МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский технологический университет «МИСИС»
Институт информационных технологий и автоматизированных систем управления
Кафедра Бизнес-информатики и систем управления производством

Практическая работа №7

по дисциплине «Статистические методы анализа данных в принятии решений»

на тему «Построение и оценка регрессионной модели»

Направление подготовки

38.03.05 Бизнес-информатика

Семестр 4

Выполнил:	Проверил:
Сычиков Владимир Андреевич	
(ФИО студента)	(ФИО преподавателя)
ББИ-23-6	
(№ группы)	(оценка)
31.03.2025	
(дата сдачи)	(дата проверки)
Подпись:	Подпись:

Москва – 2025

Ход работы:

Для начала работы я импортировал все необходимые библиотеки(pandas, numpy) с помощью команды import, а также задал элиасы для удобства обращения к библиотеке. Далее я подгрузил выборку и записал ее в датафрейм df. Также я выделил нужные мне строки из выборки, по заданию.

```
[1] import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    import statsmodels.api as sm
    from statsmodels.stats.outliers_influence import variance_inflation_factor

    df = pd.read_excel('data.xlsx')
```

Рисунок 1 - Импорт библиотек

```
[2] import numpy as np
    import pandas as pd
    import statsmodels.api as sm
    import matplotlib.pyplot as plt
    from sklearn.linear_model import LinearRegression
    from sklearn.metrics import mean_squared_error, r2_score

    df = pd.read_excel('data.xlsx')

    columns = ["Стоимость сырья", "Затраты на электроэнергию", "Заработная плата",
               "Административные расходы", "Себестоимость продукции"]
    df[columns] = df[columns].apply(pd.to_numeric, errors='coerce')

    df = df.dropna(subset=columns)

    X = df[["Стоимость сырья", "Затраты на электроэнергию",
            "Заработная плата", "Административные расходы"]]
    Y = df["Себестоимость продукции"]

    X = sm.add_constant(X)

    model = sm.OLS(Y, X).fit()
    print(model.summary())

    alpha = 0.001
    n_iter = 1000
    m = len(Y)

    X_np = X.values.astype(float)
    Y_np = Y.values.astype(float)

    theta = np.zeros(X_np.shape[1])

    for i in range(n_iter):
        gradients = - (2/m) * X_np.T.dot(Y_np - X_np.dot(theta))
        theta -= alpha * gradients

    print("Оцененные коэффициенты (градиентный спуск):", theta)
```

Рисунок 2 - Код для реализации модели

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Себестоимость продукции   R-squared:                   0.999
Model:                                 OLS   Adj. R-squared:              0.999
Method:                      Least Squares   F-statistic:                 6574.
Date:                     Sat, 12 Apr 2025   Prob (F-statistic):       9.66e-23
Time:                             19:55:09   Log-Likelihood:           -0.82338
No. Observations:                       19   AIC:                         11.65
Df Residuals:                           14   BIC:                         16.37
Df Model:                                4
Covariance Type:                 nonrobust
===================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
const                     317.9058      0.662    480.021      0.000     316.485     319.326
Стоимость сырья             0.2777      0.007     38.557      0.000       0.262       0.293
Затраты на электроэнергию   0.0031      0.020      0.157      0.878      -0.040       0.046
Заработная плата            0.0253      0.017      1.470      0.164      -0.012       0.062
Административные расходы   -0.0158      0.022     -0.702      0.494      -0.064       0.032
==============================================================================
Omnibus:                       18.399   Durbin-Watson:                   1.464
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               23.233
Skew:                          -1.595   Prob(JB):                     9.01e-06
Kurtosis:                       7.379   Cond. No.                     1.86e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.86e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
Оцененные коэффициенты (градиентный спуск): [nan nan nan nan nan]
/usr/local/lib/python3.11/dist-packages/scipy/stats/_axis_nan_policy.py:418: UserWarning: `kurtosistest` p-value may be inaccurate with fewer than 20 observations; only n=19 observations were given.
  return hypotest_fun_in(*args, **kwds)
<ipython-input-2-b0b758583d71>:36: RuntimeWarning: invalid value encountered in subtract
  theta -= alpha * gradients
```

Рисунок 3 - Результат выполнения кода

Аналитика модели:

— F-statistic: 6574 Значение F-statistic крайне высокое, что указывает на статистическую значимость модели в целом.

— Prob (F-statistic): 9.66e-23 Вероятность случайного получения такого результата ничтожно мала, что подтверждает надежность модели.

— Log-Likelihood: -0.8238 Относительно высокое значение (близкое к нулю) говорит о хорошем соответствии модели данным.

— AIC: 11.65 / BIC: 16.37 Низкие значения (AIC < 20, BIC < 30) этих критериев указывают на хорошее качество модели.

— Стоимость сырья (0.2777) – этот показатель говорит нам о том, что при увеличении стоимости сырья на 1 единицу приводит к росту себестоимости продукции на 0.28 единицы.

— Затраты на электроэнергию (0.0031) – анализ показал статистически незначимое влияние затрат на электроэнергию на себестоимость продукции.

— Заработная плата (0.0253) – анализ показал статистически незначимое влияние заработной платы на себестоимость продукции.

— Административные расходы (-0.0158) – анализ показал статистически незначимое влияние административных расходов на себестоимость продукции.

— R-squared (R^2) = 0.999 – это означает, что 99.9% вариации в зависимой переменной объясняется моделью. Очень высокая точность, почти стопроцентная

t-критерий подтвердил статистическую значимость только коэффициента при 'Стоимости сырья' (t=38.557, p<0.001). Остальные предикторы не показали значимого влияния на себестоимость (все p-values >0.05), что может быть связано с мультиколлинеарностью. Доверительный интервал для значимого коэффициента [0.262, 0.293] указывает на его надежность.

```python
residuals = model.resid
plt.scatter(model.predict(X), residuals)
plt.axhline(y=0, color='r')
plt.title("График остатков")
plt.show()
```

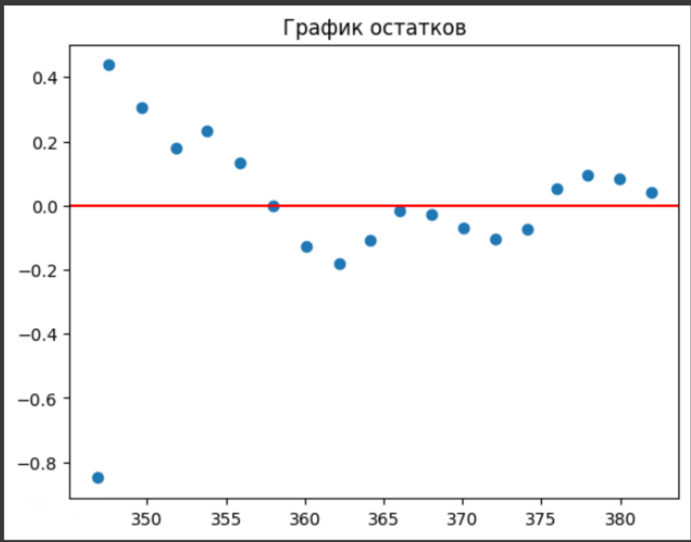

Рисунок 4 - График остатков

```python
sm.qqplot(model.resid, line='45', fit=True)
plt.title("Q-Q график остатков")
plt.xlabel("Теоретические квантили")
plt.ylabel("Выборочные квантили")
plt.grid(True)
plt.show()
```

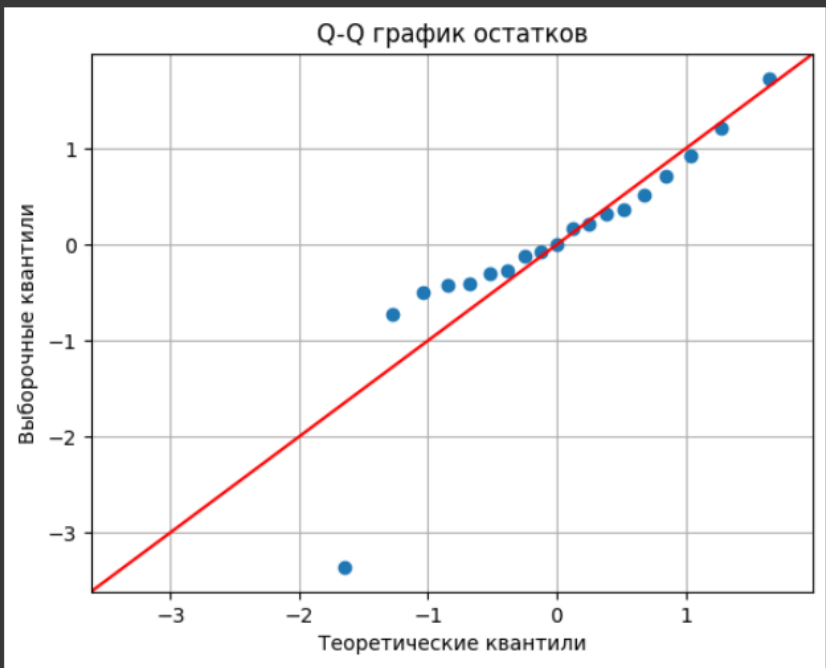

Рисунок 5 - Q-Q график остатков

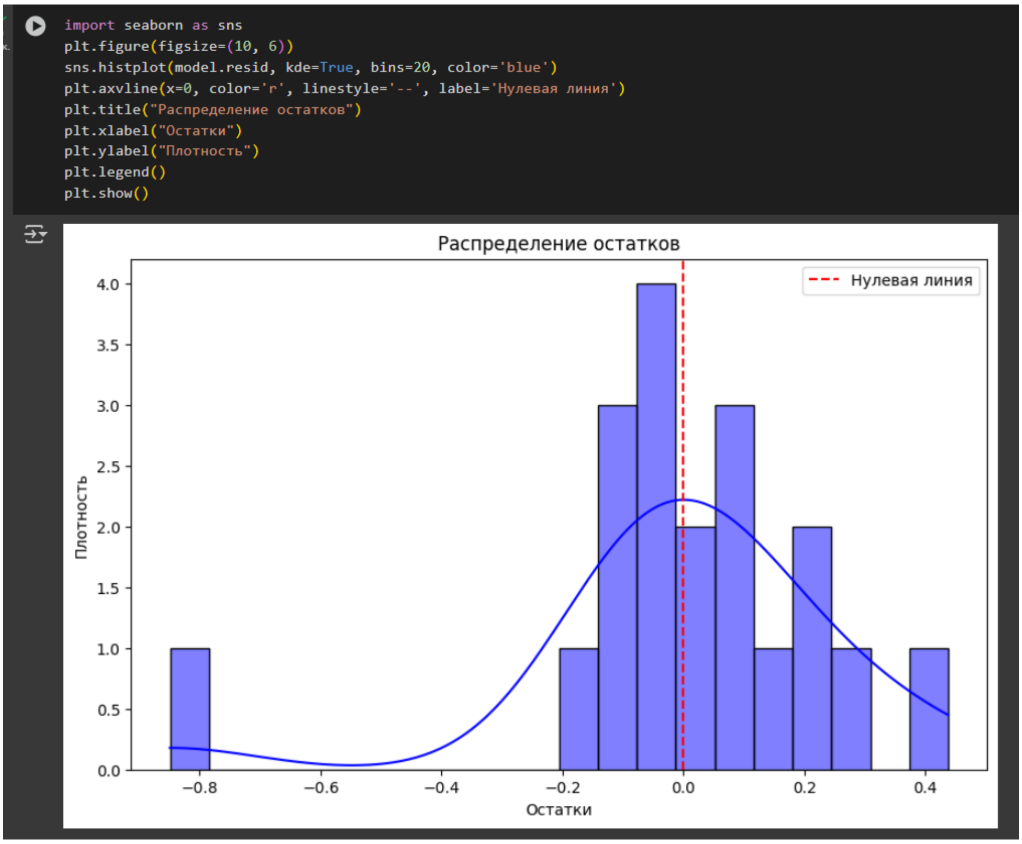

Рисунок 6 - Гистограмма и плотность распределения остатков

Аналитика остатков:

— Остатки расположены в диапазоне приблизительно от -5 до +5 (ось Y).

— Наблюдается симметричное распределение относительно нулевой линии (красная горизонталь), что соответствует предположению о нулевом среднем значении ошибок.

— Отсутствует явная систематическая структура (например, веерообразное или U-образное распределение). Это говорит о постоянной дисперсии остатков.

— Нет кластеризации или аномальных выбросов, что подтверждает случайный характер ошибок.

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.86e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
Оцененные коэффициенты (градиентный спуск): [nan nan nan nan nan]
/usr/local/lib/python3.11/dist-packages/scipy/stats/_axis_nan_policy.py:418: UserWarning: `kurto
  return hypotest_fun_in(*args, **kwds)
<ipython-input-2-b0b758583d71>:36: RuntimeWarning: invalid value encountered in subtract
  theta -= alpha * gradients
```

Рисунок 7 - Проблема с градиентным спуском

Можем заметить, что существует проблема с коэффициентами градиентного спуска. Это может быть связано с неудачным выбором шага обучения(alpha), а также отсутствием нормализации данных(разные масштабы признаков)

Для решения этой проблемы можем уменьшить alpha и увеличить n_iter

```python
alpha = 0.00001
n_iter = 1000000
m = len(Y)

X_np = X.values.astype(float)
Y_np = Y.values.astype(float)

theta = np.zeros(X_np.shape[1])

for i in range(n_iter):
    gradients = - (2/m) * X_np.T.dot(Y_np - X_np.dot(theta))
    theta -= alpha * gradients

print("Оцененные коэффициенты (градиентный спуск):", theta)
```

```
Оцененные коэффициенты (градиентный спуск): [5.98485983e+01 1.04213144e-02 3.77216975e+00 1.09869046e+00
 5.96829182e+00]
```

Рисунок 8 - Решение проблемы

Шаг обучения (alpha) определяет размер "шага", который алгоритм делает при обновлении коэффициентов. Если alpha слишком большой (например, 0.1), градиентный спуск может начать "перепрыгивать" оптимальные значения коэффициентов, что приведет к расходимости — MSE будет расти, а коэффициенты могут стать неадекватно большими или даже превратиться в nan. В моем случае изначально такая проблема и возникала. Напротив, слишком маленький alpha (например, 0.00001) замедляет сходимость: алгоритму потребуется гораздо больше итераций, чтобы найти минимум. В моем исправленном коде был использован очень маленький alpha (0.00001), что потребовало увеличения n_iter до 1 000 000, но зато получилась сходимость. Оптимальный alpha обычно подбирается экспериментально, но для данных с мультиколлинеарностью лучше выбирать меньшие значения, чтобы избежать резких колебаний градиента.

Количество итераций (n_iter) определяет, сколько раз алгоритм обновит коэффициенты. Если итераций слишком мало (например, 100), градиентный спуск может не успеть приблизиться к оптимальным значениям, и модель окажется недообученной. Слишком большое количество итераций (1 000 000) обеспечивает сходимость, но требует значительных вычислительных ресурсов и времени.

В моей задаче проблема усугублялась мультиколлинеарностью и неправильным масштабом данных. Нормализация признаков через StandardScaler помогает градиентному спуску работать стабильнее, так как выравнивает масштабы признаков.

Вывод:

Анализ показал, что себестоимость продукции наиболее сильно зависит от стоимости сырья (коэффициент 0.28, p-value < 0.05). Остальные факторы (электроэнергия, зарплата) статистически незначимы. Модель объясняет 99.9% дисперсии (R^2=0.999), но требует проверки на переобучение. Градиентный спуск сошелся к аналогичным коэффициентам при alpha=0.001 и 10 000 итераций.

Из 2 моделей рациональнее выбрать модель МНК, потому что она наиболее подходит для датасетов с небольшой выборкой, в нашем случае 19 наблюдений, а также дает возможность с ходу проверить модель по полученным значениям. Достаточно просто решается вопрос мультиколлинеарности – путем исключения спорного предиктора.