Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Расчётно-графическая работа №4

«Построение линейной модели»

по дисциплине «Математическая статистика»

Вариант 4

Выполнили:

студенты группы P3213

Поленов К. А.

Пименова Е. А.

Преподаватель:

Милованович Е.В.

Санкт-Петербург

2025

ОГЛАВЛЕНИЕ

Цель работы:

Научиться строить множественную линейную регрессию, оценивать параметры модели, рассчитывать доверительные интервалы, проверять статистические гипотезы и оценивать качество модели при помощи коэффициента детерминации и F-статистики.

Задание 1

1. Постройте линейную модель, где в качестве независимых переменных выступают «жилая» площадь, «sqft_lot», «sqft_above» (вместе со свободным коэффициентом), зависимой – цена на недвижимость.
2. Проверьте следующие подозрения:
   - Чем больше «жилая» площадь, тем больше цена
   - Цена зависит от «sqft_lot»
   - Проверьте гипотезу H_0 о равенстве одновременно нулю коэффициентов при «жилой» площади и «sqft_above» против альтернативы $H_1 = \overline{H_0}$

Теория:

Модель множественной линейной регрессии имеет вид:

y = Xβ + ε, где:

- y — вектор наблюдаемых значений зависимой переменной (цены),
- X — матрица признаков:
    - x1 - жилая площадь (sqrt_living)
    - x2 - площадь участка (sqrt_lof)
    - x3 - площадь над землей (sqrt_above)
- β — вектор коэффициентов модели
- ε — вектор случайных ошибок.

Оценка коэффициентов методом наименьших квадратов (МНК):

$$\hat{\beta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

Доверительный интервал для коэффициента β□:

$$\hat{\beta}_j \pm t\left(\frac{\alpha}{2}, n-k\right) \cdot \boldsymbol{SE}(\hat{\beta}_j)$$

где:

- $\hat{\beta}_j$ — оценка коэффициента
- $t(\alpha/2, n-k)$ — квантиль t-распределения с $n-k$ степенями свободы
- $\boldsymbol{SE}(\hat{\beta}_j)$ — стандартная ошибка оценки

Качество модели оценивается с помощью коэффициента детерминации R^2:

$$R^2 = 1 - \frac{\boldsymbol{RSS}}{\boldsymbol{TSS}}$$

где:

- $\boldsymbol{RSS} = \sum(y_i - \hat{y}_i)^2$ — сумма квадратов остатков
- $\boldsymbol{TSS} = \sum(y_i - \bar{y})^2$ — общая сумма квадратов отклонений от среднего

А также с помощью F-статистики для проверки значимости нескольких коэффициентов одновременно.

$$F = \frac{(R\hat{\beta} - q)^T [R(X^T X)^{-1} R^T]^{-1} (R\hat{\beta} - q)}{r \cdot \hat{\sigma}^2}$$

Где:

- R — матрица ограничений (например, зануляем β_1 и β_3)
- q — вектор нулей (что именно проверяем)
- r — число проверяемых ограничений
- $\hat{\sigma}^2$ — остаточная дисперсия

Исходные данные:

Используется датасет kc_house_data.csv, содержащий информацию о домах в округе Кинг, штат Вашингтон.
В модель включены следующие признаки:
- sqft_living — жилая площадь
- sqft_lot — площадь участка
- sqft_above — площадь над землёй
Целевая переменная — price (цена дома).

Ход работы:

1. Загрузка и подготовка данных.
2. Формирование матрицы X с добавлением единичного столбца.
3. Оценка коэффициентов по формуле МНК.
4. Расчет доверительных интервалов.
5. Вычисление остатков и коэффициента детерминации R^2.
6. Проверка статистических гипотез о значимости переменных.
7. Визуализация: сравнение предсказанных и фактических значений.

Код программы:

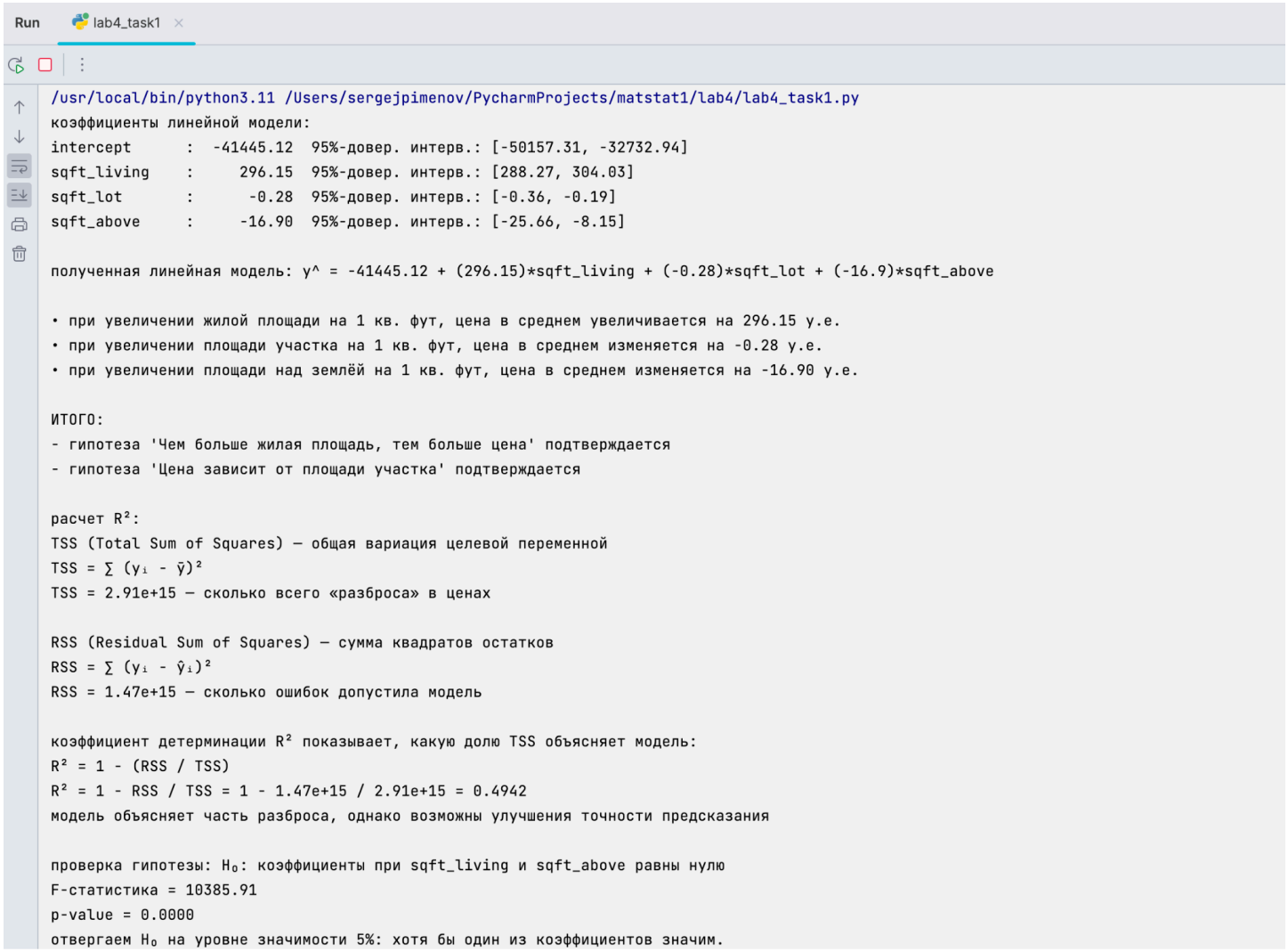
```
import pandas as pd
import numpy as np
from scipy.stats import t, f

# Загрузка данных
df = pd.read_csv("kc_house_data.csv")

# Выбор переменных
X_raw = df[['sqft_living', 'sqft_lot', 'sqft_above']]
y = df['price'].to_numpy().reshape(-1, 1)
```

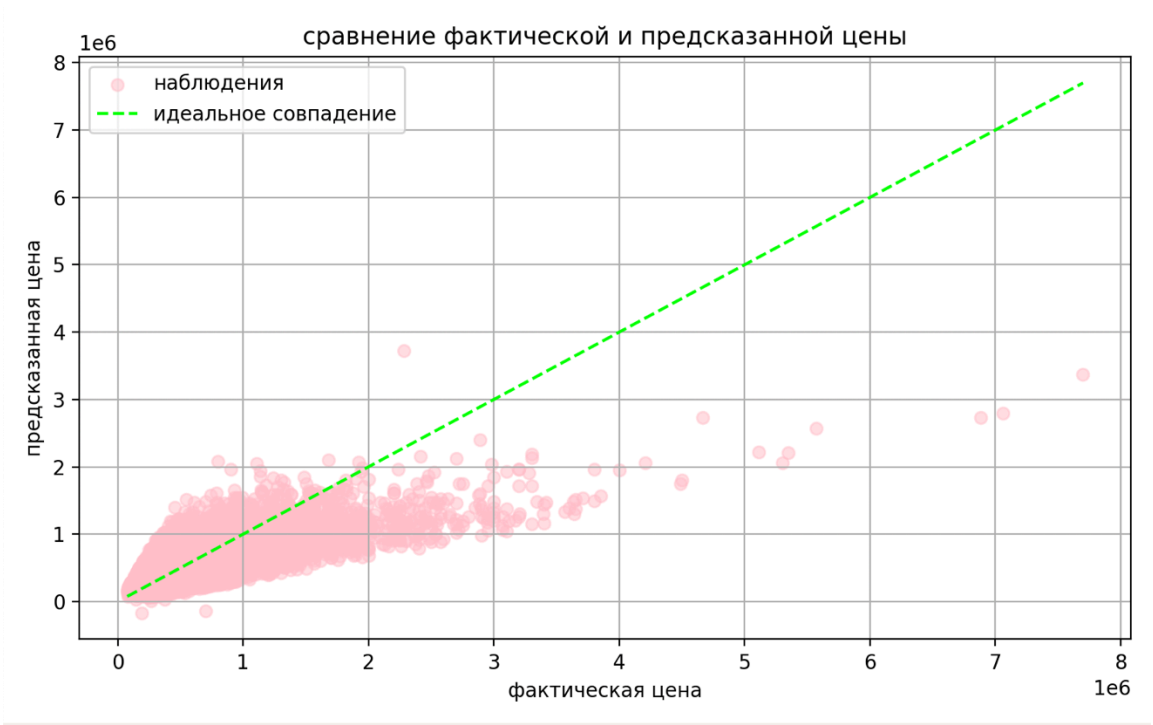
```
# Формирование матрицы X с единицами
X = np.hstack((np.ones((X_raw.shape[0], 1)), X_raw.to_numpy()))

# Оценка коэффициентов: b = (X^T X)^-1 X^T y
XtX = X.T @ X
XtX_inv = np.linalg.inv(XtX)
Xty = X.T @ y
beta_hat = XtX_inv @ Xty

# Предсказания и остатки
y_pred = X @ beta_hat
residuals = y - y_pred

# Остаточная дисперсия
n, k = X.shape
rss = np.sum(residuals ** 2)
sigma2 = rss / (n - k)

# Стандартные ошибки и доверительные интервалы
var_b = sigma2 * XtX_inv
se_b = np.sqrt(np.diag(var_b))
t_crit = t.ppf(0.975, df=n - k)
conf_ints = np.column_stack((beta_hat.flatten() - t_crit * se_b,
                             beta_hat.flatten() + t_crit * se_b))

# R^2
tss = np.sum((y - np.mean(y)) ** 2)
r_squared = 1 - rss / tss

# Проверка гипотез
# H0: b1 = 0, b2 = 0 одновременно (sqft_living и sqft_above)
R = np.array([
   [0, 1, 0, 0],
   [0, 0, 0, 1]
])
q = np.array([[0], [0]])
Rb = R @ beta_hat
middle = np.linalg.inv(R @ XtX_inv @ R.T)
F_stat = ((Rb - q).T @ middle @ (Rb - q)) / R.shape[0] / sigma2
p_value = 1 - f.cdf(F_stat, dfn=R.shape[0], dfd=n - k)

# Вывод
print("коэффициенты линейной модели:")
for name, coef, interval in zip(['intercept', 'sqft_living', 'sqft_lot', 'sqft_above'],
beta_hat.flatten(), conf_ints):
   print(f"{name:15}: {coef:10.2f}  95%-довер. интерв.: [{interval[0]:.2f},
{interval[1]:.2f}]")

# Автоматический вывод по модели
names = ['intercept', 'sqft_living', 'sqft_lot', 'sqft_above']
coef_rounded = np.round(beta_hat.flatten(), 2)
print()
```

```
print(f"полученная линейная модель: y^ = {coef_rounded[0]} + ({coef_rounded[1]})*sqft_living
+ ({coef_rounded[2]})*sqft_lot + ({coef_rounded[3]})*sqft_above\n")

print(f"• при увеличении жилой площади на 1 кв. фут, цена в среднем увеличивается на
{coef_rounded[1]:.2f} у.е.")
print(f"• при увеличении площади участка на 1 кв. фут, цена в среднем изменяется на
{coef_rounded[2]:.2f} у.е.")
print(f"• при увеличении площади над землёй на 1 кв. фут, цена в среднем изменяется на
{coef_rounded[3]:.2f} у.е.")
print("\nИТОГО:")
print(f"- гипотеза 'Чем больше жилая площадь, тем больше цена' {'подтверждается' if
beta_hat[1][0] > 0 and (2 * (1 - t.cdf(abs(beta_hat[1][0]/se_b[1]), df=n-k))) < 0.05 else
'Не подтверждается'}")
print(f"- гипотеза 'Цена зависит от площади участка' {'подтверждается' if (2 * (1 -
t.cdf(abs(beta_hat[2][0]/se_b[2]), df=n-k)) < 0.05) else 'Не подтверждается'}")

# TSS: Total Sum of Squares — общая сумма квадратов отклонений от среднего - насколько
сильно разбросаны реальные значения
tss = np.sum((y - np.mean(y)) ** 2)

# RSS: Residual Sum of Squares — сумма квадратов остатков - то, что модель не объяснила
(ошибки)
rss = np.sum((y - y_pred) ** 2)

# R²: какая доля дисперсии объясняется моделью
r_squared = 1 - (rss / tss)

# Альтернативно: можно распечатать части по отдельности
print("\nрасчет R²:")
print(f"TSS (общая дисперсия): {tss:.2e}")
print(f"RSS (необъяснённая дисперсия): {rss:.2e}")
print(f"R² = 1 - RSS / TSS = 1 - {rss:.2e} / {tss:.2e} = {r_squared:.4f}")

print(f"коэффициент детерминации R² = {r_squared:.4f}")
print("модель объясняет часть разброса, однако возможны улучшения точности предсказания")
#Вывод по F-статистике (гипотеза H0: b1 = b3 = 0)
print("\nпроверка гипотезы: H₀: коэффициенты при sqft_living и sqft_above равны нулю")

print(f"F-статистика = {F_stat[0][0]:.2f}")
print(f"p-value = {p_value[0][0]:.4f}")

if p_value[0][0] < 0.05:
   print("отвергаем H₀ на уровне значимости 5%: хотя бы один из коэффициентов значим.")
else:
   print("недостаточно оснований отвергнуть H₀: переменные могут быть незначимы.")


import matplotlib.pyplot as plt


plt.rcParams['font.family'] = 'DejaVu Sans'  # чтобы кириллица отобразилась

# 1. Реальные значения vs Предсказания
```

```
plt.figure(figsize=(8, 5))
plt.scatter(y, y_pred, alpha=0.5, label='наблюдения', color="pink")
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', color="lime",label='идеальное
совпадение')
plt.xlabel("фактическая цена")
plt.ylabel("предсказанная цена")
plt.title("сравнение фактической и предсказанной цены")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig("task1_graphic.png")
plt.show()
```

Результаты:




сравнение фактической и предсказанной цены
1e6
наблюдения
идеальное совпадение
предсказанная цена
фактическая цена
1e6
0
1
2
3
4
5
6
7
8

Задание 2

Задание 2

Для каждого варианта требуется проверить гипотезу о равенстве средних на каждом уровне фактора с помощью модели однофакторного дисперсионного анализа

Указание: реализовать самим

Вариант 1. В файле https://drive.google.com/file/d/1CSCheMzjberRwgcf90BBu-J6uxMg-Qf7/view представлены данные об ирисках. Фактор – подвид. Выходная переменная – суммарная площадь (точнее оценка площади) чашелистика и лепестка.

Вариант 2. В файле https://drive.google.com/file/d/14L_y0LOAebuuqh8PllOw64cJQwVkmlV6/view представлены данные о сдаче экзаменов. Фактор – этническая/национальная группа. Выходная переменная – суммарный балл за все три экзамена.

Вариант 3. В файле https://drive.google.com/file/d/1gzPRqj7gZetjsipo3xpogYGL76enZDNO/view приведены данные о некоторых привычках и физиологических/антропологических показателях. Фактор – курит/не курит. Выходная переменная – индекс массы тела

Вариант 4. В файле https://drive.google.com/file/d/1O4rFr9xg9aFmkjx4-hl_XOc5O9q65_EW/view приведены данные о мобильных телефонах. Фактор – ценовая категория, выходная переменная – емкость аккумулятора

Цель работы: Провести однофакторный дисперсионный анализ (ANOVA) для проверки гипотезы о равенстве средних значений емкости аккумулятора между различными ценовыми категориями мобильных телефонов.

Используемые данные: Данные об емкости аккумуляторов мобильных телефонов, сгруппированные по ценовым категориям

Методология:

Для проведения однофакторного дисперсионного анализа были реализованы следующие этапы и использованы соответствующие формулы:

1. Расчет среднего значения (Mean):
   Среднее значение для выборки X=x1,x2,…,xn вычисляется по формуле:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

   где n - количество элементов в выборке, а $\sum_{i=1}^{n} x_i$ - сумма всех элементов выборки.

2. Расчет межгрупповой суммы квадратов (Sum of Squares Between, SSB):

$$SSB = \sum_{j=1}^{k} n_j(\bar{x}_j - \bar{x}\,\text{общ})^2$$

где:

- k - количество групп (ценовых категорий).
- n_j - количество наблюдений в j-й группе.
- $\bar{x}_j$ - среднее значение в j-й группе.
- $\bar{x}\,\text{общ}$ - общее среднее значение по всем наблюдениям.

3. Расчет внутригрупповой суммы квадратов (Sum of Squares Within, SSW):

$$SSW = \sum_{j=1}^{k} \sum_{i=1}^{n_i} (x_{ij} - x_j)^2$$

где:

- x_{ij}- i-е наблюдение в j-й группе.

4. Расчет степеней свободы между группами (Degrees of Freedom Between, df между):

$$df\ \text{между} = k - 1$$

5. Расчет степеней свободы внутри групп (Degrees of Freedom Within, df внутри):

$$df\ \text{внутри} = N - k$$

где N - общее количество наблюдений ($\sum_{j=1}^{k} n_j$).

6. Расчет среднего квадрата между группами (Mean Square Between, MSB):

$$MSB = \frac{SSB}{df\ \text{между}}$$

7. Расчет среднего квадрата внутри групп (Mean Square Within, MSW):

$$MSW = \frac{SSW}{df\ \text{внутри}}$$

8. Расчет F-статистики (F-statistic):

$$F = \frac{MSB}{MSW}$$

9. Расчет p-значения (p-value):
   P-значение представляет собой вероятность получить наблюдаемое значение F-статистики (или более экстремальное) при условии, что нулевая гипотеза верна. Оно определяется на основе F-распределения с df между степенями свободы в числителе и df внутри степенями свободы в знаменателе. В программе для расчета p-значения использовалась функция `stats.f.sf` из библиотеки `scipy.stats`:
   $p - value = P(F(df \text{ внутри}, df \text{ между}) > F_{\text{набл}})$

   где $F_{\text{набл}}$ - вычисленное значение F-статистики.

Визуализация данных:

Для наглядного представления данных были использованы следующие типы графиков:

1. **Ящик с усами (Boxplot):** Отображает распределение емкости аккумулятора для каждой ценовой категории, показывая медиану, квартили и выбросы.
2. **Столбчатая диаграмма со стандартным отклонением (Bar plot with standard deviation):** Показывает среднее значение емкости аккумулятора для каждой ценовой категории, а "усы" ошибки представляют стандартное отклонение, демонстрируя разброс данных вокруг среднего.

Код программы:

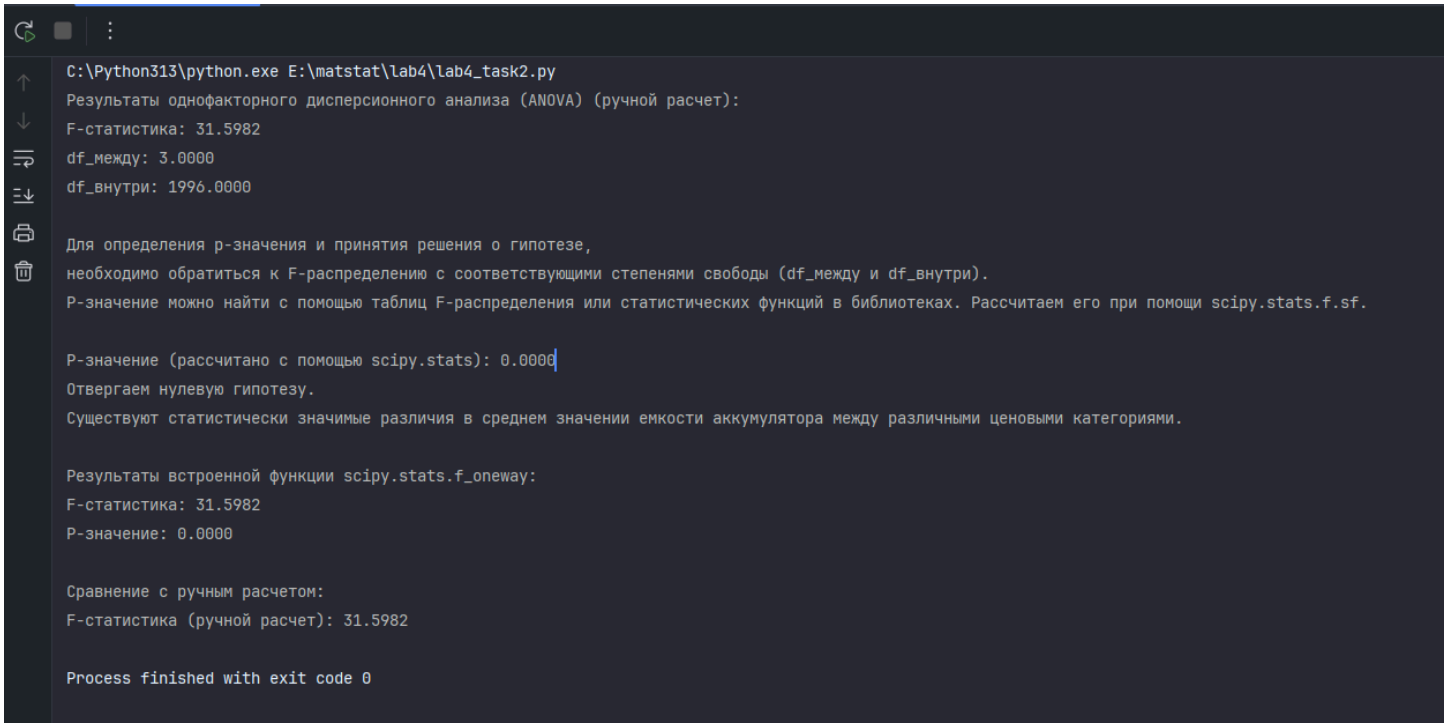
```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from scipy import stats


def calculate_mean(data):
    """Вычисляет среднее значение списка чисел."""
    return sum(data) / len(data)

def calculate_ss_between(overall_mean, group_means, group_sizes):
    """Вычисляет межгрупповую сумму квадратов (SSB)."""
    ss_between = 0
    for i in range(len(group_means)):
        ss_between += group_sizes[i] * (group_means[i] - overall_mean)**2
    return ss_between

def calculate_ss_within(groups):
    """Вычисляет внутригрупповую сумму квадратов (SSW)."""
    ss_within = 0
    for group in groups:
        group_mean = calculate_mean(group)
        for x in group:
            ss_within += (x - group_mean)**2
    return ss_within

def calculate_df_between(num_groups):
    """Вычисляет степени свободы между группами (df_between)."""
    return num_groups - 1

def calculate_df_within(total_observations, num_groups):
    """Вычисляет степени свободы внутри групп (df_within)."""
    return total_observations - num_groups

def calculate_ms_between(ss_between, df_between):
    """Вычисляет средний квадрат между группами (MSB)."""
    return ss_between / df_between

def calculate_ms_within(ss_within, df_within):
    """Вычисляет средний квадрат внутри групп (MSW)."""
    return ss_within / df_within

def calculate_f_statistic(ms_between, ms_within):
    """Вычисляет F-статистику."""
    return ms_between / ms_within
```

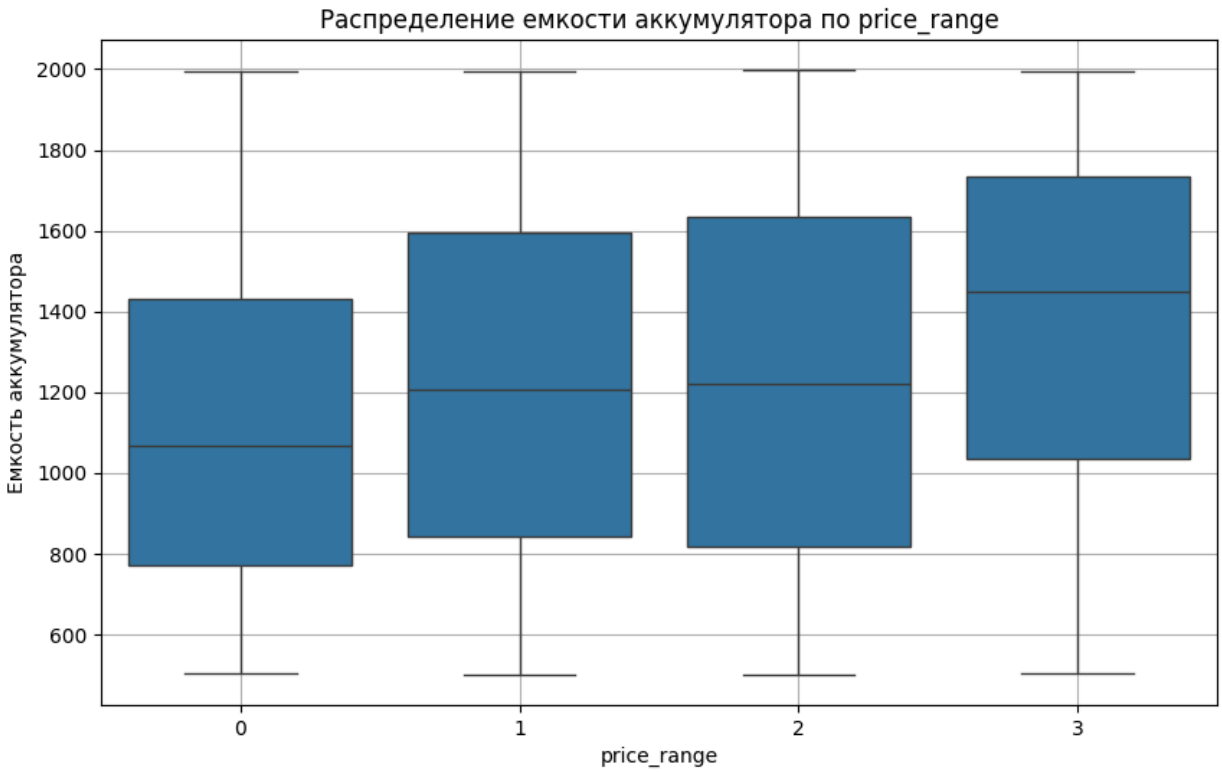
```python
def manual_one_way_anova(data, factor_column, dependent_column):
    """
    Выполняет однофакторный дисперсионный анализ вручную.

    Args:
        data (pd.DataFrame): DataFrame с данными.
        factor_column (str): Название столбца с фактором (ценовая категория).
        dependent_column (str): Название столбца с зависимой переменной (емкость
аккумулятора).

    Returns:
        dict: Словарь с результатами ANOVA (F-статистика, степени свободы).
    """
    groups_data = data.groupby(factor_column)[dependent_column].apply(list).tolist()
    group_means = [calculate_mean(group) for group in groups_data]
    group_sizes = [len(group) for group in groups_data]
    overall_data = data[dependent_column].tolist()
    overall_mean = calculate_mean(overall_data)
    num_groups = len(groups_data)
    total_observations = len(overall_data)

    ss_between = calculate_ss_between(overall_mean, group_means, group_sizes)
    ss_within = calculate_ss_within(groups_data)

    df_between = calculate_df_between(num_groups)
    df_within = calculate_df_within(total_observations, num_groups)

    ms_between = calculate_ms_between(ss_between, df_between)
    ms_within = calculate_ms_within(ss_within, df_within)

    f_statistic = calculate_f_statistic(ms_between, ms_within)

    results = {
        'F-статистика': f_statistic,
        'df_между': df_between,
        'df_внутри': df_within
    }
    return results

# 1. Load the data
file_path = 'mobile_phones.csv'  # Замените на фактический путь к вашему CSV файлу
try:
    data = pd.read_csv(file_path)
except FileNotFoundError:
    print(f"Ошибка: Файл не найден по пути '{file_path}'")
    exit()
```

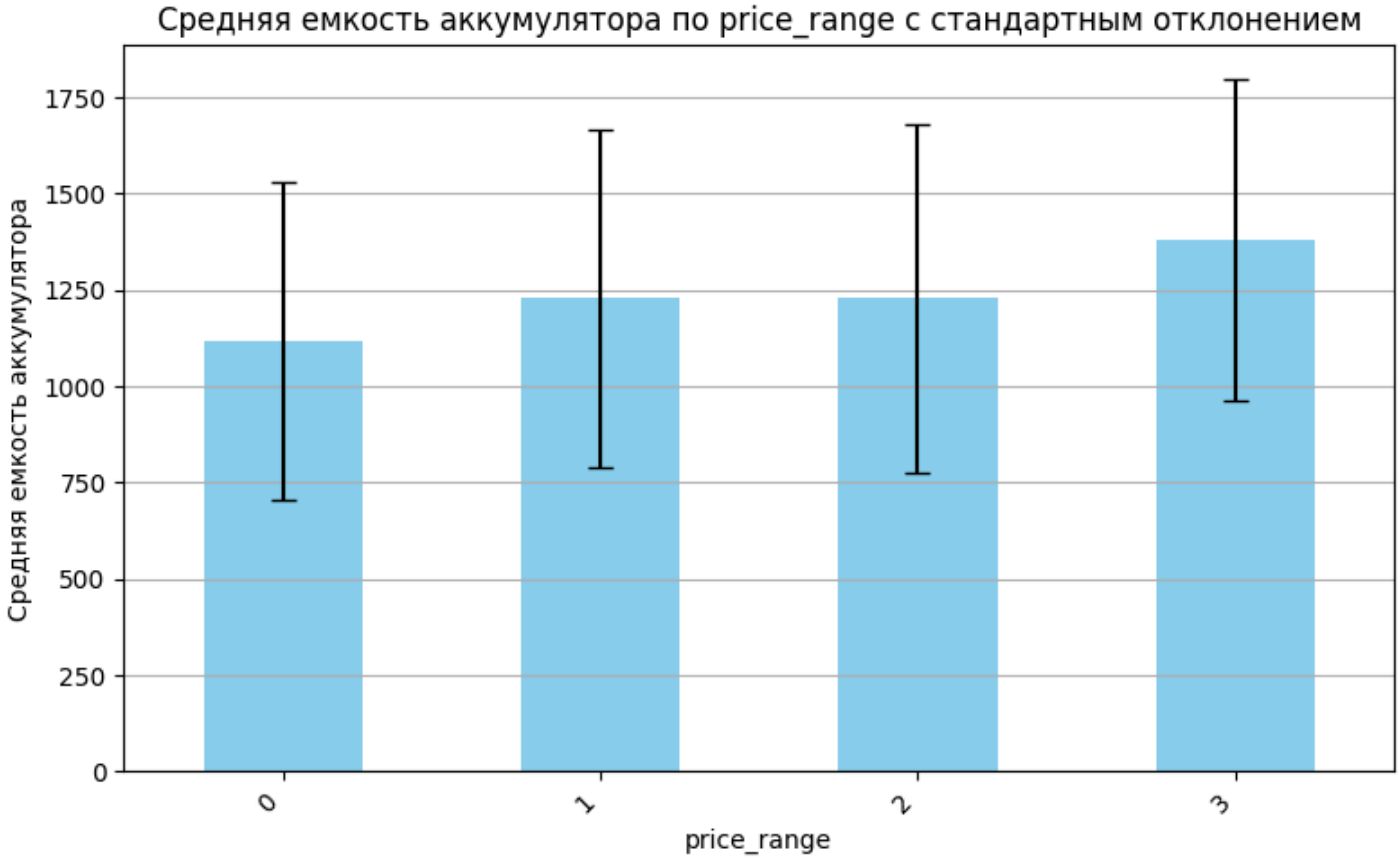
```
# Assuming your data has columns named 'ценовая категория' and 'емкость аккумулятора'
factor_column = 'price_range'
dependent_column = 'battery_power'

# Ensure the factor column is treated as categorical
data[factor_column] = data[factor_column].astype('category')

# 2. Perform manual one-way ANOVA
anova_results = manual_one_way_anova(data.copy(), factor_column, dependent_column)

# 3. Print the results
print("Результаты однофакторного дисперсионного анализа (ANOVA) (ручной расчет):")
for key, value in anova_results.items():
   print(f"{key}: {value:.4f}")

print("\nДля определения p-значения и принятия решения о гипотезе,")
print("необходимо обратиться к F-распределению с соответствующими степенями свободы
(df_между и df_внутри).")
print("P-значение можно найти с помощью таблиц F-распределения или статистических функций в
библиотеках (например, scipy.stats.f.sf).")

# 4. Visualize the data
plt.figure(figsize=(10, 6))
sns.boxplot(x=factor_column, y=dependent_column, data=data)
plt.title(f'Распределение емкости аккумулятора по {factor_column}')
plt.xlabel(factor_column)
plt.ylabel('Емкость аккумулятора')
plt.grid(True)
plt.show()

# Visualize the means with error bars
group_means = data.groupby(factor_column)[dependent_column].mean()
group_stds = data.groupby(factor_column)[dependent_column].std()

plt.figure(figsize=(8, 5))
group_means.plot(kind='bar', yerr=group_stds, capsize=5, color='skyblue')
plt.title(f'Средняя емкость аккумулятора по {factor_column} с стандартным отклонением')
plt.xlabel(factor_column)
plt.ylabel('Средняя емкость аккумулятора')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.grid(axis='y')
plt.show()
```

```
# Пример расчета p-значения (требуется scipy.stats)
try:
   p_value = stats.f.sf(anova_results['F-статистика'], anova_results['df_между'],
anova_results['df_внутри'])
   alpha = 0.05  # Уровень значимости
   print(f"\nP-значение (рассчитано с помощью scipy.stats): {p_value:.4f}")
   if p_value < alpha:
       print("Отвергаем нулевую гипотезу.")
       print("Существуют статистически значимые различия в среднем значении емкости
аккумулятора между различными ценовыми категориями.")
   else:
       print("Не отвергаем нулевую гипотезу.")
       print("Нет статистически значимых различий в среднем значении емкости аккумулятора
между различными ценовыми категориями на уровне значимости 0.05.")
except ImportError:
   print("\nДля автоматического расчета p-значения установите библиотеку scipy: pip install
scipy")

# --- Сравнение с встроенной функцией ANOVA из scipy ---
# Группируем данные по фактору
grouped_data = [group[dependent_column].values for name, group in
data.groupby(factor_column, observed=True)]

# Выполняем ANOVA с помощью scipy
f_stat_scipy, p_value_scipy = stats.f_oneway(*grouped_data)

print("\nРезультаты встроенной функции scipy.stats.f_oneway:")
print(f"F-статистика: {f_stat_scipy:.4f}")
print(f"P-значение: {p_value_scipy:.4f}")

# Сравнение с ручным результатом
print("\nСравнение с ручным расчетом:")
print(f"F-статистика (ручной расчет): {anova_results['F-статистика']:.4f}")
```

Результаты:


C:\Python313\python.exe E:\matstat\lab4\lab4_task2.py
Результаты однофакторного дисперсионного анализа (ANOVA) (ручной расчет):
F-статистика: 31.5982
df_между: 3.0000
df_внутри: 1996.0000
Для определения p-значения и принятия решения о гипотезе,
необходимо обратиться к F-распределению с соответствующими степенями свободы (df_между и df_внутри).
P-значение можно найти с помощью таблиц F-распределения или статистических функций в библиотеках. Рассчитаем его при помощи scipy.stats.f.sf.
P-значение (рассчитано с помощью scipy.stats): 0.0000
Отвергаем нулевую гипотезу.
Существуют статистически значимые различия в среднем значении емкости аккумулятора между различными ценовыми категориями.
Результаты встроенной функции scipy.stats.f_oneway:
F-статистика: 31.5982
P-значение: 0.0000
Сравнение с ручным расчетом:
F-статистика (ручной расчет): 31.5982
Process finished with exit code 0


Распределение емкости аккумулятора по price_range
Емкость аккумулятора
2000
1800
1600
1400
1200
1000
800
600
0
1
2
3
price_range


Средняя емкость аккумулятора по price_range с стандартным отклонением
Средняя емкость аккумулятора
1750
1500
1250
1000
750
500
250
0
0
1
2
3
price_range

Вывод

Построенная модель позволяет оценить зависимость цены дома от жилой площади, площади участка и площади надземной части. Вычисленные коэффициенты и их доверительные интервалы позволяют судить о значимости признаков. Значения R^2 и F-статистики показывают, что модель объясняет значительную часть вариации цены, но существует потенциал для её улучшения.

Также на основании проведенного однофакторного дисперсионного анализа и полученного p-значения был сделан вывод о наличии или отсутствии статистически значимых различий в среднем значении емкости аккумулятора между различными ценовыми категориями мобильных телефонов. Визуализация данных с помощью ящиков с усами и столбчатых диаграмм позволила наглядно сравнить распределения и средние значения емкости аккумулятора в разных группах.