|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |
| Институт информационных технологий (ИИТ) | |
| Прикладная Математика (ПМ) | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4** | | | |
| **по дисциплине «Технологии и инструментарий анализа больших данных»** | | | |
|  | | | |
| Выполнил студент группы ИКБО-13-20 | | Вакуленко А. И. | |
|  | |  | |
| Принял | | Горячев А. А. | |
| Практические работы выполнены | «\_\_\_»\_\_\_\_\_\_\_2023 г. | | (подпись студента) | |
| «Зачтено» | «\_\_\_»\_\_\_\_\_\_\_2023 г. | | (подпись руководителя) | |
|  |  | |  | |

Москва 2023

**Задание 1.**

Определить два вектора, представляющие собой число автомобилей, припаркованных в течении 5 рабочих дней у бизнес-центра на уличной стоянке и в подземном гараже (Таблица 1), (Листинг 1).

Таблица 1 – данные о хранении автомобилей

|  |  |  |
| --- | --- | --- |
| День | Улица | Гараж |
| Понедельник | 80 | 100 |
| Вторник | 98 | 82 |
| Среда | 75 | 105 |
| Четверг | 91 | 89 |
| Пятница | 78 | 102 |

|  |
| --- |
| Листинг 1 - Задание 1 |
| import numpy as np  outdoor\_storage = np.array([80, 98, 75, 91, 78])  garage\_storage = np.array([100, 82, 105, 89, 102])  print('outdoor\_storage', outdoor\_storage)  print('garage\_storage', garage\_storage) |

Результат работы программы представлены на рисунке 1.

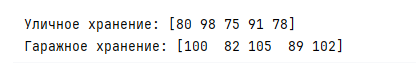


Рисунок 1 - Результат работы

**Задание 2**

Найти и интерпретировать корреляцию между переменными «Улица» и «Гараж» (подсчитать корреляцию по Пирсону) (Листинг 2).

Результат работы программы представлены на рисунке 2.

|  |
| --- |
| Листинг 2 - Задание 2 |
| print(f'Матрица коэффициентов корреляции: \n{np.corrcoef(outdoor\_storage, garage\_storage)}') print(f'Коэффициент корреляции между двумя переменными: {np.corrcoef(outdoor\_storage, garage\_storage)[0, 1]}') |

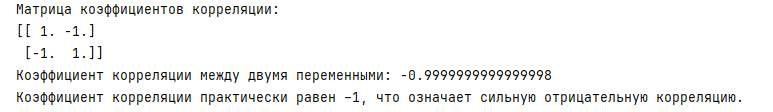


Рисунок 2 - Результат работы

**Задание 3.**

Построить диаграмму рассеяния для вышеупомянутых переменных. (Листинг 3)

|  |
| --- |
| Листинг 3 - Задание 3 |
| import matplotlib.pyplot as plt  plt.grid()  plt.scatter(outdoor\_storage, garage\_storage, marker='o')  plt.title('Диаграмма рассеивания данных о хранении автомобилей', fontsize='16')  plt.xlabel('Стоянка на улице')  plt.ylabel('Стоянка в гараже')  print('На получившимся графике мы видим сильную отрицательную корреляцию') |

Результат работы программы представлены на рисунке 3.

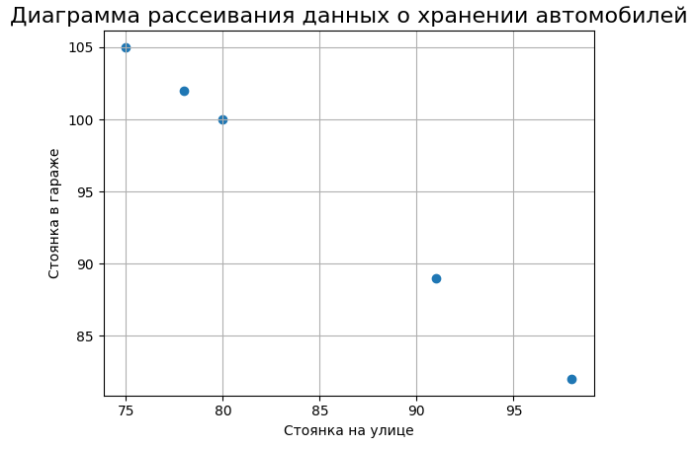


Рисунок *3* – Гистограммы числовых показателей

**Задание 4.**

Найти и выгрузить данные. Вывести, провести предобработку и описать признаки.

О наборе данных:

* phone\_name:название
* brand:марка телефона
* os:операционная система телефона
* inches size of the phone screen as inches: размер экрана телефона в дюймах
* resolution: разрешение экрана телефона
* battery: емкость батареи телефона
* battery\_type: тип батареи телефона
* ram (GB): оперативная память телефона в ГБ
* announcement\_date: дата анонса телефона
* weight(g): вес телефона в граммах
* storage (GB): объем памяти телефона в ГБ
* video\_720p: поддерживает ли камера телефона функцию 720p
* video\_1080p: поддерживает ли камера телефона функцию
* video\_4K: поддерживает ли камера телефона функцию
* video\_8K: поддерживает ли камера телефона функцию 8K
* video\_30fps: поддерживает ли камера телефона функцию 30 кадров в секунду
* video\_60fps: поддерживает ли камера телефона функцию 60 кадров в секунду
* video\_120fps: поддерживает ли камера телефона функцию 120 кадров в секунду
* video\_240fps: поддерживает ли камера телефона функцию 240 кадров в секунду
* video\_480fps: поддерживает ли камера телефона функцию 480 кадров в секунду
* video\_960fps: поддерживает ли камера телефона функцию 960 кадров в секунду
* price(USD):цена телефона в долларах США

Для решения задачи был написан код (Листинг 4)

|  |
| --- |
| Листинг 4 - Задание 4 |
| import pandas as pd  data = pd.read\_csv('cleaned\_all\_phones.csv')  data = data[data.apply(lambda row: not any(row == 'None'), axis=1)]  data.isnull().sum() |

Результат работы программы представлены на рисунках 4, 5.

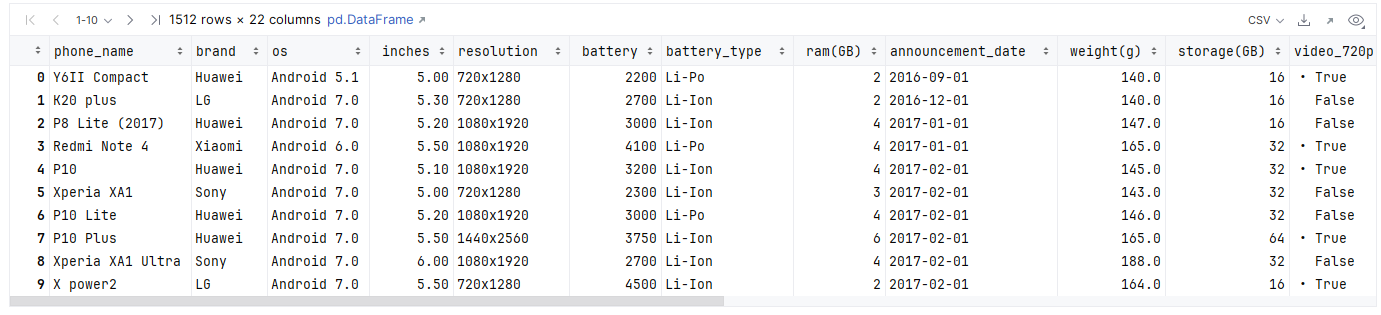


Рисунок 4 – Датасет

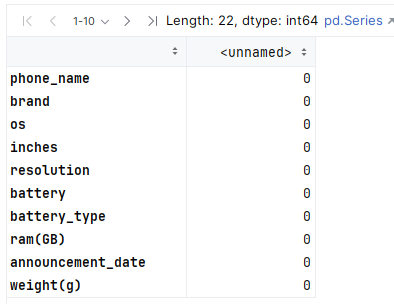


Рисунок 5 – Результат работы программы

**Задание 5.**

Построить корреляционную матрицу по одной целевой переменной. Определить наиболее коррелирующую переменную, продолжить с ней работу в следующем пункте.

Для решения задачи был написан код (Листинг 5)

|  |
| --- |
| Листинг 5 - Задание 5 |
| f\_data = pd.DataFrame()  f\_data['brand'] = pd.factorize(data['brand'])[0]  f\_data['os'] = pd.factorize(data['os'])[0]  f\_data['inches'] = pd.factorize(data['inches'])[0]  f\_data['resolution'] = pd.factorize(data['resolution'])[0]  f\_data['battery'] = data['battery']  f\_data['battery\_type'] = pd.factorize(data['battery\_type'])[0]  f\_data['ram(GB)'] = data['ram(GB)']  f\_data['weight(g)'] = data['weight(g)']  f\_data['storage(GB)'] = data['storage(GB)']  f\_data['video\_720p'] = pd.factorize(data['video\_720p'])[0]  f\_data['video\_1080p'] = pd.factorize(data['video\_1080p'])[0]  f\_data['video\_4K'] = pd.factorize(data['video\_4K'])[0]  f\_data['video\_8K'] = pd.factorize(data['video\_8K'])[0]  f\_data['video\_30fps'] = pd.factorize(data['video\_30fps'])[0]  f\_data['video\_60fps'] = pd.factorize(data['video\_60fps'])[0]  f\_data['video\_120fps'] = pd.factorize(data['video\_120fps'])[0]  f\_data['video\_240fps'] = pd.factorize(data['video\_240fps'])[0]  f\_data['video\_480fps'] = pd.factorize(data['video\_480fps'])[0]  f\_data['video\_960fps'] = pd.factorize(data['video\_960fps'])[0]  f\_data['PRICE'] = data['price(USD)']  for i in f\_data:  f\_data[f'{i}'] = f\_data[f'{i}'].astype(float)  corr\_matrix = f\_data.corr().PRICE.to\_frame().round(2) corr\_matrix.style.background\_gradient(cmap='coolwarm').to\_excel('correlation\_matrix.xlsx', engine='openpyxl') |

Результат работы программы представлены на рисунке 6.

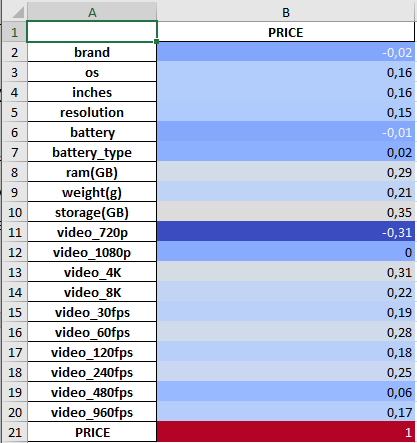


Рисунок *6* – Результат работы программы

**Вывод:** Делая вывод из полученных данных, можно сказать, что: наибольшее влияние на цену оказывает объем памяти (GB).

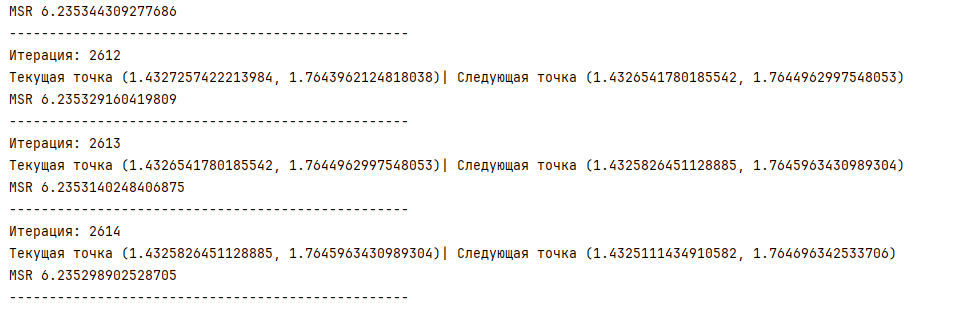
**Задание 6.**

Реализовать регрессию вручную, отобразить наклон, сдвиг и MSE.

Для решения задачи был написан код (Листинг 6)

|  |
| --- |
| Листинг 6 - Задание 6 |
| X = np.array(f\_data[['storage(GB)']], type(float))  y = np.array(f\_data['PRICE'], type(float))  X = X / 100  y = y / 100  def mserror(X, w1, w0, y):  y\_pred = w1 \* X[:, 0] + w0  return np.sum((y - y\_pred) \*\* 2 / len(y\_pred))  def gr\_mserror(X, w1, w0, y):  y\_pred = w1 \* X[:, 0] + w0  return np.array([2 / len(X) \* np.sum((y - y\_pred)) \* (-1),  2 / len(X) \* np.sum((y - y\_pred) \* (-X[:, 0]))])  eps = 0.0001  w1 = 0  w0 = 0  learning\_rate = 0.001  next\_w1 = w1  next\_w0 = w0  n = 10000  for i in range(n):  cur\_w1 = next\_w1  cur\_w0 = next\_w0  next\_w0 = cur\_w0 - learning\_rate \* gr\_mserror(X, cur\_w1, cur\_w0, y)[0]  next\_w1 = cur\_w1 - learning\_rate \* gr\_mserror(X, cur\_w1, cur\_w0, y)[1]  print(f"Итерация: {i}")  print(f"Текущая точка {cur\_w1, cur\_w0}| Следующая точка {next\_w1, next\_w0}")  print(f"MSR {mserror(X, cur\_w1, cur\_w0, y)}")  print('-' \* 50)  if (abs(cur\_w1 - next\_w1) <= eps) and (abs(cur\_w0 - next\_w0) <= eps):  break  next\_w0 = round(next\_w0, 2)  next\_w1 = round(next\_w1, 2)  f\_data['dy'] = f\_data['storage(GB)'] \* next\_w1 + next\_w0  # Регрессия sklearn  from sklearn.linear\_model import LinearRegression  model = LinearRegression()  X = np.array(f\_data[['storage(GB)']], type(float))  y = np.array(f\_data['PRICE'], type(float))  model.fit(X, y)  model.coef\_, model.intercept\_  # Визуализация регрессий  model\_a = model.coef\_[0]  model\_b = model.intercept\_  model\_y\_sk = model\_a \* X + model\_b  fig = plt.figure(figsize=(16, 9))  plt.plot(X, model\_y\_sk, linewidth='2', color='red', label=f'Модель регресии sklearn \n{model\_a} + {model\_b}x')  plt.plot(X, f\_data.dy, '--g', linewidth='2', color='blue',  label=f'Модель регресии Методичка \n{next\_w0}+ {next\_w1}x')  plt.scatter(X, y)  plt.xlabel('storage(GB)')  plt.ylabel('Price')  plt.grid()  plt.legend(prop={'size': 20}) |

Результат работы программы представлены на рисунке 7.

****

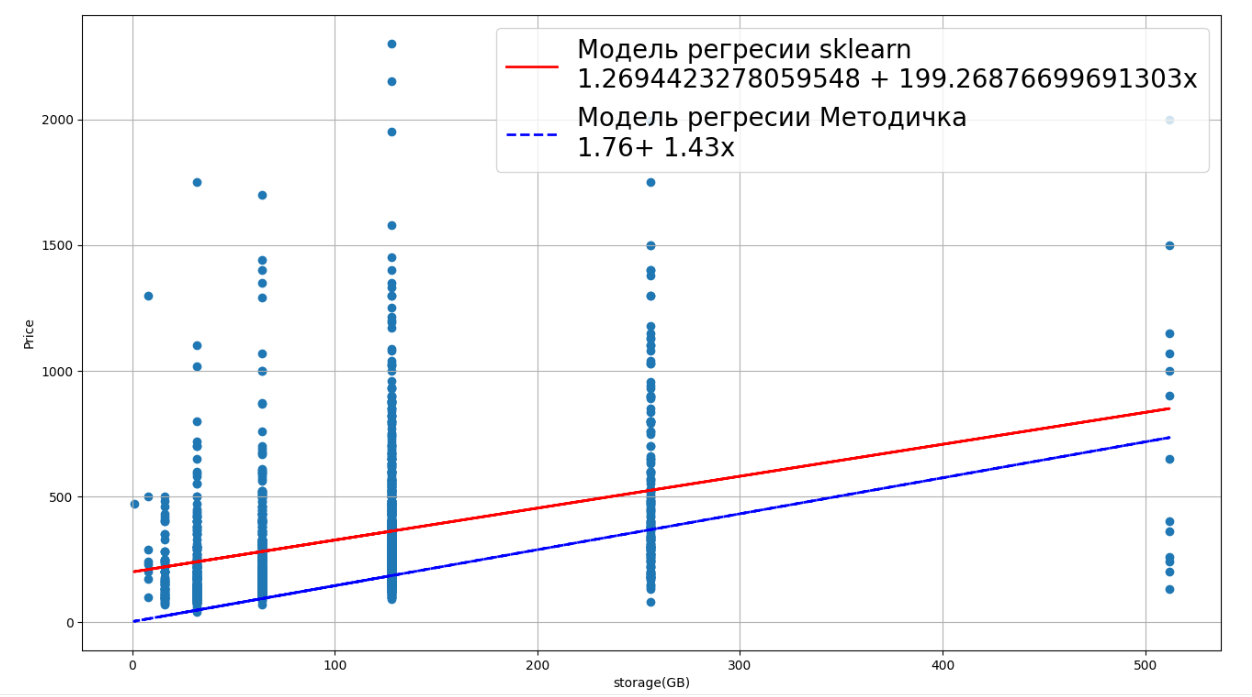
****

Рисунок 7 – Регрессия и mse

**Задание 7.**

Загрузить данные: 'insurance.csv'. Вывести и провести предобработку. Вывести список уникальных регионов.

Для решения задачи был написан код (Листинг 7)

|  |
| --- |
| Листинг 7 - Задание 7 |
| import pandas as pd  dataset = pd.read\_csv('insurance.csv')  dataset  #%%  dataset.describe()  #%%  dataset.isnull().sum()  #%%  dataset.info() |

Результат работы программы представлены на рисунке 8.

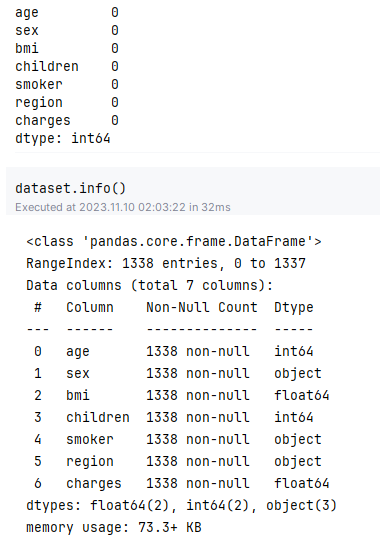
****

Рисунок 8 – Результат работы программы

**Задание 8.**

Выполнить однофакторный ANOVA тест, чтобы проверить влияние региона на индекс массы тела (BMI), используя первый способ, через библиотеку Scipy.

Для решения задачи был написан код (Листинг 8)

|  |
| --- |
| Листинг 8 - Задание 8 |
| import scipy.stats as stats  np.random.seed(12)  regions = dataset.region.unique()  frame = pd.DataFrame({'region': dataset.region, 'bmi': dataset.bmi})  frame  #%%  groups = frame.groupby('region').groups  #%%  southeast = dataset.bmi[groups["southeast"]]  southwest = dataset.bmi[groups['southwest']]  northwest = dataset.bmi[groups['northwest']]  northeast = dataset.bmi[groups['northeast']]  #%%  f\_statistic, p\_value = stats.f\_oneway(southeast, southwest, northwest, northeast)  # Вывод результатов теста  print("\nРезультаты однофакторного ANOVA-теста:")  print(f"F-статистика: {f\_statistic}")  print(f"P-значение: {format(p\_value, '.30f')}")  print(  'Поскольку p-значение меньше 0,05. Это означает, что фактор region оказывает статистически значимое влияние на bmi!!!') |

Результат работы программы представлены на рисунке 9.

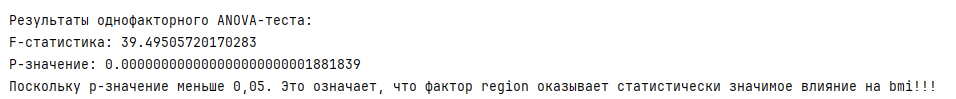
****

Рисунок 9 – Распределения BMI и Charges

**Задание 9.**

Выполнить однофакторный ANOVA тест, чтобы проверить влияние региона на индекс массы тела (BMI), используя второй способ, с помощью функции anova\_lm() из библиотеки statsmode.

Для решения задачи был написан код (Листинг 9)

|  |
| --- |
| Листинг 9 - Задание 9 |
| import statsmodels.api as sm  from statsmodels.formula.api import ols  #%%  model = ols('bmi ~ region', data=frame).fit()  anova\_result = sm.stats.anova\_lm(model, typ=2)  print(anova\_result) |

Результат работы программы представлены на рисунке 10.

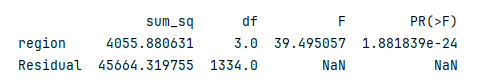


Рисунок 10 – Результат работы программы

**Задание 10.**

С помощью t критерия Стьюдента перебрать все пары. Определить поправку Бонферрони. Сделать выводы.

Для решения задачи был написан код (Листинг 10)

|  |
| --- |
| Листинг 10 - Задание 10 |
| t\_dataframe = pd.DataFrame(columns=['Группы', 'p-value', 'Гипотеза'])  pairs\_for\_comparison = []  for i in regions:  for j in regions:  if i != j and [i, j] not in pairs\_for\_comparison and [j, i] not in pairs\_for\_comparison:  pairs\_for\_comparison.append([i, j])  m = len(pairs\_for\_comparison)  for i, j in pairs\_for\_comparison:  group = f'{i} - {j}'  p\_value = stats.ttest\_ind(dataset.bmi[groups[i]], dataset.bmi[groups[j]])  if p\_value[1] > (0.05 / m):  hypothesis = 'Принимается'  else:  hypothesis = 'Отклоняется'  # new\_row = {'Группы':group, 'p-value':p\_value[1], 'Гипотеза':hypothesis}  new\_row = [group, format(p\_value[1], '.10f'), hypothesis]  t\_dataframe.loc[len(t\_dataframe.index)] = new\_row  print(group)  print('statistic', format(p\_value[0], '.10f'))  print('pvalue', format(p\_value[1], '.10f'))  print() |

Результат работы программы представлены на рисунке 11.

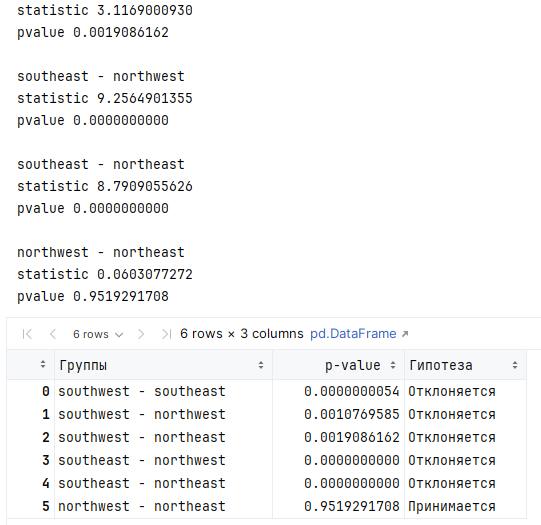
****

Рисунок 11 – Результат кода

**Задание 11.**

Выполнить пост-хок тесты Тьюки и построить график.

Для решения задачи был написан код (Листинг 11)

|  |
| --- |
| Листинг 11 - Задание 11 |
| tukey = pairwise\_tukeyhsd(endog=dataset.bmi, groups=dataset.region, alpha=0.5)  tukey.plot\_simultaneous()  plt.vlines(x=29, ymin=-0.5, ymax=4.5, color='red')  print(tukey.summary()) |

Результат работы программы представлены на рисунке 12.

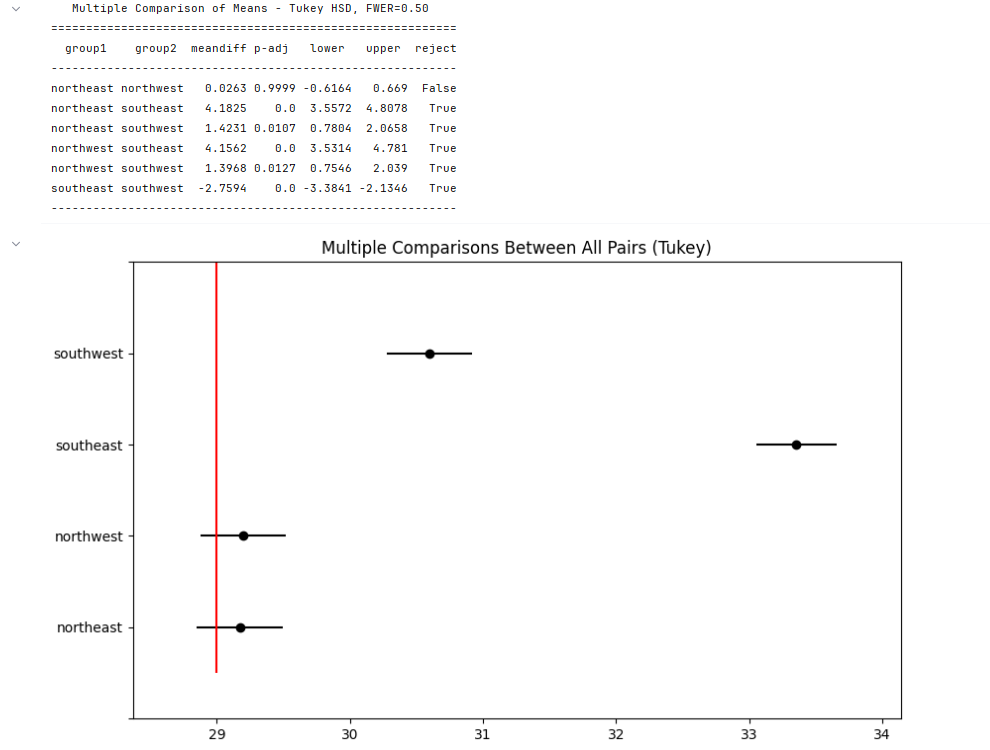


Рисунок 12 – Статистика данных через describe().

**Задание 12.**

Выполнить двухфакторный ANOVA тест, чтобы проверить влияние региона и пола на индекс массы тела (BMI), используя функцию anova\_lm() из библиотеки statsmodels.

Для решения задачи был написан код (Листинг 12)

|  |
| --- |
| Листинг 12 - Задание 12 |
| anova\_two = pd.DataFrame()  anova\_two['region'] = dataset.region  anova\_two['sex'] = dataset.sex  anova\_two['bmi'] = dataset.bmi  import statsmodels.api as sm  from statsmodels.formula.api import ols  model = ols('bmi ~ C(region) + C(sex) + C(region):C(sex)', data=anova\_two).fit()  sm.stats.anova\_lm(model, typ=2) |

Результат работы программы представлены на рисунке 13.

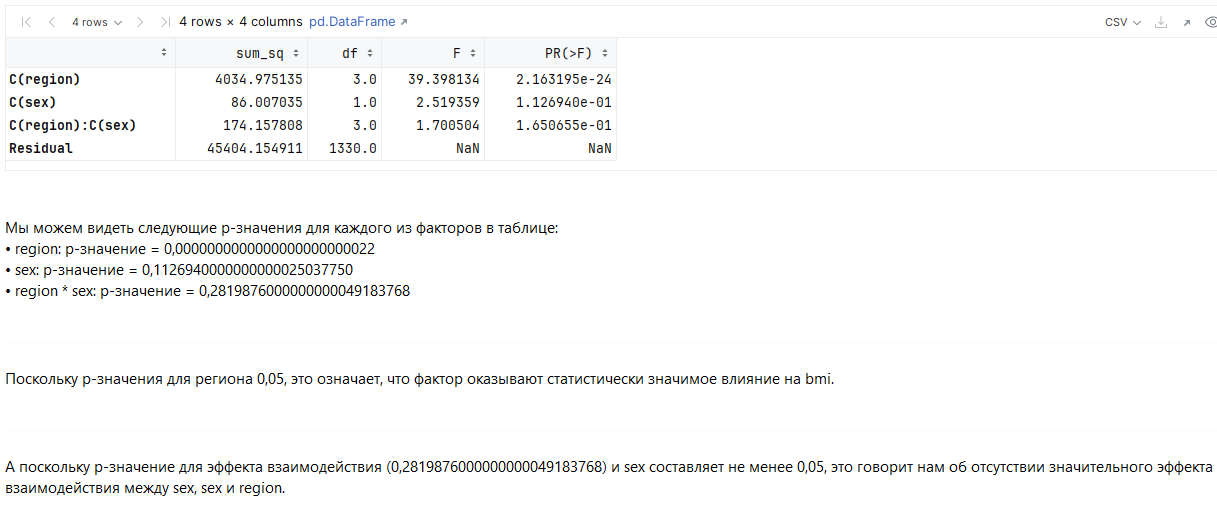


Рисунок 13 – Результат кода

**Задание 13.**

Загрузить данные из файла “bmi.csv”. Взять оттуда две выборки. Одна выборка – это индекс массы тела людей c региона northwest, вторая выборка – это индекс массы тела людей с региона southwest. Сравнить средние значения этих выборок, используя t-критерий Стьюдента. Предварительно проверить выборки на нормальность (критерий Шопиро-Уилка) и на гомогенность дисперсии (критерий Бартлетта).

Для решения задачи был написан код (Листинг 13)

|  |
| --- |
| Листинг 13 - Задание 13 |
| anova\_two['combo'] = anova\_two['region'] + ' \ ' + anova\_two['sex']  anova\_two  tukey = pairwise\_tukeyhsd(endog = anova\_two['bmi'], groups=anova\_two['combo'], alpha=0.05)  tukey.plot\_simultaneous()  print(tukey.summary()) |

Результат работы программы представлены на рисунке 13.

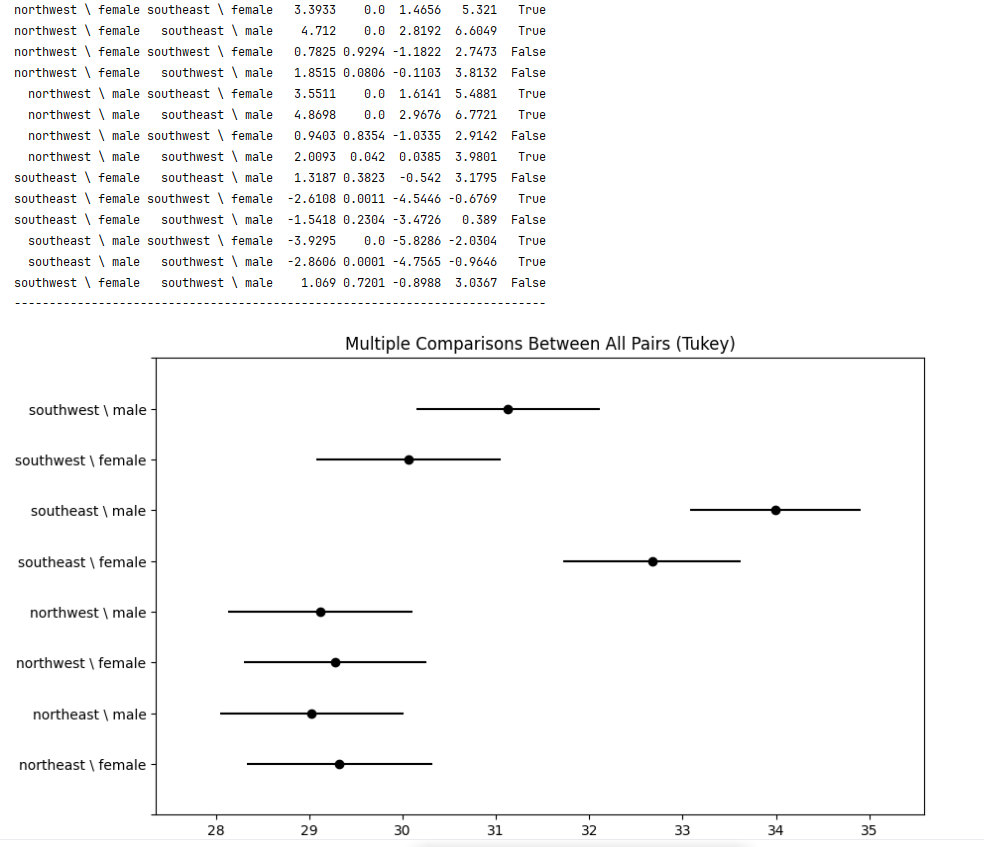


Рисунок 18 – Результат кода