu/ml/datasets/Glass+Identification
12	http://archive.ics.uci.edu/ml/datasets/Flags
13	http://archive.ics.uci.edu/ml/datasets/Ecoli
14	http://archive.ics.uci.edu/ml/datasets/Echocardiogram
15	http://archive.ics.uci.edu/ml/datasets/Dermatology
16	http://archive.ics.uci.edu/ml/datasets/Credit+Approval
17	http://archive.ics.uci.edu/ml/datasets/Pittsburgh+Bridges
18	http://archive.ics.uci.edu/ml/datasets/Spambase
19	http://archive.ics.uci.edu/ml/datasets/University
20	http://archive.ics.uci.edu/ml/datasets/statlog+(australian+credit+approval)

2. Наборы данных с *неизвестными* метками классов объектов.

Таблица 11. Варианты наборов данных

Вариант	Источник набора данных
1	https://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities
2	https://archive.ics.uci.edu/ml/datasets/Breath+Metabolomics
3	https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT
4	https://archive.ics.uci.edu/ml/datasets/DrivFace
5	https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
6	https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition
7	https://archive.ics.uci.edu/ml/datasets/FMA%3A+A+Dataset+For+Music+Analysis
8	https://archive.ics.uci.edu/ml/datasets/Gas+Sensor+Array+Drift+Dataset+at+Different+Concentrations
9	https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq
10	https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones
11	https://archive.ics.uci.edu/ml/datasets/Kitsune+Network+Attack+Dataset
12	https://archive.ics.uci.edu/ml/datasets/MEx
13	https://archive.ics.uci.edu/ml/datasets/Mturk+User-Perceived+Clusters+over+Images
14	https://archive.ics.uci.edu/ml/datasets/Repeat+Consumption+Matrices
15	https://archive.ics.uci.edu/ml/datasets/Reuter_50_50
16	https://www.kaggle.com/arjunbhasin2013/ccdata
17	https://www.kaggle.com/mrisdal/open-exoplanet-catalogue
18	https://www.kaggle.com/imdevskp/corona-virus-report
19	https://www.kaggle.com/roshansharma/mall-customers-clustering-analysis
20	https://www.kaggle.com/karnikakapoor/customer-segmentation-clustering

СПИСОК ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Тьюки Дж. Анализ результатов наблюдений. Разведочный анализ. – М.: Мир, 1981. – 696 с.
2. Брюс П., Брюс Э. Практическая статистика для специалистов Data Science. СПб.: БХВ-Петербург. 2018. С. 19–58. 304 с.
3. John W. Tukey. The Future of Data Analysis // The Annals of Mathematical Statistics, Vol. 33. No. 1. 1962. pp. 1-67.
4. Tukey J.W. Exploratory data analysis. Reading, PA: Addison-Wesley. 1977. 711 p.
5. Behrens J.T. Principles and Procedures of Exploratory Data Analysis // Psychological Methods Copyright 1997 by the American Psychological Association, Inc. 1997. Vol. 2. No. 2. pp. 131-160.
6. Exploratory data analysis [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Exploratory_data_analysis, свободный (дата обращения 12.10.2022).
7. Sailem H.Z., Sero J.E., Bakal C. Visualizing cellular imaging data using Pheno Plot // Nature Communication. 2015. Jan 8; 6: 5825. doi: 10.1038/ncomms682.
8. pandas [Электронный ресурс]. – Режим доступа: https://pandas.pydata.org/, свободный (дата обращения 12.10.2022).
9. statistics 1.0.3.5 [Электронный ресурс]. – Режим доступа: https://pypi.org/project/statistics/, свободный (дата обращения 12.10.2022).
10. numpy 1.23.3 [Электронный ресурс]. – Режим доступа: https://numpy.org/, свободный (дата обращения 12.10.2022).
11. SciPy [Электронный ресурс]. – Режим доступа: https://pypi.org/project/scipy/, свободный (дата обращения 12.10.2022).
12. seaborn: statistical data visualization [Электронный ресурс]. – Режим доступа: https://seaborn.pydata.org/, свободный (дата обращения 12.10.2022).
13. matplotlib: Visualization with Python [Электронный ресурс]. – Режим доступа: https://matplotlib.org/, свободный (дата обращения 12.10.2022).
14. Iris Data Set [Электронный ресурс]. – Режим доступа: http://archive.ics.uci.edu/ml/datasets/Iris, свободный (дата обращения 12.10.2022).
15. Лутц М. Изучаем Python. М.: Диалектика. 2020. Том. 1. 832 с.; Том. 2. 720 с.
16. Доусон М. Программируем на Python. СПб.: Питер. 2020. 416 с.

Сведения об авторах

Демидова Лилия Анатольевна, доктор технических наук, профессор, профессор кафедры корпоративных информационных систем Института информационных технологий РТУ МИРЭА.