МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)

Кафедра прикладной математики

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 4

по дисциплине «Технологии и инструментарий анализа больших данных»

Выполнил студент группы ИКБО-20-21 Фомичев Р.А.

Проверил ассистент кафедры ПМ ИИТ Тетерин Н.Н.

Москва 2024

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

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

Код программы представлен на рисунке 1.

```
data = {
    'День': ['Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница']
    'Улица': [80, 98, 75, 91, 78],
    'Гараж': [100, 82, 105, 89, 102]
}

df = pd.DataFrame(data)
correlation = df['Улица'].corr(df['Гараж'])
print(f'Корреляция по Пирсону между улицей и гаражом: {correlation}')
```

Рисунок 1 – Код программы

Вывод программы представлен на рисунке 2.

```
Корреляция по Пирсону между улицей и гаражом: -0.9999999999999998
```

Рисунок 2 – Вывод программы

1.2. Построить диаграмму рассеяния для вышеупомянутых переменных.

Код программы представлен на рисунке 3.

```
plt.scatter(df['Улица'], df['Гараж'])
plt.xlabel('Автомобили на Улице')
plt.ylabel('Автомобили в Гараже')
plt.title('Диаграмма рассеяния между Улицей и Гаражом')
plt.show()
```

Рисунок 3 – Код программы

Вывод программы представлен на рисунке 4.

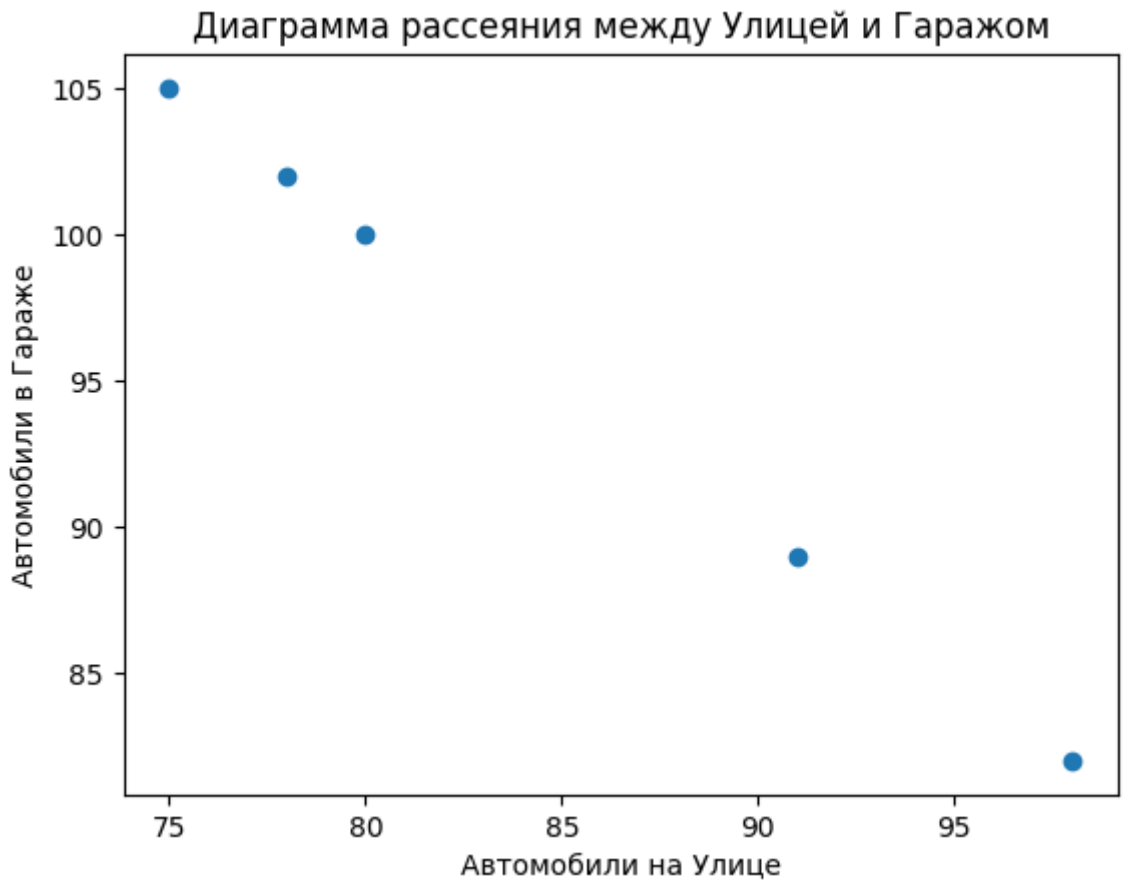


Рисунок 4 – Вывод программы

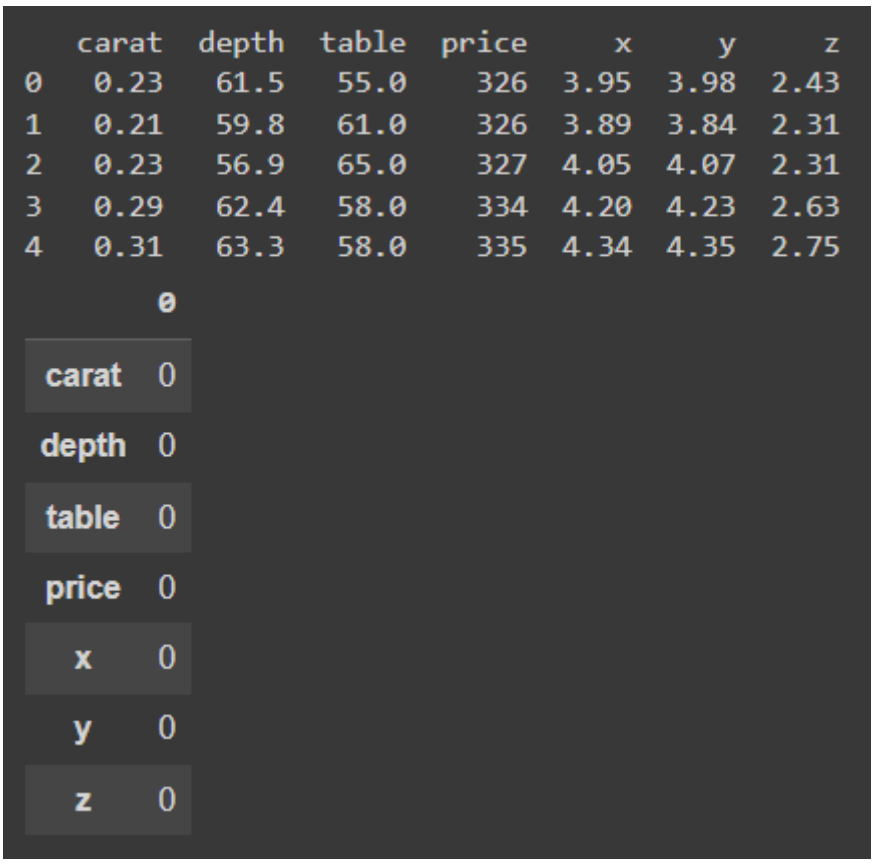
2. Найти и выгрузить данные. Вывести, провести предобработку и описать признаки.

Код программы представлен на рисунке 5.

```
df = pd.read_csv('sample_data/diamonds_prices.csv')
df = df.drop(["cut", "color", "clarity"], axis=1).drop(df.columns [0], axis=1)
print(df.head())
df.isnull().sum()
```

Рисунок 5 – Код программы

Вывод программы представлен на рисунке 6.



	carat	depth	table	price	x	y	z
0	0.23	61.5	55.0	326	3.95	3.98	2.43
1	0.21	59.8	61.0	326	3.89	3.84	2.31
2	0.23	56.9	65.0	327	4.05	4.07	2.31
3	0.29	62.4	58.0	334	4.20	4.23	2.63
4	0.31	63.3	58.0	335	4.34	4.35	2.75

	0
carat	0
depth	0
table	0
price	0
x	0
y	0
z	0

Рисунок 6 – Код программы

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

Код программы представлен на рисунке 7.

```
corr_matrix = df.corr()
print(corr_matrix)

corr_target = corr_matrix["price"].sort_values(ascending=False)
print(corr_target)
```

Рисунок 7 – Код программы

Вывод программы представлен на рисунке 8.

```
          carat     depth     table     price         x         y         z
carat  1.000000  0.028234  0.181602  0.921591  0.975093  0.951721  0.953387
depth  0.028234  1.000000 -0.295798 -0.010630 -0.025289 -0.029340  0.094927
table  0.181602 -0.295798  1.000000  0.127118  0.195333  0.183750  0.150915
price  0.921591 -0.010630  0.127118  1.000000  0.884433  0.865419  0.861249
x      0.975093 -0.025289  0.195333  0.884433  1.000000  0.974701  0.970771
y      0.951721 -0.029340  0.183750  0.865419  0.974701  1.000000  0.952005
z      0.953387  0.094927  0.150915  0.861249  0.970771  0.952005  1.000000
price    1.000000
carat    0.921591
x        0.884433
y        0.865419
z        0.861249
table    0.127118
depth   -0.010630
```

Рисунок 8 – Вывод программы

2.2. Реализовать регрессию вручную, отобразить наклон, сдвиг и MSE.

Код программы представлен на рисунке 9.

```python
X = df["carat"].values
y = df["price"].values

def mserror(X, w1, w0, y):
    y_pred = w1 * X[:] + w0
    return np.sum((y - y_pred) ** 2) / len(y_pred)


def gr_mserror(X, w1, w0, y):
    y_pred = w1 * X[:] + w0
    return np.array([2 / len(X) * np.sum((y - y_pred)) * (-1),
                     2 / len(X) * np.sum((y - y_pred) * (-X[:]))])


eps = 0.0001
w1 = 0
w0 = 0

learning_rate = 0.1

next_w1 = w1
next_w0 = w0
n = 100000
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"Итерации: {i}")
    print(f"Текущиая точка {cur_w1, cur_w0}| Следующая точка {next_w1, next_w0}")
    print(f"MSE {mserror(X, cur_w1, cur_w0, y)}")

    if (abs(cur_w1 - next_w1) <= eps) and (abs(cur_w0 - next_w0) <= eps):
        break
```

Рисунок 9 – Код программы

Вывод программы представлен на рисунке 10.

```
np.float64(-2256.391238024158))
MSE 2397833.972704674
Итерации: 535
Текущиая точка (np.float64(7756.4320047234305), np.float64(-2256.391238024158))| Следующая точка (np.float64(7756.432112536326),
(-2256.39133687759))
MSE 2397833.9727044515
Итерации: 536
Текущиая точка (np.float64(7756.432112536326), np.float64(-2256.39133687759))| Следующая точка (np.float64(7756.432217551574), np
(-2256.3914331658652))
MSE 2397833.9727042406
Итерации: 537
Текущиая точка (np.float64(7756.432217551574), np.float64(-2256.3914331658652))| Следующая точка (np.float64(7756.4323198417715),
np.float64(-2256.391526955547))
MSE 2397833.97270404
Итерации: 538
Текущиая точка (np.float64(7756.4323198417715), np.float64(-2256.391526955547))| Следующая точка (np.float64(7756.432419477631),
(-2256.391618311471))
MSE 2397833.9727038504
```

Рисунок 10 – Вывод программы

2.3. Визуализировать регрессию на графике.

Код программы представлен на рисунке 11.

```
fig = plt.figure(figsize=(10,6))

x = np.arange(0, 5)

man_model = cur_w1 * x + cur_w0

plt.plot(x, man_model, linewidth=2, color = "r", label=f'вручную = {cur_w1:.2f}x + {cur_w0:.2f}')
plt.scatter(X, y, label='Исходные данные')
plt.grid()
plt.xlabel("Карат")
plt.ylabel("Цена")
plt.legend (prop={'size': 16})
plt.show()
```

Рисунок 11 – Код программы

Вывод программы представлен на рисунке 12.

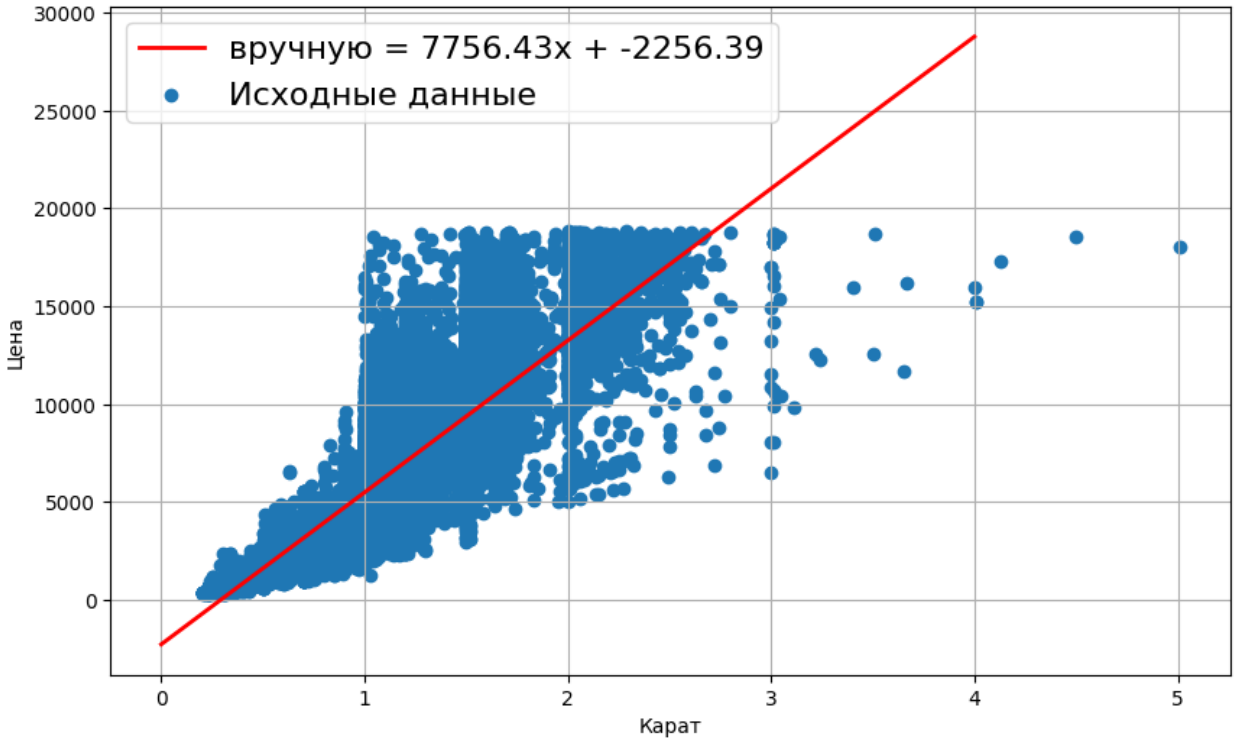


Рисунок 12 – Вывод программы

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

Код программы представлен на рисунке 13.

```
df = pd.read_csv('sample_data/insurance.csv')
unique_regions = df['region'].unique()
print("Уникальные регионы: ", unique_regions)
```

Рисунок 13 – Код программы

Вывод программы представлен на рисунке 14.

```
Уникальные регионы:  ['southwest' 'southeast' 'northwest' 'northeast']
```

Рисунок 14 – Вывод программы

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

Код программы представлен на рисунке 15.

```
regions = df['region'].unique()

grouped_data = [df['bmi'][df['region'] == region] for region in regions]
f_statistic, p_value = scipy.stats.f_oneway(*grouped_data)
print("Результаты однофакторного ANOVA теста:")
print("F-статистика:", f_statistic)
print("p-значение:", p_value)
```

Рисунок 15 – Код программы

Вывод программы представлен на рисунке 16.

```
Результаты однофакторного ANOVA теста:
F-статистика: 39.49505720170283
p-значение: 1.881838913929143e-24
```

Рисунок 16 – Вывод программы

3.2. Выполнить однофакторный ANOVA тест, чтобы проверить влияние региона на индекс массы тела (BMI), используя второй способ, с помощью функции anova_lm() из библиотеки statsmodels.

Код программы представлен на рисунке 17.

```
model = ols('bmi ~ region', data=df).fit()
anova_table = sm.stats.anova_lm(model)
print(anova_table)
```

Рисунок 17 – Код программы

Вывод программы представлен на рисунке 18.

```
                df        sum_sq      mean_sq          F        PR(>F)
region         3.0   4055.880631  1351.960210  39.495057  1.881839e-24
Residual    1334.0  45664.319755    34.231124        NaN           NaN
```

Рисунок 18 – Вывод программы

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

Код программы представлен на рисунке 19.

```
regions = df['region'].unique()
results = []
for i in range(len(regions)):
    for j in range(i + 1, len(regions)):
        region1 = regions[i]
        region2 = regions[j]
        group1 = df['bmi'][df['region'] == region1]
        group2 = df['bmi'][df['region'] == region2]
        t_statistic, p_value = ttest_ind(group1, group2)
        results.append((region1, region2, t_statistic, p_value))

# Поправка Бонферрони
p_values = [result[3] for result in results]
corrected_p_values = multipletests(p_values, method='bonferroni')[1]
print("Значимые различия между регионами с поправкой Бонферрони:")
for i, result in enumerate(results):
    region1, region2, t_statistic, p_value = result
    corrected_p_value = corrected_p_values[i]
    print(
        f"{region1} vs {region2}: t-статистика = {t_statistic}, p-значение = {p_value}, Поправленное p-значение (Бонферрони) = {corrected_p_value}")
```

Рисунок 19 – Код программы

Вывод программы представлен на рисунке 20.

```
Значимые различия между регионами с поправкой Бонферрони:
southwest vs southeast: t-статистика = -5.908373821545118, p-значение = 5.4374009639680636e-09, Поправленное p-значение (Бонферрони) = 3.2624405783808385e-08
southwest vs northwest: t-статистика = 3.2844171500398582, p-значение = 0.001076958496307695, Поправленное p-значение (Бонферрони) = 0.006461750977846171
southwest vs northeast: t-статистика = 3.1169000930045923, p-значение = 0.0019086161671573072, Поправленное p-значение (Бонферрони) = 0.011451697002943843
southeast vs northwest: t-статистика = 9.25649013552548, p-значение = 2.643571405230106e-19, Поправленное p-значение (Бонферрони) = 1.5861428431380637e-18
southeast vs northeast: t-статистика = 8.790905562598699, p-значение = 1.186014937424813e-17, Поправленное p-значение (Бонферрони) = 7.116089624548878e-17
northwest vs northeast: t-статистика = 0.0603077271832931 85, p-значение = 0.951929170821864, Поправленное p-значение (Бонферрони) = 1.0
```

Рисунок 20 – Вывод программы

3.4. Выполнить пост-хок тесты Тьюки и построить график.

Код программы представлен на рисунке 21.

```
results = pairwise_tukeyhsd(df['bmi'], df['region'])
print(results)

results.plot_simultaneous()
plt.show()
```

Рисунок 21 – Код программы

Вывод программы представлен на рисунках 22, 23.

```
   Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
  group1    group2  meandiff p-adj   lower   upper  reject
-----------------------------------------------------
northeast northwest   0.0263 0.9999 -1.1552  1.2078  False
northeast southeast   4.1825    0.0   3.033   5.332   True
northeast southwest   1.4231 0.0107  0.2416  2.6046   True
northwest southeast   4.1562    0.0  3.0077  5.3047   True
northwest southwest   1.3968 0.0127  0.2162  2.5774   True
southeast southwest  -2.7594    0.0 -3.9079 -1.6108   True
-----------------------------------------------------
```

Рисунок 22 – Вывод программы

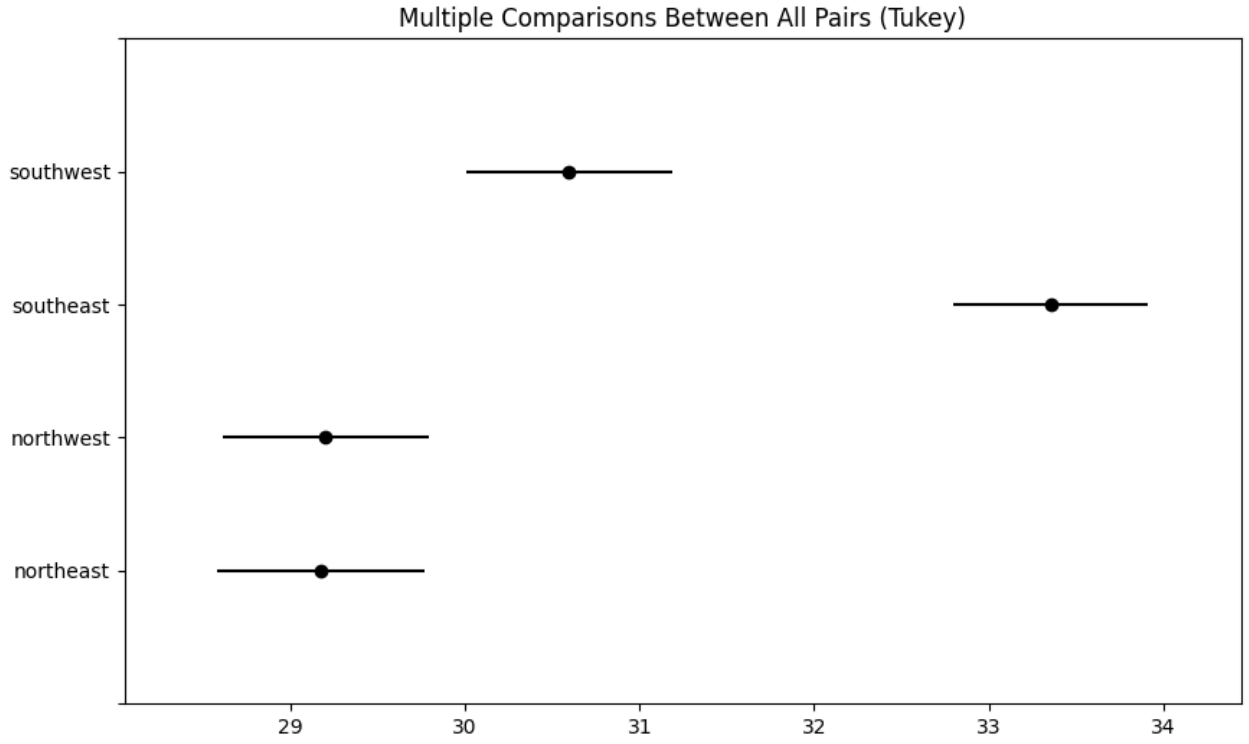


Рисунок 23 – Вывод программы

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

Код программы представлен на рисунке 24.

```
model = ols('bmi ~ region + sex', data=df).fit()

print("Результаты двухфакторного ANOVA теста регион и пол:")
anova_table = anova_lm(model)
print(anova_table)
```

Рисунок 24 – Код программы

Вывод программы представлен на рисунке 25.

```
Результаты двухфакторного ANOVA теста регион и пол:
               df        sum_sq      mean_sq          F        PR(>F)
region        3.0   4055.880631  1351.960210  39.539923  1.773031e-24
sex           1.0     86.007035    86.007035   2.515393  1.129767e-01
Residual   1333.0  45578.312720    34.192283        NaN           NaN
```

Рисунок 25 – Вывод программы

3.6. Выполнить пост-хок тесты Тьюки и построить график.

Код программы представлен на рисунке 26.

```
tukey_results = pairwise_tukeyhsd(df['bmi'], df['region'])
print(tukey_results)
tukey_results.plot_simultaneous()
plt.show()
```

Рисунок 26 – Код программы

Вывод программы представлен на рисунках 27, 28.

```
   Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
  group1    group2  meandiff p-adj   lower   upper  reject
-----------------------------------------------------
northeast northwest   0.0263 0.9999 -1.1552  1.2078  False
northeast southeast   4.1825    0.0   3.033   5.332   True
northeast southwest   1.4231 0.0107  0.2416  2.6046   True
northwest southeast   4.1562    0.0  3.0077  5.3047   True
northwest southwest   1.3968 0.0127  0.2162  2.5774   True
southeast southwest  -2.7594    0.0 -3.9079 -1.6108   True
-----------------------------------------------------
```

Рисунок 27 – Вывод программы

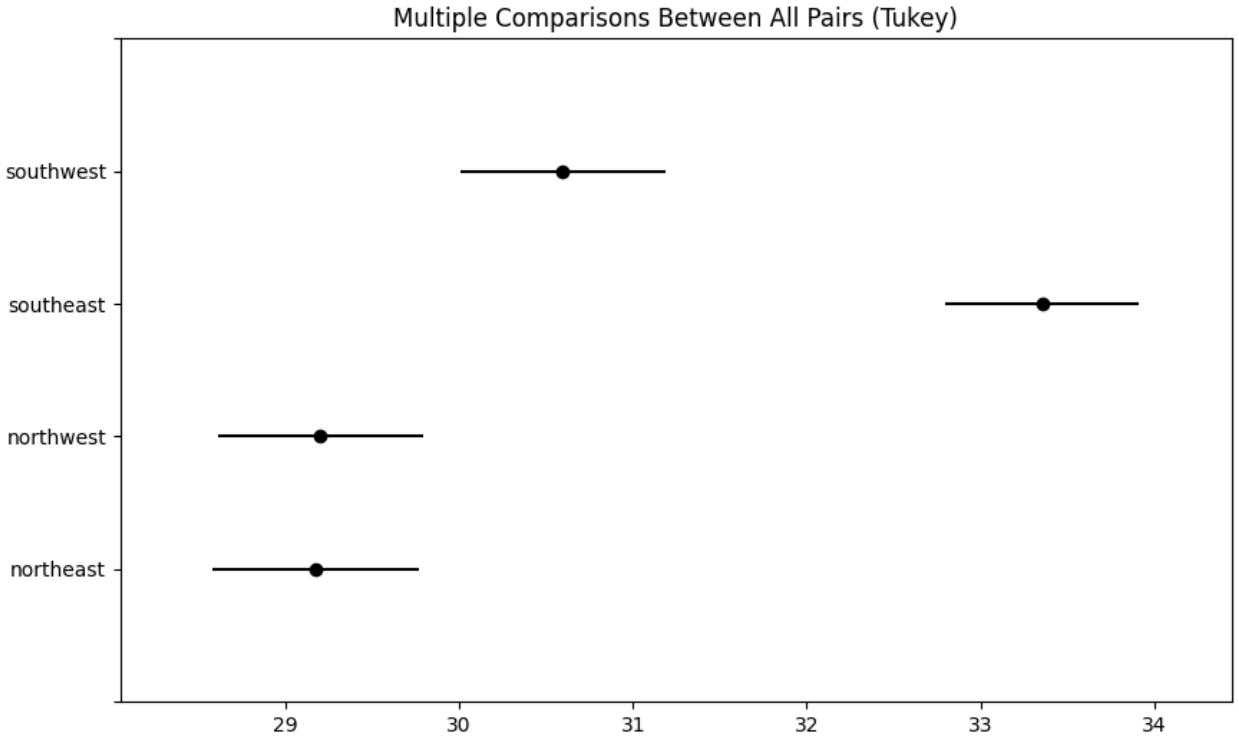


Рисунок 28 – Вывод программы