

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
**«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ имени В. И. ВЕРНАДСКОГО»**  
(ФГАОУ ВО «КФУ им. В. И. Вернадского»)  
**Физико-технический институт**  
Кафедра прикладной математики

Манамс Александр Николай Павел

**СОЗДАНИЕ УМНОГО ПОМОЩНИКА ДЛЯ СПЕЦИАЛИСТОВ,  
ПРОВОДЯЩИХ СОЦИОЛОГИЧЕСКИЕ ИССЛЕДОВАНИЯ**

Выпускная квалификационная работа

Обучающегося 4 курса

Направления подготовки 01.03.04. Прикладная математика

Форма обучения очная

Научный руководитель  
Старший преподаватель  
кафедры прикладной математики E. A. Стус

К ЗАЩИТЕ ДОПУСКАЮ:

Заведующий кафедрой,  
кандидат педагогических наук,  
доцент E. A. Косова

Симферополь, 2024

## Содержание

Введение .....	3
РАЗДЕЛ 1. СТАТИСТИЧЕСКИЕ МЕТОДЫ ОБРАБОТКИ ЭМПИРИЧЕСКИХ ДАННЫХ .....	5
1.1. Основные понятия .....	5
1.2. Язык программирования для анализа данных.....	7
1.3. Правила и методы сбора эмпирических данных .....	8
1.4. Методы обработки социологических данных .....	11
1.5. Расчёт описательных статистик распределения и их реализация в умном помощнике.....	17
1.5. Способы визуализации данных .....	21
1.6. Проверка статистических гипотез.....	24
РАЗДЕЛ 2. СОЗДАНИЕ УМНОГО ПОМОЩНИКА ДЛЯ АНАЛИЗА ДАННЫХ С ПОМОЩЬЮ СРЕДСТВ ЯЗЫКА PYTHON .....	30
2.1. Сбор данных .....	30
2.2. Обработка данных умным помощником.....	31
2.3. Анализ данных умным помощником .....	54
ЗАКЛЮЧЕНИЕ .....	62
Список использованной литературы.....	65
Приложения .....	67
Приложение А .....	67
Приложение Б.....	68
Приложение В .....	69
Приложение Г .....	71
Приложение Д .....	74

## Введение

Социологические исследования имеют крайне широкое распространение, их целью является сбор информации о различных социальных процессах и явлениях, а также связях между ними.

Целью работы является изучение методов обработки и анализа данных социологических исследований, создание на их основе умного помощника для специалистов, проводящих социологические исследования, и его практическое применение к эмпирическим данным, полученным в результате социологического опроса трудоустроенных представителей молодёжи Крыма.

Задачи работы:

- 1) Изучение правил проведения и методов выборки в социологических исследованиях.
- 2) Изучение методов обработки и анализа эмпирических данных социологических исследований.
- 3) Создание умного помощника для специалистов, проводящих социологические исследования.
- 4) Сбор данных среди трудоустроенных представителей молодёжи Крыма для социологического исследования.
- 5) Обработка и анализ эмпирических данных социологического исследования с помощью умного помощника, в ходе которого будет получено представление об участии граждан в общественной жизни их региона.

Объектом исследования являются методы обработки и анализа эмпирических данных, полученных в ходе проведения социологического исследования.

Предметом исследования является реализация методов обработки социологических данных с помощью средств языка программирования высокого уровня Python.

Для проведения социологического исследования необходимо выполнить следующие этапы: провести статистическое наблюдение, обработать полученные данные, провести их анализ, проверки статистических гипотез.

Работа состоит из двух разделов. Первый раздел называется «Статистические методы обработки эмпирических данных» и

рассматривает правила и методы проведения выборки, методы обработки и анализа данных социологических исследований.

Второй раздел «Создание умного помощника для анализа данных с помощью средств языка Python» - включает в себя практическую часть, в которой применяются методы, рассмотренные в первом разделе при создании умного помощника для специалистов, проводящих социологические исследования, и их применение к эмпирическим данным, полученным в ходе проведения социологического исследования «Изучение участия трудоустроенных представителей молодёжи Крыма в общественной жизни их региона».

# РАЗДЕЛ 1. СТАТИСТИЧЕСКИЕ МЕТОДЫ ОБРАБОТКИ ЭМПИРИЧЕСКИХ ДАННЫХ

## 1.1. Основные понятия

Социологическое исследование – системный процесс изучения различных процессов и явлений в обществе в целях получения новой научной информации об общественных явлениях и процессах в обществе. Эта информация может позволить предугадывать возможные итоги происходящих в обществе процессов, находить их причины, и показывать, как можно повлиять на эти процессы и явления [1].

Методы обработки и анализа эмпирических данных в социологии могут давать прогнозы только с той или иной степенью вероятности.

Для осуществления любого социологического исследования необходимо выполнить следующие этапы:

- 1) Статистическое наблюдение, которое можно представить в виде совокупности случайных событий и случайных величин.
- 2) Обработка данных, полученных в ходе статистического наблюдения, и их представление в удобном для последующего анализа данных виде.
- 3) Анализ обработанных данных, работа с целью проверки статистических гипотез, изучения результатов исследования и создания выводов на основе проведённого анализа данных. Данные выводы позволяют далее оказывать влияние на происходящие явления и процессы.

В процессе социологического исследования производится измерение, то есть приданье, согласно правилам, зависящим от постановки задачи, некоторых числовых значений признакам объектов и объектам. Вышеуказанные действия необходимы для получения требуемых результатов социологического исследования в виде математической модели исследования.

В ходе социологического исследования используются разнообразные измерительные шкалы. В процессе социологического исследования шкалу измерений называют основным инструментом измерения. Она служит эталоном для определённого набора значений, которые рассматриваются в

процессе исследования. В результате своих действий с её помощью исследователь может привести к количественным показателям, которые будут сопоставимыми, значения, которые ранее не могли сопоставляться, так как были качественно различными. Характер признаков, которые планируется измерять, и постановка задачи определяют то, какой тип шкалы будет применён. Существуют номинальные, ранговые (порядковые) и метрические шкалы [2].

Метрические шкалы делятся на два подтипа: шкала отношений и интервальная шкала. Для отражения отношений пропорции используется шкала отношений. Во время анализа силы проявления силы свойств объектов, выражаемых некоторыми величинами, которые разделены на интервалы равной длины, используется интервальная шкала. При этом нуль на данном типе шкалы является условным [2].

Ранговая (порядковая) шкала, как правило, используется при сравнении силы проявления признаков по убыванию и возрастанию. При использовании порядковой шкалы каждому рангу присваивается некоторое число. Вместе с этим, если числа на шкале заменить на иные, то порядок расположения рангов на шкале сохраняется, не изменяя уже имеющегося порядка [2].

Номинальная шкала, как правило, используется при классификации объектов и их характеристик. Она используется для определения различных групп объектов, когда значения на шкале не поддаются сравнению между собой, то есть каждой группе присваивается определённая позиция на шкале [2].

Для сбора информации в социологических исследованиях применяют различные методы. Основные из них: анкетирование, интервью, наблюдение, эксперимент, анализ документов. Каждый из этих методов имеет свои преимущества и недостатки. В данной работе применяется только анкетирование.

В социологическом исследовании для построения графиков и различных расчётов можно не использовать программное обеспечение, однако его использование позволяет существенно ускорить ход вычислений, увеличить точность вычислений. Для подобных целей может применяться множество прикладных программ, например, Libre Office, Microsoft Office, STATISTICA, SPSS и другие, так же возможно применение языков программирования со специальными библиотеками, как Python, R и другие.

## **1.2. Язык программирования для анализа данных**

Существует множество языков программирования, которые имеют готовые библиотеки для специализированных методов обработки и анализа данных, а также их отображения, которые применяются при проведении социологического исследования.

Одними из наиболее часто применяемых языков программирования при проведении социологических исследований используют языки программирования Python и R. Оба языка имеют широкий выбор уже существующих библиотек, которые позволяют упростить написание кода, который будет использоваться при проведении социологического исследования.

Язык программирования R был специально создан для математических расчётов, статистического анализа данных и машинного обучения.

Язык программирования Python является универсальным языком программирования, имея множество сфер применения, и при этом не уступает языку R в функциональности при использовании в математических расчётах, статистическом анализе и машинном обучении.

В данной работе был выбран язык программирования Python, так как он ни в чём значительно не уступает другим языкам программирования при использовании в социологическом исследовании, и именно в Python имелся наибольший опыт написания кода.

При создании умного помощника использовались библиотеки Python: numpy, pandas, matplotlib, math, scipy, warnings, copy. В программе это реализовано следующим образом:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from math import *
import scipy as sci
import warnings
import copy
```

Листинг 1.2.1. Подключение библиотек.

### **1.3. Правила и методы сбора эмпирических данных**

Генеральная совокупность в социологии является совокупностью всех людей с некоторой определенной характеристикой. Опросить всех представителей генеральной совокупности часто физически или по разумным причинам невозможно. Тогда применяются методы выборочного опроса, он заключается в отборе небольшого числа людей из общей совокупности по определенным правилам в качестве социальной модели, воспроизводящей структуру объектов исследования. Процесс отбора группы таких людей, а также сама группа людей называется выборкой [2].

В то же время результаты исследования зависят от правильного проведения выборки, поскольку выборка представляет собой упрощенную форму общей совокупности и не в полной мере отражает ее разнообразие.

Поэтому необходимо соблюдать правила составления выборочной совокупности:

- 1) В качественном исследовании выборка должна быть не однородной (гетерогенной) [2].
- 2) В количественном исследовании выборка должна быть однородной (гомогенной) [2].
- 3) Выборка должна быть репрезентативной [2].
- 4) Каждый элемент генеральной совокупности должен иметь одинаковую вероятность попадания в выборку [2].

В социологии под репрезентативностью понимается свойства выборки, которые позволяет ей выступать в качестве модели генеральной совокупности на момент переписи. Репрезентативными выборками считаются выборки, основные характеристики которых соответствуют аналогичным характеристикам генеральной совокупности [2].

Однородные группы людей должны совпадать по основным характеристикам, а разнородные группы должны отличаться по основным характеристикам.

Метод выборки является способом создания выборки. Каждый тип выборки имеет свои собственные математические инструменты и процедуры [2].

Методы выборки делятся на вероятностные (статистические) и целевые (не статистические) [2].

Вероятностные методы включают в себя:

- 1) Простой случайный отбор.
- 2) Стратифицированный отбор.
- 3) Кластерный (гнездовой) случайный отбор.
- 4) Систематический отбор [2].

Простой случайный отбор из генеральной совокупности заключается в том, что генеральная совокупность однородна, и что все ее элементы также могут быть использованы в исследованиях в равной степени, имеется список всей совокупности элементов, составляющих генеральную совокупность. Для получения полного списка используются процедуры случайного отбора (в частности, применение генератора случайных чисел) [2].

Стратифицированная выборка делит объём выборки между всеми стратами пропорционально их численности, и извлекает простые случайные выборки из каждой страты. Этот метод обеспечивает равномерно распределённое представительство всевозможных групп и (или) типов населения в выборочной совокупности [2].

Кластерная выборка (гнездовая) является типом выборки, при котором выбранными объектами образуется кластер (гнездо, группа) из меньших единиц. Группы выбираются случайным образом (в некоторых случаях с вероятностью, пропорциональной их количеству), и подвергаются изучению полностью или выборочно [2, 4].

Системный отбор заключается в выборе из списка представителей генеральной совокупности людей с помощью определённого количества номеров [2].

Целевые методы включают в себя:

- 1) Квотную выборку.
- 2) Метод «снежного кома».
- 3) Метод типичных представителей.
- 4) Метод стихийного отбора на основе принципа удобства.
- 5) Метод на основе суждений [2].

Квотная выборка является уменьшенной моделью объекта социологического исследования. Она устанавливается на основе статистических данных (квотных параметров) о социальном и демографическом характере элементов генеральной совокупности. Методология основан на намеренном установлении структуры выборочной совокупности. Например, в ходе исследования была предпринята попытка опросить некоторое количество людей определенных возрастов, пола,

уровней образования и профессий. Удельные квоты в выборочной совокупности обязаны быть приведены в соответствие к её удельному весу в генеральной совокупности. Обычно квотная выборка используется при последних стадиях проведения отбора [2].

Метод "снежного кома" применим тогда, когда существует предположение о том, что отбор дополнительных (последующих) респондентов производится после ссылки на них ранее отобранных. Этот метод применяется для изучения особенных, редких и неслучайных совокупностей [3].

Метод типичных представителей лучше всего применим на последних стадиях отбора, в случае, когда нужно использовать небольшую численность объектов. Только тогда, когда существует обоснование выбора объектов, данный метод может в значительной мере обеспечить репрезентативность выборки. Для этого нужно собрать дополнительную информацию о тех признаках, которые позволяют рассматривать их в качестве контрольных [3].

Метод стихийного отбора аналогичен случайному отбору, но в случае стихийного отбора необходимо опрашивать тех, кто внешне похож на представителей генеральной совокупности [3].

Смысл метода отбора на основе принципа удобства сводится к тому, чтобы создать экземпляр наиболее удобным способом с точки зрения исследователя, например, с точки зрения доступности респондентов и (или) с меньшими затратами времени и усилий.

Создание выборки на основе суждений основывается на учёте суждений квалифицированных экспертов о составе выборки. С использованием этого подхода часто создаются элементы фокус-групп [4].

На практике часто используется многоступенчатая выборка, при которой набор объектов, выбранных на предыдущем этапе, используется в качестве исходного объекта на следующем этапе. Объекты самого низкого этапа, полученные из непосредственно собранных данных, называются единицами наблюдения. Многоуровневая выборка используется, когда генеральная совокупность велика и разнообразна, а рандомизация, достигаемая другими методами, приводит к чрезмерной дисперсии выборки.

После сбора эмпирических данных каким-либо из выше представленных методов перед их использованием с помощью умного блокнота для помещения их в форму, которую сможет обработать

программа умного помощника, может применяться множество прикладных программ, например, Libre Office, Microsoft Office, в которых необходимо создать таблицу с собранными данными в формате .csv.

Необходимо поместить данные о каждом критерии об исследуемом объекте в отдельный столбец с соответствующим названием. Столбцы должны располагаться вплотную друг к другу в левой верхней части таблицы. Используется кодировка UTF-8.

#### 1.4. Методы обработки социологических данных

Характеристика, рассматриваемая в социологическом исследовании, принимается за  $X$ . Значения  $X$  (т. е. случайной величины), получаемые в процессе сбора эмпирических данных, обозначаются через  $x^{(1)}, x^{(2)}, \dots, x^{(n)}$  и называются вариантами – значениями признака. Количество объектов, имеющих данный признак, называется частотой варианты. Множество всех возможных вариант называется генеральной совокупностью. Любое конечное подмножество из генеральной совокупности называется выборкой [4].

$\{x^{(1)}, x^{(2)}, \dots, x^{(n)}\} = X$ , где  $n$  – объём выборки. Значения  $x^{(i)}$  располагают в порядке возрастания:

$$x_1, x_2, \dots, x_n \quad (x_1 < x_2 < \dots < x_n).$$

Некоторые варианты  $x_i$  могут встречаться в выборке несколько раз.

Предположим, было опрошено некоторое количество респондентов с помощью анкеты.

1. Пол:
    - 1) Мужской
    - 2) Женский
  2. Сколько вам полных лет?
  3. Как часто вы посещаете мероприятия культурной направленности в вашем регионе?
    - 1) Постоянно
    - 2) Часто
    - 3) Редко
    - 4) Никогда
- ...

Здесь представлены признаки, которые можно распределить на шкалах трёх типов: результаты первого вопроса можно изобразить на номинальной шкале, второй – на метрической, третий – на порядковой. Необходимо перенести данные из анкет в специальную табл. 1.4.1.

Таблица 1.4.1

Условный пример: данные социологического опроса 30 респондентов

Номер анкеты	Пол	Возраст	Частота просмотра новостей культурной направленности о регионе респондента
	1	1	3
1	1	1	3
2	2	4	1
...	...		...
30	1	6	4

Такие таблицы называют матрицами данных.

В программе входные данные представляются в виде объекта типа DataFrame библиотеки Pandas:

```
df_start = pd.read_csv(path_empiric_data)
```

Листинг 1.4.1. Загрузка данных из таблицы.

Тут df\_start – переменная типа pandas.DataFrame для хранения загруженной таблицы, path\_empiric\_data – переменная типа string для хранения пути к файлу на диске.

Вариационным рядом называют ряд, разделённый в порядке возрастания или убывания вариант с соответствующими им весами. Различают непрерывные и дискретные вариационные ряды.

В табл. 1.4.1 представлены дискретные вариационные ряды различных признаков.

Так же можно составить табл. 1.4.2, которая называется дискретным вариационным рядом выборки. В ней содержатся частоты, частности и проценты.

Таблица 1.4.2

### Дискретный вариационный ряд выборки

Варианты, $x_i$	$x_1$	$x_2$	...	$x_k$
Частоты, $n_i$	$n$	$n_2$	...	$n_k$
Относительные частоты, $w_i$	$w_1$	$w_2$	...	$w_k$

Для отображения, какое количество раз в данных выборки встречается варианта  $x_i$  используется эмпирическая частота выборки  $n_j$ . Вместе с этим

$$\sum_{i=1}^k n_i = n \quad (1.4.1)$$

Отношение

$$w_i = \frac{n_i}{n} \quad (1.4.2)$$

называется относительной частотой варианты  $x_i$ . Тут  $n_i$  – частота появления варианты  $x_i$ , а  $n$  – объём выборки [5].

Пусть имеется дискретный вариационный ряд выборки, тогда ломаная линия с вершинами  $(x_k, n_k)$  называется полигоном частот. А ломаная линия с вершинами  $(x_k, w_k)$  – полигоном относительных частот [6].

Чаще всего при  $n > 30$  данные помещают в интервальный вариационный ряд.

Для создания вариационного ряда вычисляется диапазон изменения  $R$  признака  $X$ , как разность между значениями признака - наибольшим  $x_{max}$  и наименьшим  $x_{min}$ :

$$R = x_{max} - x_{min} \quad (1.4.3)$$

$R$  разбивается на  $k$  равных интервалов. Как правило, пользуются одним из правил для выбора числа  $k$ , хотя теоретически можно взять любое  $k$ :

- 1)  $6 \leq k \leq 20$
- 2)  $k \approx \sqrt{n}$
- 3)  $k \approx 1 + \log_2 n \approx 1 + 3,221 \lg n = 1 + 1.44 \cdot \ln n$

$k$  приравнивается от 6 до 10, а каждому интервалу присваивается ширина  $h=R/k$  в том случае, когда объём выборки не большой. Как правило,  $h$  округляется до некого значения  $d$  [6].

Составляется таблица 1.4.3, называемая интервальным вариационным рядом выборки.

Таблица 1.4.3

### Интервальный вариационный ряд выборки

Интервалы	$[x_0; x_1)$	$[x_1; x_2)$	...	$[x_{k-1}; x_k)$
Частоты, $n_j$	$n_1$	$n_2$	...	$n_k$
Относительные частоты, $w_i$	$w_1$	$w_2$	...	$w_k$

Можно составить по полученным данным гистограмму выборки, или гистограмму относительных частот выборки.

$F_n^*(x)$  является эмпирической функцией распределения выборки  $\{x_k\}_1^n$ , и определяется равенством:

$$F_n^*(x) = \frac{n_x}{n}, \quad (1.4.5)$$

где  $n$  – объём выборки,  $n_x$  – число вариант выборки, меньших, чем  $x$ .

Теоретическая функция определяет вероятность события  $X < x$ . Эмпирическая функции различаются тем, что теоретическая функция определяет относительную частоту этого события.

Эмпирическая функция всегда принимает значения в диапазоне значений  $[0; 1]$  (рис. 1.4.1).

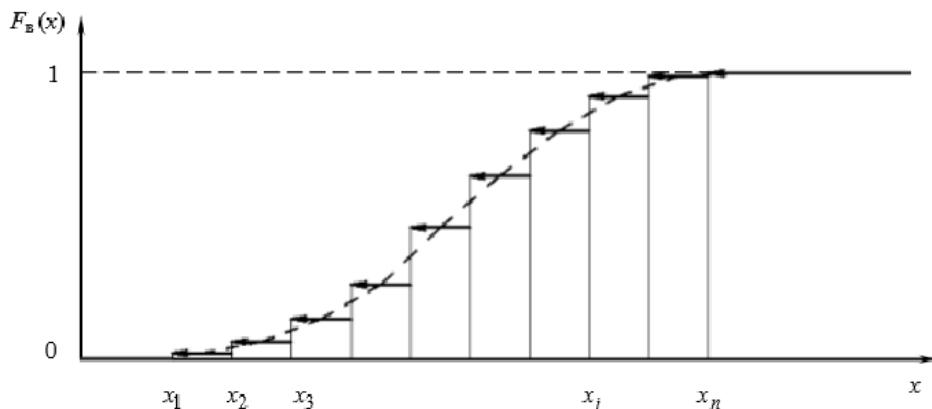


Рис. 1.2.1 Функции распределения - эмпирическая и теоретическая.

Выброс (экстремальное значение, аномальное значение) – редкое значение характеристики, значительно отличающееся от других значений.

Выбросы могут быть вызваны как ошибкой статистического сбора данных (например, неточность приборов измерения), так и значительно отличающимися редкими крайне значениями, которые реально встречаются на практике.

Для того, чтобы решить, какие данные необходимо считать выбросами, определяются границы выбросов. Они могут устанавливаться по разным правилам с учётом исследуемых данных и задач исследования.

Удаляются выбросы потому, что оказывают значительное воздействие на различные метрики и результаты анализа данных, которое не пропорционально относительной частоте их наблюдения.

Распространённой практикой считаются границы выбросов, равные не попадают в интервал

$$[Q1 - 1,5 * IQR; Q3 + 1,5 * IQR], \quad (1.4.6)$$

где  $Q1$  и  $Q3$  – первый и третий квартили,  $IQR$  – расстояние между первым и третьим квартилем.

Все значения, не попадающие в этот интервал, считаются выбросами.

В программе нахождение границ выбросов реализовано следующим образом:

```
# Первый и третий квартили:  
q1 = table[s].quantile(0.25)  
q3 = table[s].quantile(0.75)  
# Межквартальный размах:  
iqr = q3 - q1  
# Границы выбросов:  
lower_bound = q1 - 1.5 * iqr  
upper_bound = q3 + 1.5 * iqr
```

Листинг 1.4.2. Вычисление границ выбросов.

Тут  $table$  – переменная типа `pandas.DataFrame` для хранения таблицы,  $s$  – переменная типа `str` для хранения строки с названием признака – столбца таблицы,  $q1$ ,  $q3$  – переменные типа `numpy.float64` для хранения значений первого и третьего квартилей,  $iqr$  – переменная типа `numpy.float64` для хранения значения межквартального интервала,  $lowerbound$  и  $upper_bound$  – переменные типа `numpy.float64` для хранения значений границ выбросов.

Существует три метода обработки выбросов: удаление, трансформация, вменение в вину.

Самым простым методом можно считать удаление. Этот метод обычно применяется тогда, когда значения уверенно можно считать ошибочными, или же можно пренебречь крайне редкими и значительно отличающимися значениями. В этом случае удаляются данные, содержащие выбросы для того, чтобы избежать их какого бы то ни было воздействия на дальнейшие вычисления [9].

В случае, когда необходимо учесть значения выбросов, и вместе с этим снизить их воздействие на всю выборку, может применяться трансформация. Этот метод может подразумевать различные преобразования, в частности, нахождения значений натурального или десятичного логарифмов, квадратных корней, обратных величин [9].

В случае, когда необходимо сохранить значения, содержащие выбросы, учесть тенденции, на которые могут указывать выбросы, и вместе с этим снизить из воздействие на всю выборку в ещё большей степени, чем в случае трансформации, применяется вменение в вину [9].

Для дальнейшего анализа данных необходимо сделать сопоставимыми эмпирические значения признаков. В этих целях данные можно подвергнуть нормализации или стандартизации.

Нормализация – способ преобразования эмпирических данных путём изменения масштаба значений, в ходе которого значения  $x_i$  признака  $X$  располагаются в интервале  $[0; 1]$ .

Нормализованные данные рассчитываются по формуле:

$$x'_i = \frac{(x_i - \bar{x})}{x_{max} - x_{min}}, \quad (1.4.7)$$

где  $x'_i$  – нормализованное значение,  $x_{max}$  и  $x_{min}$  – максимальные значения признака  $X$ .

В программе нормализация реализована следующим образом:

```
table[s] = (table[s] - x_min) / (x_max - x_min)
```

Листинг 1.4.3. Нормализация данных.

Тут  $table$  – переменная типа `pandas.DataFrame` для хранения таблицы со значениями,  $s$  – переменная типа `string` для хранения имени столбца таблицы,  $x\_min$  и  $x\_max$  – переменные типа `numpy.int64` или `numpy.float64` для хранения максимального и минимального значений в столбце таблицы.

Стандартизация – способ преобразования эмпирических данных путём изменения масштаба значений, в ходе которого значения  $x_i$  признака  $X$  располагаются таким образом, что их среднее арифметическое приравнивается 0, а среднее квадратическое отклонение приравнивается единице.

Стандартизованные данные рассчитываются по формуле:

$$X'_i = \frac{(x_i - \bar{x})}{s}, \quad (1.4.8)$$

где  $x'_i$  нормализованное значение,  $S$ -среднее квадратическое отклонение признака  $X$ .

В программе стандартизация реализована следующим образом:

```
table[s] = (table[s] - mean) / std
```

Листинг 1.4.4. Стандартизация данных.

Тут `table` – переменная типа `pandas.DataFrame` для хранения таблицы со значениями, `s` переменная типа `string` для хранения имени столбца таблицы, `mean` и `std` – переменные типа `numpy.float64` для хранения среднего арифметического и среднего квадратического отклонения.

## 1.5. Расчёт описательных статистик распределения и их реализация в умном помощнике

Для характеристики распределения значений случайной величины применяют описательные статистики, которые используют различные вычисляемые показатели. Это мода, медиана, математическое ожидание, дисперсия и среднее отклонение, максимум, минимум и размах, асимметрия  $A_s$  и эксцесс  $E_x$ .

Выборочными показателями называют различные средние показатели, используемые в качестве характеристики свойств статистического распределения.

Пусть будет выборка объёма  $n$  со значениями  $x_1, x_2, \dots, x_n$  признака  $X$ .

Мода выборки – это то значение варианты выборки, которое имеет наибольшую частоту.

В программе мода вычисляется следующим образом:

```
mode = sci.stats.mode(np_data) [0]
```

Листинг 1.5.1. Вычисление моды.

Тут `mode` – переменная типа `numpy.float64` для хранения значения моды, `np_data` – переменная типа `numpy.ndarray` для хранения массива `numpy` со значениями некоторой характеристики.

Медиана выборки – серединное значение вариационного ряда значений случайной величины.

Медиана обозначается как  $Me$ .

Если  $n=2k+1$ , то:

$$Me = x_{k+1} \quad (1.5.1)$$

Если  $n=2k$ , то:

$$Me = \frac{x_k + x_{k+1}}{2} \quad (1.5.2)$$

В программе медиана вычисляется следующим образом:

```
median = np.median(np_data)
```

Листинг 1.5.2. Вычисление медианы.

Тут  $median$  – переменная типа `numpy.float64` для хранения значения медианы, `np_data` – переменная типа `numpy.ndarray` для хранения массива `numpy` со значениями некоторой характеристики.

В случае, если признак  $X$  имеет значения  $x_i$ , которые не сгруппированы в вариационные ряды (табл. 1.4.2, 1.4.3), а объём выборки  $n$  не большой, то математическое ожидание  $a_n^*$  находится по формуле:

$$a_n^* = \frac{1}{n} \sum_{i=1}^n x_i \quad (1.5.3)$$

В программе математическое ожидание вычисляется следующим образом:

```
mean = round(np_data.mean(), 3)
```

Листинг 1.5.3. Вычисление математического ожидания.

Тут  $mean$  – переменная типа `numpy.float64` для хранения значения математического ожидания, `np_data` – переменная типа `numpy.ndarray` для хранения массива `numpy` со значениями некоторой характеристики.

Дисперсия  $(\delta_n^*)^2$  находится по формуле:

$$(\delta_n^*)^2 = \frac{1}{n} \sum_{i=1}^n (x_i - a_n^*)^2 \quad (1.5.4)$$

Однако, если был образован дискретный вариационный ряд (табл. 1.4.2), то формулы имеют следующий вид:

$$a_n^* = \frac{1}{n} \sum_{k=1}^n x_k, \quad n = \sum_{k=1}^k n_k \quad (1.5.5)$$

$$(\delta_n^*)^2 = \frac{1}{n} \sum_{k=1}^n (x_k - a_n^*)^2 n_k \quad (1.5.6)$$

В программе дисперсия вычисляется следующим образом:

```
variance = round(np.var(np_data, ddof = 1), 3)
```

#### Листинг 1.5.4. Вычисление дисперсии.

Тут variance – переменная типа numpy.float64 для хранения значения дисперсии, np\_data – переменная типа numpy.ndarray для хранения массива numpy со значениями некоторой характеристики.

Очевидно, что формулы (1.5.1) и (1.5.3) дают такие же результаты, как (1.5.2) и (1.5.4) для  $a_n^*$  и  $(\delta_n^*)^2$  соответственно.

Стандартная ошибка – характеристика точности величины, для которой она вычисляется. Ошибка для математического ожидания считается по формуле:

$$m_{a_n^*} = \frac{a_n^*}{\sqrt{n}} \quad (1.5.7)$$

Формулу (1.5.4) используют, когда объём выборки  $n \leq 50$ , в противном случае используют исправленную дисперсию для простой выборки:

$$(S_n^*)^2 \frac{1}{n-1} \sum_{i=1}^n (x_i - a_n^*)^2 \quad (1.5.8)$$

или для взвешенной выборки:

$$(\hat{S}_n^*)^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - a_n^*)^2 n_i \quad (1.5.9)$$

Чем слабее разброс значений признака относительно своего математического ожидания, тем слабее варьируются результаты респондентов в данной группе.

Среднее квадратичное отклонение показывает, насколько отдельные члены ряда отклоняются от среднего значения при различных объемах выборки:

$$S = \sqrt{(S_n^*)^2} \text{ или } \hat{S} = \sqrt{(\hat{S}_n^*)^2} \quad (1.5.10)$$

В программе среднее квадратическое вычисляется следующим образом:

```
std = round(np.std(np_data), 3)
```

#### Листинг 1.5.5. Вычисление дисперсии.

Тут std – переменная типа numpy.float64 для хранения значения среднего квадратического, np\_data – переменная типа numpy.ndarray для хранения массива numpy со значениями некоторой характеристики.

Ошибка среднего квадратичного отклонения вычисляется по формуле:

$$m_s = \frac{s}{\sqrt{2n}} \quad (1.5.11)$$

Коэффициент вариации – это отношение среднего квадратичного отклонения к средней арифметической, которое выражено в процентах:

$$c_v = \frac{s}{\bar{a}_n^*} \cdot 100\% \quad (1.5.12)$$

Ошибки коэффициента вариации:

$$m_v = \frac{c_v}{\sqrt{2n}} \quad (1.5.13)$$

Для характеристики форм распределения используется асимметрия. Она показывает, в какую сторону относительно среднего сдвинуто большинство значений случайной величины. Симметричность эмпирического распределения относительно среднего значения характеризуется нулевым значением. На сдвиг распределения в сторону больших значений указывает  $A_s < 0$ , в сторону меньших  $A_s > 0$ . В большинстве случаев за нормальное распределение принимается распределение с асимметрией  $A_s \in [-1; +1]$ .

Формула асимметрии: [7]

$$\frac{1}{n} \sum_{i=1}^n \left( \frac{x_i - \bar{x}}{s} \right)^3 \quad (1.5.14)$$

В программе асимметрия вычисляется следующим образом:

```
A = round(np.mean(((np_data - mean) ** 3) / variance ** (3 / 2)), 3)
```

#### Листинг 1.5.6. Вычисление асимметрии.

Тут A – переменная типа `numpy.float64` для хранения значения асимметрии, mean – переменная типа `numpy.float64` для хранения значения среднего арифметического, variance – переменная типа `numpy.float64` для хранения значения дисперсии, np\_data – переменная типа `numpy.ndarray` для хранения массива `numpy` со значениями некоторой характеристики.

Для характеристики формы распределения применяют такую характеристику как эксцесс. Если эксцесса близко к нулю, то форма распределения близка к теоретическому виду. Если  $E_s > 0$ , следовательно, распределение имеет плоскую вершину. Если  $E_s \gg 5.0$ , то распределение имеет острую вершину и его график вытянут по вертикальной оси. Если  $E_s \in [-1; +1]$ , то распределение соответствует нормальному виду. Формулы, по которым происходит вычисление данных характеристик формы распределения описаны в п. 1.4 (1.5.12 и 1.5.13).

Формула эксцесса [7]:

$$\frac{1}{n} \sum_{i=1}^n \left( \frac{x_i - \bar{x}}{s} \right)^4, \quad (1.5.15)$$

В программе эксцесс вычисляется следующим образом:

```
E = round(np.mean(((np_data - mean) ** 4) / (variance ** 2)),  
3)
```

Листинг 1.5.6. Вычисление эксцесса.

Тут E – переменная типа `numpy.float64` для хранения значения эксцесса, mean – переменная типа `numpy.float64` для хранения значения среднего арифметического, variance – переменная типа `numpy.float64` для хранения значения дисперсии, np\_data – переменная типа `numpy.ndarray` для хранения массива `numpy` со значениями некоторой характеристики.

## 1.5. Способы визуализации данных

Для визуализации собранных статистических данных может использоваться множество способов. Наиболее распространёнными способами отображения данных являются гистограммы, коробчатые диаграммы (диаграммы размаха), точечные диаграммы.

Гистограмма является способом отображения данных, в ходе которого отображается координатная плоскость с прямоугольниками, характеризующими количество нахождений значений  $x_i$ .

Обычно горизонтальная ось (ось абсцисс) отображает значения признака  $X$ , а вертикальная ось (ось ординат) отображает количество нахождений значения  $x_i$ , представленного на оси абсцисс. Для отображения на гистограмме значения признака  $X$  делятся на равные интервалы. На координатной плоскости изображаются прямоугольники, чья длина характеризует количество значений  $x_i$  признака  $X$  в указанном интервале.

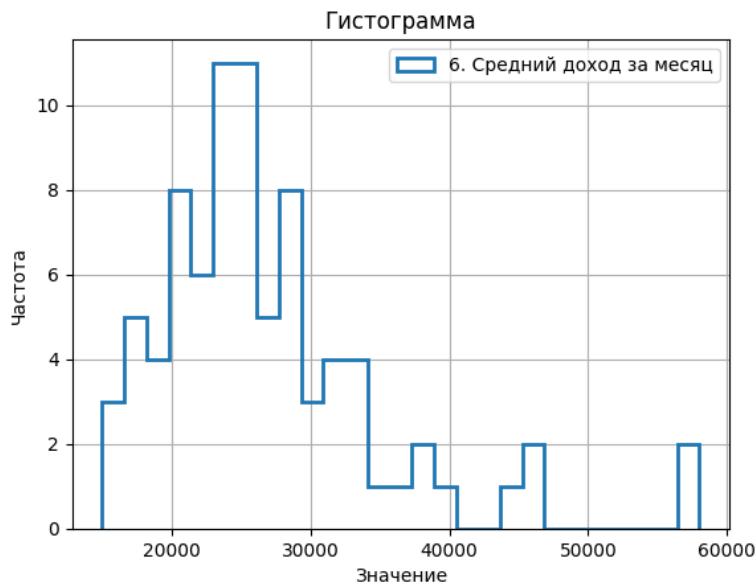


Рис. 1.6.1. Пример гистограммы данных.

Диаграмма размаха является способом отображения данных, который позволяет изобразить основные данные о распределении признака  $X$  с помощью построения на координатной плоскости прямоугольника с двумя отрезками, исходящими из противоположных сторон прямоугольника. На оси, вдоль которой расположен прямоугольник и отрезки, изображены значения  $x_i$  признака  $X$ .

Стороны прямоугольника, из которых исходят отрезки, расположены на уровне значений, указывающих на границы между первым и вторым квартилями и третьим и четвёртым квартилями признака  $X$ . Чёрта внутри прямоугольника указывает на расположение медианы признака  $X$ . Конец линий, выходящих из прямоугольника влево и вправо, указывают на наблюдаемый максимум и минимум значений признака  $X$ , без учёта выбросов. Возможные кружки за пределами отрезков указывают на выбросы.

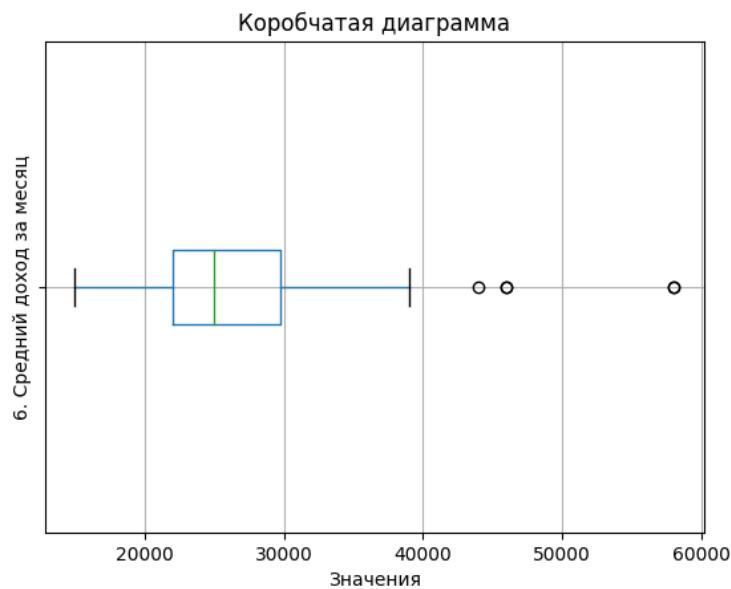


Рис. 1.6.2. Пример коробчатой диаграммы данных.

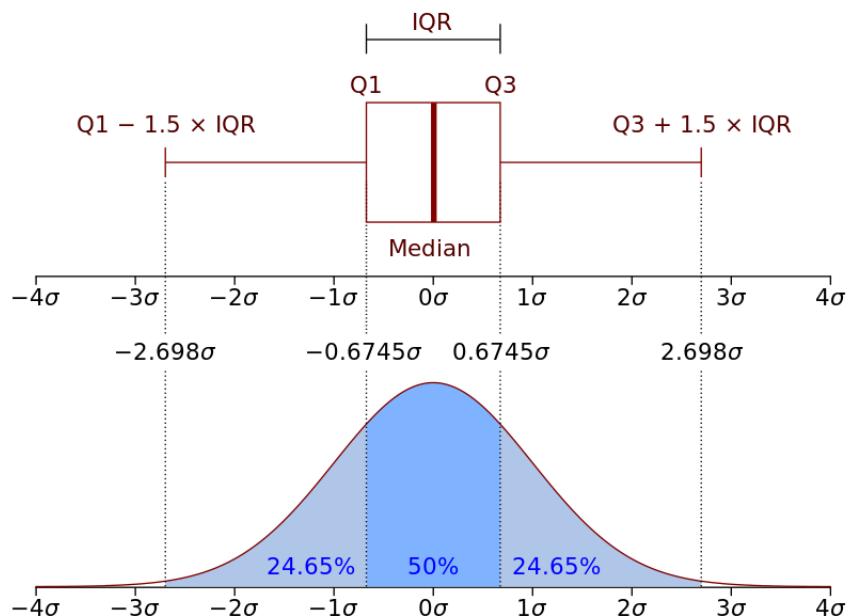


Рис. 1.6.3. Сравнение коробчатой диаграммы и нормального распределения.

Точечная диаграмма является способом отображения данных, в ходе которого можно наглядно наблюдать зависимость значений одного признака от значений другого признака.

На координатной плоскости на горизонтальной оси (оси абсцисс) и вертикальной оси (оси ординат) отображаются значения двух некоторых признаков  $X$  и  $Y$ , а точки на плоскости указывают на наблюдения,

имеющие значения  $x_i$  и  $y_i$ , соответствующие шкалам на координатных осях.

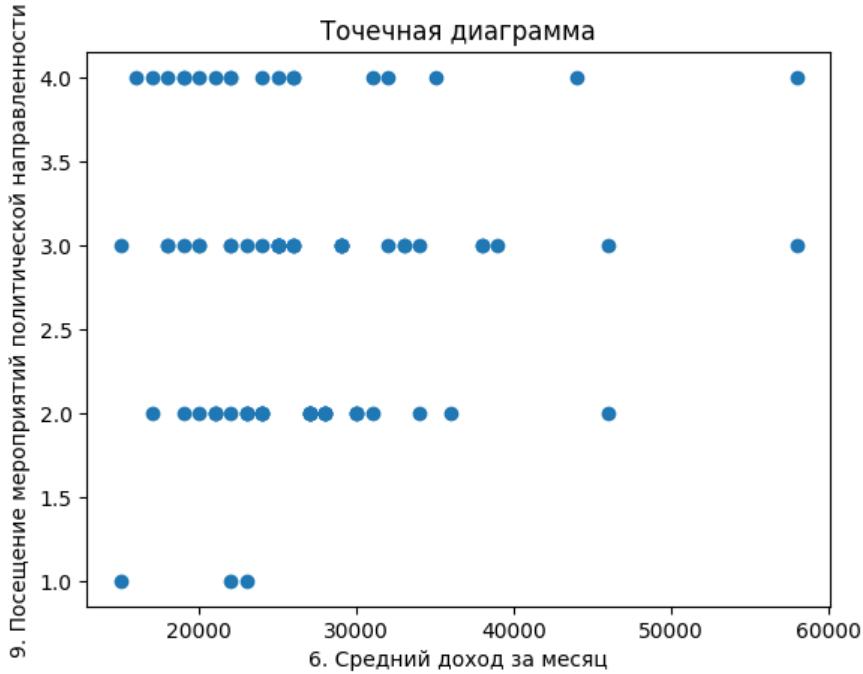


Рис. 1.6.3. Пример точечной диаграммы.

## 1.6. Проверка статистических гипотез

Нулевая гипотеза - гипотеза об отсутствии различий, её обозначают через  $H_0$ . Нулевая гипотеза – это предположение, которое мы хотим опровергнуть. Нулевую гипотезу можно опровергнуть только с определённой вероятностью, зависящей от расчётов [10].

Альтернативная гипотеза – это гипотеза о значимости различий, её обозначают через  $H_1$ . Альтернативная гипотеза или экспериментальная гипотеза – это предположение, которое мы хотим доказать [10].

Зачастую производится проверка близости эмпирического распределения и нормального распределения, как наиболее часто встречающегося распределения в природе.

Функция нормального распределения имеет вид:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}}, \quad (1.7.1)$$

где  $a$  – математическое ожидание,  $\sigma$  – среднее квадратическое отклонение.

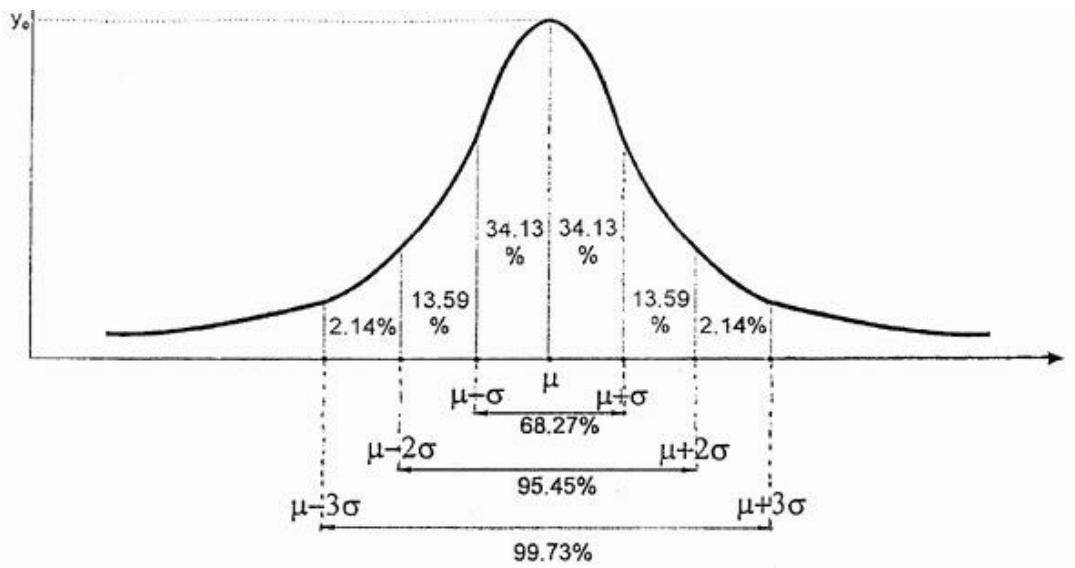


Рис. 1.7.1. График нормального распределения.

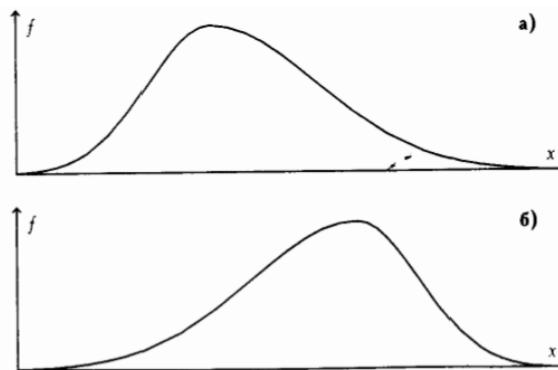


Рис. 1.7.2 Асимметрия распределений: а) положительная, б) отрицательная.

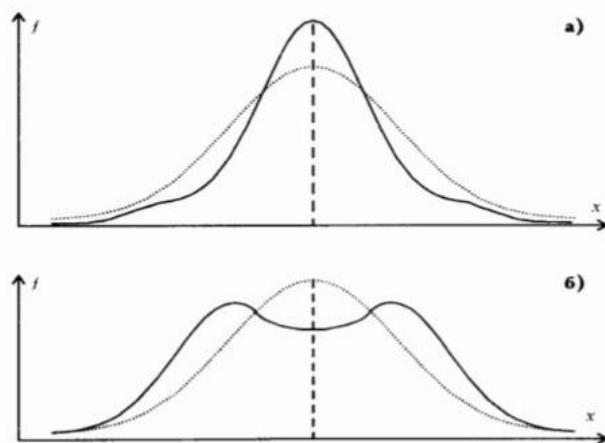


Рис. 1.7.3. Преобладание крайних значений в распределении. Эксцесс: а) положительный; б) отрицательный.

Для проверки гипотез могут применяться множество критериев, например, критерий Эпсса-Палли, критерий  $\chi^2$  Пирсона и др.

Преимущество критерия Эпсса-Палли заключается в том, что он имеет статус одного из наиболее мощных критериев для проверки распределения на нормальность [11].

Критерий Эпсса-Палли применим при объёме выборки  $n \geq 8$ . Выборки при  $n < 8$  при обнаружении отклонений от нормального распределения не дают достоверных результатов [8].

В качестве результата применения критерия Эпсса-Палли к эмпирическим данным возможны следующие гипотезы:

- 1)  $H_0$ : полученное эмпирическое распределение признака не отличается от теоретического распределения.
- 2)  $H_1$ : полученное эмпирическое распределение признака отличается от теоретического распределения.

Статистику критерия  $T_{EP}$  Эпсса-Палли вычисляют по формуле

$$T_{EP} = 1 + \frac{n}{\sqrt{3}} + \frac{2}{n} \sum_{k=2}^n \sum_{j=1}^{k-1} \exp \left\{ \frac{-(x_j - x_k)^2}{2m_2} \right\} - \sqrt{2} \sum_{j=1}^n \left\{ \frac{-(x_j - \bar{x})^2}{4m_2} \right\}, \quad (1.7.2)$$

где  $m_2$  – выборочный центральный момент второго порядка,  $n$  объём выборки,  $\bar{x}$  – среднее арифметическое [8].

Выборочный центральный момент второго порядка  $m_2$  рассчитывается по формуле:

$$m_2 = \frac{1}{n} \sum_{j=1}^n (x_j - \bar{x})^2 \quad (1.7.3)$$

В программе критерий  $T_{EP}$  Эпсса-Палли вычисляется следующим образом:

```
m_2 = np.var(np_data, ddof = 0)
A = sqrt(2) * np.sum([exp(-(np_data[i] - mean)**2 / (4*m_2)) for i in range(n)])
B = 2/n * np.sum([
    np.sum([
        exp(-(np_data[j] - np_data[k])**2 / (2*m_2)) for j in range(0, k)
    ]) for k in range(1, n)])
T_EP_empiric = round(1 + n / sqrt(3) + B - A, 3)
```

Листинг 1.7.1. Вычисление значения критерия Эпсса-Палли.

Тут `pr_data` – переменная типа `numpy.ndarray`, хранящая массив значений признака, `n` – переменная типа `int`, хранящая длину выборки, `mean` – переменная типа `numpy.float64`, хранящая среднее арифметическое.

Нулевую гипотезу отклоняют, если вычисленное значение статистики  $T_{EP}$  превышает р-квантиль при данных уровне значимости  $\alpha$  и объёме выборки  $n$ .  $p$ -Квантили статистики критерия  $T_{EP}$  при  $p = 1 - \alpha = 0,90; 0,95; 0,975$  и  $0,99$  приведены в приложении А [8].

Нередко в статистике необходимо установить корреляцию двух переменных в эмпирическом распределении.

Корреляция – это взаимосвязь между значениями двух и более переменных.

Для проверки гипотез может применяться множество методов, например, использование коэффициента корреляции Пирсона.

Коэффициент корреляции Пирсона может принимать значения от 1 до -1:

- 1) Если коэффициент корреляции меньше 0, то корреляция является отрицательной, т. е. при увеличении одной переменной другая уменьшается [13].
- 2) Если коэффициент корреляции больше 0, то корреляция является положительной, т. е. при увеличении одной переменной увеличивается и другая [13].
- 3) Чем ближе коэффициент корреляции к 0, тем меньше выражена зависимость одной переменной от другой [13].

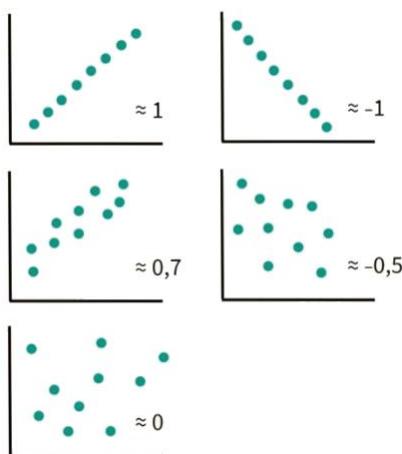


Рис. 1.7.4. Сравнение значений коэффициентов корреляции с соответствующими точечными диаграммами.

Коэффициент корреляции Пирсона вычисляется по формуле:

$$r_{x,y} = \frac{cov(x,y)}{\sqrt{s_x^2 s_y^2}}, \quad (1.7.3)$$

где  $cov(x, y)$  –ковариация,  $s_x^2$  и  $s_y^2$  -квадраты средних квадратических отклонений.

Ковариация находится по формуле:

$$cov(x, y) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}), \quad (1.7.4)$$

$\bar{x}$  и  $\bar{y}$  - средние арифметические признаков  $X$  и  $Y$ ,  $x_i$  и  $y_i$  – значения признаков,  $n$  – объём выборки.

В программе коэффициент корреляции Пирсона вычисляется следующим образом:

```
r      = round(float(sci.stats.pearsonr(data_list_1,
data_list_2)[0]), 3)
```

Листинг 1.7.2. Вычисление коэффициента корреляции Пирсона.

Тут переменные `data_list_1` и `data_list_2` типа `list` содержат в себе значения признаков  $X$  и  $Y$ , переменная `r` типа `float` хранит значение коэффициента корреляции Пирсона.

Для проверки значимости можно использовать  $t$ -распределение Стьюдента. Значение  $t$ -распределения вычисляется по формуле:

$$t = \frac{r_{xy}\sqrt{n-2}}{1-r_{xy}^2}, \quad (1.7.5)$$

Где  $r_{xy}$  – значение коэффициента корреляции Пирсона,  $n$  – длина выборки.

Так как распределение Стьюдента симметрично, и рассматривается возможность отклонения вычисленного значения  $r_{x,y}$  от истинного как в большую, так и в меньшую стороны, то двустороннее  $t$ -распределение будет зависеть от  $\alpha/2$  [12]:

$$t \in [t_{\alpha/2}, t_{1-\alpha/2}] \quad (1.7.6)$$

В программе значение  $t$ -распределения вычисляется следующим образом:

```
t = r * np.sqrt((df) / (1 - r**2))
```

Листинг 1.7.3. Вычисление значения  $t$ -распределения.

Тут `df` – переменная типа `int`, хранящая количество степеней свободы, `r` – переменная типа `float` для хранения значения корреляции

Пирсона,  $t$  – переменная типа `numpy.float64` для хранения значения статистики  $t$ -распределения.

Полученное значение  $t$ -распределения сравнивается со значением в таблице значений  $t$ -распределения Стьюдента. Таблица приведена в приложении Б.

В программе нахождение табличного значения  $t$ -распределения реализовано следующим образом:

```
t_critical = sci.stats.t.ppf(1 - alpha/2, df)
```

Листинг 1.7.4. Вычисление табличного значения  $t$ -распределения.

Тут переменная `t_critical` типа `numpy.float64` хранит табличное значение  $t$ -критерия, переменная `alpha` типа `float` содержит значение  $\alpha$ , переменная `df` типа `int` содержит количество степеней свободы, равное  $n - 2$ .

Если табличное значение меньше модуля вычисленного значения  $t$ -статистики, то коэффициент корреляции считается значимым.

Для классификации значений коэффициента корреляции часто используется шкала Чеддока [15].

Таблица 1.6.1

Шкала Чеддока

Значение коэффициента корреляции	Качественная характеристика силы связи
0.1-0.3	Слабая
0.3-0.5	Умеренная
0.5-0.7	Заметная
0.7-0.9	Высокая
0.9-0.99	Весьма высокая

Следует помнить, что это деление весьма условно, и может давать результаты, не корректные по отношению к генеральной совокупности. Его границы могут пересматриваться в зависимости от объекта и задач исследования [15].

## РАЗДЕЛ 2. СОЗДАНИЕ УМНОГО ПОМОЩНИКА ДЛЯ АНАЛИЗА ДАННЫХ С ПОМОЩЬЮ СРЕДСТВ ЯЗЫКА PYTHON

### 2.1. Сбор данных

Сбор информации производился добровольным и анонимным анкетированием среди трудоустроенных представителей молодёжи Крыма методом простого случайного отбора в интернете с помощью сервиса <https://www.google.com/intl/ru/forms/about>. Ссылка на анкету, используемую при опросе: <https://forms.gle/gksFRTBbWiy9xseB6>.

Для создания анкеты применялись вопросы закрытого типа. После этого собранные эмпирические данные были закодированы с помощью порядковой кодировки. В ходе использования анкеты (приложение В) были представлены признаки, порождающие порядковые, метрические и номинальные шкалы [1].

Объектом исследования являлись представители трудоустроенной молодёжи Крыма, то есть граждане возрастом от 18 до 35 лет [14].

Предметом исследования являлось участие представителей трудоустроенной молодёжи Крыма в общественной жизни региона.

Собранные эмпирические данные представлены в приложении Г.

Были собраны эмпирические данные по следующим признакам:

- 1)  $X_1$  - «1. Пол».
- 2)  $X_2$  - «2. Возраст».
- 3)  $X_3$  - «3. Образование».
- 4)  $X_4$  - «4. Наличие брака».
- 5)  $X_5$  - «5. Средний доход в месяц».
- 6)  $X_6$  - «6. Членство в молодёжной организации».
- 7)  $X_7$  - «7. Посещение мероприятий культурной направленности».
- 8)  $X_8$  - «8. Посещение мероприятий политической направленности».
- 9)  $X_9$  - «9. Посещение мероприятий развлекательной направленности».

В ходе анкетирования была получена выборка объёмом  $n = 160$ .

Для значений признаков  $X_1$ ,  $X_4$ ,  $X_6$  использовались номинальные измерительные шкалы.

Для значений признаков  $X_2$ ,  $X_3$ ,  $X_7$ ,  $X_8$ ,  $X_9$  использовались ранговые измерительные шкалы.

Для значений признака  $X_5$  использовалась метрическая измерительная шкала.

Данные были собраны в файле с расширением .csv в виде таблицы с кодировкой UTF-8 с запятой в качестве разделителя значений.

1	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9
							5. Посещение мероприятий	8. Посещение мероприятий	9. Посещение мероприятий
2	1. Пол	2. Возраст	3. Образование	4. Наличие доход за брака	5. Средний доход за месяц	6. Членство в молодёжной организациии	7. Посещение мероприятий направленности	8. Посещение мероприятий направленности	9. Посещение мероприятий направленности
3	Мужской	21-23	Среднее общее	Нет	Нет	Нет	Редко	Редко	Редко
4	Мужской	18-20	Среднее общее	Нет	Нет	Да	Часто	Редко	Часто
5	Мужской	27-29	Высшее	Нет	Нет	Нет	Редко	Постоянно	Редко
6	Мужской	33-35	Высшее	Да	32000	Нет	Часто	Постоянно	Редко
7	Мужской	27-29	Высшее	Нет	33000	Нет	Постоянно	Часто	Постоянно
8	Мужской	24-26	Высшее	Нет	17000	Нет	Часто	Постоянно	Постоянно
9	Мужской	24-26	Высшее	Нет	35000	Нет	Постоянно	Постоянно	Редко
10	Мужской	21-23	Высшее	Нет	Нет	Нет	Редко	Часто	Редко
11	Мужской	33-35	Среднее профессиональное	Да	58000	Да	Редко	Часто	Редко
12	Мужской	33-35	Высшее	Нет	46000	Нет	Часто	Редко	Часто
13	Мужской	30-32	Среднее профессиональное	Да	39000	Да	Часто	Часто	Часто
14	Мужской	24-26	Среднее профессиональное	Нет	34000	Да	Часто	Часто	Часто
15	Мужской	30-32	Высшее	Нет	Нет	Да	Часто	Постоянно	Часто
16	Женский	21-23	Среднее общее	Нет	Нет	Нет	Редко	Часто	Редко

Рис. 2.1.1. Фрагмент таблицы с собранными данными.

## 2.2. Обработка данных умным помощником

Собранные эмпирические данные были загружены в программу (приложение Д), и далее обрабатывались и анализировались в ней.

Значения признаков, для которых использовались номинальные и ранговые шкалы, в целях дальнейшей обработки и анализа были зашифрованы. Для признаков с различными данными и типами шкал использовались разные шифры. Для значений, использующих метрические шкалы – значений признака  $X_5$  - шифр не использовался.

Данные признака  $X_1$  были зашифрованы следующим образом:

Таблица 2.2.1

### Шифр для признака $X_1$

Не зашифрованное значение	Зашифрованное значение
Мужской	1
Женский	2

Данные признака  $X_2$  были зашифрованы следующим образом:

Таблица 2.2.2

### Шифр для признака $X_2$

Не зашифрованное значение	Зашифрованное значение
18-20	19
21-23	22
24-26	25
27-29	28
30-32	31
33-35	34

Данные признака  $X_3$  были зашифрованы следующим образом:

Таблица 2.2.3

Шифр для признака  $X_3$

Не зашифрованное значение	Зашифрованное значение
Общее	1
Среднее профессиональное	2
Высшее	3

Данные признаков  $X_4$  и  $X_6$  были зашифрованы следующим образом:

Таблица 2.2.4

Шифр для признаков  $X_4$  и  $X_6$

Не зашифрованное значение	Зашифрованное значение
Да	1
Нет	0

Данные признаков  $X_7$ ,  $X_8$  и  $X_9$  были зашифрованы следующим образом:

Таблица 2.2.5

Шифр для признаков  $X_7$ ,  $X_8$  и  $X_9$

Не зашифрованное значение	Зашифрованное значение
Постоянно	4
Часто	3
Редко	2
Никогда	1

Пример процесса создания шифров:

```

14 dict_1 = {
15     'Мужской': 1,
16     'Женский': 2
17 }
18
19 dict_4_6 = {
20     'Да': 1,
21     'Нет': 0
22 }
23

```

Рис. 2.2.1. Создание шифров.

Тут переменные `dict_1` и `dict_2` типа `dict` содержат словари с парами «значение – шифр для значения».

Пример процесса применения шифров:

```

21 df_data["1. Пол"] = \
22 df_data["1. Пол"].replace(dict_1).infer_objects(copy=False)
23
24 question_4_6 = [
25     '4. Наличие брака',
26     '6. Членство в молодёжной организации'
27 ]
28 for i in question_4_6:
29     df_data[i] = df_data[i].replace(dict_4_6).infer_objects(copy=False)
30

```

Рис. 2.2.2. Применение шифров.

Тут переменная `df_data` типа `pandas.DataFrame` хранит таблицу со значениями, `dict_1` и `dict_2` – словари, содержащие пары «значение – шифр для значения», `question_4_6` – переменная типа `list` для хранения списка названий столбцов таблицы, для которых применяется шифр.

После применения шифров можно изобразить полученные данные на гистограммах и диаграммах размаха:

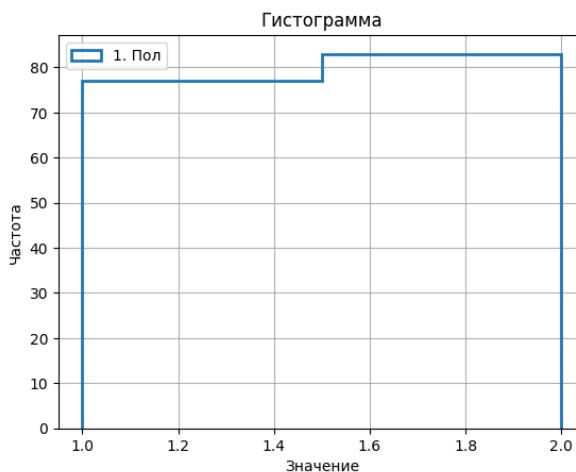


Рис. 2.2.3. Гистограмма значений признака  $X_1$ .

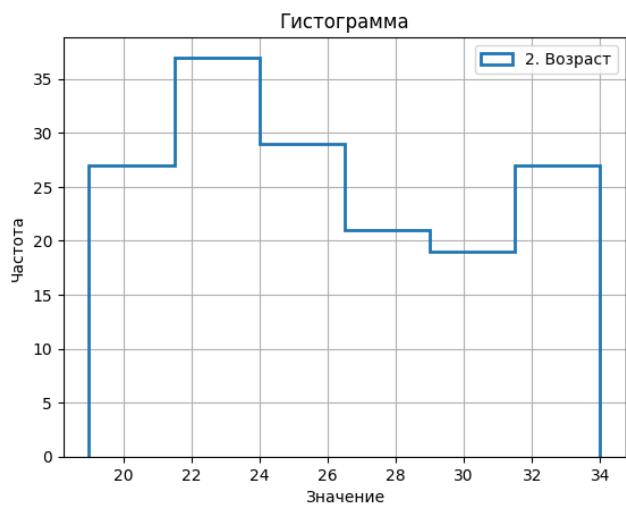


Рис. 2.2.4. Гистограмма значений признака  $X_2$ .

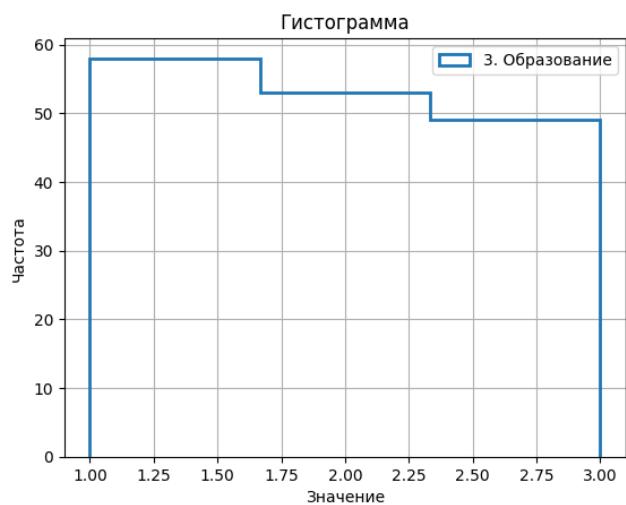


Рис. 2.2.5. Гистограмма значений признака  $X_3$ .

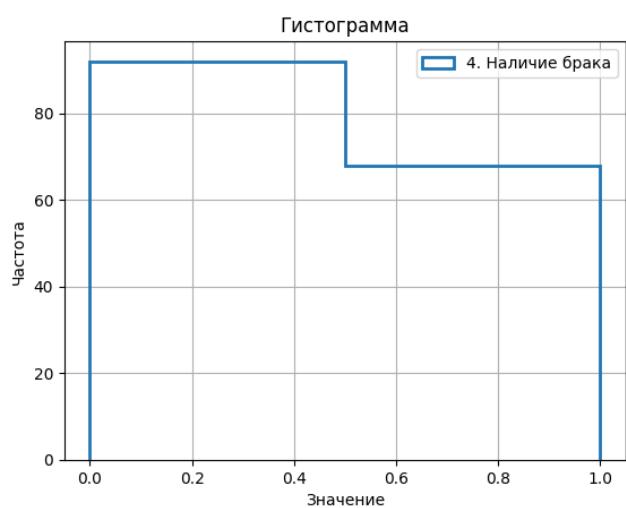


Рис. 2.2.6. Гистограмма значений признака  $X_4$ .

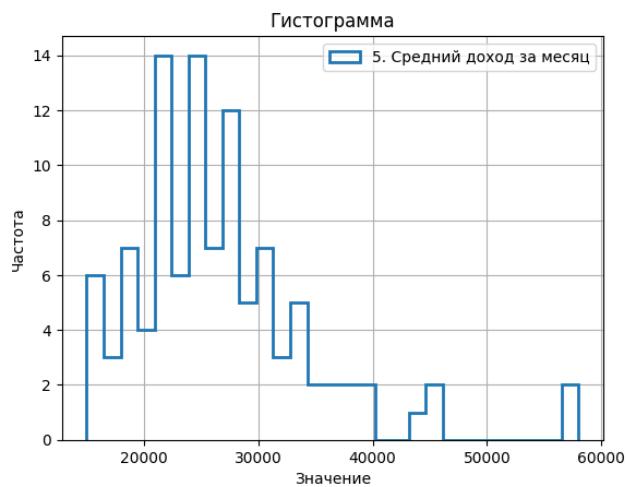


Рис. 2.2.7. Гистограмма значений признака  $X_5$ .

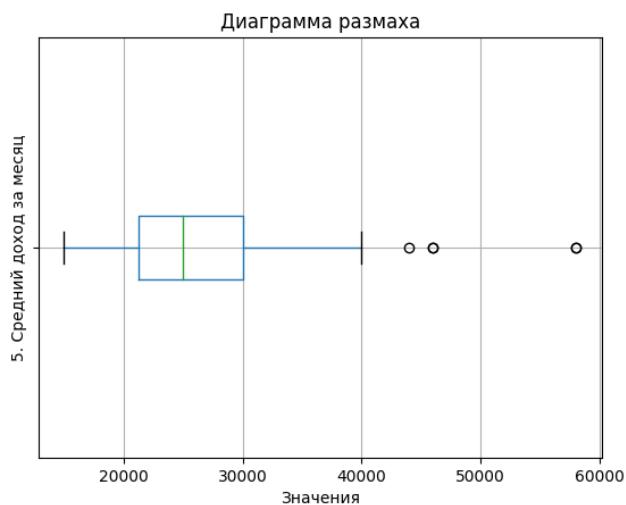


Рис. 2.2.8. Диаграмма размаха значений признака  $X_5$ .

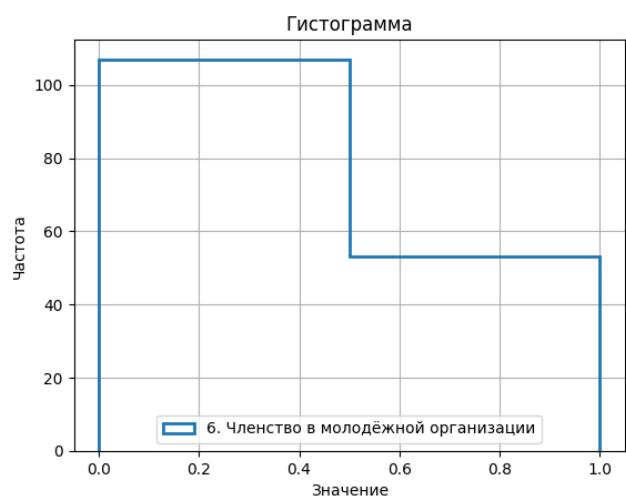


Рис. 2.2.9. Гистограмма значений признака  $X_6$ .

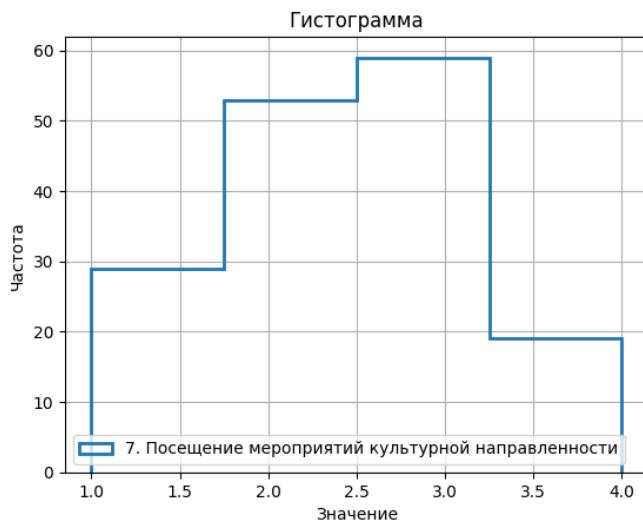


Рис. 2.2.10. Гистограмма значений признака  $X_7$ .

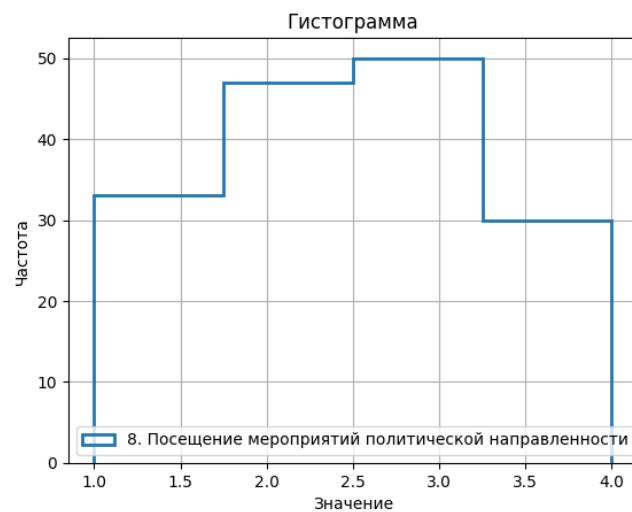


Рис. 2.2.11. Гистограмма значений признака  $X_8$ .

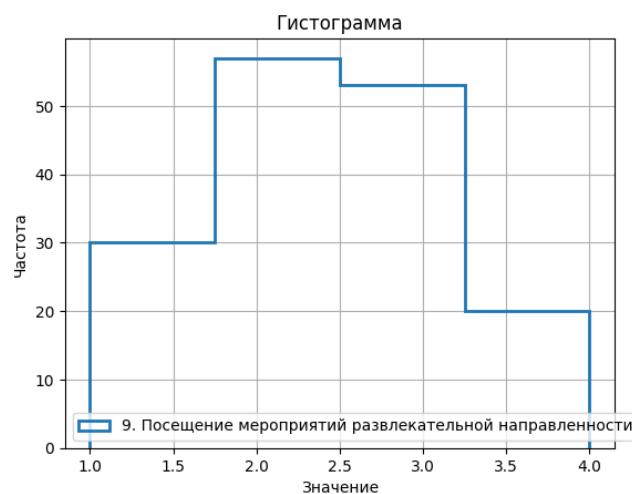


Рис. 2.2.12. Гистограмма значений признака  $X_9$ .

Для построения гистограммы применялась следующая функция:

```
1 def histogram(table: pd.DataFrame, series: list, path_to_save: str = ''):  
2     """  
3         Построение гистограммы распределения.  
4         Аргументы функции:  
5             table - таблица, содержащая столбец с данными,  
6             series - список названий столбцов с данными,  
7             path_to_save - по ум. = '', путь для сохранения файлов.  
8         """  
9  
10    # Создание папки для изображений:  
11    if path_to_save != '':  
12        # Нумерация построенных гистограмм в сохранённых файлах:  
13        hist_number = 0  
14        # Создание подпапки:  
15        path_to_save = path_to_save + '/hist'  
16        output_dir = pathlib.Path(path_to_save)  
17        # Проверка существования папки:  
18        if not output_dir.exists():  
19            # Создание папки для изображений:  
20            output_dir.mkdir(parents=True, exist_ok=True)  
21    # Перебор всех выбранных столбцов:  
22    for s in series:  
23        # Выделение координатных осей:  
24        fig, ax = plt.subplots()  
25        # Добавление сетки:  
26        ax.grid(True)  
27        # Помещение данных столбца таблицы в список:  
28        data_list = table[s].tolist()  
29        # Количество интервалов на гистограмме:  
30        # Тут можно указать жаемое количество интервалов, на которые будут  
31        # делиться значения признака (т. е. количество столбцов в гистограмме),  
32        # например, 5.  
33        # При этом будет необходимо закомментировать последние 3 строки кода.  
34        # Это может быть полезно при желании сделать столбцы гистограммы шире,  
35        # избежать пустого пространства между столбцами и т. п.  
36        # bins = 5  
37        bins = table[s].nunique()  
38        # Установка предела количества прямоугольников на гистограмме:  
39        if bins > 30:  
40            bins = 30  
41        # Построение гистограммы:  
42        plt.hist(data_list, bins, histtype='step', linewidth=2, label=s)  
43        # Добавление легенды:  
44        plt.legend()  
45        # Добавление подписей к осям и заголовка:  
46        plt.xlabel('Значение')  
47        plt.ylabel('Частота')  
48        plt.title('Гистограмма')  
49        # Сохранение изображения:  
50        if path_to_save != '':  
51            # Установка имени файла и пути к нему:  
52            output_file = output_dir / ('hist_' + str(hist_number) + '.png')  
53            # Сохранение файла:  
54            plt.savefig(output_file)  
55            # Изменение значения количества построенных изображений:  
56            hist_number = hist_number + 1  
57        # Отображение гистограммы:  
58        plt.show()
```

Рис. 2.2.13. Функция построения гистограмм.

Тут функция принимает в качестве аргументов переменную table типа pandas.DataFrame для хранения таблицы со значениями, переменную series типа list для хранения списка с заголовками столбцов таблицы, для которых строится гистограмма, переменную типа path\_to\_save типа string для хранения имени подпапки для сохранения изображений, или пустого значения в случае, если сохранять изображения не требуется.

Для построения диаграммы размаха применялась следующая функция:

```
1 def box_diagram(
2     table: pd.DataFrame,
3     series: list,
4     path_to_save: str = ''
5 ):
6 """
7 Построение коробчатой диаграммы (диаграммы размаха, ящика с усами).
8 Аргументы функции:
9 table - таблица, содержащая столбец с данными,
10 series - список названий столбцов с данными,
11 path_to_save - по ум. = '', путь для сохранения файлов.
12 """
13 # Создание папки для изображений:
14 if path_to_save != '':
15     # Нумерация построенных изображений:
16     box_number = 0
17     # Создание подпапки:
18     path_to_save = path_to_save + '/box'
19     output_dir = pathlib.Path(path_to_save)
20     # Проверка существования папки:
21     if not output_dir.exists():
22         # Создание папки для изображений:
23         output_dir.mkdir(parents=True, exist_ok=True)
24     # Перебор всех выбранных столбцов:
25     for s in series:
26         # Построение горизонтальной коробчатой диаграммы. При желании
27         # изобразить диаграмму вертикально, последний параметр надо
28         # изменить с False на True:
29         table.boxplot(s, vert=False)
30         # Добавление подписей к осям и заголовка:
31         plt.title('Диаграмма размаха')
32         plt.xlabel('Значения')
33         # Поворот подписи слева от диаграммы для экономии места:
34         plt.xticks(rotation=90, va='center')
35         # Сохранение изображения:
36         if path_to_save != '':
37             # Установка имени файла и пути к нему:
38             output_file = output_dir / ('box_' + str(box_number) + '.png')
39             # Сохранение файла:
40             plt.savefig(output_file)
41             # Изменение значения количества построенных изображений:
42             box_number = box_number + 1
43         # Отображение диаграммы:
44         plt.show()
45 
```

Рис. 2.2.14. Функция построения диаграмм размаха.

Тут функция принимает в качестве аргументов переменную `table` типа `pandas.DataFrame` для хранения таблицы со значениями, переменную `series` типа `list` для хранения списка с заголовками столбцов таблицы, для которых строится диаграмма размаха, переменную типа `path_to_save` типа `string` для хранения имени подпапки для сохранения изображений, или пустого значения в случае, если сохранять изображения не требуется.

Для построения точечной диаграммы применялась следующая функция:

```

1 def scatter_diagram(
2     table: pd.DataFrame,
3     series: list,
4     path_to_save: str = ''
5     ):
6     """
7     Построение точечной диаграммы.
8     Аргументы функции:
9     table - таблица, содержащая столбец с данными,
10    series - список названий столбцов с данными,
11    path_to_save - по ум. = '', путь для сохранения файлов.
12    """
13    # Создание папки для изображений:
14    if path_to_save != '':
15        # Нумерация построенных изображений:
16        dot_number = 0
17        # Создание подпапки:
18        path_to_save = path_to_save + '/dot'
19        output_dir = pathlib.Path(path_to_save)
20        # Проверка существования папки:
21        if not output_dir.exists():
22            # Создание папки для изображений:
23            output_dir.mkdir(parents=True, exist_ok=True)
24        # Перебор всех пар указанных столбцов без повторений:
25        for i in column_headers_show_dot:
26            for j in column_headers_show_dot:
27                if i >= j: continue
28                # Подготовка значений для построения диаграммы:
29                data_list_1 = table[i].tolist()
30                data_list_2 = table[j].tolist()
31                # Построение точечной диаграммы:
32                plt.scatter(data_list_1, data_list_2)
33                # Добавление подписей к осям и заголовка:
34                plt.xlabel(i)
35                plt.ylabel(j)
36                plt.title('Точечная диаграмма')
37                # Сохранение изображения:
38                if path_to_save != '':
39                    # Установка имени файла и пути к нему:
40                    output_file = output_dir / ('dot_' + str(dot_number) + '.png')
41                    # Сохранение файла:
42                    plt.savefig(output_file)
43                    # Изменение значения количества построенных изображений:
44                    dot_number = dot_number + 1
45                # Отображение диаграммы:
46                plt.show()
47

```

Рис. 2.2.15. Функция построения точечных диаграмм.

Собранные данные были проверены на наличие пустых значений. После применения шифра среди значений признака  $X_5$  присутствовали строчные значения «NaN» и числовые значения. Значения «NaN» были получены из незашифрованных значений «Нет», и считались пустыми значениями. Строки, содержащие эти значения, были удалены из таблицы.

Для обработки пустых значений применялась следующая функция:

```
1 def void_killer(
2     table: pd.DataFrame,
3     series: list,
4     mod: str = 'delete'
5     ) -> pd.DataFrame:
6 """
7 Удаление пустых значений.
8 Аргументы функции:
9 table - таблица, содержащая столбец с данными,
10 series - список названий столбцов с данными,
11 mod - по ум. 'delete', так же возможно 'mean', 'median'.
12 Возвращает pd.DataFrame.
13 """
14 for s in series:
15     # Получение индексов строк с пропусками:
16     # void_r = table[s][table[s].isnull().any(axis=1)].index.tolist()
17     void_i = table[s].loc[table[s].isna()].index
18     # Удаление строк с пустыми значениями:
19     if mod == 'delete':
20         for i in void_i:
21             print(f'Пустое значение "{s}[{i}]" удаляется со строкой.')
22         table = table.drop(void_i)
23     # Замена пустых значений медианой:
24     if mod == 'mean':
25         for i in void_i:
26             print(f'Пустое значение "{s}[{i}]" \
27 заменяется на среднее арифметическое столбца.')
28         # Преобразование данных в массив numpy:
29         np_data = table[s].to_numpy()
30         # Удаление значений, равных NaN:
31         np_data = np_data[~np.isnan(np_data)]
32         # Вычисление среднего арифметического:
33         mean = np_data.mean()
34         table.loc[void_i, s] = mean
35     if mod == 'median':
36         for i in void_i:
37             print(f'Пустое значение "{s}[{i}]" заменяется на медиану \
38 столбца.')
39         # Преобразование данных в массив numpy:
40         np_data = table[s].to_numpy()
41         # Удаление значений, равных NaN:
42         np_data = np_data[~np.isnan(np_data)]
43         # Вычисление медианы:
44         median = np.median(np_data)
45         table.loc[void_i, s] = median
46     return table
47 
```

Рис. 2.2.16. Функция удаления пустых значений.

Тут функция принимает в качестве аргументов переменную `table` типа `pandas.DataFrame` для хранения таблицы со значениями, переменную `series` типа `list` для хранения списка с заголовками столбцов таблицы, в которых обрабатываются пустые значения, переменную `mod` типа `string`, определяющую то, как будут обрабатываться пустые значения – удаляться, или заменяться на медиану или среднее арифметическое.

Собранные данные были проверены на наличие выбросов. Границы выбросов рассчитывались по формуле (1.4.6). Среди значений признака  $X_5$  были обнаружены выбросы, которые были заменены на среднее арифметическое, так как они могли указывать на некоторую тенденцию значений признака в выборке.

После удаления пустых значений и выбросов мощность выборки стала равна  $n = 106$ .

Для удаления выбросов применялась следующая функция:

```
1  def filter_outliers(
2      table: pd.DataFrame,
3      series: list,
4      mod: str = 'mean'
5      ) -> pd.DataFrame:
6
7      """  
8          Удаление выбросов.  
9          Аргументы функции:  
10         table - таблица, содержащая столбец с данными,  
11         series - список названий столбцов с данными,  
12         mod - по ум. 'mean', так же возможно 'square', 'delete'.  
13         Возвращает pd.DataFrame.  
14     """  
15
16     for s in series:  
17         # ОПРЕДЕЛЕНИЕ ГРАНИЦ ВЫБРОСОВ  
18         # Первый и третий квартили:  
19         q1 = table[s].quantile(0.25)  
20         q3 = table[s].quantile(0.75)  
21         # Межквартальный размах:  
22         iqr = q3 - q1  
23         # Границы выбросов:  
24         lower_bound = q1 - 1.5 * iqr  
25         upper_bound = q3 + 1.5 * iqr  
26         # Определение индексов строк с выбросами:  
27         i_outliers = table[
28             (table[s] < lower_bound) | (table[s] > upper_bound)
29         ].index  
30         # УДАЛЕНИЕ ВЫБРОСОВ  
31         # Проверка, надо ли удалять выбросы в рассматриваемом столбце s:
32         if not i_outliers.empty:
33             # Замена значений выбросов на среднее арифметическое значений
34             # столбца:
35             if mod == 'mean':
```

Рис. 2.2.17. Функция удаления выбросов (часть 1).

```

34         mean = round(table[s].mean(), 3)
35         for i in i_outliers:
36             print(f'Заменята значение "{s} [{i}]" = {table.loc[i, s]} \
37 на среднее арифметическое значений столбца, равное {mean}.')
38             table.loc[i, s] = mean
39             # Применение преобразования квадратного корня значений столбца:
40             if mod == 'square':
41                 print(f'Применение преобразования квадратного корня к столбцу \
42 "{s}" из-за встреченных значений:')
43                 for i in i_outliers:
44                     print(f'{s} [{i}]' = {table.loc[i, s]})'
45                     table[s] = np.sqrt(table[s])
46             # Удаление строк с выбросами:
47             if mod == 'delete':
48                 for i in i_outliers:
49                     print(f'Удаление строки "{i}" из-за встреченного \
50 "{table.loc[i, s]}" в столбце "{s}"')
51                 table = table[
52                     (table[s] >= lower_bound) & (table[s] <= upper_bound)
53                 ]
54             # Защита от столбцов с одинаковым содержанием ячеек
55             # после удаления выбросов:
56             # Список столбцов для удаления:
57             no_sence_columns_to_drop = []
58             # Проверка каждого столбца на наличие одинаковых значений:
59             for column in table.columns:
60                 if table[column].nunique() == 1:
61                     print(f'Столбец "{column}" после удаления выбросов \
62 содержит одинаковые значения и удаляется.')
63                     no_sence_columns_to_drop.append(column)
64             # Удаляем столбцы с одинаковыми значениями:
65             table = table.drop(columns=no_sence_columns_to_drop)
66     return table
67

```

Рис. 2.2.18. Функция удаления выбросов (часть 2).

Тут функция принимает в качестве аргументов переменную `table` типа `pandas.DataFrame` для хранения таблицы со значениями, переменную `series` типа `list` для хранения списка с заголовками столбцов таблицы, в которых обрабатываются выбросы, переменную `mod` типа `string`, определяющую то, как будут обрабатываться выбросы – заменяются на среднее арифметическое, удаляются, или подвергаются преобразованию квадратного корня.

Для полученных значений можно сформировать интервалльные и дискретные вариационные ряды.

Для вычисления частоты и относительной частоты использовались формулы (1.4.1) и (1.4.2).

Полученные значения для возможности сравнивать их между собой и дальнейшего анализа данных необходимо нормализовать. Для нормализации использовалась формула (1.4.7).

После нормализации данные готовы для их анализа.

Ниже представлены вариационные ряды, гистограммы и диаграммы размаха для обработанных значений признаков.

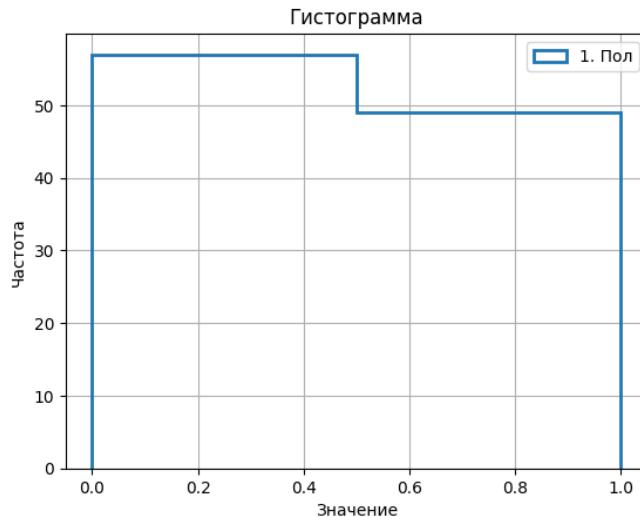


Рис. 2.2.19. Гистограмма значений признака  $X_1$ .

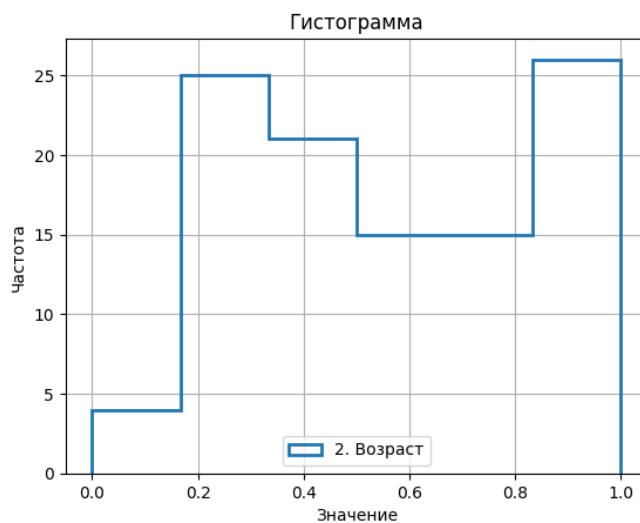


Рис. 2.2.20. Гистограмма значений признака  $X_2$ .

Таблица 2.2.1

Дискретные вариационные ряды для признаков  $X_1$  и  $X_2$ .

Показатель	Признак		Всего
	$X_1$	$X_2$	
	Номер варианты ответа		

	1	2	19	22	25	28	31	34	
Частоты, $n_i$	57	49	4	25	21	15	15	26	106
Относительные частоты, $w_i$	0,54	0,46	0,04	0,24	0,2	0,14	0,14	0,25	1
Процент, %	54%	46%	4%	24%	20%	14%	14%	25%	100%

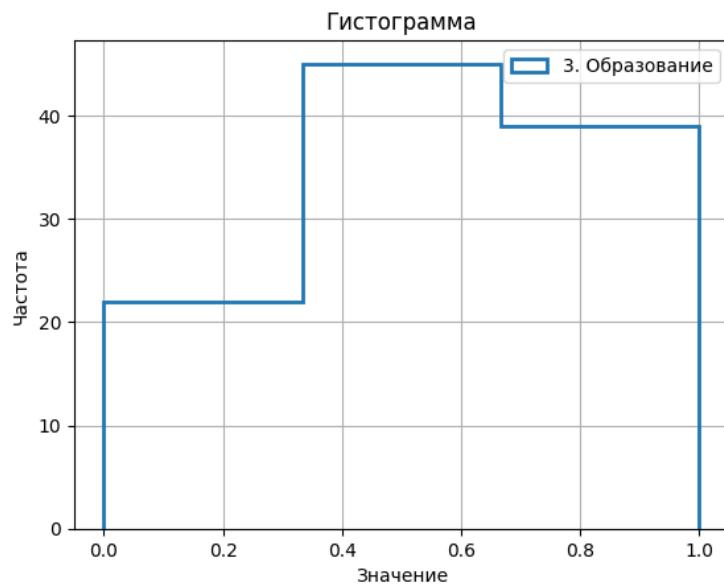


Рис. 2.2.21. Гистограмма значений признака  $X_3$ .

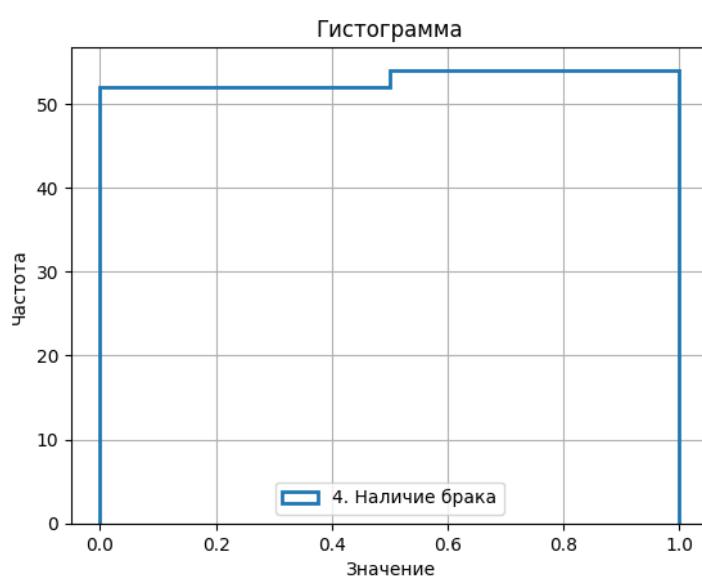


Рис. 2.2.22. Гистограмма значений признака  $X_4$ .

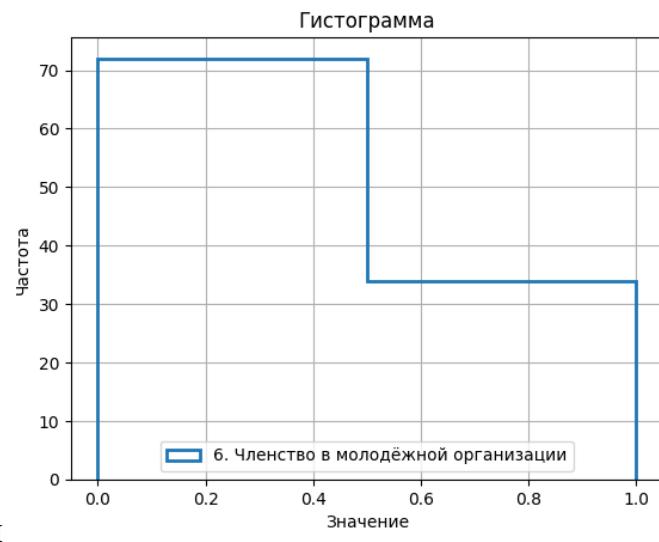


Рис. 2.2.23. Гистограмма значений признака  $X_6$ .

Таблица 2.2.2

Дискретные вариационные ряды для признаков  $X_3, X_4, X_6$ .

Показатель	Признак							Всего	
	$X_3$		$X_4$		$X_6$				
	Номер варианты ответа								
	1	2	3	0	1	0	1		
Частоты, $n_i$	22	45	39	52	54	72	34	106	
Относительные частоты, $w_i$	0,21	0,42	0,37	0,49	0,51	0,68	0,32	1	
Процент, %	21%	42%	37%	49%	51%	68%	32%	100%	

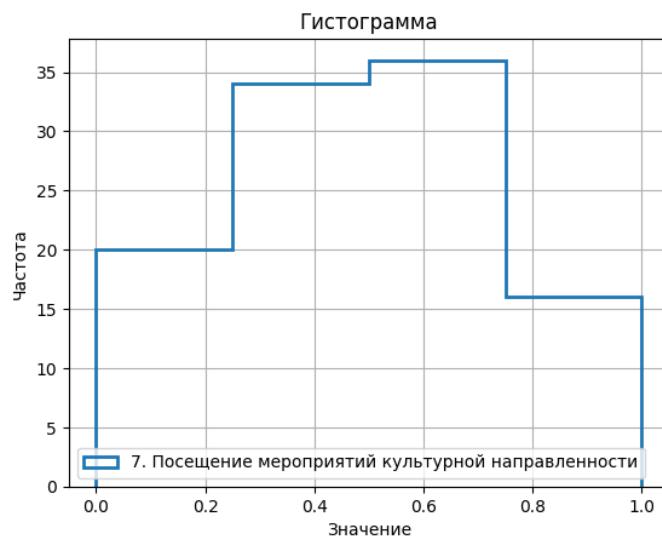


Рис. 2.2.24. Гистограмма значений признака  $X_7$ .

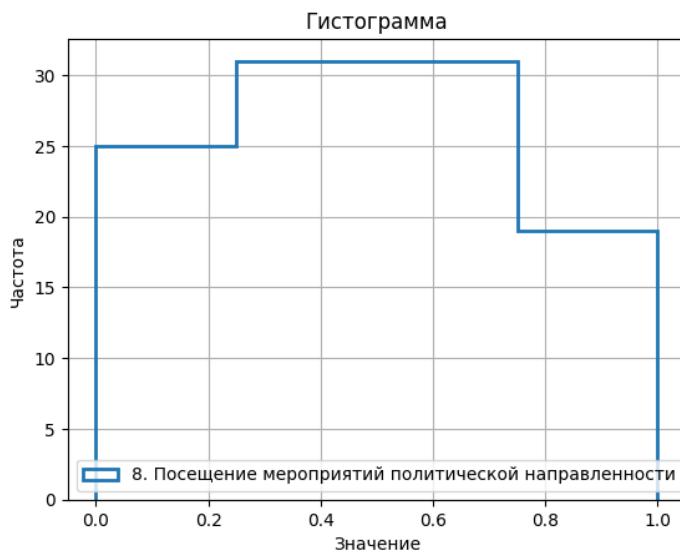


Рис. 2.2.25. Гистограмма значений признака  $X_8$ .

Таблица 2.2.3

Дискретные вариационные ряды для признаков  $X_7$  и  $X_8$ .

Показатель	Признак								Всего	
	$X_7$				$X_8$					
	Номер варианты ответа									
	1	2	3	4	1	2	3	4		
Частоты, $n_i$	20	34	36	16	25	31	31	19	106	
Относительные частоты, $w_i$	0,19	0,32	0,34	0,15	0,24	0,29	0,29	0,18	1	
Процент, %	19%	32%	34%	15%	24%	29%	29%	18%	100%	

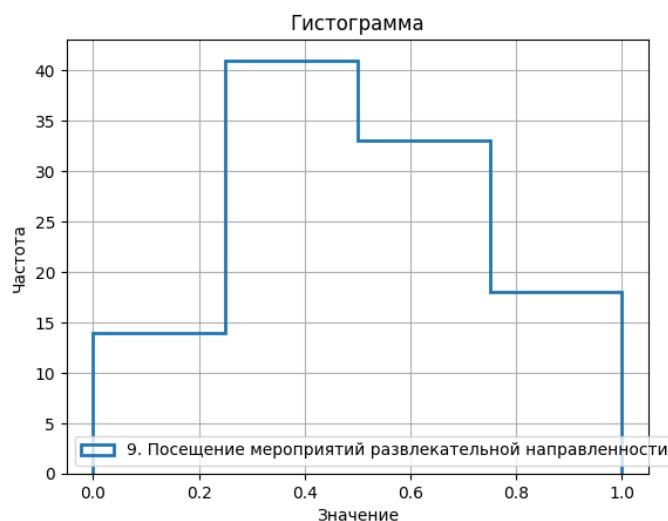


Рис. 2.2.26. Гистограмма значений признака  $X_9$ .

Таблица 2.2.4

Дискретный вариационный ряд для признака  $X_9$ .

Показатель	Признак				Всего	
	$X_9$					
	Номер варианты ответа					
	1	2	3	4		
Частоты, $n_i$	14	41	33	18	106	
Относительные частоты, $w_i$	0,13	0,39	0,31	0,17	1	
Процент, %	13%	39%	31%	17%	100%	

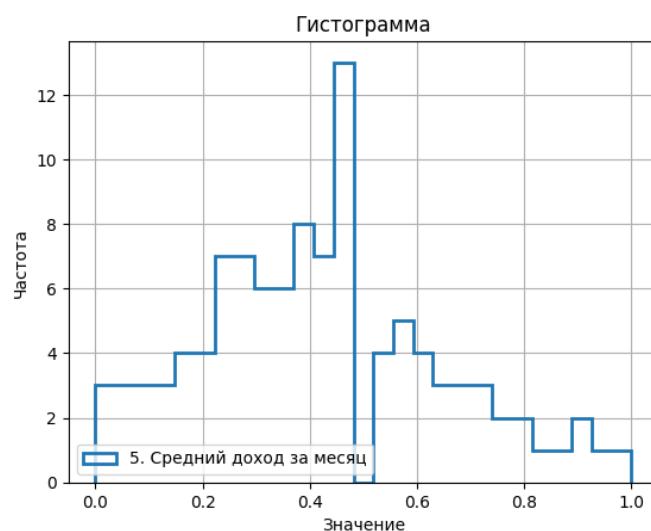
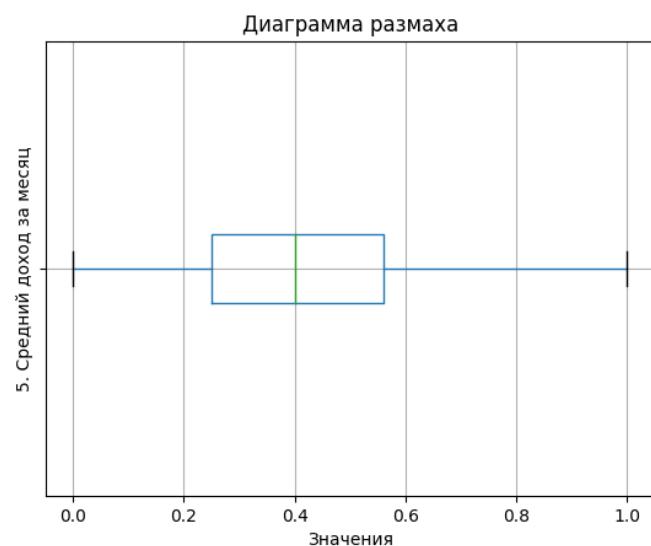
Рис. 2.2.27. Гистограмма значений признака  $X_5$ .Рис. 2.2.28. Диаграмма размаха признака  $X_5$ .

Таблица 2.2.5

Интервальный вариационный ряд для признака  $X_5$ .

Показатель	Признак					Всего	
	$X_5$						
	Интервал						
	15000- 20000	20000- 25000	25000- 30000	30000- 35000	35000- 40000		
Частоты, $n_i$	16	30	37	15	8	106	
Относительные частоты, $w_i$	0,15	0,28	0,35	0,14	0,07	1	
Процент, %	15%	28%	35%	14%	7%	100%	

Для получения вариационных рядов применялась следующая функция:

```

1 def frequency_table(
2     table: pd.DataFrame,
3     series: str,
4     mod: str = 'interval',
5     path_to_save: str = ''
6     ):
7     """
8         Функция создаёт таблицу со значениями вариационного ряда.
9         Аргументы функции:
10            table - таблица, содержащая столбец с данными,
11            series - список названий столбцов с данными,
12            mod - по ум. = 'interval', так же возможно 'describe'.
13            При 'interval' строится вариационный ряд.
14            При 'describe' строится дискретный ряд.
15            path_to_save - по ум. = '', путь для сохранения файлов.
16        """
17        # Создание папки для изображений:
18        if path_to_save != '':
19            # Настройка подпапки:
20            path_to_save = path_to_save + '/' + mod
21            # Нумерация построенных изображений:
22            freq_number = 0
23            output_dir = pathlib.Path(path_to_save)
24            # Проверка существования папки:
25            if not output_dir.exists():
26                # Создание папки для изображений:
27                output_dir.mkdir(parents=True, exist_ok=True)
28        # Перебор всех выбранных столбцов:
29        for s in series:
30            # Дискретный вариационный ряд:
31            if mod == 'describe':
32                # Получение уникальных значений столбца и их сортировка:
33                unique_values = sorted((table[s].unique()).tolist())
34                # Получение частот значений столбца:
35                frequency = [table[table[s] == x].shape[0] for x in unique_values]
36                # Получение относительных частот столбца:

```

Рис. 2.2.29. Функция получения вариационных рядов (часть 1).

```

37 |     relative_frequency = [f / table.shape[0] for f in frequency]
38 | # Получение процентных значений:
39 | percent_frequency = [str(round(f * 100, 0)) + '%' \
40 | for f in relative_frequency]
41 | # Создание таблицы с результатами:
42 | table_frequency = pd.DataFrame({
43 |     "Варианты x_i": unique_values,
44 |     "Частоты n_i": frequency,
45 |     "Относительные частоты w_i": relative_frequency,
46 |     "Процент, %": percent_frequency
47 | })
48 | print(f'Вариационный ряд значений столбца "{s}":')
49 | display(table_frequency)
50 | # Интервальный вариационный ряд:
51 | if mod == 'interval':
52 |     # Получение уникальных значений столбца и их сортировка:
53 |     unique_values = sorted(table[s].unique())
54 |     # Вычисление минимального и максимального значений:
55 |     min_value = min(unique_values)
56 |     max_value = max(unique_values)
57 |     # Вычисление количества интервалов:
58 |     if len(unique_values) > 20:
59 |         num_intervals = int(np.sqrt(len(unique_values)))
60 |     else:
61 |         num_intervals = len(unique_values)
62 |     # Вычисление ширины интервала:
63 |     interval_width = (max_value - min_value) / num_intervals
64 |     # Создание интервалов в списке:
65 |     intervals = [[
66 |         min_value + i * interval_width, \
67 |         min_value + (i + 1) * interval_width
68 |     ] for i in range(num_intervals)]
69 |     # Редактирование последнего интервала для включения в него
#самых больших значений:
70 |     intervals[-1][-1] = intervals[-1][-1] + 0.0001
71 |     # Вычисление значений для каждого интервала:
72 |     # Перебор всех интервалов:
73 |     result = []
74 |     interval_labels = []
75 |     frequency = []
76 |     relative_frequency = []
77 |     for i in intervals:
78 |         # Добавление интервала:
79 |         interval_labels.append(f'{round(i[0], 3)} - {round(i[1], 3)}')
80 |         # Вычисление частоты значений в интервале:
81 |         count = len(table[(table[s] >= i[0]) & (table[s] < i[1])])
82 |         frequency.append(count)
83 |         # Вычисление относительной частоты значений в интервале:
84 |         relative_freq = count / len(table)
85 |         relative_frequency.append(relative_freq)
86 |         # Получение процентных значений:
87 |         percent_frequency = [str(round(f * 100, 0)) + '%' \
88 | for f in relative_frequency]
89 |     # Создание таблицы с результатами:
90 |     table_frequency = pd.DataFrame({
91 |         "Интервал": interval_labels,
92 |         "Частоты n_i": frequency,
93 |         "Относительные частоты w_i": relative_frequency,
94 |         "Процент, %": percent_frequency
95 |     })

```

Рис. 2.2.30. Функция получения вариационных рядов (часть 2).

```

97     print(f'Интервальный ряд значений столбца "{s}"')
98     display(table_frequency)
99
100    # Сохранение таблицы в файл:
101    if path_to_save != '':
102        # Изменение значения количества построенных таблиц:
103        freq_number = freq_number + 1
104
105        # Сохранение таблицы в текущую директорию (папку):
106        if mod == 'interval':
107            file_name = \
108                '/result_table_frequency_interval_' + str(freq_number) + '.csv'
109        if mod == 'describe':
110            file_name = \
111                '/result_table_frequency_describe_' + str(freq_number) + '.csv'
112
113        table_frequency.to_csv(
114            path_to_save + file_name,
115            index=False
116        )

```

Рис. 2.2.31. Функция получения вариационных рядов (часть 3).

Тут функция принимает в качестве аргументов переменную `table` типа `pandas.DataFrame` для хранения таблицы со значениями, переменную `series` типа `list` для хранения списка с заголовками столбцов таблицы, для которых строятся вариационные ряды, переменную `mod` типа `string`, определяющую, будет строиться дискретный или интервальный вариационный ряд, переменную `path_to_save` типа `string` для хранения имени подпапки для сохранения таблиц, или пустого значения в случае, если сохранять изображения не требуется.

Для нормализации применялась следующая функция:

```

1 def normalization(table: pd.DataFrame, series: list) -> pd.DataFrame:
2     """
3         Нормализация данных.
4         Аргументы функции:
5             table - таблица, содержащая столбец с данными,
6             series - список названий столбцов с данными.
7             Возвращает pd.DataFrame.
8         """
9
10    for s in series:
11        # Минимальное и максимальное значения:
12        x_min = table[s].min()
13        x_max = table[s].max()
14        # Вычисление нормализованного значения:
15        table[s] = (table[s] - x_min) / (x_max - x_min)
16

```

Рис. 2.2.32. Функция нормализации.

Тут функция принимает в качестве аргументов переменную `table` типа `pandas.DataFrame` для хранения таблицы со значениями, переменную

series типа list для хранения списка с заголовками столбцов таблицы, данные которых подвергаются нормализации.

Для стандартизации использовалась следующая функция:

```
1 def standardization(table: pd.DataFrame, series: list) -> pd.DataFrame:
2     """
3         Нормализация данных.
4         Аргументы функции:
5             table - таблица, содержащая столбец с данными,
6             series - список названий столбцов с данными.
7             Возвращает pd.DataFrame.
8         """
9     for s in series:
10         # Среднее арифметическое:
11         mean = table[s].mean()
12         # Стандартное квадратическое отклонение:
13         std = table[s].std()
14         # Вычисление стандартизованного значения:
15         table[s] = (table[s] - mean) / std
16
17     return table
```

Рис. 2.2.33. Функция стандартизации.

Тут функция принимает в качестве аргументов переменную table типа pandas.DataFrame для хранения таблицы со значениями, переменную series типа list для хранения списка с заголовками столбцов таблицы, данные которых подвергаются стандартизации.

После нормализации описательные метрики различных признаков можно сравнивать между собой.

Вычисление описательных метрик эмпирических данных.

Медиана вычисляется согласно формулам (1.5.1, 1.5.2).

Среднее арифметическое вычисляется согласно формуле (1.5.3).

Дисперсия вычисляется согласно формулам (1.5.4, 1.5.8).

Среднее квадратическое отклонение вычисляется согласно формуле (1.5.10).

Асимметрия вычисляется согласно формуле (1.5.14).

Эксцесс вычисляется согласно формуле (1.5.15).

Табл. 2.2.6

Описательные метрики признаков(начало).

Признак	n	Мода	Медиана	Среднее арифметическое
$X_1$	106	0	0	0.462
$X_2$	106	1	0.6	0.570
$X_3$	106	0.5	0.5	0.580

$X_4$	106	1	1	0.509
$X_5$	106	0.4	0.4	0.418
$X_6$	106	0	0	0.321
$X_7$	106	0.667	0.333	0.484
$X_8$	106	0.333	0.333	0.472
$X_9$	106	0.333	0.333	0.506

Табл. 2.2.7

Описательные метрики признаков (окончание).

Признак	Дисперсия	Среднее квадратическое отклонение	Асимметрия	Эксцесс
$X_1$	0.251	0.499	0.149	1.004
$X_2$	0.104	0.321	0.041	1.597
$X_3$	0.139	0.371	-0.262	1.816
$X_4$	0.252	0.500	-0.037	0.983
$X_5$	0.052	0.227	0.343	2.728
$X_6$	0.220	0.467	0.757	1.560
$X_7$	0.104	0.321	0.007	1.997
$X_8$	0.120	0.345	0.074	1.807
$X_9$	0.096	0.308	0.087	2.108

Для вычисления описательных метрик применялась следующая функция:

```

1 def metrics(
2     table: pd.DataFrame,
3     series: list,
4     path_to_save: str = ''
5     ) -> pd.DataFrame:
6
7     """
8         Нахождение описательных метрик данных.
9         Аргументы функции:
10            table - таблица, содержащая столбец с данными,
11            series - список названий столбцов с данными,
12            path_to_save - по ум. = '', путь для сохранения файлов.
13            Возвращает pd.DataFrame.
14
15            # ВЫЧИСЛЕНИЕ МЕТРИК
16            # Создание таблицы, куда будет размещаться ответ:

```

Рис. 2.2.34. Функция вычисления описательных метрик (часть 1).

```

16     column_names_metrics = [
17         'данные',
18         'n',
19         'Мода',
20         'Медиана',
21         'Среднее арифметическое',
22         'Дисперсия',
23         'Среднее квадратическое отклонение',
24         'Асимметрия',
25         'Эксцесс'
26     ]
27     result_metrics = pd.DataFrame(columns = column_names_metrics)
28     for s in series:
29         # Преобразование данных в массив пимпу:
30         np_data = table[s].to_numpy()
31         # РАСЧЁТ ОПИСАТЕЛЬНЫХ СТАТИСТИК
32         # Вычисление моды:
33         mode = sci.stats.mode(np_data)[0]
34         # Вычисление медианы:
35         median = np.median(np_data)
36         # Вычисление среднего квадратического:
37         std = np.std(np_data)
38         # Объём выборки:
39         n = len(np_data)
40         # Вычисление среднего арифметического:
41         mean = np_data.mean()
42         # Вычисление выборочной дисперсии:
43         # Для вычисления генеральной дисперсии необходимо
44         # приравнять значение ddof нулю: ddof = 0
45         variance = np.var(np_data, ddof = 1)
46         # Вычисление асимметрии:
47         A = np.mean(((np_data - mean) ** 3) / variance ** (3 / 2))
48         # Вычисление эксцесса:
49         E = np.mean(((np_data - mean) ** 4) / (variance ** 2))
50         # Создание таблицы с результатами:
51         result = pd.DataFrame({
52             'данные': [s],
53             'n': [n],
54             'Мода': [round(mode, 3)],
55             'Медиана': [round(median, 3)],
56             'Среднее арифметическое': [round(mean, 3)],
57             'Дисперсия': [round(variance, 3)],
58             'Среднее квадратическое отклонение': [round(std, 3)],
59             'Асимметрия': [round(A, 3)],
60             'Эксцесс': [round(E, 3)]
61         })
62         result_metrics = pd.concat([result_metrics, result])
63         # Восстановление индексов таблицы:
64         result_metrics = result_metrics.reset_index(drop=True)
65         # СОЗДАНИЕ ФАЙЛА ТАБЛИЦЫ С РЕЗУЛЬТАТОМ
66         if path_to_save != '':
67             output_dir = pathlib.Path(path_to_save)
68             # Проверка существования папки:
69             if not output_dir.exists():

```

Рис. 2.2.35. Функция вычисления описательных метрик (часть 2).

```

70     # Создание папки для таблицы:
71     output_dir.mkdir(parents=True, exist_ok=True)
72     # Сохранение таблицы с метриками в текущую директорию (папку):
73     result_metrics.to_csv(
74         path_to_save + '/result_table_metrics.csv',
75         index=False
76     )
77     return result_metrics
78

```

Рис. 2.2.36. Функция вычисления описательных метрик (часть 3).

Тут функция принимает в качестве аргументов переменную `table` типа `pandas.DataFrame` для хранения таблицы со значениями, переменную `series` типа `list` для хранения списка с заголовками столбцов таблицы, для которых вычисляются метрики, переменную `path_to_save` типа `string` для хранения имени подпапки для сохранения таблицы, или пустого значения в случае, если сохранять изображения не требуется.

### 2.3. Анализ данных умным помощником

После обработки данные можно использовать для их анализа и проверки статистических гипотез.

Пусть будет проведена проверка гипотезы для всех полученных признаков о том, что между хотя бы одной парой признаков есть статистически значимая корреляция, являющаяся хотя бы заметной согласно шкале Чеддока, т. е.  $|r| \geq 0,5$ . Тогда альтернативной гипотезой будет отсутствие статистически значимой корреляции, которая будет хотя бы заметной согласно шкале Чеддока.

Выдвинем 2 гипотезы:

$H_0$ : между хотя бы одной парой признаков из полученных  $X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9$  есть статистически значимая корреляция со значением коэффициента корреляции  $|r| \geq 0,5$ .

$H_1$ : между хотя бы одной парой признаков из полученных  $X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9$  нет статистически значимой корреляции со значением коэффициента корреляции  $|r| \geq 0,5$ .

Найдём коэффициент корреляции Пирсона для всех пар признаков, проверим их статистическую значимость с помощью t-распределения, и выведем информацию только о значимых коэффициентах корреляции.

В рамках этой задачи пусть будет  $\alpha = 0,05$ .

Выведем таблицу с парами признаков и соответствующими им значениями коэффициентов корреляции, в которые по модулю будут больше 0.5, а также некоторые соответствующие им точечные диаграммы:

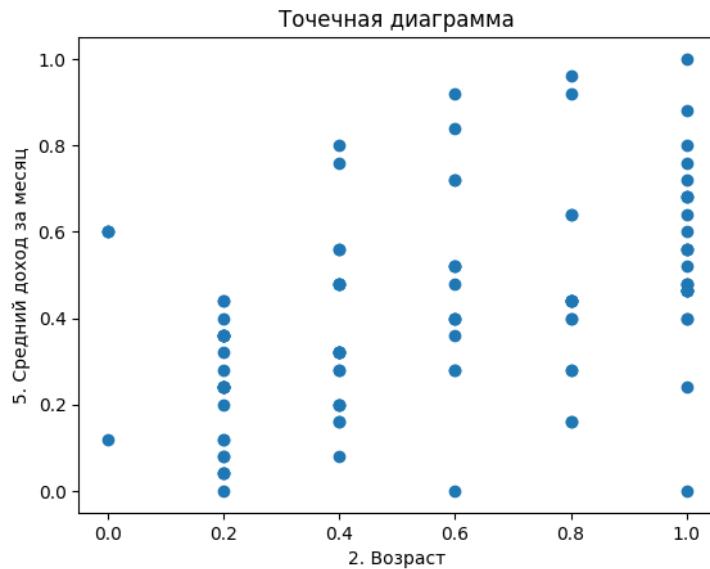


Рис. 2.3.1. Точечная диаграмма для признаков  $X_2$  и  $X_5$ .

Таблица 2.3.1  
Вычисление коэффициентов корреляции.

$X_i$	$X_j$	n	t-статистика (t)	Табличное критическое значение t-статистики (t крит.)	$ t  > t \text{ крит.}$	r
$X_2$	$X_3$	106	8.276	1.983	ИСТИНА	0.63

Из результатов работы программы можно видеть, что в выборочной совокупности наблюдается заметная положительная корреляция между признаками  $X_2$  и  $X_3$ .

Указанная пара признаков имеет между собой коэффициенты корреляции, удовлетворяющие условию  $|r| \geq 0.5$ .

Вывод: гипотеза  $H_0$  принимается, отклоняется гипотеза  $H_1$ .

Для вычисления коэффициентов корреляции применяется следующая функция:

```

1 def correlation_pearson(
2     table: pd.DataFrame,
3     series: list,
4     alpha: float = 0.95,
5     only_meaning: bool = True,
6     critical: float = 0,
```

Рис. 2.3.2. Функция коэффициента корреляции Пирсона (часть 1).

```

7 |         scatter_diagram_plotting: bool = False,
8 |         path_to_save: str = ''
9 |     ) -> pd.DataFrame:
10 |
11 |     """"
12 |     Вычисление коэффициента корреляции Пирсона.
13 |     Аргументы функции:
14 |     table - таблица, содержащая столбец с данными,
15 |     series - список названий столбцов с данными,
16 |     alpha - по ум. = 0.95, возвм. знач.: 0,1; 0,5; 0,25; 0,1; 0,001,
17 |     only_meaning - по ум. = True, если True, то выводятся только значимые
18 |     значения коэффициента корреляции,
19 |     critical - по ум. = 0, если не равен 0, то выводятся только значения
20 |     коэффициента корреляции, большие critical,
21 |     scatter_diagram_plotting - по ум. False, если True, то выводится точечная
22 |     диаграмма scatter_diagram(),
23 |     path_to_save - по ум. = '', путь для сохранения файлов.
24 |     Возвращает pd.DataFrame.
25 |
26 |     # ВЫЧИСЛЕНИЕ КОРРЕЛЯЦИИ
27 |     # Создание таблицы, куда будет размещаться ответ:
28 |     column_names_corr = [
29 |         'Первый столбец',
30 |         'Второй столбец',
31 |         'n',
32 |         't-статистика (t)',
33 |         'Табличное критическое значение t-статистики (t крит.)',
34 |         '|t| > t крит.',
35 |         'Качественная характеристика',
36 |         'Значимость корреляции',
37 |         'Коэффициент корреляции'
38 |     ]
39 |     result_corr = pd.DataFrame(columns = column_names_corr)
40 |     for i, series_1 in enumerate(series):
41 |         for j, series_2 in enumerate(series):
42 |             if i >= j: continue
43 |             # Подготовка значений для построения диаграммы:
44 |             data_list_1 = table[series_1].tolist()
45 |             data_list_2 = table[series_2].tolist()
46 |             # Вычисление коэффициента корреляции Пирсона:
47 |             r = float(sci.stats.pearsonr(data_list_1, data_list_2)[0])
48 |             # Классификация корреляции по шкале Чеддока:
49 |             ch_r = abs(r)
50 |             if ch_r < 0.1:
51 |                 cheddok = ''
52 |             elif ch_r < 0.3:
53 |                 cheddok = '*'
54 |             elif ch_r < 0.5:
55 |                 cheddok = '**'
56 |             elif ch_r < 0.7:
57 |                 cheddok = '***'
58 |             elif ch_r < 0.9:
59 |                 cheddok = '****'
60 |             else:
61 |                 cheddok = '*****'
62 |             # Определение объема выборки
63 |             n = len(data_list_1)
64 |             # Определение степеней свободы:
65 |             df = n - 2
66 |             # Вычисление t-статистики:

```

Рис. 2.3.3. Функция коэффициента корреляции Пирсона (часть 2).

```

67     # Нахождение критического значения t-распределения:
68     t_critical = sci.stats.t.ppf(1 - alpha/2, df)
69     # Проверка значимости:
70     if abs(t) > t_critical:
71         Conclusion = 'Значима'
72     else:
73         Conclusion = 'Не значима'
74     # Проверка, удовлетворяют ли значения условиям отображения:
75     raw_show = False
76     if only_meaning:
77         if Conclusion == 'Значима':
78             if critical != 0:
79                 if abs(r) > critical:
80                     raw_show = True
81                 else:
82                     raw_show = True
83             else:
84                 if critical != 0:
85                     if abs(r) > critical:
86                         raw_show = True
87                     else:
88                         raw_show = True
89     if raw_show:
90         # Создание строки с вычисленным коэффициентом корреляции:
91         result = pd.DataFrame({
92             'первый столбец': [series_1],
93             'второй столбец': [series_2],
94             'n': [n],
95             't-статистика (t)': [round(t, 3)],
96             'Табличное критическое значение t-статистики (t крит.)': \
97             [round(t_critical, 3)],
98             '|t| > t крит.': [abs(t) > t_critical],
99             'Качественная характеристика': [cheddok],
100            'Значимость корреляции': [Conclusion],
101            'Коэффициент корреляции': [round(r, 3)]
102        })
103        # Добавление строки в таблицу с результатами:
104        result_corr = pd.concat([result_corr, result])
105    # Вывод точечной диаграммы анализируемых данных:
106    if raw_show and scatter_diagram_plotting:
107        # Построение точечной диаграммы:
108        plt.scatter(data_list_1, data_list_2)
109        # Добавление подписей к осям и заголовка:
110        plt.xlabel(series_1)
111        plt.ylabel(series_2)
112        plt.title('Точечная диаграмма')
113        # Отображение диаграммы:
114        plt.show()
115    # Восстановление индексов таблицы:
116    result_corr = result_corr.reset_index(drop=True)
117    # СОХРАНЕНИЕ ФАЙЛА ТАБЛИЦЫ С РЕЗУЛЬТАТОМ
118    if path_to_save != '':
119        # Создание подпапки:
120        path_to_save = path_to_save + '/correlation'
121        output_dir = pathlib.Path(path_to_save)
122        # Проверка существования папки:
123        if not output_dir.exists():
124            # Создание папки для таблицы:
125            output_dir.mkdir(parents=True, exist_ok=True)
126        # Сохранение таблицы с метриками в текущую директорию (папку):
127        result_corr.to_csv(

```

Рис. 2.3.4. Функция коэффициента корреляции Пирсона (часть 3).

```

128     path_to_save + '/result_table_correlation.csv',
129     index=False
130   )
131   return result_corr
132

```

Рис. 2.3.5. Функция коэффициента корреляции Пирсона (часть 4).

Тут функция принимает в качестве аргументов переменную `table` типа `pandas.DataFrame` для хранения таблицы со значениями, переменную `series` типа `list` для хранения списка с заголовками столбцов таблицы, для которых вычисляются коэффициенты корреляции, переменную `alpha` типа `float` для хранения значения  $\alpha$ , переменную `only_meaning` типа `bool` для хранения информации о необходимости вывода строк с только статистически значимыми коэффициентами корреляции, переменную `critical` типа `float` для хранения значения, при модуле коэффициента корреляции ниже которого строка не будет выводиться, переменную `scatter_diagram_plotting` типа `bool` для хранения информации о необходимости построения точечных диаграмм, переменную `path_to_save` типа `string` для хранения имени подпапки для сохранения таблицы, или пустого значения в случае, если сохранять изображения не требуется.

Пусть будет проведена проверка гипотезы о соответствия эмпирического распределения нормальному распределению для признака  $X_5$ .

Выдвинем 2 гипотезы:

$H_0$ : распределение признака  $X_5$  является нормальным.

$H_1$ : распределение признака  $X_5$  не является нормальным.

Вычислим значение критерия Эппса-Палли для признака  $X_5$ , сравним его с табличным значением, и выведем результаты проверки распределения на нормальность.

В рамках этой задачи пусть будет  $\alpha = 0.05$ .

Выведем таблицу с результатами работы программы, а также гистограмму значений признака  $X_5$ .

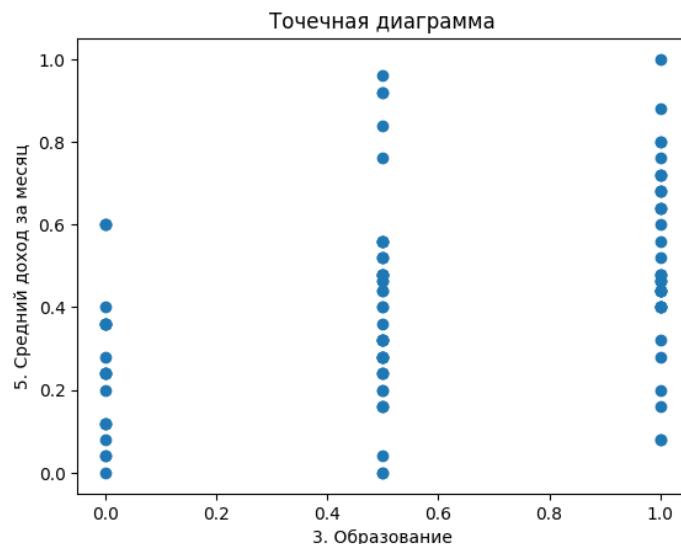


Рис. 2.3.6. Гистограмма значений признака  $X_5$ .

Таблица 2.3.2

Проверка распределения на нормальность.

$X_i$	n	p	Значение критерия (T_EP)	Табличное значение критерия (T_EP крит.)	T_EP < T_EP крит.	Нормальное распределение
$X_5$	106	0.95	0.182	0.376	ИСТИНА	Выполняется

Из результатов работы программы можно видеть, что распределение признака  $X_5$  является нормальным.

Вывод: гипотеза  $H_0$  принимается, отклоняется гипотеза  $H_1$ .

Для проверки распределения на нормальность применяется следующая функция:

```

1 def epps_pally(
2     table: pd.DataFrame,
3     series: list,
4     p_level: float = 0.95,
5     path_to_save: str = ''
6     ) -> pd.DataFrame:
7
8     """
9         Критерий Эппса-Палли для проверки нормального распределения данных.
10        Аргументы функции:
11        table - табл., содержащая столбец с анализируемыми данными,
12        series - список названий столбцов с данными,
13        p-level - вероятность получить верное значение

```

Рис. 2.3.7. Функция критерия Эппса-Палли (часть 1).

```

13     p = alpha - 1, по ум. = 0.95, возмож. знач.: 0,9; 0,95; 0,975; 0,99,
14     path_to_save - по ум. = '', путь для сохранения файлов.
15     Возвращает: pd.DataFrame.
16     """
17     # Вычисление критерия Эппса-Палли
18     # Создание таблицы, куда будет размещаться ответ:
19     column_names_T_EP = [
20         'данные',
21         'n',
22         'p',
23         'Значение критерия (T_EP)',
24         'Табличное значение критерия (T_EP крит.)',
25         'T_EP < Table',
26         'Нормальное распределение'
27     ]
28     result_T_EP = pd.DataFrame(columns = column_names_T_EP)
29     # Таблица для критерия Эппса-Палли:
30     # p-квантили статистики критерия T_EP для
31     # p = 1 - alpha = 0,90; 0,95; 0,975; 0,99
32     n_index = [8, 9, 10, 15, 20, 30, 50, 100, 200]
33     data = {
34         'p=0.9': [0.271, 0.275, 0.279, 0.284, 0.287,
35                     0.289, 0.290, 0.291, 0.290],
36         'p=0.95': [0.347, 0.350, 0.357, 0.366, 0.368,
37                     0.371, 0.374, 0.376, 0.379],
38         'p=0.975': [0.426, 0.428, 0.437, 0.447, 0.450,
39                     0.459, 0.460, 0.464, 0.467],
40         'p=0.99': [0.526, 0.537, 0.545, 0.560, 0.564,
41                     0.569, 0.574, 0.583, 0.590]
42     }
43     T_EP_table = pd.DataFrame(data, index=n_index)
44     T_EP_table.index.name = 'n'
45     # Выбор величины доверительной вероятности:
46     p_level_dict = {
47         0.9: T_EP_table.columns[0],
48         0.95: T_EP_table.columns[1],
49         0.975: T_EP_table.columns[2],
50         0.99: T_EP_table.columns[3]
51     }
52     for s in series:
53         # Преобразование данных в массив numpy:
54         np_data = table[s].to_numpy()
55         # Вычисление значения критерия
56         # Объём выборки:
57         n = len(np_data)
58         # Критерий применим для n >= 8:
59         if n < 8:
60             print(f'Столбец {s} не обработан, так как n < 8 не возможно \
61 получение достоверных результатов. n = {n}')
62             continue
63         # Среднее арифметическое:
64         mean = np_data.mean()
65         # Центральный момент 2-го порядка:
66         m_2 = np.var(np_data, ddof = 0)
67         A = sqrt(2) * np.sum([exp(-(np_data[i] - mean)**2 / (4*m_2))
68             for i in range(n)])
69         B = 2/n * np.sum([
70             np.sum([
71                 exp(-(np_data[j] - np_data[k])**2 / (2*m_2)) for j in \
72                 range(0, k)
73             ]) for k in range(1, n)])

```

Рис. 2.3.8. Функция критерия Эппса-Палли (часть 2).

```

74     T_EP_empiric = 1 + n / sqrt(3) + B - A
75     # НАХОЖДЕНИЕ ТАБЛИЧНОГО ЗНАЧЕНИЯ
76     # Линейная интерполяция для нахождения р для
77     # различных значений выборки n:
78     N_index = T_EP_table.index
79     T = T_EP_table[p_level_dict[p_level]]
80     f_lin = sci.interpolate.interp1d(N_index, T)
81     T_EP_teoretic = float(f_lin(n))
82     # СРАВНЕНИЕ ПОЛУЧЕННЫХ ЗНАЧЕНИЙ
83     if T_EP_empiric < T_EP_teoretic:
84         Conclusion = 'Выполняется'
85     else:
86         Conclusion = 'Не выполняется'
87     # СОЗДАНИЕ ТАБЛИЦЫ С РЕЗУЛЬТАТАМИ
88     result = pd.DataFrame({
89         'Данные': [s],
90         'n': [n],
91         'p': [p_level],
92         'Значение критерия (T_EP)': [round(T_EP_empiric, 3)],
93         'Табличное значение критерия (T_EP крит.)': \
94             [round(T_EP_teoretic, 3)],
95         'T_EP < Table': [T_EP_empiric < T_EP_teoretic],
96         'Нормальное распределение': [Conclusion]
97     })
98     result_T_EP = pd.concat([result_T_EP, result])
99     # Восстановление индексов таблицы:
100    result_T_EP = result_T_EP.reset_index(drop=True)
101    # СОХРАНЕНИЕ ФАЙЛА ТАБЛИЦЫ С РЕЗУЛЬТАТОМ
102    if path_to_save != '':
103        # Создание подпапки:
104        path_to_save = path_to_save + '/T_EP'
105        output_dir = pathlib.Path(path_to_save)
106        # Проверка существования папки:
107        if not output_dir.exists():
108            # Создание папки для таблицы:
109            output_dir.mkdir(parents=True, exist_ok=True)
110        # Сохранение таблицы с метриками в текущую директорию (папку):
111        result_T_EP.to_csv(
112            path_to_save + '/result_table_T_EP.csv',
113            index=False
114        )
115    return result_T_EP
116

```

Рис. 2.3.9. Функция критерия Эпписа-Палли (часть 3).

Тут функция принимает в качестве аргументов переменную `table` типа `pandas.DataFrame` для хранения таблицы со значениями, переменную `series` типа `list` для хранения списка с заголовками столбцов таблицы, для которых вычисляются коэффициенты корреляции, переменную `p_level` типа `float` для хранения значения  $p$ , переменную `path_to_save` типа `string` для хранения имени подпапки для сохранения таблицы, или пустого значения в случае, если сохранять изображения не требуется.

## ЗАКЛЮЧЕНИЕ

Социологические исследования имеют широкую сферу применения, их проводят с целью сбора информации о различных социальных процессах и явлениях и связях между ними, а также для получения возможных выводов о возможности действий с целью влияния на социальные процессы и явления.

В любом социологическом исследовании проводятся следующие этапы: проведение статистического наблюдения, обработка полученных данных, анализ обработанных данных, проверка статистических гипотез.

Первый раздел работы содержит описание некоторых правил и методов проведения выборки в социологических исследованиях, методов обработки и анализа эмпирических данных социологических исследований, а также описание коэффициента корреляции Пирсона и критерия Эппса-Палли, используемых для проверки статистических гипотез.

Второй раздел включает в себя обработку и анализ эмпирических данных социологического исследования, проведённого среди представителей трудоустроенной молодёжи Крыма. В исследовании участвовало 160 человек.

Сбор данных производился методом анкетирования, выборка среди респондентов проводилась методом простого случайного отбора.

Были изучены методы обработки и анализа данных социологических исследований, в ходе работы был разработан умный помощник на языке программирования высокого уровня Python, и практически применён к эмпирическим данным, полученным в результате социологического опроса трудоустроенных представителей молодёжи Крыма.

В ходе работы был разработан умный помощник на языке программирования высокого уровня Python. Эмпирические данные, полученные в результате анкетирования, были обработаны умным помощником. В результате обработки и анализа ответов респондентов программой стало известно, что:

- 1) Среди респондентов находилось 54% мужчин и 46% женщин.
- 2) 4% респондентов возрастом 18-20 лет, 24% респондентов возрастом 21-23 года, 20% респондентов возрастом 24-26 лет, 14% респондентов возрастом 27-29 лет, 14% респондентов возрастом 30-32 года, 25% респондентов возрастом 33-35 лет.

- 3) 21% респондентов имеет среднее общее образование, 42% респондентов имеет среднее профессиональное образование, 37% респондентов имеет высшее образование.
- 4) 51% респондент находится в официально зарегистрированном или официально не зарегистрированном браке, а 49% респондентов не находятся в официально зарегистрированном или официально не зарегистрированном браке.
- 5) 32% респондентов состоят в какой-либо молодёжной организации, а 68% респондентов не состоят ни в какой молодёжной организации.
- 6) 15% респондентов постоянно посещает общественные мероприятия культурной направленности в регионе, 34% респондентов делает это часто, 34% респондентов делает это редко, а 19% респондентов никогда не посещает общественные мероприятия культурной направленности.
- 7) 18% респондентов постоянно посещает общественные мероприятия политической направленности в регионе, 29% делают это часто, 29% делают это редко, а 24% респондентов никогда не посещает общественные мероприятия политической направленности в регионе.
- 8) 17% респондентов постоянно посещает общественные мероприятия политической культурной направленности в регионе, 31% делает это часто, 39% делает это редко, а 13% никогда не постоянно посещает общественные мероприятия политической культурной направленности в регионе.

Коэффициент корреляции Пирсона показал, что между уровнями образования и возрастом респондентов присутствует заметная положительная корреляция со значением  $r = 0,63$ .

Критерий Эпса-Палли показал, что уровень среднего дохода респондентов имеет нормальное распределение.

В разработанном умном помощнике поэтапно реализованы различные методы обработки и анализа данных, каждый из которых сопровождается подробными пояснениями и комментариями в коде.

Основными преимуществами по сравнению с существующими программными продуктами являются встроенное поэтапное руководство для пользователя по использованию умного помощника вместе с подробным комментированием кода, наличие программной реализации

критерия Эппса-Палли, которая отсутствует в широко известных библиотеках языка Python, открытый код, который свободен для модификации пользователем.

В дальнейшем результаты данного социологического исследования могут применяться в других социологических исследованиях, для разработки программных продуктов, или для других целей.

## Список использованной литературы

10. Губин В. И., Осташков В. Н., 2007, Статистические методы обработки экспериментальных данных: Учеб. пособие для студентов технических вузов – 202 с.
11. А. А. Типикин, А. А. Прусаков, Н. А. Тимошенко. Программная реализация критерия Эпса-Палли в среде моделирования Matlab. [Электронный ресурс] / Режим доступа: <https://openedu.rea.ru/jour/article/download/1028/627>
12. t-статистика Стьюдента в Excel. [Электронный ресурс] / Режим доступа: <https://baguzin.ru/wp/wp-content/uploads/2018/10/t-statistika-Styudenta-v-Excel.pdf>
13. Стус Е. А. Методические рекомендации и задания к практическим занятиям по дисциплине «Языки программирования для анализа данных»: учебно-методическое пособие.
14. Федеральный закон от 30.12.2020 № 489-ФЗ "О молодежной политике в Российской Федерации" [Электронный ресурс], Режим доступа: <http://actual.pravo.gov.ru/content/content.html#pnum=0001202012300003>
15. Орлов Александр Иванович, ВЕРОЯТНОСТНО-СТАТИСТИЧЕСКИЕ МОДЕЛИ КОРРЕЛЯЦИИ И РЕГРЕССИИ [Электронный ресурс] / Режим доступа: <http://ej.kubagro.ru/2020/06/pdf/11.pdf>

## Приложения

### Приложение А

Критерий Эпса-Палли:  $p$ -квантили статистики критерия  $T_{EP}$  для  $p = 1 - \alpha = 0,90; 0,95; 0,975$  и  $0,99$ :

$n$	$p$				$n$	$p$			
	0,9	0,95	0,975	0,99		0,9	0,95	0,975	0,99
8	0,271	0,347	0,426	0,526	30	0,289	0,371	0,459	0,569
9	0,275	0,350	0,428	0,537	50	0,290	0,374	0,460	0,574
10	0,279	0,357	0,447	0,545		0,291	0,376	0,464	0,583
15	0,284	0,366	0,447	0,560		0,290	0,379	0,467	0,590
20	0,287	0,368	0,450	0,564					

## Приложение Б

t-распределение Стьюдента.

df	p				df	p			
	0,10	0,05	0,01	0,001		0,10	0,05	0,01	0,001
1	6,314	12,70	63,65	636,61	46	1,679	2,013	2,687	3,515
2	2,920	4,303	9,925	31,602	47	1,678	2,012	2,685	3,510
3	2,353	3,182	5,841	12,923	48	1,677	2,011	2,682	3,505
4	2,132	2,776	4,604	8,610	49	1,677	2,010	2,680	3,500
5	2,015	2,571	4,032	6,869	50	1,676	2,009	2,678	3,496
6	1,943	2,447	3,707	5,959	51	1,675	2,008	2,676	3,492
7	1,895	2,365	3,499	5,408	52	1,675	2,007	2,674	3,488
8	1,860	2,306	3,355	5,041	53	1,674	2,006	2,672	3,484
9	1,833	2,262	3,250	4,781	54	1,674	2,005	2,670	3,480
10	1,812	2,228	3,169	4,587	55	1,673	2,004	2,668	3,476
11	1,796	2,201	3,106	4,437	56	1,673	2,003	2,667	3,473
12	1,782	2,179	3,055	4,318	57	1,672	2,002	2,665	3,470
13	1,771	2,160	3,012	4,221	58	1,672	2,002	2,663	3,466
14	1,761	2,145	2,977	4,140	59	1,671	2,001	2,662	3,463
15	1,753	2,131	2,947	4,073	60	1,671	2,000	2,660	3,460
16	1,746	2,120	2,921	4,015	61	1,670	2,000	2,659	3,457
17	1,740	2,110	2,898	3,965	62	1,670	1,999	2,657	3,454
18	1,734	2,101	2,878	3,922	63	1,669	1,998	2,656	3,452
19	1,729	2,093	2,861	3,883	64	1,669	1,998	2,655	3,449
20	1,725	2,086	2,845	3,850	65	1,669	1,997	2,654	3,447
21	1,721	2,080	2,831	3,819	66	1,668	1,997	2,652	3,444
22	1,717	2,074	2,819	3,792	67	1,668	1,996	2,651	3,442
23	1,714	2,069	2,807	3,768	68	1,668	1,995	2,650	3,439
24	1,711	2,064	2,797	3,745	69	1,667	1,995	2,649	3,437
25	1,708	2,060	2,787	3,725	70	1,667	1,994	2,648	3,435
26	1,706	2,056	2,779	3,707	71	1,667	1,994	2,647	3,433
27	1,703	2,052	2,771	3,690	72	1,666	1,993	2,646	3,431
28	1,701	2,049	2,763	3,674	73	1,666	1,993	2,645	3,429

## **Приложение В**

Анкета.

Изучение участия молодёжи Крыма в общественной жизни их региона.

Уважаемый респондент, мы проводим социологическое исследование на тему "Изучение участия молодёжи Крыма в общественной жизни их региона" и будем признательны, если вы ответите на вопросы нашей анкеты. Опрос проводится добровольно и анонимно.

1. Укажите ваш пол.

- 1) Мужской
- 2) Женский

2. Сколько вам полных лет?

- 1) 18-20
- 2) 21-23
- 3) 24-26
- 4) 27-29
- 5) 30-32
- 6) 33-35

3. Какое образование у вас имеется?

- 1) Общее
- 2) Среднее профессиональное
- 3) Высшее

4. Вы состоите в официально зарегистрированном или не зарегистрированном браке?

- 1) Да
- 2) Нет

5. Укажите ваш доход в среднем за месяц в рублях РФ.

---

6. Вы состоите в какой-либо молодёжной организации?

- 1) Да
- 2) Нет

7. Как часто вы посещаете общественные мероприятия культурной направленности в вашем регионе?

- 1) Постоянно
- 2) Часто
- 3) Редко
- 4) Никогда

8. Как часто вы посещаете общественные мероприятия политической направленности в вашем регионе?

- 1) Постоянно
- 2) Часто
- 3) Редко
- 4) Никогда

9. Как часто вы посещаете общественные мероприятия развлекательной направленности в вашем регионе?

- 1) Постоянно
- 2) Часто
- 3) Редко
- 4) Никогда

## Приложение Г

### Собранные эмпирические данные.

1	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9
2	1. Пол	2. Возраст	3. Образование	4. Наличие брака	5. Средний доход за месяц	6. Членство в молодёжной организации	7. Посещение мероприятий	8. Посещение мероприятий политической направленности	9. Посещение мероприятий развлекательной направленности
3	Мужской	21-23	Среднее общее	Нет	Нет	Нет	Редко	Редко	Редко
4	Мужской	18-20	Среднее общее	Нет	Нет	Да	Часто	Редко	Часто
5	Мужской	27-29	Высшее	Нет	Нет	Нет	Редко	Постоянно	Редко
6	Мужской	33-35	Высшее	Да	32000	Нет	Часто	Постоянно	Редко
7	Мужской	27-29	Высшее	Нет	33000	Нет	Постоянно	Часто	Постоянно
8	Мужской	24-26	Высшее	Нет	17000	Нет	Часто	Постоянно	Постоянно
9	Мужской	24-26	Высшее	Нет	35000	Нет	Постоянно	Постоянно	Редко
10	Мужской	21-23	Высшее	Нет	Нет	Нет	Редко	Часто	Редко
11	Мужской	33-35	Среднее профессиональное	Да	58000	Да	Редко	Часто	Редко
12	Мужской	33-35	Высшее	Нет	46000	Нет	Часто	Редко	Часто
13	Мужской	30-32	Среднее профессиональное	Да	39000	Да	Часто	Часто	Часто
14	Мужской	24-26	Среднее профессиональное	Нет	34000	Да	Часто	Часто	Часто
15	Мужской	30-32	Высшее	Нет	Нет	Да	Часто	Постоянно	Часто
16	Женский	21-23	Среднее общее	Нет	Нет	Нет	Редко	Часто	Редко
17	Женский	33-35	Высшее	Да	46000	Нет	Редко	Часто	Редко
18	Женский	21-23	Высшее	Нет	Нет	Нет	Часто	Редко	Редко
19	Женский	21-23	Среднее общее	Нет	18000	Нет	Никогда	Постоянно	Никогда
20	Мужской	21-23	Среднее общее	Нет	Нет	Нет	Часто	Постоянно	Часто
21	Женский	21-23	Среднее общее	Нет	Нет	Да	Редко	Редко	Часто
22	Мужской	21-23	Среднее общее	Нет	18000	Нет	Часто	Часто	Часто
23	Женский	18-20	Среднее общее	Нет	Нет	Нет	Редко	Никогда	Никогда
24	Женский	21-23	Среднее общее	Нет	Нет	Нет	Часто	Редко	Часто
25	Женский	18-20	Среднее общее	Нет	Нет	Нет	Редко	Часто	Редко
26	Женский	24-26	Высшее	Нет	Нет	Нет	Часто	Часто	Часто
27	Мужской	18-20	Среднее общее	Нет	Нет	Нет	Часто	Постоянно	Часто
28	Женский	18-20	Среднее общее	Нет	Нет	Нет	Редко	Редко	Редко
29	Женский	21-23	Среднее общее	Нет	Нет	Нет	Редко	Часто	Редко
30	Женский	21-23	Высшее	Да	17000	Нет	Редко	Редко	Редко
31	Женский	21-23	Среднее общее	Нет	20000	Да	Редко	Редко	Часто
32	Женский	18-20	Среднее общее	Нет	Нет	Да	Редко	Редко	Часто
33	Женский	18-20	Среднее общее	Нет	Нет	Нет	Редко	Часто	Редко
34	Женский	18-20	Среднее общее	Нет	Нет	Да	Часто	Часто	Часто
35	Женский	18-20	Среднее общее	Нет	Нет	Нет	Часто	Часто	Часто
36	Женский	18-20	Среднее общее	Нет	Нет	Нет	Часто	Редко	Редко
37	Женский	18-20	Среднее общее	Нет	Нет	Нет	Часто	Часто	Часто
38	Женский	21-23	Среднее общее	Нет	Нет	Да	Никогда	Никогда	Никогда
39	Женский	21-23	Среднее общее	Нет	Нет	Нет	Редко	Часто	Никогда
40	Мужской	21-23	Среднее общее	Нет	Нет	Нет	Редко	Часто	Часто
41	Мужской	18-20	Среднее общее	Нет	Нет	Нет	Редко	Часто	Редко
42	Женский	18-20	Среднее общее	Нет	Нет	Да	Часто	Часто	Часто
43	Женский	24-26	Среднее профессиональное	Нет	Нет	Да	Постоянно	Постоянно	Постоянно
44	Мужской	27-29	Среднее профессиональное	Да	15000	Нет	Часто	Часто	Постоянно
45	Мужской	27-29	Среднее профессиональное	Да	38000	Нет	Постоянно	Часто	Часто
46	Мужской	27-29	Среднее профессиональное	Нет	36000	Да	Часто	Редко	Часто
47	Мужской	27-29	Среднее профессиональное	Да	24000	Да	Редко	Постоянно	Редко
48	Мужской	27-29	Высшее	Нет	33000	Да	Часто	Часто	Редко
49	Мужской	27-29	Высшее	Нет	22000	Нет	Редко	Часто	Часто
50	Мужской	27-29	Высшее	Нет	Нет	Нет	Часто	Часто	Часто
51	Женский	27-29	Среднее общее	Да	Нет	Нет	Редко	Постоянно	Редко
52	Женский	27-29	Среднее профессиональное	Нет	22000	Нет	Никогда	Редко	Редко
53	Женский	27-29	Среднее профессиональное	Нет	27000	Нет	Никогда	Редко	Никогда
54	Мужской	30-32	Среднее профессиональное	Да	38000	Нет	Редко	Часто	Никогда
55	Мужской	30-32	Среднее профессиональное	Да	19000	Нет	Никогда	Часто	Редко
56	Мужской	30-32	Среднее профессиональное	Да	19000	Нет	Часто	Постоянно	Часто
57	Мужской	30-32	Среднее профессиональное	Да	22000	Нет	Редко	Часто	Редко
58	Мужской	30-32	Среднее профессиональное	Да	22000	Да	Часто	Постоянно	Часто
59	Мужской	30-32	Высшее	Нет	25000	Нет	Часто	Часто	Редко
60	Мужской	30-32	Высшее	Нет	Нет	Нет	Часто	Постоянно	Часто



131	Мужской	21-23	Среднее общее	Нет	17000	Да	Никогда	Никогда	Редко
132	Женский	33-35	Высшее	Да	37000	Нет	Редко	Постоянно	Никогда
133	Мужской	21-23	Среднее общее	Нет	21000	Да	Никогда	Никогда	Редко
134	Женский	33-35	Высшее	Да	35000	Нет	Редко	Никогда	Постоянно
135	Мужской	21-23	Среднее профессиональное	Нет	22000	Нет	Никогда	Никогда	Редко
136	Женский	33-35	Высшее	Да	33000	Нет	Постоянно	Никогда	Постоянно
137	Мужской	21-23	Среднее профессиональное	Нет	23000	Нет	Никогда	Никогда	Редко
138	Женский	33-35	Высшее	Да	32000	Нет	Постоянно	Никогда	Постоянно
139	Мужской	21-23	Среднее профессиональное	Нет	25000	Нет	Никогда	Никогда	Постоянно
140	Женский	33-35	Высшее	Да	30000	Нет	Редко	Никогда	Часто
141	Мужской	21-23	Среднее профессиональное	Нет	26000	Нет	Редко	Никогда	Часто
142	Женский	18-20	Среднее общее	Нет	Нет	Да	Никогда	Никогда	Часто
143	Женский	18-20	Среднее общее	Нет	Нет	Нет	Никогда	Никогда	Часто
144	Мужской	21-23	Среднее общее	Нет	15000	Да	Никогда	Никогда	Никогда
145	Женский	33-35	Высшее	Да	29000	Нет	Редко	Никогда	Постоянно
146	Мужской	21-23	Среднее общее	Нет	16000	Да	Никогда	Никогда	Редко
147	Мужской	24-26	Среднее профессиональное	Да	Нет	Да	Часто	Редко	Никогда
148	Женский	33-35	Высшее	Да	31000	Нет	Редко	Никогда	Часто
149	Женский	18-20	Среднее общее	Нет	Нет	Нет	Никогда	Никогда	Никогда
150	Женский	33-35	Высшее	Да	27000	Нет	Редко	Никогда	Постоянно
151	Мужской	21-23	Среднее общее	Нет	21000	Да	Никогда	Никогда	Редко
152	Женский	33-35	Высшее	Да	27000	Нет	Редко	Никогда	Постоянно
153	Женский	33-35	Высшее	Да	27000	Нет	Постоянно	Никогда	Постоянно
154	Мужской	21-23	Среднее профессиональное	Нет	26000	Нет	Редко	Никогда	Часто
155	Мужской	27-29	Среднее профессиональное	Да	Нет	Да	Часто	Редко	Никогда
156	Женский	18-20	Среднее общее	Да	Нет	Да	Никогда	Часто	Никогда
157	Мужской	27-29	Среднее профессиональное	Да	Нет	Да	Часто	Постоянно	Никогда
158	Мужской	21-23	Среднее профессиональное	Нет	16000	Нет	Никогда	Никогда	Никогда
159	Мужской	33-35	Среднее профессиональное	Нет	25000	Нет	Никогда	Никогда	Никогда
160	Мужской	21-23	Среднее общее	Нет	21000	Да	Никогда	Никогда	Никогда
161	Мужской	24-26	Среднее профессиональное	Да	Нет	Да	Часто	Редко	Никогда
162	Женский	18-20	Среднее общее	Да	Нет	Да	Никогда	Часто	Никогда
163									

## Приложение Д

### Код программы

#### Содержание

##### Умный помощник для специалистов, проводящих социологические исследования

Первоначальная настройка

Обработка данных

Создание и использование шифра для кодировки данных

Представление необработанных данных

Гистограммы

Диаграммы размаха

Точечные диаграммы

Удаление пустых значений

Удаление выбросов

Вариационные ряды данных

Нормализация и стандартизация данных

Представление обработанных данных

Гистограммы

Диаграммы размаха

Точечные диаграммы

Вычисление описательных метрик

Анализ данных

Настройка перед анализом данных

Корреляция значений признаков

Проверка распределения на соответствие нормальному

##### Умный помощник для специалистов, проводящих социологические исследования

Программа рассчитана на помощь специалистам, проводящим социологические исследования. Так же она может применяться в других целях, если для них необходима программная реализация методов анализа данных, рассматриваемых далее.

Код написан на языке Python.

В некоторых местах в коде расположены комментарии. Они не влияют на выполнение кода. Комментарии обозначены символом `#`. Комментарии могут объяснять устройство кода. Так же в комментариях может помещаться код с пояснениями, который изменяет работу программы. Для того, чтобы сделать код активным, необходимо удалить символ `#` (и, возможно, пробел после него) перед кодом.

Над блоками кода, где новичкам рекомендуется вносить изменения, размещена надпись "РЕДАКТИРОВАНИЕ КОДА".

```
✓ [1] 1 # Подключение необходимых для работы программы библиотек.  
2 # При локальном использовании необходимо установить эти библиотеки.  
3 import numpy as np  
4 import pandas as pd  
5 import matplotlib.pyplot as plt  
6 from math import *  
7 import scipy as sci  
8 import warnings  
9 import copy  
10 import pathlib  
11  
✓ [2] 1 # Отключение предупреждений:  
2 warnings.filterwarnings('ignore')  
3
```

## ▼ Первоначальная настройка

Для работы программы нужно загрузить файл таблицы с данными.

Данные в таблице должны соответствовать ряду правил:

- **Соответствие столбцам.** Статистические данные должны располагаться в столбцах с соответствующими им названиями.
- **Отсутствие пустых строк.** Между строками и столбцами таблицы не должно быть пустых строк и столбцов, т. е. все данные должны находиться в левом верхнем углу таблицы.
- **Одинаковая длина.** Заполненные столбцы должны иметь одинаковую длину.
- **Строка - объект.** В одной строке должна находиться данные только об одном объекте исследования (например, участнику опроса), и наоборот.

Необходимо подключить файл в коде программы. Ниже в коде программы приведён пример этой операции.

- Если вы работаете с локальным файлом, то вам необходимо загрузить файл в рабочую директорию (папку), и поместить название файла в кавычках вместо значения переменной path, как указанно в примере. Если вы работаете с Google colab, то ваша рабочая директория - "/content/".
- Если вы подключаете файл через Google Disk, то ваша рабочая директория - "[/content/drive/MyDrive/](#)".

## РЕДАКТИРОВАНИЕ КОДА

```
✓ [3] 1 # Подключение файла через загрузку в Google Colab :  
2 # path_empric_data = "/content/<Название файла>.csv"  
3  
4 # Подключение файла из текущей директории:  
5 # path_empric_data = "./<Название файла>.csv"  
6  
7 # Подключение файла с Google Disk:  
8 # path_empric_data = "/content/drive/MyDrive/Colab Notebooks/<Название файла>.csv"  
9  
10 # Подключение файла с Google Disk:  
11 path_empric_data = "/content/drive/MyDrive/Colab Notebooks/Diplom/Diplom\_dataset.csv"  
12  
13 # Загрузка данных в программу, и помещение их в переменную df_start  
14 df_start = pd.read_csv(path_empric_data)  
15
```

```
✓ [4] 1 from google.colab import drive  
2 drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount)

Тут и далее в некоторых местах расположены блоки кода, предназначенные для вывода данных о таблице, с которой работает программа, а так же краткой информации о данных в таблице.

В частности, строка

```
print(df_start.info(), end = '\n\n')
```

выводит краткую информацию о загруженных данных, давая возможность убедиться в том, что была загружена нужная таблица.

Помимо прочего, можно увидеть количество строк в загруженной таблице в строке

```
"RangeIndex: <число строк> entries, <индекс первой строки> to <индекс последней строки>"
```

Ниже можно видеть информацию о столбцах в таблице: номер столбца указываются в колонке "#", их называния в колонке "Column", количество не нулевых значений в столбце в колонке "Non-Null Count". В последнем столбце указан тип данных, т. е. то, в качестве каких значений программа воспринимает данные. В начале работы программы там будет отображаться "object", но при дальнейшей работе программы это изменится.

#	Column	Non-Null Count	Dtype
0	<Название столбца 1>	<n> non-null	object
1	<Название столбца 2>	<n> non-null	object

Рядом располагаются и другая техническая информация. Так, в самом низу указывается, какой объём памяти занимает таблица.

Для вывода фрагмента самой таблицы можно использовать целый ряд возможных строк с командами. Пояснения для них приведены в комментариях.

## РЕДАКТИРОВАНИЕ КОДА

```
[5] 1 # Вывод общей информации о таблице:
2 print(df_start.info(), end = '\n\n')
3
4 # Вывод первых 5 строк таблицы:
5 display(df_start.head(5))
6
7 # Вывод последних 5 строк таблицы:
8 # display(df_start.tail(5))
9
10 # Вывод всей таблицы:
11 # display(df_start)
12
13 # Вывод среза таблицы, например, с 10 по 20 строки:
14 # df_start[10:20]
15
```

```
→ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 160 entries, 0 to 159
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   1. Пол            160 non-null    object 
 1   2. Возраст         160 non-null    object 
 2   3. Образование     160 non-null    object 
 3   4. Наличие брака    160 non-null    object 
 4   5. Средний доход за месяц  160 non-null    object 
 5   6. Членство в молодёжной организаци 160 non-null    object 
 6   7. Посещение мероприятий культурной направленности 160 non-null    object 
 7   8. Посещение мероприятий политической направленности 160 non-null    object 
 8   9. Посещение мероприятий развлекательной направленности 160 non-null    object 
dtypes: object(9)
memory usage: 11.4+ KB
None
```

	1. Пол	2. Возраст	3. Образование	4. Наличие брака	5. Средний доход за месяц	6. Членство в молодёжной организаци	7. Посещение мероприятий культурной направленности	8. Посещение мероприятий политической направленности	9. Посещение мероприятий развлекательной направленности
0	Мужской	21-23	Среднее общее	Нет	Нет	Нет	Редко	Редко	Редко
1	Мужской	18-20	Среднее общее	Нет	Нет	Да	Часто	Редко	Часто
2	Мужской	27-29	Высшее	Нет	Нет	Нет	Редко	Постоянно	Редко
3	Мужской	33-35	Высшее	Да	32000	Нет	Часто	Постоянно	Редко
4	Мужской	27-29	Высшее	Нет	33000	Нет	Постоянно	Часто	Постоянно

Все изменения будут производиться с копией/копиями исходной таблицы или её фрагментов, чтобы загруженные данные не изменились. Сделаем копию имеющейся таблицы. Далее она будет храниться в переменной df\_data.

```
[6] 1 df_data = df_start.copy(deep=True)
2
```

В ходе работы программы будут создаваться изображения и таблицы с результатами обработки и анализа загруженных данных.

Можно выбрать одно из трёх действий:

- Сохранять файлы на Google Disk, подключив его в настройках Google Colab.
- Сохранять файлы в текущую папку (директорию). Это будет хранилище Google Colab, если работа происходит в нём, или папка на компьютере пользователя, если блокнот запущен локально.
- Не сохранять файлы. Они будут только отображаться в выводе блокнота.

## РЕДАКТИРОВАНИЕ КОДА

```
✓ [7] 1 # Сохранение файлов будет производиться в директорию (папку) на Google Disk.
2 # Необходимо дополнить строку, указав необходимую папку на Google Disk.
3 # files_storage = '/content/drive/MyDrive/<Папка для сохранения>/results/'
4
5 # Сохранение файлов будет производиться в текущую директорию (папку):
6 # files_storage = './results/'
7
8 # Для отсутствия сохранения файлов, выбирать эту строку:
9 # files_storage = ''
10
11 files_storage = \
12 '/content/drive/MyDrive/Colab Notebooks/Diplom latest files/results/'
```

### ▼ Обработка данных

#### ▼ Создание и использование шифра для кодировки данных

Для анализа данных необходимо преобразовать данные из текста в числа путём создания шифра.

Ниже приведён вывод названий всех столбцов для удобства. Это может быть полезно для указания при дальнейшей работе программы названий столбцов, в конце которых может находиться пробел (в таком случае он является частью названий столбца, и его необходимо будет указывать внутри кавычек вместе с названием).

```
✓ [8] 1 print('Вывод всех столбцов таблицы:\n')
2 for i in df_data.columns:
3     print(f'{i}')
4
→ Вывод всех столбцов таблицы:
"1. Пол"
"2. Возраст"
"3. Образование"
"4. Наличие брака"
"5. Средний доход за месяц"
"6. Членство в молодёжной организации"
"7. Посещение мероприятий культурной направленности"
"8. Посещение мероприятий политической направленности"
"9. Посещение мероприятий развлекательной направленности"
```

Далее создаются словари, с помощью которых имеющиеся в таблице данные кодируются некоторыми числами.

- В случае, если рассматриваемые данные можно представить на метрической шкале, то рекомендуется использовать или шкалу отношений, или интервальную шкалу с интервалами одинаковой длины.
- В случае, если рассматриваемые данные можно представить на ранговой (порядковой) шкале, то рекомендуется придавать каждому рангу некоторое число. При этом, если числа заменить на шкале изменить, то порядок расположения рангов на шкале сохраняется.
- В случае, если рассматриваемые данные напрямую не сравниваются, и используются для классификации объектов с их характеристиками, то используется номинальная шкала.

Пример создания словарей для двух или более признаков, для которых планируется использовать один шифр, приведён в комментариях в блоке ниже.

## РЕДАКТИРОВАНИЕ КОДА

```
✓ [9] 1 # Пример создания словарей для шифрования данных.
2 # Тут присваиваются числа 1 и 2 значению некоторого признака:
3 # dict_1 = {
4 #     'Первый вариант значения': 1,
5 #     'Второй вариант значения': 2
6 # }
```

```

8 # dict_2 = {
9 #     'Первый вариант значения': 1,
10 #     'Второй вариант значения': 2,
11 #     'Третий вариант значения': 3
12 # }
13
14 dict_1 = {
15     'Мужской': 1,
16     'Женский': 2
17 }
18
19 dict_2 = {
20     '18-20': 19,
21     '21-23': 22,
22     '24-26': 25,
23     '27-29': 28,
24     '30-32': 31,
25     '33-35': 34
26 }
27
28 dict_3 = {
29     'Среднее общее': 1,
30     'Среднее профессиональное': 2,
31     'Высшее': 3
32 }
33
34 dict_4_6 = {
35     'Да': 1,
36     'Нет': 0
37 }
38
39 dict_7_8_9 = {
40     'Постоянно': 4,
41     'Часто': 3,
42     'Редко': 2,
43     'Никогда': 1
44 }
45

```

Далее созданные выше словари применяются к загруженной таблице.

Возможно не только применять по одному словарю для каждого столбца, но и использовать один словарь для множества признаков. Так же возможно преобразовывать уже записанные числа в таблице, которые ранее воспринимались, как обычные строки, в числа, пригодные для вычислений.

Примеры применения словарей к таблице приведены в комментариях ниже.

#### РЕДАКТИРОВАНИЕ КОДА

```

✓ [10] 1 # Пример применения словаря к одному столбцу:
2 # df_data["Название столбца 1"] = \
3 # df_data["Название столбца 1"].replace(dict_1).infer_objects(copy=False)
4
5 # Для применения словаря ко множеству столбцов удобнее использовать такой код:
6 # question_2_3 = [
7 #     "Название столбца 2",
8 #     "Название столбца 3"
9 # ]
10 # for i in question_2_3:
11 #     df_data[i] = df_data[i].replace(dict_1).infer_objects(copy=False)
12
13 # Для преобразования чисел в специальный тип данных,
14 # чтобы программа правильно воспринимала загруженные числа,
15 # словарь не требуется:
16 # df_data['Название столбца 3'] = \
17 # pd.to_numeric(df_data['Название столбца 3'], errors='coerce')
18 # Приравнивание пустых значений к некоторому числу, например, 0:
19 # df_data['Название столбца 4'] = df_data['Название столбца 4'].fillna(0)
20
21 df_data["1. Пол"] = \
22 df_data["1. Пол"].replace(dict_1).infer_objects(copy=False)
23
24 df_data["2. Возраст"] = \
25 df_data["2. Возраст"].replace(dict_2).infer_objects(copy=False)
26
27 df_data["3. Образование"] = \

```

```

28 df_data["3. Образование"].replace(dict_3).infer_objects(copy=False)
29
30 question_4_6 = [
31     '4. Наличие брака',
32     '6. Членство в молодёжной организации'
33 ]
34 for i in question_4_6:
35     df_data[i] = df_data[i].replace(dict_4_6).infer_objects(copy=False)
36
37 df_data['5. Средний доход за месяц'] = \
38 pd.to_numeric(df_data['5. Средний доход за месяц'], errors='coerce')
39
40 question_7_8_9 = [
41     '7. Посещение мероприятий культурной направленности',
42     '8. Посещение мероприятий политической направленности',
43     '9. Посещение мероприятий развлекательной направленности'
44 ]
45 for i in question_7_8_9:
46     df_data[i] = df_data[i].replace(dict_7_8_9).infer_objects(copy=False)
47

```

Просмотр информации о таблице после применения словарей.

#### РЕДАКТИРОВАНИЕ КОДА

```

[11]  1 # Вывод общей информации о таблице:
2 print(df_data.info(), end = '\n\n')
3
4 # Вывод первых 5 строк таблицы:
5 display(df_data.head(5))
6
7 # Вывод последних 5 строк таблицы:
8 # display(df_data.tail(5))
9
10 # Вывод всей таблицы:
11 # display(df_data)
12
13 # Вывод среза таблицы, например, с 10 по 20 строки:
14 # df_data[10:20]
15

```

```

→ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 160 entries, 0 to 159
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   1. Пол            160 non-null    int64  
 1   2. Возраст         160 non-null    int64  
 2   3. Образование      160 non-null    int64  
 3   4. Наличие брака      160 non-null    int64  
 4   5. Средний доход за месяц  106 non-null    float64
 5   6. Членство в молодёжной организации 160 non-null    int64  
 6   7. Посещение мероприятий культурной направленности 160 non-null    int64  
 7   8. Посещение мероприятий политической направленности 160 non-null    int64  
 8   9. Посещение мероприятий развлекательной направленности 160 non-null    int64  
dtypes: float64(1), int64(8)
memory usage: 11.4 KB
None

```

	1. Пол	2. Возраст	3. Образование	4. Наличие брака	5. Средний доход за месяц	6. Членство в молодёжной организации	7. Посещение мероприятий культурной направленности	8. Посещение мероприятий политической направленности	9. Посещение мероприятий развлекательной направленности
0	1	22	1	0	NaN	0	2	2	2
1	1	19	1	0	NaN	1	3	2	3
2	1	28	3	0	NaN	0	2	4	2
3	1	34	3	1	32000.0	0	3	4	2
4	1	28	3	0	33000.0	0	4	3	4

#### ▼ Представление необработанных данных

Выбор столбцов для представления в виде:

1) Гистограммы.

2) Диаграммы размаха (коробчатой диаграммы).

3) Точечной диаграммы.

#### РЕДАКТИРОВАНИЕ КОДА

```
[12] 1 # ВЫБОР СТОЛБЦОВ ДЛЯ ПРЕДСТАВЛЕНИЯ В ВИДЕ ГИСТОГРАММ
2
3 # Для обработки малого числа столбцов:
4
5 # Тут необходимо перечислить столбцы, которые требуется использовать:
6 # column_headers_show_hist = [
7 #     "Название первого столбца",
8 #     "Название второго столбца"
9 # ]
10
11 # Для обработки большого числа столбцов:
12
13 # column_headers_show_hist = df_data.columns.tolist()
14 # Тут можно перечислить столбцы, которые не требуется использовать при
15 # построении гистограмм:
16 # data_remove = [
17 #     "Название первого столбца",
18 #     "Название второго столбца"
19 # ]
20 # for i in data_remove:
21 #     column_headers_show_hist.remove(i)
22
23 column_headers_show_hist = df_data.columns.tolist()
24
25 # ВЫБОР СТОЛБЦОВ ДЛЯ ПРЕДСТАВЛЕНИЯ В ВИДЕ КОРОБЧАТЫХ ДИАГРАММ
26
27 # Для обработки малого числа столбцов:
28
29 # Тут необходимо перечислить столбцы, которые требуется использовать:
30 # column_headers_show_box = [
31 #     "Название первого столбца",
32 #     "Название второго столбца"
33 # ]
34
35 # Для обработки большого числа столбцов:
36
37 # column_headers_show_box = df_data.columns.tolist()
38 # Тут можно перечислить столбцы, которые не требуется использовать при
39 # построении гистограмм:
40 # data_remove = [
41 #     "Название первого столбца",
42 #     "Название второго столбца"
43 # ]
44 # for i in data_remove:
45 #     column_headers_show_hist.remove(i)
46
47 column_headers_show_box = ['5. Средний доход за месяц']
48
49 # ВЫБОР СТОЛБЦОВ ДЛЯ ПРЕДСТАВЛЕНИЯ В ВИДЕ ТОЧЕЧНЫХ ДИАГРАММ
50
51 # Для обработки малого числа столбцов:
52
53 # Тут необходимо перечислить столбцы, которые требуется использовать:
54 # column_headers_show_dot = [
55 #     "Название первого столбца",
56 #     "Название второго столбца"
57 # ]
58
59 # Для обработки большого числа столбцов:
60
61 # column_headers_show_dot = df_data.columns.tolist()
62 # Тут можно перечислить столбцы, которые не требуется использовать при
63 # построении гистограмм:
64 # data_remove = [
65 #     "Название первого столбца",
66 #     "Название второго столбца"
67 # ]
68 # for i in data_remove:
69 #     column_headers_show_hist.remove(i)
```

```

70
71 column_headers_dot = df_data.columns.tolist()
72 data_remove = [
73     '1. Пол',
74     '2. Возраст',
75     '4. Наличие брака',
76     '6. Членство в молодёжной организации'
77 ]
78 for i in data_remove:
79     column_headers_dot.remove(i)
80

```

#### ▼ Гистограммы

Гистограмма является способом отображения данных, в ходе которого отображается координатная плоскость с прямоугольниками, характеризующими количество нахождений всех представленных в выборке значений рассматриваемого признака.

Горизонтальная ось отображает представленные в выборке значения признака, а вертикальная ось отображает количество нахождений значений признака. Для отображения на гистограмме значения признака X делятся на равные интервалы. На координатной плоскости изображаются прямоугольники, чья длина характеризует количество значений признака в указанном интервале.

Для отображения гистограмм используется функция `histogram()`.

```

[13] 1 def histogram(table: pd.DataFrame, series: list, path_to_save: str = ''):
2
3     """
4         Построение гистограммы распределения.
5
6         Аргументы функции:
7
8             table - таблица, содержащая столбец с данными,
9
10            series - список названий столбцов с данными,
11
12            path_to_save - по ум. = '', путь для сохранения файлов.
13        """
14
15    # Создание папки для изображений:
16    if path_to_save != '':
17
18        # Нумерация построенных гистограмм в сохранённых файлах:
19        hist_number = 0
20
21        # Создание подпапки:
22        path_to_save = path_to_save + '/hist'
23
24        output_dir = pathlib.Path(path_to_save)
25
26        # Проверка существования папки:
27        if not output_dir.exists():
28
29            # Создание папки для изображений:
30            output_dir.mkdir(parents=True, exist_ok=True)
31
32    # Перебор всех выбранных столбцов:
33    for s in series:
34
35        # Выделение координатных осей:
36        fig, ax = plt.subplots()
37
38        # Добавление сетки:
39        ax.grid(True)
40
41        # Помещение данных столбца таблицы в список:
42        data_list = table[s].tolist()
43
44        # Количество интервалов на гистограмме:
45        # Тут можно указать желающее количество интервалов, на которые будут
46        # делиться значения признака (т. е. количество столбцов в гистограмме),
47        # например, 5.
48        # При этом будет необходимо закомментировать последние 3 строки кода.
49        # Это может быть полезно при желании сделать столбцы гистограммы шире,
50        # избежать пустого пространства между столбцами и т. п.
51        # bins = 5

```

```

52     bins = table[s].nunique()
53     # Установка предела количества прямоугольников на гистограмме:
54     if bins > 30:
55         bins = 30
56
57     # Построение гистограммы:
58     plt.hist(data_list, bins, histtype='step', linewidth=2, label=s)
59
60     # Добавление легенды:
61     plt.legend()
62
63     # Добавление подписей к осям и заголовка:
64     plt.xlabel('Значение')
65     plt.ylabel('Частота')
66     plt.title('Гистограмма')
67
68     # Сохранение изображения:
69     if path_to_save != '':
70
71         # Установка имени файла и пути к нему:
72         output_file = output_dir / ('hist_' + str(hist_number) + '.png')
73
74         # Сохранение файла:
75         plt.savefig(output_file)
76
77         # Изменение значения количества построенных изображений:
78         hist_number = hist_number + 1
79
80     # Отображение гистограммы:
81     plt.show()

```

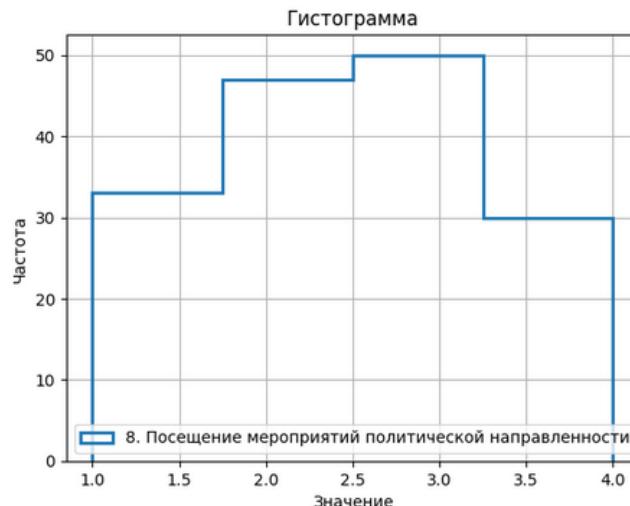
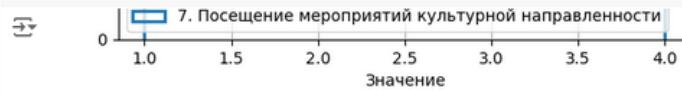
82

Построение гистограмм.

```

✓ [14] 1 # Создание подпапки для сохранения файлов (при необходимости).
2 if files_storage != '':
3     path = files_storage + 'image/raw'
4 else:
5     path = ''
6
7 # Построение гистограмм для выбранных данных:
8 histogram(
9     table=df_data,
10    series=column_headers_show_hist,
11    path_to_save=path
12 )
13

```



#### ▼ Диаграммы размаха

Диаграмма размаха (коробчатая диаграмма, ящик с усами) является способом отображения данных, который позволяет

изобразить основные данные о распределении признака с помощью построения на координатной плоскости прямоугольника с двумя отрезками, исходящими из противоположных сторон прямоугольника. На оси, вдоль которой расположен прямоугольник и отрезки, изображены значения признака.

Стороны прямоугольника, из которых исходят отрезки, расположены на уровне значений, указывающих на границы между первым и вторым квартилями и третьим и четвёртым квартилями значений признака. Черта внутри прямоугольника указывает на расположение медианы значений признака. Концы линий, выходящих из прямоугольника влево и вправо, указывают на наблюдаемые максимум и минимум значений признака X, без учёта выбросов. Возможные кружки за пределами отрезков указывают на выбросы (редкие и сильно отличающиеся значения).

Для построения диаграмм размаха используется функция `box_diagram()`.

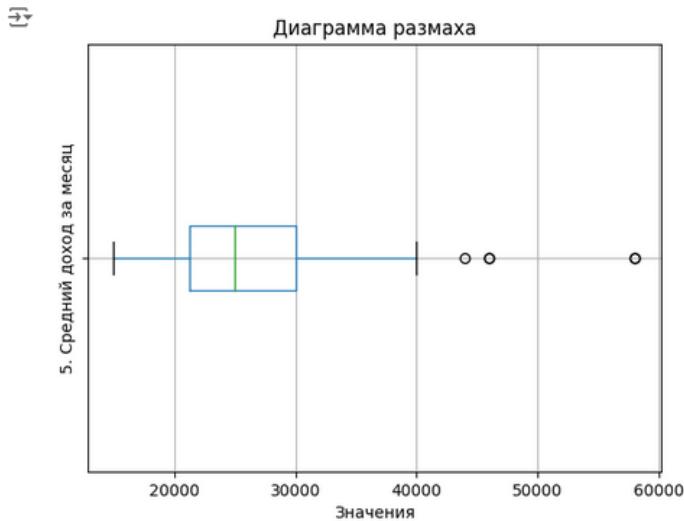
```
[15] 1 def box_diagram(table: pd.DataFrame, series: list, path_to_save: str = ''):  
2  
3     """  
4     Построение коробчатой диаграммы (диаграммы размаха, ящика с усами).  
5  
6     Аргументы функции:  
7  
8     table - таблица, содержащая столбец с данными,  
9  
10    series - список названий столбцов с данными,  
11  
12    path_to_save - по ум. = '', путь для сохранения файлов.  
13    """  
14  
15    # Создание папки для изображений:  
16    if path_to_save != '':  
17  
18        # Нумерация построенных изображений:  
19        box_number = 0  
20  
21        # Создание подпапки:  
22        path_to_save = path_to_save + '/box'  
23  
24        output_dir = pathlib.Path(path_to_save)  
25  
26        # Проверка существования папки:  
27        if not output_dir.exists():  
28  
29            # Создание папки для изображений:  
30            output_dir.mkdir(parents=True, exist_ok=True)  
31  
32        # Перебор всех выбранных столбцов:  
33        for s in series:  
34  
35            # Построение горизонтальной коробчатой диаграммы. При желании изобразить  
36            # диаграмму вертикально, последний параметр надо изменить с False на True:  
37  
38            table.boxplot(s, vert=False)  
39  
40            # Добавление подписей к осям и заголовка:  
41            plt.title('Диаграмма размаха')  
42            plt.xlabel('Значения')  
43  
44            # Поворот подписи слева от диаграммы для экономии места:  
45            plt.yticks(rotation=90, va='center')  
46  
47            # Сохранение изображения:  
48            if path_to_save != '':  
49  
50                # Установка имени файла и пути к нему:  
51                output_file = output_dir / ('box_' + str(box_number) + '.png')  
52  
53                # Сохранение файла:  
54                plt.savefig(output_file)  
55  
56                # Изменение значения количества построенных изображений:  
57                box_number = box_number + 1  
58  
59                # Отображение диаграммы:  
60                plt.show()
```

Построение диаграмм размаха.

```

[16] 1 # Создание подпапки для сохранения файлов (при необходимости).
0 сек.
2 if files_storage != '':
3     path = files_storage + 'image/raw'
4 else:
5     path = ''
6
7 # Построение коробчатых диаграмм для выбранных данных:
8 box_diagram(
9     table=df_data,
10    series=column_headers_show_box,
11    path_to_save=path
12 )
13

```



#### ▼ Точечные диаграммы

Точечная диаграмма является способом отображения данных, в ходе которого можно наглядно наблюдать зависимость значений одного признака от значений другого признака.

На координатной плоскости на горизонтальной и вертикальной осях отображаются значения двух признаков, а точки на плоскости указывают на наблюдения, имеющие значения каждого из признаков, соответствующие шкалам на координатных осях.

Для построения точечной диаграммы используется функция scatter\_diagram().

```

[17] 1 def scatter_diagram(table: pd.DataFrame, series: list, path_to_save: str = ''):
2
3     """
4         Построение точечной диаграммы.
5
6         Аргументы функции:
7
8             table - таблица, содержащая столбец с данными,
9
10            series - список названий столбцов с данными,
11
12            path_to_save - по ум. = '', путь для сохранения файлов.
13        """
14
15         # Создание папки для изображений:
16         if path_to_save != '':
17
18             # Нумерация построенных изображений:
19             dot_number = 0
20
21             # Создание подпапки:
22             path_to_save = path_to_save + '/dot'
23
24             output_dir = pathlib.Path(path_to_save)
25
26             # Проверка существования папки:
27             if not output_dir.exists():

```

```

28     # Создание папки для изображений:
29     output_dir.mkdir(parents=True, exist_ok=True)
30
31     # Перебор всех пар указанных столбцов без повторений:
32     for i in column_headers_show_dot:
33         for j in column_headers_show_dot:
34             if i >= j: continue
35
36             # Подготовка значений для построения диаграммы:
37             data_list_1 = table[i].tolist()
38             data_list_2 = table[j].tolist()
39
40             # Построение точечной диаграммы:
41             plt.scatter(data_list_1, data_list_2)
42
43             # Добавление подписей к осям и заголовка:
44             plt.xlabel(i)
45             plt.ylabel(j)
46             plt.title('Точечная диаграмма')
47
48             # Сохранение изображения:
49             if path_to_save != '':
50
51                 # Установка имени файла и пути к нему:
52                 output_file = output_dir / ('dot_' + str(dot_number) + '.png')
53
54                 # Сохранение файла:
55                 plt.savefig(output_file)
56
57                 # Изменение значения количества построенных изображений:
58                 dot_number = dot_number + 1
59
60             # Отображение диаграммы:
61             plt.show()
62
63

```

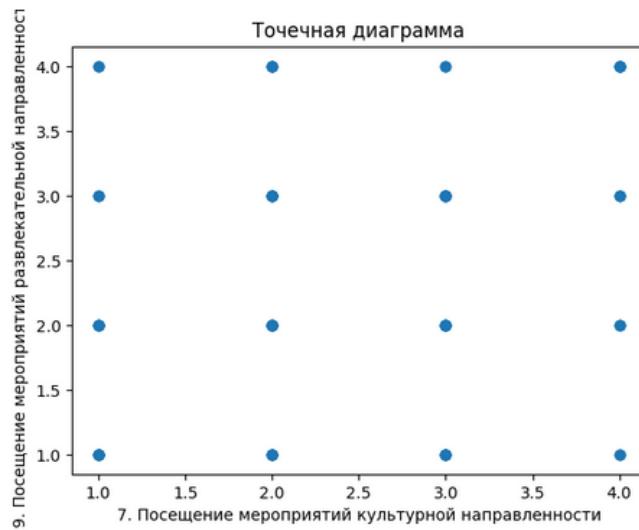
Построение точечной диаграммы.

```

[ ] 1  # Создание подпапки для сохранения файлов (при необходимости).
2  if files_storage != '':
3      path = files_storage + 'image/raw'
4
4 else:
5      path = ''
6
7 # Построение точечных диаграмм для выбранных данных:
8 scatter_diagram(
9     table=df_data,
10    series=column_headers_show_dot,
11    path_to_save=path
12 )
13

```





#### ▼ Удаление пустых значений

Пустые значения не пригодны для вычислений, и могут оказывать воздействие на вычисление различных метрик и проведение анализа данных. Поэтому пустые значения обычно удаляются из таблицы.

Выбор столбцов для удаления пустых значений. Примеры выбора столбцов приведены ниже в комментариях.

#### РЕДАКТИРОВАНИЕ КОДА

```
✓ [19] 1 # Для обработки малого числа столбцов:
2
3 # Тут необходимо перечислить столбцы, которые требуется использовать:
4 # column_headers_void = [
5 #     "Название первого столбца",
6 #     "Название второго столбца"
7 # ]
8
9 # Для обработки большого числа столбцов:
10
11 # column_headers_void = df_data.columns.tolist()
12 # Тут можно перечислить столбцы, которые не требуется использовать при
13 # построении гистограмм:
14 # data_remove = [
15 #     "Название первого столбца",
16 #     "Название второго столбца"
17 # ]
18 # for i in data_remove:
19 #     column_headers_s.remove(i)
20
21 column_headers_void = df_data.columns.tolist()
```

Для удаления пустых значений используется функция void\_killer().

```
✓ [20] 1 def void_killer(
2         table: pd.DataFrame,
3         series: list,
4         mod: str = 'delete'
5         ) -> pd.DataFrame:
6
7     """
8     Удаление пустых значений.
9
10    Аргументы функции:
11
12    table - таблица, содержащая столбец с данными,
13
14    series - список названий столбцов с данными,
15
16    mod - по ум. 'delete', так же возможно 'mean', 'median'.
17
```

```

18     Возвращает pd.DataFrame.
19     """
20
21     for s in series:
22
23         # Получение индексов строк с пропусками:
24         # void_r = table[s][table[s].isnull().any(axis=1)].index.tolist()
25         void_i = table[s].loc[table[s].isna()].index
26
27         # Удаление строк с пустыми значениями:
28         if mod == 'delete':
29
30             for i in void_i:
31                 print(f'Пустое значение "{s} [{i}]" удаляется со строкой.')
32
33             table = table.drop(void_i)
34
35         # Замена пустых значений медианой:
36         if mod == 'mean':
37
38             for i in void_i:
39
40                 print(f'Пустое значение "{s} [{i}]" \
41 заменяется на среднее арифметическое столбца.')
42
43             # Преобразование данных в массив numpy:
44             np_data = table[s].to_numpy()
45
46             # Удаление значений, равных NaN:
47             np_data = np_data[~np.isnan(np_data)]
48
49             # Вычисление среднего арифметического:
50             mean = np_data.mean()
51
52             table.loc[void_i, s] = mean
53
54         if mod == 'median':
55
56             for i in void_i:
57                 print(f'Пустое значение "{s} [{i}]" заменяется на медиану столбца.')
58
59             # Преобразование данных в массив numpy:
60             np_data = table[s].to_numpy()
61
62             # Удаление значений, равных NaN:
63             np_data = np_data[~np.isnan(np_data)]
64
65             # Вычисление медианы:
66             median = np.median(np_data)
67
68             table.loc[void_i, s] = median
69
70     return table
71

```

Удаление пустых значений.

#### РЕДАКТИРОВАНИЕ КОДА

```

[21] 1  # Удаление пустых значений.
2  # Для удаления строк, содержащих пустые значения, установить значение аргумента
3  # mod='delete'.
4  # Для замены пустых значений на среднее арифметическое, установить значение
5  # аргумента mod='mode'.
6  # Для замены пустых значений на медиану, установить значение
7  # аргумента mod='median'.
8
9  df_data = void_killer(df_data, column_headers_void, mod='delete')
10

```

Пустое значение "5. Средний доход за месяц[0]" удаляется со строкой.  
Пустое значение "5. Средний доход за месяц[1]" удаляется со строкой.  
Пустое значение "5. Средний доход за месяц[2]" удаляется со строкой.  
Пустое значение "5. Средний доход за месяц[7]" удаляется со строкой.  
Пустое значение "5. Средний доход за месяц[12]" удаляется со строкой.  
Пустое значение "5. Средний доход за месяц[13]" удаляется со строкой.

Просмотр данных после удаления пустых значений.

#### РЕДАКТИРОВАНИЕ КОДА

```
[22] 1 # Вывод общей информации о таблице:  
2 print(df_data.info(), end = '\n\n')  
3  
4 # Вывод первых 5 строк таблицы:  
5 display(df_data.head(5))  
6  
7 # Вывод последних 5 строк таблицы:  
8 # display(df_data_ns.tail(5))  
9  
10 # Вывод всей таблицы:  
11 # display(df_data_ns)  
12  
13 # Вывод среза таблицы, например, с 10 по 20 строки:  
14 # df_data_ns[10:20]  
15
```

```
→ <class 'pandas.core.frame.DataFrame'>  
Index: 106 entries, 3 to 157  
Data columns (total 9 columns):  
 # Column Non-Null Count Dtype  
--- ---  
 0 1. Пол 106 non-null int64  
 1 2. Возраст 106 non-null int64  
 2 3. Образование 106 non-null int64  
 3 4. Наличие брака 106 non-null int64  
 4 5. Средний доход за месяц 106 non-null float64  
 5 6. Членство в молодёжной организации 106 non-null int64  
 6 7. Посещение мероприятий культурной направленности 106 non-null int64  
 7 8. Посещение мероприятий политической направленности 106 non-null int64  
 8 9. Посещение мероприятий развлекательной направленности 106 non-null int64  
dtypes: float64(1), int64(8)  
memory usage: 8.3 KB  
None
```

1. Пол	2. Возраст	3. Образование	4. Наличие брака	5. Средний доход за месяц	6. Членство в молодёжной организации	7. Посещение мероприятий культурной направленности	8. Посещение мероприятий политической направленности	9. Посещение мероприятий развлекательной направленности
3	1	34	3	1	32000.0	0	3	4
4	1	28	3	0	33000.0	0	4	3
5	1	25	3	0	17000.0	0	3	4
6	1	25	3	0	35000.0	0	4	2
8	1	34	2	1	58000.0	1	2	3

#### Удаление выбросов

Выброс (экстремальное значение, аномальное значение) – редкое значение характеристики, значительно отличающееся от других значений.

Выбросы могут быть вызваны как ошибкой статистического сбора данных (например, неточность приборов измерения), так и значительно отличающимися крайне редкими значениями, которые реально встречаются на практике.

Удаляются выбросы потому, что оказывают значительное воздействие на различные метрики и результаты анализа данных, которое не пропорционально относительной частоте их наблюдения.

Выбор столбцов для удаления выбросов. Примеры выбора столбцов приведены ниже в комментариях.

#### РЕДАКТИРОВАНИЕ КОДА

```
[23] 1 # Для обработки малого числа столбцов:  
2  
3 # Тут необходимо перечислить столбцы, которые требуется использовать:  
4 # column_headers_trash_clean = [  
5 #     "Название первого столбца",  
6 #     "Название второго столбца"  
7 # ]
```

```

8
9 # Для обработки большого числа столбцов:
10
11 # column_headers_trash_clean = df_data.columns.tolist()
12 # Тут необходимо перечислить столбцы, которые не требуется использовать:
13 # data_remove = [
14 #     "Название первого столбца",
15 #     "Название второго столбца"
16 # ]
17 # for i in data_remove:
18 #     column_headers_trash_clean.remove(i)
19
20 column_headers_trash_clean = ["5. Средний доход за месяц"]
21

```

Для удаления выбросов используется функция filter\_outliers().

```

✓ [24] 1 def filter_outliers(
2         table: pd.DataFrame,
3         series: list,
4         mod: str = 'mean'
5         ) -> pd.DataFrame:
6
7     """
8     Удаление выбросов.
9
10    Аргументы функции:
11
12    table - таблица, содержащая столбец с данными,
13
14    series - список названий столбцов с данными,
15
16    mod - по ум. 'mean', так же возможно 'square', 'delete'.
17
18    Возвращает pd.DataFrame.
19    """
20
21    for s in series:
22
23        # ОПРЕДЕЛЕНИЕ ГРАНИЦ ВЫБРОСОВ
24
25        # Первый и третий квартили:
26        q1 = table[s].quantile(0.25)
27        q3 = table[s].quantile(0.75)
28
29        # Межквартальный размах:
30        iqr = q3 - q1
31
32        # Границы выбросов:
33        lower_bound = q1 - 1.5 * iqr
34        upper_bound = q3 + 1.5 * iqr
35
36        # Определение индексов строк с выбросами:
37        i_outliers = table[
38            (table[s] < lower_bound) | (table[s] > upper_bound)
39            ].index
40
41        # УДАЛЕНИЕ ВЫБРОСОВ
42
43        # Проверка, надо ли удалять выбросы в рассматриваемом столбце s:
44        if not i_outliers.empty:
45
46            # Замена значений выбросов на среднее арифметическое значений
47            # столбца:
48            if mod == 'mean':
49                mean = round(table[s].mean(), 3)
50                for i in i_outliers:
51                    print(f'Заменяется значение "{s}[{i}]" = {table.loc[i, s]} \
52 на среднее арифметическое значений столбца, равное {mean}.')
53                    table.loc[i, s] = mean
54
55            # Применение преобразования квадратного корня значений столбца:
56            if mod == 'square':
57                print(f'Применение преобразования квадратного корня к столбцу \
58 "{s}" из-за встреченных значений:')
59                for i in i_outliers:
60                    print(f'"({s})[{i}]" = {table.loc[i, s]}')

```

```

61     table[s] = np.sqrt(table[s])
62
63     # Удаление строк с выбросами:
64     if mod == 'delete':
65         for i in i_outliers:
66             print(f'Удаление строки "{i}" из-за встреченного \
67 "{table.loc[i, s]}" в столбце "{s}"')
68             table = table[
69                 (table[s] >= lower_bound) & (table[s] <= upper_bound)
70             ]
71
72     # Защита от столбцов с одинаковым содержанием ячеек
73     # после удаления выбросов:
74
75     # Список столбцов для удаления:
76     no_sence_columns_to_drop = []
77
78     # Проверка каждого столбца на наличие одинаковых значений:
79     for column in table.columns:
80         if table[column].nunique() == 1:
81             print(f'Столбец "{column}" после удаления выбросов \
82 содержит одинаковые значения и удаляется.')
83             no_sence_columns_to_drop.append(column)
84
85     # Удаляем столбцы с одинаковыми значениями:
86     table = table.drop(columns=no_sence_columns_to_drop)
87
88     return table
89

```

Применение функции filter\_outliers() для нахождения и обработки найденных выбросов. Возможные варианты обработки выбросов:

- Замена значений выбросов на среднее арифметическое значений столбца. Позволяет избавиться от выбросов, при этом отразив тенденцию, на которую выбросы, возможно, указывают.
- Применение преобразования квадратного корня заменяет все значения в столбце на квадратные корни этих значений. Позволяет избавиться от выбросов, сводя их влияние на выборку к минимуму.
- Удалить строки таблицы с выбросами. При этом варианте выбросы не будут никак влиять на всю выборку в целом.

Решение о том, какой вариант использовать, остаётся за исследователем, и зависит от обрабатываемых значений, задач исследования и т. п.

Удаление выбросов.

#### РЕДАКТИРОВАНИЕ КОДА

```

✓ 0 [25] 1  # Для замены выбросов на среднее арифметическое значение столбца установить
    2  # значение аргумента mod='mean'.
    3
    4  # Для применения преобразования квадратного корня установить значение аргумента
    5  # mod='sqrt'.
    6
    7  # Для удаления строк с выбросами установить значение аргумента mod='delete'.
    8  # При этом могут возникнуть столбцы с одинаковыми значениями, которые будут
    9  # удалены, о чём будет выведено сообщение.
10
11 df_data = filter_outliers(df_data, column_headers_trash_clean, mod='mean')
12

```

Заменя значения "5. Средний доход за месяц[8]" = 58000.0 на среднее арифметическое значение столбца, равное 26584.9  
 Замена значения "5. Средний доход за месяц[9]" = 46000.0 на среднее арифметическое значение столбца, равное 26584.9  
 Замена значения "5. Средний доход за месяц[14]" = 46000.0 на среднее арифметическое значение столбца, равное 26584.  
 Замена значения "5. Средний доход за месяц[77]" = 44000.0 на среднее арифметическое значение столбца, равное 26584.  
 Замена значения "5. Средний доход за месяц[80]" = 58000.0 на среднее арифметическое значение столбца, равное 26584.

Просмотр данных после удаления выбросов.

#### РЕДАКТИРОВАНИЕ КОДА

```

✓ 0 [26] 1  # Вывод общей информации о таблице:
    2  print(df_data.info(), end = '\n\n')
    3
    4  # Вывод первых 5 строк таблицы:
    5  display(df_data.head(5))

```

```

6
7 # Вывод последних 5 строк таблицы:
8 # display(df_data.tail(5))
9
10 # Вывод всей таблицы:
11 # display(df_data)
12
13 # Вывод среза таблицы, например, с 10 по 20 строки:
14 # df_data[10:20]

```

→ <class 'pandas.core.frame.DataFrame'>  
Index: 106 entries, 3 to 157  
Data columns (total 9 columns):  
# Column Non-Null Count Dtype  
-- --  
0 1. Пол 106 non-null int64  
1 2. Возраст 106 non-null int64  
2 3. Образование 106 non-null int64  
3 4. Наличие брака 106 non-null int64  
4 5. Средний доход за месяц 106 non-null float64  
5 6. Членство в молодёжной организации 106 non-null int64  
6 7. Посещение мероприятий культурной направленности 106 non-null int64  
7 8. Посещение мероприятий политической направленности 106 non-null int64  
8 9. Посещение мероприятий развлекательной направленности 106 non-null int64  
dtypes: float64(1), int64(8)  
memory usage: 12.4 KB  
None

	1. Пол	2. Возраст	3. Образование	4. Наличие брака	5. Средний доход за месяц	6. Членство в молодёжной организации	7. Посещение мероприятий культурной направленности	8. Посещение мероприятий политической направленности	9. Посещение мероприятий развлекательной направленности
3	1	34	3	1	32000.000	0	3	4	2
4	1	28	3	0	33000.000	0	4	3	4
5	1	25	3	0	17000.000	0	3	4	4
6	1	25	3	0	35000.000	0	4	4	2
8	1	34	2	1	26584.906	1	2	3	2

#### ✓ Вариационные ряды данных

Вариационные ряды позволяют увидеть частоту значений и относительную частоту значений некоторого признака. Вариационные ряды бывают интервальными и дискретными.

Частота значения - то, сколько раз некоторое значение встречалось среди значений признака.

Относительная частота значения - результат деления частоты значения на количество значений признака в дискретном вариационном ряде, и на количество интервалов значений признака в интервальном вариационном ряде.

Дискретный вариационный ряд содержит каждое значение признака. Он хорошо подходит для признаков с номинальными или ранговыми шкалами значений, при которых существует небольшое количество значений.

Интервальный вариационный ряд группирует все значения по интервалам. Он хорошо подходит для признаков с метрической шкалой значений, при которых существует большое количество значений.

#### РЕДАКТИРОВАНИЕ КОДА

```

✓ [27] 1 # ВЫБОР СТОЛБЦОВ ДЛЯ ПОЛУЧЕНИЯ ДИСКРЕТНЫХ ВАРИАЦИОННЫХ РЯДОВ ЗНАЧЕНИЙ
2
3 # Для обработки малого числа столбцов:
4
5 # Тут необходимо перечислить столбцы, которые требуется использовать:
6 # column_headers_show_table_frequency_describe = [
7 #     "Название первого столбца",
8 #     "Название второго столбца"
9 # ]
10
11 # Для обработки большого числа столбцов:
12
13 # column_headers_show_table_frequency_describe = df_data.columns.tolist()
14 # Тут можно перечислить столбцы, которые не требуется использовать при
15 # построении гистограмм:
16 # data_remove = [
17 #     "Название первого столбца",
18 #     "Название второго столбца"
19 # ]
20 # for i in data_remove:
21 #     column_headers_show_table_frequency_describe.remove(i)

```

```

22
23 column_headers_show_table_frequency_describe = df_data.columns.tolist()
24 data_remove = ["5. Средний доход за месяц"]
25 for i in data_remove:
26     column_headers_show_table_frequency_describe.remove(i)
27
28 # ВЫБОР СТОЛБЦОВ ДЛЯ ПОЛУЧЕНИЯ ИНТЕРВАЛЬНЫХ ВАРИАЦИОННЫХ РЯДОВ ЗНАЧЕНИЙ
29
30 # Для обработки малого числа столбцов:
31
32 # Тут необходимо перечислить столбцы, которые требуется использовать:
33 # column_headers_show_table_frequency_interval = [
34 #     "Название первого столбца",
35 #     "Название второго столбца"
36 # ]
37
38 # Для обработки большого числа столбцов:
39
40 # column_headers_show_table_frequency_interval = df_data.columns.tolist()
41 # Тут можно перечислить столбцы, которые не требуется использовать при
42 # построении гистограмм:
43 # data_remove = [
44 #     "Название первого столбца",
45 #     "Название второго столбца"
46 # ]
47 # for i in data_remove:
48 #     column_headers_show_table_frequency_interval.remove(i)
49
50 column_headers_show_table_frequency_interval = ["5. Средний доход за месяц"]
51

```

Для построения таблиц вариационных рядов используется функция frequency\_table().

```

✓ [28] 1 def frequency_table(
2     table: pd.DataFrame,
3     series: str,
4     mod: str = 'interval',
5     path_to_save: str = ''
6     ):
7     """
8         Функция создаёт таблицу со значениями вариационного ряда.
9
10    Аргументы функции:
11
12        table - таблица, содержащая столбец с данными,
13
14        series - список названий столбцов с данными,
15
16        mod - по ум. = 'interval', так же возможно 'describe'.
17        При 'interval' строится вариационный ряд.
18        При 'describe' строится дискретный ряд.
19
20        path_to_save - по ум. = '', путь для сохранения файлов.
21
22
23 # Создание папки для изображений:
24 if path_to_save != '':
25
26     # Настройка подпапки:
27     path_to_save = path_to_save + '/' + mod
28
29     # Нумерация построенных изображений:
30     freq_number = 0
31
32     output_dir = pathlib.Path(path_to_save)
33
34     # Проверка существования папки:
35     if not output_dir.exists():
36
37         # Создание папки для изображений:
38         output_dir.mkdir(parents=True, exist_ok=True)
39
40     # Перебор всех выбранных столбцов:
41     for s in series:
42
43         # Дискретный вариационный ряд:
44         if mod == 'describe':

```

```

45
46     # Получение уникальных значений столбца и их сортировка:
47     unique_values = sorted(table[s].unique().tolist())
48
49     # Получение частот значений столбца:
50     frequency = [table[table[s] == x].shape[0] for x in unique_values]
51
52     # Получение относительных частот столбца:
53     relative_frequency = [f / table.shape[0] for f in frequency]
54
55     # Получение процентных значений:
56     percent_frequency = [str(round(f * 100, 0)) + '%' \
57                           for f in relative_frequency]
58
59     # Создание таблицы с результатами:
60     table_frequency = pd.DataFrame({
61         "Варианты x_i": unique_values,
62         "частоты n_i": frequency,
63         "Относительные частоты w_i": relative_frequency,
64         "Процент, %": percent_frequency
65     })
66
67     print(f'Вариационный ряд значений столбца "{s}":')
68     display(table_frequency)
69
70     # Интервальный вариационный ряд:
71     if mod == 'interval':
72
73         # Получение уникальных значений столбца и их сортировка:
74         unique_values = sorted(table[s].unique())
75
76         # Вычисление минимального и максимального значений:
77         min_value = min(unique_values)
78         max_value = max(unique_values)
79
80         # Вычисление количества интервалов:
81         if len(unique_values) > 20:
82             num_intervals = int(np.sqrt(len(unique_values)))
83         else:
84             num_intervals = len(unique_values)
85
86         # Вычисление ширины интервала:
87         interval_width = (max_value - min_value) / num_intervals
88
89         # Создание интервалов в списке:
90         intervals = [
91             min_value + i * interval_width, \
92             min_value + (i + 1) * interval_width
93             ] for i in range(num_intervals)]
94         # Редактирование последнего интервала для включения в него
95         # самых больших значений:
96         intervals[-1][-1] = intervals[-1][-1] + 0.0001
97
98         # Вычисление значений для каждого интервала:
99         # Перебор всех интервалов:
100        result = []
101        interval_labels = []
102        frequency = []
103        relative_frequency = []
104
105        for i in intervals:
106
107            # Добавление интервала:
108            interval_labels.append(f'{round(i[0], 3)} - {round(i[1], 3)}')
109
110            # Вычисление частоты значений в интервале:
111            count = len(table[(table[s] >= i[0]) & (table[s] < i[1])])
112            frequency.append(count)
113
114            # Вычисление относительной частоты значений в интервале:
115            relative_freq = count / len(table)
116            relative_frequency.append(relative_freq)
117
118            # Получение процентных значений:
119            percent_frequency = [str(round(f * 100, 0)) + '%' \
120                                  for f in relative_frequency]
121
122        # Создание таблицы с результатами:

```

```

123     table_frequency = pd.DataFrame({
124         "Интервал": interval_labels,
125         "Частоты n_i": frequency,
126         "Относительные частоты w_i": relative_frequency,
127         "Процент, %": percent_frequency
128     }
129
130     print(f'Интервальный ряд значений столбца "{s}"')
131     display(table_frequency)
132
133     # Сохранение таблицы в файл:
134     if path_to_save != '':
135
136         # Изменение значения количества построенных таблиц:
137         freq_number = freq_number + 1
138
139         # Сохранение таблицы в текущую директорию (папку):
140         if mod == 'interval':
141             file_name = \
142                 '/result_table_frequency_interval_' + str(freq_number) + '.csv'
143         if mod == 'describe':
144             file_name = \
145                 '/result_table_frequency_describe_' + str(freq_number) + '.csv'
146
147         table_frequency.to_csv(
148             path_to_save + file_name,
149             index=False
150         )
151

```

#### РЕДАКТИРОВАНИЕ КОДА

```

[29] 1 # Создание подпапки для сохранения файлов (при необходимости).
2 if files_storage != '':
3     path = files_storage + 'frequency/'
4 else:
5     path = ''
6
7 # Создание таблиц с интервальными вариационными рядами:
8 frequency_table(
9     table=df_data,
10    series=column_headers_show_table_frequency_describe,
11    mod='describe',
12    path_to_save=path)
13
14 # Создание таблиц с интервальными вариационными рядами:
15 frequency_table(
16     table=df_data,
17    series=column_headers_show_table_frequency_interval,
18    mod='interval',
19    path_to_save=path)

```

→ Вариационный ряд значений столбца "1. Пол":

	Варианты x_i	Частоты n_i	Относительные частоты w_i	Процент, %
0	1	57	0.537736	54.0%
1	2	49	0.462264	46.0%

Вариационный ряд значений столбца "2. Возраст":

	Варианты x_i	Частоты n_i	Относительные частоты w_i	Процент, %
0	19	4	0.037736	4.0%
1	22	25	0.235849	24.0%
2	25	21	0.198113	20.0%
3	28	15	0.141509	14.0%
4	31	15	0.141509	14.0%
5	34	26	0.245283	25.0%

Вариационный ряд значений столбца "3. Образование":

	Варианты x_i	Частоты n_i	Относительные частоты w_i	Процент, %
0	1	22	0.207547	21.0%
1	2	45	0.424528	42.0%
2	3	39	0.367925	37.0%

Вариационный ряд значений столбца "4. Наличие брака":

Варианты x_i Частоты n_i Относительные частоты w_i Процент, %				
0	0	52	0.490566	49.0%
1	1	54	0.509434	51.0%
Вариационный ряд значений столбца "6. Членство в молодёжной организации":				
Варианты x_i Частоты n_i Относительные частоты w_i Процент, %				
0	0	72	0.679245	68.0%
1	1	34	0.320755	32.0%
Вариационный ряд значений столбца "7. Посещение мероприятий культурной направленности":				
Варианты x_i Частоты n_i Относительные частоты w_i Процент, %				
0	1	20	0.188679	19.0%
1	2	34	0.320755	32.0%
2	3	36	0.339623	34.0%
3	4	16	0.150943	15.0%
Вариационный ряд значений столбца "8. Посещение мероприятий политической направленности":				
Варианты x_i Частоты n_i Относительные частоты w_i Процент, %				
0	1	25	0.235849	24.0%
1	2	24	0.235849	24.0%

#### ▼ Нормализация и стандартизация данных

Для того, чтобы полученные значения можно было сравнивать между собой, их нужно привести к некой общей шкале.

Для категориальных данных применяется стандартизация, для числовых - нормализация.

Категориальные данные (красный, зелёный, синий), в отличии от числовых (большой, средний, малый), не имеют количественного выражения.

Выбор столбцов для нормализации и (или) стандартизации.

#### РЕДАКТИРОВАНИЕ КОДА

```
[30] 1 # ВЫБОР СТОЛБЦОВ ДЛЯ НОРМАЛИЗАЦИИ
2
3 # Для обработки малого числа столбцов:
4
5 # Тут необходимо перечислить столбцы, которые требуется использовать:
6 # column_headers_n = [
7 #     "Название первого столбца",
8 #     "Название второго столбца"
9 # ]
10
11 # Для обработки большого числа столбцов:
12
13 # column_headers_n = df_data.columns.tolist()
14 # Тут можно перечислить столбцы, которые не требуется использовать при
15 # построении гистограмм:
16 # data_remove = [
17 #     "Название первого столбца",
18 #     "Название второго столбца"
19 # ]
20 # for i in data_remove:
21 #     column_headers_n.remove(i)
22
23 column_headers_n = df_data.columns.tolist()
24
25 # ВЫБОР СТОЛБЦОВ ДЛЯ СТАНДАРТИЗАЦИИ
26
27 # Для обработки малого числа столбцов:
28
29 # Тут необходимо перечислить столбцы, которые требуется использовать:
30 # column_headers_s = [
31 #     "Название первого столбца",
32 #     "Название второго столбца"
33 # ]
34
35 # Для обработки большого числа столбцов:
36
37 # column_headers_s = df_data.columns.tolist()
38 # Тут можно перечислить столбцы, которые не требуется использовать при
39 # построении гистограмм:
```

```
40 # data_remove = [
41 #     "Название первого столбца",
42 #     "Название второго столбца"
43 # ]
44 # for i in data_remove:
45 #     column_headers_s.remove(i)
46
47 column_headers_s = []
48
```

Для нормализации данных используется функция normalization().

```
[31] 1 def normalization(table: pd.DataFrame, series: list) -> pd.DataFrame:
2
3     """
4     Нормализация данных.
5
6     Аргументы функции:
7
8     table - таблица, содержащая столбец с данными,
9
10    series - список названий столбцов с данными.
11
12    Возвращает pd.DataFrame.
13    """
14
15    for s in series:
16
17        # Минимальное и максимальное значения:
18        x_min = table[s].min()
19        x_max = table[s].max()
20
21        # Вычисление нормализованного значения:
22        table[s] = (table[s] - x_min) / (x_max - x_min)
23
24    return table
25
```

Для стандартизации используется функция standardization().

```
[32] 1 def standardization(table: pd.DataFrame, series: list) -> pd.DataFrame:
2
3     """
4     Нормализация данных.
5
6     Аргументы функции:
7
8     table - таблица, содержащая столбец с данными,
9
10    series - список названий столбцов с данными.
11
12    Возвращает pd.DataFrame.
13    """
14
15    for s in series:
16
17        # Среднее арифметическое:
18        mean = table[s].mean()
19
20        # Стандартное квадратическое отклонение:
21        std = table[s].std()
22
23        # Вычисление стандартизованного значения:
24        table[s] = (table[s] - mean) / std
25
26    return table
27
```

Для данных, к части из которых (или всем) была применена стандартизация и (или) нормализация, будет использоваться новая таблица df\_data\_ns .

Применение нормализации и (или) стандартизации.

```
[33] 1 # Создание таблицы для нормализованных и стандартизованных данных:
```

```

2 df_data_ns = df_data.copy(deep=True)
3
4 # Нормализация выбранных данных:
5 df_data_ns = normalization(df_data_ns, column_headers_n)
6
7 # Стандартизация выбранных данных:
8 df_data_ns = standardization(df_data_ns, column_headers_s)
9

```

Просмотр данных после нормализации и (или) стандартизации.

```

✓ [34] 1 # Вывод общей информации о таблице:
2 print(df_data_ns.info(), end = '\n\n')
3
4 # Вывод первых 5 строк таблицы:
5 display(df_data_ns.head(5))
6
7 # Вывод последних 5 строк таблицы:
8 # display(df_data_ns.tail(5))
9
10 # Вывод всей таблицы:
11 # display(df_data_ns)
12
13 # Вывод среза таблицы, например, с 10 по 20 строки:
14 # df_data_ns[10:20]
15

```

```

→ <class 'pandas.core.frame.DataFrame'>
Index: 106 entries, 3 to 157
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   1. Пол            106 non-null    float64
 1   2. Возраст         106 non-null    float64
 2   3. Образование     106 non-null    float64
 3   4. Наличие брака    106 non-null    float64
 4   5. Средний доход за месяц  106 non-null    float64
 5   6. Членство в молодёжной организации 106 non-null    float64
 6   7. Посещение мероприятий культурной направленности 106 non-null    float64
 7   8. Посещение мероприятий политической направленности 106 non-null    float64
 8   9. Посещение мероприятий развлекательной направленности 106 non-null    float64
dtypes: float64(9)
memory usage: 12.4 KB
None

```

	1. Пол	2. Возраст	3. Образование	4. Наличие брака	5. Средний доход за месяц	6. Членство в молодёжной организации	7. Посещение мероприятий культурной направленности	8. Посещение мероприятий политической направленности	9. Посещение мероприятий развлекательной направленности
3	0.0	1.0	1.0	1.0	0.680000	0.0	0.666667	1.000000	0.333333
4	0.0	0.6	1.0	0.0	0.720000	0.0	1.000000	0.666667	1.000000
5	0.0	0.4	1.0	0.0	0.080000	0.0	0.666667	1.000000	1.000000
6	0.0	0.4	1.0	0.0	0.800000	0.0	1.000000	1.000000	0.333333
8	0.0	1.0	0.5	1.0	0.463396	1.0	0.333333	0.666667	0.333333

#### ▼ Представление обработанных данных

При представлении обработанных данных можно получить иную информацию, в отличии от представления необработанных данных (например, из-за удаления выбросов и пустых значений).

Выбор столбцов для представления в виде:

- 1) Гистограммы.
- 2) Диаграммы размаха (коробчатой диаграммы).
- 3) Точечной диаграммы.

#### РЕДАКТИРОВАНИЕ КОДА

```

✓ [35] 1 # ВЫБОР СТОЛБЦОВ ДЛЯ ПРЕДСТАВЛЕНИЯ В ВИДЕ ГИСТОГРАММ
2
3 # Для обработки малого числа столбцов:
4
5 # Тут необходимо перечислить столбцы, которые требуется использовать:

```

```

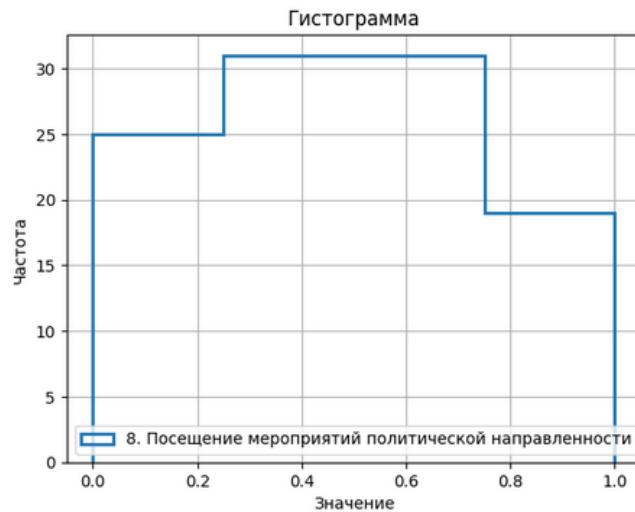
6 # column_headers_show_hist = [
7 #     "Название первого столбца",
8 #     "Название второго столбца"
9 # ]
10
11 # Для обработки большого числа столбцов:
12
13 # column_headers_show_hist = df_data_ns.columns.tolist()
14 # Тут можно перечислить столбцы, которые не требуется использовать при
15 # построении гистограмм:
16 # data_remove = [
17 #     "Название первого столбца",
18 #     "Название второго столбца"
19 # ]
20 # for i in data_remove:
21 #     column_headers_show_hist.remove(i)
22
23 column_headers_show_hist = df_data_ns.columns.tolist()
24
25 # ВЫБОР СТОЛБЦОВ ДЛЯ ПРЕДСТАВЛЕНИЯ В ВИДЕ КОРОБЧАТЫХ ДИАГРАММ
26
27 # Для обработки малого числа столбцов:
28
29 # Тут необходимо перечислить столбцы, которые требуется использовать:
30 # column_headers_show_box = [
31 #     "Название первого столбца",
32 #     "Название второго столбца"
33 # ]
34
35 # Для обработки большого числа столбцов:
36
37 # column_headers_show_box = df_data_ns.columns.tolist()
38 # Тут можно перечислить столбцы, которые не требуется использовать при
39 # построении гистограмм:
40 # data_remove = [
41 #     "Название первого столбца",
42 #     "Название второго столбца"
43 # ]
44 # for i in data_remove:
45 #     column_headers_show_hist.remove(i)
46
47 column_headers_show_box = ['5. Средний доход за месяц']
48
49 # ВЫБОР СТОЛБЦОВ ДЛЯ ПРЕДСТАВЛЕНИЯ В ВИДЕ ТОЧЕЧНЫХ ДИАГРАММ
50
51 # Для обработки малого числа столбцов:
52
53 # Тут необходимо перечислить столбцы, которые требуется использовать:
54 # column_headers_show_dot = [
55 #     "Название первого столбца",
56 #     "Название второго столбца"
57 # ]
58
59 # Для обработки большого числа столбцов:
60
61 # column_headers_show_dot = df_data_ns.columns.tolist()
62 # Тут можно перечислить столбцы, которые не требуется использовать при
63 # построении гистограмм:
64 # data_remove = [
65 #     "Название первого столбца",
66 #     "Название второго столбца"
67 # ]
68 # for i in data_remove:
69 #     column_headers_show_hist.remove(i)
70
71 column_headers_show_dot = df_data_ns.columns.tolist()
72 data_remove = [
73     '1. Пол',
74     '2. Возраст',
75     '4. Наличие брака',
76     '6. Членство в молодёжной организации'
77 ]
78 for i in data_remove:
79     column_headers_show_dot.remove(i)
80

```

▼ Гистограммы

Построение гистограмм.

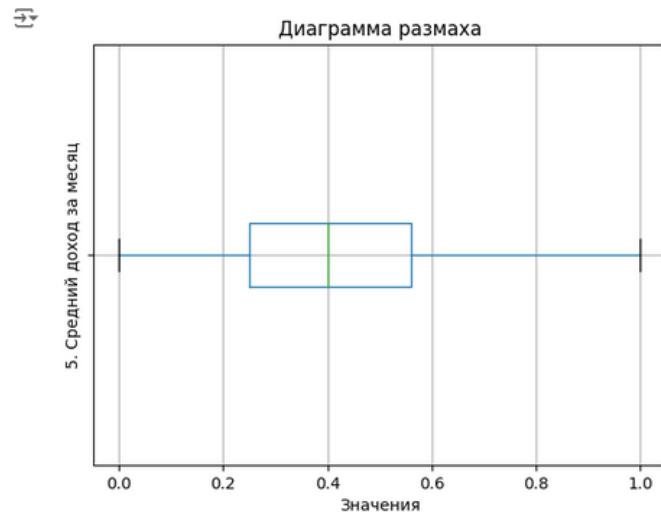
```
[36] 1 # Создание подпапки для сохранения файлов (при необходимости).
2 if path != '':
3     path = files_storage + 'image/processed/'
4
5 # Построение гистограмм для выбранных данных:
6 histogram(
7     table=df_data_ns,
8     series=column_headers_show_hist,
9     path_to_save=path
10 )
11
```



#### ▼ Диаграммы размаха

Построение диаграмм размаха.

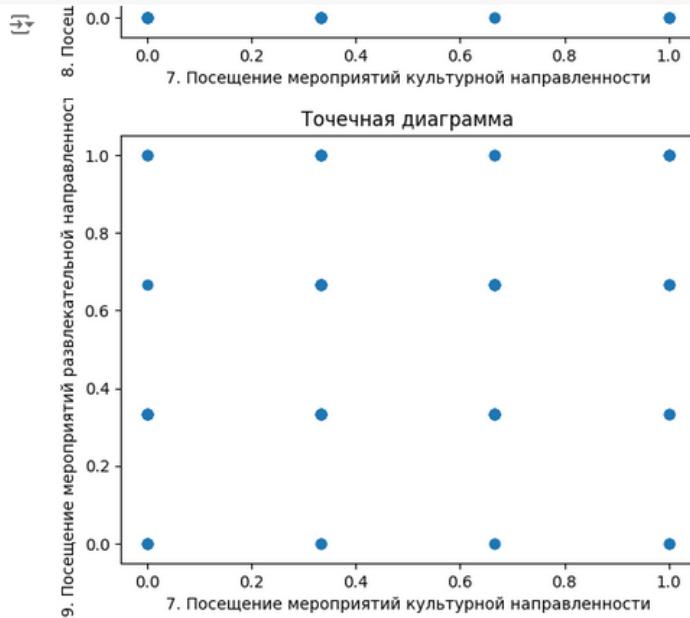
```
[37] 1 # Создание подпапки для сохранения файлов (при необходимости).
2 if path != '':
3     path = files_storage + 'image/processed/'
4
5 # Построение коробчатых диаграмм для выбранных данных:
6 box_diagram(
7     table=df_data_ns,
8     series=column_headers_show_box,
9     path_to_save=path
10 )
11
```



## ▼ Точечные диаграммы

Построение точечной диаграммы.

```
[38] 1 # Создание подпапки для сохранения файлов (при необходимости).
2 if path != '':
3     path = files_storage + 'image/processed/'
4
5 # Построение точечных диаграмм для выбранных данных:
6 scatter_diagram(
7     table=df_data_ns,
8     series=column_headers_show_dot,
9     path_to_save=path
10    )
11
```



## ▼ Вычисление описательных метрик

Описательные метрики позволяют получить некоторое представление о собранных данных, не прибегая к методам анализа данных.

**Мода** - наиболее часто встречающееся значение среди случайных величин выборки.

- Если существует лишь одно наиболее часто встречающееся значение, то считается, что дискретное распределение является **унимодальным**.
- Если существует два наиболее часто встречающихся значения, то считается, что дискретное распределение является **двуимодальным**.
- Если существует более двух наиболее часто встречающихся значения, то считается, что дискретное распределение является **мультимодальным**.

**Медиана** - значение  $[(n + 1)/2]$ -й порядковой статистики при нечетном объеме выборки  $n$ ; значение суммы  $(n/2)$ -й и  $[(n/2) + 1]$ -й порядковых статистик, деленной на два, при четном объеме выборки  $n$ .

**Среднее арифметическое** (выборочное среднее) - сумма случайных величин выборки, деленная на число слагаемых в этой сумме.

**Дисперсия** - сумма квадратов отклонений случайных величин выборки от их выборочного среднего, деленная на число слагаемых в этой сумме минус один.

**Среднее квадратическое отклонение** - положительный квадратный корень из дисперсии.

Чем среднее квадратическое отклонение больше, тем более рассеянные данные в выборке.

**Асимметрия и эксцесс** позволяют оценить "на глаз", насколько близко распределение к нормальному.

Интерпретация значения коэффициента асимметрии:

- Чем коэффициент асимметрии меньше 0, тем более "левосторонняя" асимметрия, и тем более удлинён левый "хвост" распределения.
- Чем коэффициент асимметрии больше 0, тем более "правосторонняя" асимметрия, и тем более удлинён правый "хвост" распределения.

- Чем коэффициент асимметрии по модулю меньше, тем ближе распределение к нормальному.

Интерпретация значения коэффициента эксцесса:

- Чем коэффициент эксцесса больше нуля, тем график распределения более "островершинный".
- Чем коэффициент эксцесса меньше нуля, тем график распределения более пологий.
- Чем коэффициент эксцесса ближе к нулю, тем ближе распределение к нормальному.

## РЕДАКТИРОВАНИЕ КОДА

```
✓ [39] 1 # ВЫБОР СТОЛБЦОВ ДЛЯ ВЫЧИСЛЕНИЯ ОПИСАТЕЛЬНЫХ МЕТРИК
2
3 # Для обработки малого числа столбцов:
4
5 # Тут необходимо перечислить столбцы, которые требуется использовать:
6 # column_headers_analysis_metrics = [
7 #     "Название первого столбца",
8 #     "Название второго столбца"
9 # ]
10
11 # Для обработки большого числа столбцов:
12
13 # column_headers_analysis_metrics = df_data_ns.columns.tolist()
14 # Тут можно перечислить столбцы, которые не требуется использовать при
15 # построении гистограмм:
16 # data_remove = [
17 #     "Название первого столбца",
18 #     "Название второго столбца"
19 # ]
20 # for i in data_remove:
21 #     column_headers_analysis_AE.remove(i)
22
23 column_headers_analysis_metrics = df_data_ns.columns.tolist()
24 # Тут необходимо перечислить столбцы, которые не требуется использовать
25
```

Для вычисления описательных метрик используется функция metrics().

```
✓ [40] 1 def metrics(
2         table: pd.DataFrame,
3         series: list,
4         path_to_save: str = ''
5         ) -> pd.DataFrame:
6
7     """
8         Нахождение описательных метрик данных.
9
10        Аргументы функции:
11
12        table - таблица, содержащая столбец с данными,
13
14        series - список названий столбцов с данными,
15
16        path_to_save - по ум. = '', путь для сохранения файлов.
17
18        Возвращает pd.DataFrame.
19    """
20
21    # ВЫЧИСЛЕНИЕ МЕТРИК
22
23    # Создание таблицы, куда будет размещаться ответ:
24    column_names_metrics = [
25        'Данные',
26        'n',
27        'Мода',
28        'Медиана',
29        'Среднее арифметическое',
30        'Дисперсия',
31        'Среднее квадратическое отклонение',
32        'Асимметрия',
33        'Эксцесс'
34    ]
35    result_metrics = pd.DataFrame(columns = column_names_metrics)
36
37    for s in series:
```

```

38     # Преобразование данных в массив numpy:
39     np_data = table[s].to_numpy()
40
41     # РАСЧЁТ ОПИСАТЕЛЬНЫХ СТАТИСТИК
42
43     # Вычисление моды:
44     mode = sci.stats.mode(np_data)[0]
45
46     # Вычисление медианы:
47     median = np.median(np_data)
48
49     # Вычисление среднего квадратического:
50     std = np.std(np_data)
51
52     # Объём выборки:
53     n = len(np_data)
54
55     # Вычисление среднего арифметического:
56     mean = np_data.mean()
57
58     # Вычисление выборочной дисперсии:
59     # Для вычисления генеральной дисперсии необходимо
60     # приравнять значение ddof нулю: ddof = 0
61     variance = np.var(np_data, ddof = 1)
62
63     # Вычисление асимметрии:
64     A = np.mean(((np_data - mean) ** 3) / variance ** (3 / 2))
65
66     # Вычисление эксцесса:
67     E = np.mean(((np_data - mean) ** 4) / (variance ** 2))
68
69     # Создание таблицы с результатами:
70     result = pd.DataFrame({
71         'Данные': [s],
72         'n': [n],
73         'Мода': [round(mode, 3)],
74         'Медиана': [round(median, 3)],
75         'Среднее арифметическое': [round(mean, 3)],
76         'Дисперсия': [round(variance, 3)],
77         'Среднее квадратическое отклонение': [round(std, 3)],
78         'Асимметрия': [round(A, 3)],
79         'Эксцесс': [round(E, 3)]
80     })
81
82     result_metrics = pd.concat([result_metrics, result])
83
84
85     # Восстановление индексов таблицы:
86     result_metrics = result_metrics.reset_index(drop=True)
87
88     # СОЗДАНИЕ ФАЙЛА ТАБЛИЦЫ С РЕЗУЛЬТАТОМ
89
90     if path_to_save != '':
91
92         output_dir = pathlib.Path(path_to_save)
93
94         # Проверка существования папки:
95         if not output_dir.exists():
96
97             # Создание папки для таблицы:
98             output_dir.mkdir(parents=True, exist_ok=True)
99
100        # Сохранение таблицы с метриками в текущую директорию (папку):
101        result_metrics.to_csv(
102            path_to_save + '/result_table_metrics.csv',
103            index=False
104        )
105
106    return result_metrics
107

```

Вычисление описательных метрик.

```

✓ [41] 1  # Создание подпапки для сохранения файлов (при необходимости).
0      2  if files_storage != '':
1      |  path = files_storage + 'metrics/'
2  else:
3
4

```

```

5     path = ''
6
7 # Вычисление описательных матрик для нормализованных значений:
8 result_table_metrics = metrics(
9     table=df_data_ns,
10    series=column_headers_analysis_metrics,
11    path_to_save=path
12 )
13
14 print('Результат нахождения описательных метрик для данных в выбранных \
15 столбцах.\n')
16 display(result_table_metrics)
17

```

→ Результат нахождения описательных метрик для данных в выбранных столбцах.

	Данные	n	Мода	Медиана	Среднее арифметическое	Дисперсия	Среднее квадратическое отклонение	Асимметрия	Эксцесс
0	1. Пол	106	0.000	0.000	0.462	0.251	0.499	0.149	1.004
1	2. Возраст	106	1.000	0.600	0.570	0.104	0.321	0.041	1.597
2	3. Образование	106	0.500	0.500	0.580	0.139	0.371	-0.262	1.816
3	4. Наличие брака	106	1.000	1.000	0.509	0.252	0.500	-0.037	0.983
4	5. Средний доход за месяц	106	0.400	0.400	0.418	0.052	0.227	0.343	2.728
5	6. Членство в молодёжной организации	106	0.000	0.000	0.321	0.220	0.467	0.757	1.560
6	7. Посещение мероприятий культурной направленн...	106	0.667	0.333	0.484	0.104	0.321	0.007	1.997
7	8. Посещение мероприятий политической направле...	106	0.333	0.333	0.472	0.120	0.345	0.074	1.807
8	9. Посещение мероприятий развлекательной напра...	106	0.333	0.333	0.506	0.096	0.308	0.087	2.108

## ▼ Анализ данных

### ▼ Настройка перед анализом данных

Выбор данных для анализа:

- 1) Вычисление коэффициента корреляции Пирсона.
- 2) Проверка распределения на соответствие нормальному критерием Эпса-Палли.

## РЕДАКТИРОВАНИЕ КОДА

✓ [42] 1 # ВЫБОР СТОЛБЦОВ ДЛЯ ВЫЧИСЛЕНИЯ КОЭФФИЦИЕНТА КОРРЕЛЯЦИИ ПИРСОНА
2
3 # Для обработки малого числа столбцов:
4
5 # Тут необходимо перечислить столбцы, которые требуется использовать:
6 # column\_headers\_analysis\_corr = [
7 # "Название первого столбца",
8 # "Название второго столбца"
9 # ]
10
11 # Для обработки большого числа столбцов:
12
13 # column\_headers\_analysis\_corr = df\_data.columns.tolist()
14 # Тут можно перечислить столбцы, которые не требуется использовать при
15 # построении гистограмм:
16 # data\_remove = [
17 # "Название первого столбца",
18 # "Название второго столбца"
19 # ]
20 # for i in data\_remove:
21 # column\_headers\_analysis\_corr.remove(i)
22
23 column\_headers\_analysis\_corr = df\_data\_ns.columns.tolist()
24 # Тут необходимо перечислить столбцы, которые не требуется использовать

```

25 # ВЫБОР СТОЛБЦОВ ДЛЯ ПРОВЕРКИ РАСПРЕДЕЛЕНИЯ
26 # НА СООТВЕТСТВИЕ НОРМАЛЬНОМУ КРИТЕРИЮ ЭППСА-ПАЛЛИ
27
28 # Для обработки малого числа столбцов:
29
30 # Тут необходимо перечислить столбцы, которые требуется использовать:
31 # column_headers_analysis_EP = [
32 #     "Название первого столбца",
33 #     "Название второго столбца"
34 # ]
35
36 # Для обработки большого числа столбцов:
37
38 # column_headers_analysis_EP = df_data.columns.tolist()
39 # Тут можно перечислить столбцы, которые не требуется использовать при
40 # построении гистограмм:
41 # data_remove = [
42 #     "Название первого столбца",
43 #     "Название второго столбца"
44 # ]
45 # for i in data_remove:
46 #     column_headers_analysis_EP.remove(i)
47
48 column_headers_analysis_EP = df_data_ns.columns.tolist()
49 # Тут необходимо перечислить столбцы, которые не требуется использовать
50
51

```

#### ▼ Корреляция значений признаков

Для того, чтобы установить, существует ли зависимость между двумя переменными, применяется коэффициент корреляции Пирсона.

В результате своей работы он возвращает коэффициент корреляции - число в диапазоне от 1 до -1.

- Если коэффициент корреляции меньше 0, то корреляция является отрицательной, т. е. при увеличении одной переменной другая уменьшается.
- Если коэффициент корреляции больше 0, то корреляция является положительной, т. е. при увеличении одной переменной увеличивается и другая.
- Чем ближе коэффициент корреляции к 0, тем меньше выражена зависимость одной переменной от другой. Ноль означает отсутствие корреляции.

Для вычисления значения коэффициента Пирсона используется функция correlation().

```

[43] 1 def correlation_pearson(
2     table: pd.DataFrame,
3     series: list,
4     alpha: float = 0.95,
5     only_meaning: bool = True,
6     critical: float = 0,
7     scatter_diagram_plotting: bool = False,
8     path_to_save: str = ''
9     ) -> pd.DataFrame:
10
11     """
12     Вычисление коэффициента корреляции Пирсона.
13
14     Аргументы функции:
15
16     table - таблица, содержащая столбец с данными,
17
18     series - список названий столбцов с данными,
19
20     alpha - по ум. = 0.95, возмож. знач.: 0,1; 0,5; 0,25; 0,1; 0,001,
21
22     only_meaning - по ум. = True, если True, то выводятся только значимые
23     значения коэффициента корреляции,
24
25     critical - по ум. = 0, если не равен 0, то выводятся только значения
26     коэффициента корреляции, большие critical,
27
28     scatter_diagram_plotting - по ум. False, если True, то выводится точечная
29     диаграмма scatter_diagram(),
30

```

```

31     path_to_save - по ум. = '', путь для сохранения файлов.
32
33     Возвращает pd.DataFrame.
34
35
36     # ВЫЧИСЛЕНИЕ КОРРЕЛЯЦИИ
37
38     # Создание таблицы, куда будет размещаться ответ:
39     column_names_corr = [
40         'Первый столбец',
41         'Второй столбец',
42         'n',
43         't-статистика (t)',
44         'Табличное критическое значение t-статистики (t крит.)',
45         '|t| > t крит.',
46         'Качественная характеристика',
47         'Значимость корреляции',
48         'Коэффициент корреляции'
49     ]
50     result_corr = pd.DataFrame(columns = column_names_corr)
51
52     for i, series_1 in enumerate(series):
53         for j, series_2 in enumerate(series):
54             if i >= j: continue
55
56             # Подготовка значений для построения диаграммы:
57             data_list_1 = table[series_1].tolist()
58             data_list_2 = table[series_2].tolist()
59
60             # Вычисление коэффициента корреляции Пирсона:
61             r = float(sci.stats.pearsonr(data_list_1, data_list_2)[0])
62
63             # Классификация корреляции по шкале Чеддока:
64             ch_r = abs(r)
65             if ch_r < 0.1:
66                 cheddok = ''
67             elif ch_r < 0.3:
68                 cheddok = '*'
69             elif ch_r < 0.5:
70                 cheddok = '**'
71             elif ch_r < 0.7:
72                 cheddok = '***'
73             elif ch_r < 0.9:
74                 cheddok = '****'
75             else:
76                 cheddok = '*****'
77
78             # Определение объёма выборки
79             n = len(data_list_1)
80
81             # Определение степеней свободы:
82             df = n - 2
83
84             # Вычисление t-статистики:
85             t = r * np.sqrt(df) / (1 - r**2)
86
87             # Нахождение критического значения t-распределения:
88             t_critical = sci.stats.t.ppf(1 - alpha/2, df)
89
90             # Проверка значимости:
91             if abs(t) > t_critical:
92                 Conclusion = 'Значима'
93             else:
94                 Conclusion = 'Не значима'
95
96             # Проверка, удовлетворяют ли значения условиям отображения:
97             raw_show = False
98             if only_meaning:
99                 if Conclusion == 'Значима':
100                     if critical != 0:
101                         if abs(r) > critical:
102                             raw_show = True
103                         else:
104                             raw_show = True
105             else:
106                 if critical != 0:
107                     if abs(r) > critical:
108                         raw_show = True

```

```

109     else:
110         raw_show = True
111
112     if raw_show:
113         # Создание строки с вычисленным коэффициентом корреляции:
114         result = pd.DataFrame({
115             'Первый столбец': [series_1],
116             'Второй столбец': [series_2],
117             'n': [n],
118             't-статистика (t)': [round(t, 3)],
119             'Табличное критическое значение t-статистики (t крит.)': \
120             [round(t_critical, 3)],
121             '|t| > t крит.': [abs(t) > t_critical],
122             'Качественная характеристика': [cheddok],
123             'Значимость корреляции': [Conclusion],
124             'Коэффициент корреляции': [round(r, 3)]
125         })
126
127         # Добавление строки в таблицу с результатами:
128         result_corr = pd.concat([result_corr, result])
129
130     # Вывод точечной диаграммы анализируемых данных:
131     if raw_show and scatter_diagram_plotting:
132         # Построение точечной диаграммы:
133         plt.scatter(data_list_1, data_list_2)
134
135         # Добавление подписей к осям и заголовка:
136         plt.xlabel(series_1)
137         plt.ylabel(series_2)
138         plt.title('Точечная диаграмма')
139
140         # Отображение диаграммы:
141         plt.show()
142
143     # Восстановление индексов таблицы:
144     result_corr = result_corr.reset_index(drop=True)
145
146     # СОХРАНИЕНИЕ ФАЙЛА ТАБЛИЦЫ С РЕЗУЛЬТАТОМ
147
148     if path_to_save != '':
149
150         # Создание подпапки:
151         path_to_save = path_to_save + '/correlation'
152
153         output_dir = pathlib.Path(path_to_save)
154
155         # Проверка существования папки:
156         if not output_dir.exists():
157
158             # Создание папки для таблицы:
159             output_dir.mkdir(parents=True, exist_ok=True)
160
161         # Сохранение таблицы с метриками в текущую директорию (папку):
162         result_corr.to_csv(
163             path_to_save + '/result_table_correlation.csv',
164             index=False
165         )
166
167     return result_corr
168

```

Нахождение коэффициента корреляции Пирсона.

```

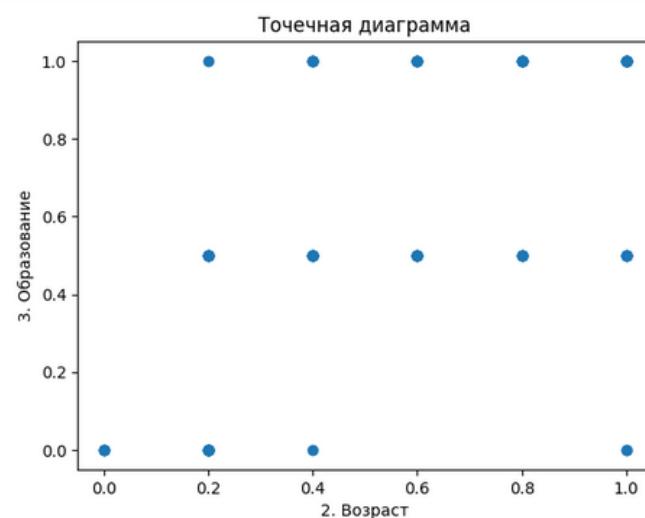
✓ 0 [44] 1 # Создание подпапки для сохранения файлов (при необходимости).
2 if files_storage != '':
3     path = files_storage + 'analysis/'
4 else:
5     path = ''
6
7 # При значении коэффициента корреляции по модулю меньшим, чем это значение,
8 # Стока с ответом выводиться не будет. Для показа всех строк приравнять 0:
9 restrict = 0.5
10
11 # Для вывода всех значений корреляции, а не только значимых,
12 # измените значение аргумента only_meaning на False.
13
14 # Для построения точечных диаграмм для всех рассматриваемых пар столбцов,

```

```

15 # измените значение аргумента scatter_diagram_plotting на True.
16
17 # Нахождение коэффициента корреляции для нормализованных значений:
18 result_table_corr = correlation_pearson(
19     table=df_data_ns,
20     series=column_headers_analysis_corr,
21     alpha=0.05,
22     only_meaning=True,
23     critical=restrict,
24     scatter_diagram_plotting=True,
25     path_to_save=path
26 )
27
28 print(f'Результат нахождения коэффициента корреляции Пирсона для данных \
29 в выбранных столбцах.')
30 if restrict != 0:
31     print(f'Выводятся строки со значимыми значениями коэффициента корреляции \
32 по модулю большими, чем {restrict}.')
33 display(result_table_corr)
34

```



Результат нахождения коэффициента корреляции Пирсона для данных в выбранных столбцах.  
Выводятся строки со значимыми значениями коэффициента корреляции по модулю большими, чем 0.5.

Первый столбец	Второй столбец	n	t-статистика (t)	Табличное критическое значение t-статистики (t крит.)		Качественная характеристика	Значимость корреляции	Коэффициент корреляции
				t  > t крит.	True			
0	2. Возраст	3. Образование	106	8.276	1.983	True	***	Значима

#### ▼ Проверка распределения на соответствие нормальному

Проверка подчинения данных нормальному распределению с помощью критерия Эпса-Палли, описанного в ГОСТ Р ИСО 5479-2002.

Критерий Эпса-Палли применим для выборки объёмом  $n \geq 8$ , при работе с выборками меньшего объёма с помощью данного критерия невозможно получить достоверные результаты.

Для проверки соответствию нормальному распределению используется функция `epps_pally()`.

```

✓ [45] 1 def epps_pally(
2     table: pd.DataFrame,
3     series: list,
4     p_level: float = 0.95,
5     path_to_save: str = ''
6     ) -> pd.DataFrame:
7
8     """
9         Критерий Эпса-Палли для проверки нормального распределения данных.
10
11     Аргументы функции:
12
13     table - табл., содержащая столбец с анализируемыми данными,

```

```

14     series - список названий столбцов с данными,
15
16     p-level - вероятность получить верное значение
17     p = alpha - 1, по ум. = 0.95, возмож. знач.: 0.9; 0.95; 0.975; 0.99,
18
19     path_to_save - по ум. = '', путь для сохранения файлов.
20
21     Возвращает: pd.DataFrame.
22
23
24
25     # ВЫЧИСЛЕНИЕ КРИТЕРИЯ ЭППСА-ПАЛЛИ
26
27     # Создание таблицы, куда будет размещаться ответ:
28     column_names_T_EP = [
29         'Данные',
30         'n',
31         'p',
32         'Значение критерия (T_EP)',
33         'Табличное значение критерия (T_EP крит.)',
34         'T_EP < Table',
35         'Нормальное распределение'
36     ]
37     result_T_EP = pd.DataFrame(columns = column_names_T_EP)
38
39     # Таблица для критерия Эппса-Палли:
40     # p-квантили статистики критерия T_EP для
41     # p = 1 - alpha = 0.90; 0.95; 0.975; 0.99
42     n_index = [8, 9, 10, 15, 20, 30, 50, 100, 200]
43     data = {
44         'p=0.9': [0.271, 0.275, 0.279, 0.284, 0.287,
45                     0.289, 0.290, 0.291, 0.290],
46         'p=0.95': [0.347, 0.350, 0.357, 0.366, 0.368,
47                     0.371, 0.374, 0.376, 0.379],
48         'p=0.975': [0.426, 0.428, 0.437, 0.447, 0.450,
49                     0.459, 0.460, 0.464, 0.467],
50         'p=0.99': [0.526, 0.537, 0.545, 0.560, 0.564,
51                     0.569, 0.574, 0.583, 0.590]
52     }
53     T_EP_table = pd.DataFrame(data, index=n_index)
54     T_EP_table.index.name = 'n'
55
56     # Выбор величины доверительной вероятности:
57     p_level_dict = {
58         0.9: T_EP_table.columns[0],
59         0.95: T_EP_table.columns[1],
60         0.975: T_EP_table.columns[2],
61         0.99: T_EP_table.columns[3]
62     }
63
64     for s in series:
65
66         # Преобразование данных в массив numpy:
67         np_data = table[s].to_numpy()
68
69         # ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ КРИТЕРИЯ
70
71         # Объём выборки:
72         n = len(np_data)
73
74         # Критерий применим для n >= 8:
75         if n < 8:
76             print(f'Столбец {s} не обработан, так как n < 8 не возможно \\\
77 получение достоверных результатов. n = {n}')
78             continue
79
80         # Среднее арифметическое:
81         mean = np_data.mean()
82
83         # Центральный момент 2-го порядка:
84         m_2 = np.var(np_data, ddof = 0)
85
86         A = sqrt(2) * np.sum([exp(-(np_data[i] - mean)**2 / (4*m_2))
87             for i in range(n)])
88         B = 2/n * np.sum([
89             np.sum([
90                 exp(-(np_data[j] - np_data[k])**2 / (2*m_2)) for j in \
91                 range(0, k)

```

```

92     |     ]) for k in range(1, n)])
93     T_EP_empiric = 1 + n / sqrt(3) + B - A
94
95     # НАХОЖДЕНИЕ ТАБЛИЧНОГО ЗНАЧЕНИЯ
96
97     # Линейная интерполяция для нахождения р для
98     # различных значений выборки n:
99     N_index = T_EP_table.index
100    T = T_EP_table[p_level_dict[p_level]]
101    f_lin = sci.interpolate.interp1d(N_index, T)
102    T_EP_teoretic = float(f_lin(n))
103
104    # СРАВНЕНИЕ ПОЛУЧЕННЫХ ЗНАЧЕНИЙ
105
106    if T_EP_empiric < T_EP_teoretic:
107        Conclusion = 'Выполняется'
108    else:
109        Conclusion = 'Не выполняется'
110
111    # СОЗДАНИЕ ТАБЛИЦЫ С РЕЗУЛЬТАТАМИ
112
113    result = pd.DataFrame({
114        'Данные': [s],
115        'n': [n],
116        'p': [p_level],
117        'Значение критерия (T_EP)': [round(T_EP_empiric, 3)],
118        'Табличное значение критерия (T_EP крит.)': \
119        [round(T_EP_teoretic, 3)],
120        'T_EP < Table': [T_EP_empiric < T_EP_teoretic],
121        'Нормальное распределение': [Conclusion]
122    })
123
124    result_T_EP = pd.concat([result_T_EP, result])
125
126    # Восстановление индексов таблицы:
127    result_T_EP = result_T_EP.reset_index(drop=True)
128
129    # СОХРАНЕНИЕ ФАЙЛА ТАБЛИЦЫ С РЕЗУЛЬТАТОМ
130
131    if path_to_save != '':
132
133        # Создание подпапки:
134        path_to_save = path_to_save + '/T_EP'
135
136        output_dir = pathlib.Path(path_to_save)
137
138        # Проверка существования папки:
139        if not output_dir.exists():
140
141            # Создание папки для таблицы:
142            output_dir.mkdir(parents=True, exist_ok=True)
143
144        # Сохранение таблицы с метриками в текущую директорию (папку):
145        result_T_EP.to_csv(
146            path_to_save + '/result_table_T_EP.csv',
147            index=False
148        )
149
150    return result_T_EP
151

```

Применение критерия Эппса-Палли

```

[46] 1  # Создание подпапки для сохранения файлов (при необходимости).
2  if files_storage != '':
3  |  path = files_storage + 'analysis/'
4  else:
5  |  path = ''
6
7  # Вычисление критерия Эппса-Палли для нормализованных значений:
8  result_table_T_EP = epps_pally(
9      table=df_data_ns,
10     series=column_headers_analysis_EP,
11     p_level=0.95,
12     path_to_save=path
13   )
14

```

```

15 print('Результат применения критерия Эпсса-Палли для данных \
16 в выбранных столбцах.\n')
17 display(result_table_T_EP)
18

```

→ Результат применения критерия Эпсса-Палли для данных в выбранных столбцах.

	Данные	n	p	Значение критерия (T_EP)	Табличное значение критерия (T_EP крит.)	T_EP < Table	Нормальное распределение
0	1. Пол	106	0.95	4.721	0.376	False	Не выполняется
1	2. Возраст	106	0.95	1.203	0.376	False	Не выполняется
2	3. Образование	106	0.95	0.709	0.376	False	Не выполняется
3	4. Наличие брака	106	0.95	4.631	0.376	False	Не выполняется
4	5. Средний доход за месяц	106	0.95	0.182	0.376	True	Выполняется
5	6. Членство в молодёжной организации	106	0.95	6.861	0.376	False	Не выполняется
6	7. Посещение мероприятий культурной направленн...	106	0.95	0.278	0.376	True	Выполняется
7	8. Посещение мероприятий политической направле...	106	0.95	0.551	0.376	False	Не выполняется
8	9. Посещение мероприятий развлекательной напра...	106	0.95	0.256	0.376	True	Выполняется