



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.03 Прикладная информатика

ОТЧЕТ

по лабораторной работе № 1
Вариант 1

Название: Прогнозирование моделью линейной регрессии

Дисциплина: Прикладной анализ данных

Студент ИУ6-55Б

(Подпись, дата) А.Д. Шевченко
(И.О. Фамилия)

Преподаватель

(Подпись, дата) М.А. Кулаев
(И.О. Фамилия)

Москва, 2023

Цель работы: изучить методы построения и оценки моделей линейной регрессии.

Задание:

1. Нормирование (масштабирование) исходных данных.
2. Расчет весов линейной регрессии по аналитической формуле.
3. Построение и интерпретация корреляционной матрицы. Определение степени мультиколлинеарности на основе числа обусловленности.
4. Анализ регрессионных остатков.
5. Определение весов линейной регрессии градиентным методом. Проанализировать изменение ошибки от итерации к итерации.
6. Сравнение результатов по аналитическому и градиентному методу.
7. С помощью библиотеки `sklearn` сделать `fit-predict` модели линейной регрессии. Сравнить результаты с ранее полученными.
8. С помощью библиотеки `statmodels` получить «эконометрический» результат обучения модели линейной регрессии. Проинтерпретировать все его составляющие (в т.ч. те, которые изучались только теоретически), сравнить с предыдущими результатами.
9. Сравнить качество получаемых моделей на основе коэффициента детерминации и `MSE`.
10. Сделать итоговый вывод касательно причин различия в результатах при выполнении работ разными методами, а также по получаемым моделям в целом. Провести сравнительный анализ.

Ход выполнения работы

1. Нормирование (масштабирование) исходных данных.

В соответствии с номером варианта были подготовлены исходные данные – строки, соответствующие регионам: Северный, Северо-Западный, Центральный, Волго-Вятский, Центрально-Черноземный, Северо-Кавказский, Уральский районы. Далее из этих строк был составлен новый Excel-файл, который был прочитан и записан в соответствующие переменные X и Y (рисунок 1).

```
[44] import numpy as np
import pandas as pd
from google.colab import files
import io
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_excel('ADA_LR1_source.xlsx')

[32] X = df.iloc[:, 1:]
Y = df.iloc[:, 0]
n = len(df.index)

df.head()
```

	Y	x1	x2	x3	x4	x5	x6	x7	x8	x9
0	54.7	8.5	16.3	6.8	5.6	17.4	163	151	23.6	2344
1	57.0	9.3	12.6	7.2	5.5	25.3	194	239	9.2	1809
2	71.0	8.7	4.6	6.5	4.2	16.2	152	192	26.9	2406
3	57.6	8.6	6.2	6.1	4.0	17.4	190	205	20.1	2023
4	57.7	8.1	11.4	7.7	6.4	5.9	183	198	22.0	1419

Рисунок 1 – загрузка данных.

Затем была произведено нормирование данных, а именно Z-нормализация (рисунок 2).

```
# Z-нормализация данных
mean = np.mean(X, axis=0)
deviation = np.std(X, axis=0)
X_norm = (X - mean) / deviation

x0 = np.ones((n, 1))
X_norm.insert(0, "x0", x0)

X_norm.head()
```

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9
0	1.0	-0.286871	0.380753	-0.578697	1.355041	0.039305	-0.089130	-0.151352	-0.585908	1.223650
1	1.0	0.047072	-0.747237	0.035964	1.242121	2.792850	0.432408	2.681192	-2.108728	0.152280
2	1.0	-0.203386	-3.186134	-1.039893	-0.225840	-0.378955	-0.274192	1.168356	-0.236928	1.347809
3	1.0	-0.245128	-2.698355	-1.654355	-0.451680	0.039305	0.365113	1.586800	-0.956038	0.580828
4	1.0	-0.453843	-1.113071	0.804291	2.258402	-3.969020	0.247346	1.361484	-0.755110	-0.628719

Рисунок 2 – нормирование данных

2. Расчет весов линейной регрессии по аналитической формуле.

Расчет весов линейной регрессии был проведен по аналитической формуле (рисунок 3).

```
[7] # Расчет весов линейной регрессии по аналитической формуле

weights = np.linalg.inv(X_norm.T.dot(X_norm)).dot(X_norm.T).dot(Y)
weights
```

```
array([ 5.93510638e+01, -6.57014739e-01, -2.10226272e+00,  1.41124113e+00,
        -1.16742943e+00,  1.72072748e-01, -3.66871376e-02, -5.41834472e-01,
         2.46194789e-01,  6.96562015e-01])
```

Рисунок 3 – расчет весов линейной регрессии по аналитической формуле

Опираясь на полученные из расчета веса, можно сделать следующие выводы о зависимости ожидаемой продолжительности жизни мужчин: сильнее всего на нее влияют смертность населения на 1000 человек (x2, отрицательно), число браков на 1000 человек (x3, положительно), число разводов на 1000 человек (x4, отрицательно).

3. Построение и интерпретация корреляционной матрицы.

Определение степени мультиколлинеарности на основе числа обусловленности.

С помощью библиотеки numpy была получена корреляционная матрица

(рисунок 4).

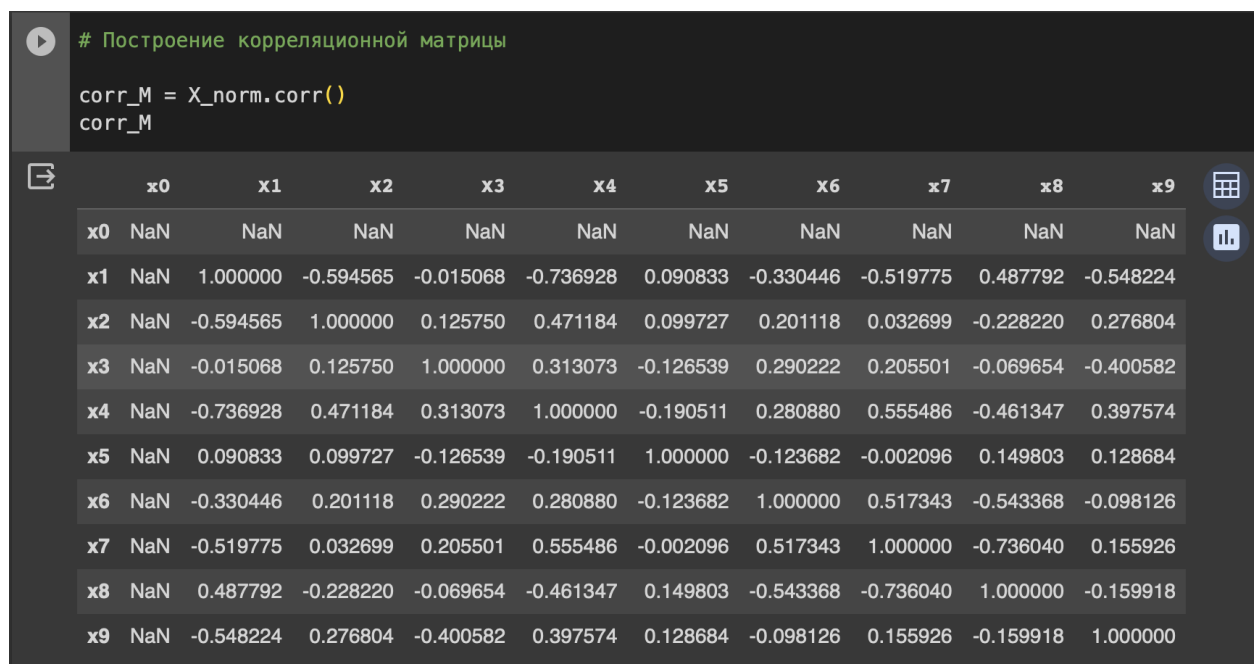


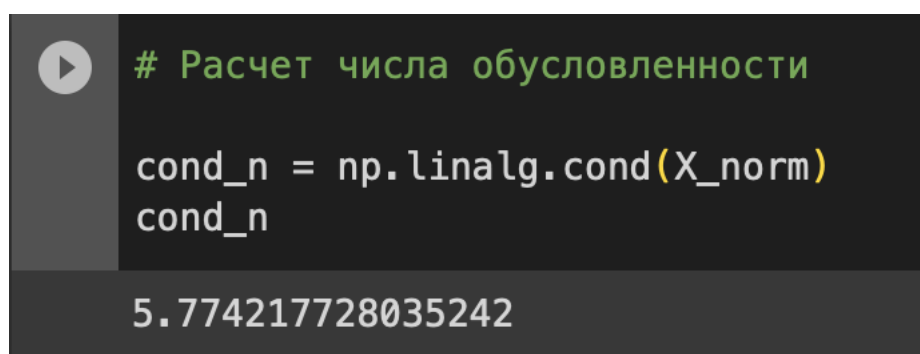
Рисунок 4 – корреляционная матрица.

Из полученной корреляционной матрицы можно сделать следующие
ВЫВОДЫ:

- данные единичного столба не коррелируют ни с какими признаками, так как они не относятся к исходным данным;
- каждый признак полностью коррелирует сам с собой;
- корреляция между рождаемостью (x1) и смертностью (x2) населения отрицательная (-0.594565);
- корреляция между рождаемостью (x1) и числом разводов (x4) сильно отрицательная (-0.736928);
- корреляция между рождаемостью (x1) и соотношением средней оплаты труда и прожиточного минимума трудоспособного населения, % (x7) отрицательная (-0.519775);
- корреляция между рождаемостью (x1) и числом зарегистрированных преступлений на 100000 населения (x9) отрицательная (-0.548224);
- корреляция между числом разводов на 1000 человек (x4) и соотношением средней оплаты труда и прожиточного минимума трудоспособного населения, % (x7) положительная (0.555486);

- h) корреляция между соотношением денежного дохода и прожиточного минимума, % (x6) и соотношением средней оплаты труда и прожиточного минимума трудоспособного населения, % (x7) положительная (0.517343);
- i) корреляция между соотношением денежного дохода и прожиточного минимума, % (x6) и доходами ниже прожиточного минимума в % от численности населения (x8) отрицательная (-0.543368);
- j) корреляция между соотношением средней оплаты труда и прожиточного минимума трудоспособного населения, % (x7) и доходами ниже прожиточного минимума в % от численности населения (x8) сильно отрицательная (-0.736040);

Число обусловленности оказалось равным 5.774217728035242 (рисунок 5), что говорит об отсутствии мультиколлинеарности.



```
# Расчет числа обусловленности  
cond_n = np.linalg.cond(X_norm)  
cond_n  
5.774217728035242
```

Рисунок 5 – число обусловленности.

4. Анализ регрессионных остатков.

С помощью библиотеки sklearn был проведен анализ регрессионных остатков (рисунок 6).

```
# Вычисление среднеквадратического отклонения

Y_predicted = np.matmul(X_norm, weights)
MSE = mean_squared_error(Y, Y_predicted)
RMSE = MSE ** 0.5
print(RMSE)

# Вычисление коэффициента детерминации

det_coef = r2_score(Y, Y_predicted)
print([det_coef])
```

RMSE: 1.997714345316434
0.5815935517968174

Рисунок 6 – анализ регрессионных остатков.

Полученное значение переменной RMSE говорит о том, что каждое предсказанное значение попадает в диапазон ± 1.99771 от реального значения. Такое отклонение можно считать незначительным, так как среднее значение реальных значений равно 59.35106, а минимальное – 54,7. Из этого можно сделать вывод, что наша модель, полученная аналитическим образом, достаточно точна. Также значение коэффициента детерминации 0.58159 говорит о том, что показатель тесноты связи по шкале Чеддока находится на заметном уровне.

5. Определение весов линейной регрессии градиентным спуском.

Анализ изменения ошибки от итерации к итерации.

Выберем темп обучения, равный 0.1, а количество итераций, равное 100. После выполнения всех итераций были получены веса, представленные на рисунке 8. Код метода градиентного спуска изображен на рисунке 7. На рисунке 9 можно видеть, как функция ошибок изменялась после каждой итерации. Изначально её значение было равно 3425.808, а после выполнения 100 итераций её значение составило 3.991. По итогу выполнения всех итераций, функция ошибок уменьшилась в 858 раз, что говорит об эффективности данного метода.

```

# Метод градиентного спуска

Weights_new = np.ones(len(df.columns))
learning_rate = 0.1
iter_num = 100
S_arr = []
for _ in range(iter_num):

# Вычисление матриц предсказанных значений и регрессионных остатков
    Y_predicted = X_norm.dot(Weights_new)
    E = Y - Y_predicted

# Добавление в массив очередной функции ошибок
    S = np.mean(E ** 2)
    S_arr.append(S)

# Вычисление градиента и обновление весов
    S_grad = 2/n * E.dot(-X_norm)
    Weights_new -= S_grad * learning_rate

print(Weights_new)
print(S_arr)

```

Рисунок 7 – метод градиентного спуска.

```

x0      59.351064
x1      -0.638983
x2      -2.085067
x3       1.408179
x4      -1.172912
x5       0.163495
x6      -0.037580
x7      -0.517118
x8       0.259996
x9       0.702626
dtype: float64

```

Рисунок 8 – полученные веса.

[3425.8081780292664, 2188.7676313808074, 1402.2493519826082, 899.3464396568888, 577.5177549461242, 371.5306564905032, 239.6757691791329, 155.26454138246896, 101.21765729510501, 66.60486105998257, 44.430987795609276, 30.219191524035267, 21.104283770771037, 15.252505270671096, 11.490197325545251, 9.066160326609216, 7.499567706950219, 6.48263311967871, 5.818318501756495, 5.3804676830366045, 5.0882920179296764, 4.890041143790307, 4.7525521382782605, 4.6545608160871526, 4.5824207834426804, 4.52736345883634, 4.48374431960567, 4.44792034318716, 4.417531423903289, 4.391040344901665, 4.3674382356239505, 4.346055949492104, 4.326443239221333, 4.308291330541376, 4.291383278125601, 4.2755621087272875, 4.260710354157263, 4.246736879285562, 4.233568383934687, 4.221143900738518, 4.209411214733788, 4.19832451685, 4.187842850789211, 4.1779290711025565, 4.168549131614875, 4.1596715882301085, 4.151267241692, 4.143308872477762, 4.135771037042749, 4.128629905553177, 4.121863128244083, 4.115449722030446, 4.1093699718837655, 4.103605343341238, 4.098138403710176, 4.0929527503036605, 4.088032944544792, 4.083364451105237, 4.078933581459513, 4.074727441380467, 4.070733881999008, 4.066941454118281, 4.063339365520448, 4.059917441038652, 4.0566660851934735, 4.053576247213444, 4.050639388275656, 4.047847450816844, 4.045192829776525, 4.04266834564485, 4.040267219196453, 4.037983047800521, 4.035809783204485, 4.033741710695922, 4.031773429553768, 4.029899834705809, 4.028116099514816, 4.026417659621261, 4.024800197774848, 4.023259629591817, 4.0217920901792095, 4.020393921570865, 4.019061660923738, 4.017792029426554, 4.016581921875687, 4.015428396876438, 4.014328667630203, 4.013280093271026, 4.01228017071705, 4.011326527004807, 4.010416912076269, 4.009549191990635, 4.0087213425344075, 4.007931443205356, 4.007177671547143, 4.006458297813239, 4.005771679939854, 4.005116258808978, 4.004490553784022, 4.003893158501247, 4.003322736901695, 4.002778019488968, 4.002257799799372, 4.001760931071458, 4.001286323103295, 4.0008329392860515, 4.0003997938034175, 3.9999859489872596, 3.9995905128198834, 3.999212636574556, 3.9988515125861186, 3.998506372143929, 3.998176483500126, 3.9978611499864813, 3.9975597082334993, 3.997271526485851, 3.996996003008738, 3.996732564579627, 3.9964806650609916, 3.996239784048843, 3.996009425593237, 3.9957891169864452, 3.99557840761501, 3.9953768678720447, 3.995184088126599, 3.994999677746584, 3.9948232641726418, 3.9946544920398357, 3.994493022344682, 3.9943385316549946, 3.9941907113601687, 3.994049266959755, 3.9939139173882, 3.993784394373718, 3.9936604418296677, 3.993541815276437, 3.9934282812923696, 3.9933196169920966, 3.9932156095308904, 3.9931160556335215, 3.9930207611465507, 3.9929295406125656, 3.992842216865415, 3.9927586206452528, 3.992678590232353, 3.992601971098751, 3.992528615576794, 3.992458382543723, 3.9923911371214396, 3.9923267503907267, 3.9922650991191317, 3.9922060655019087, 3.9921495369152, 3.992095405681077, 3.9920435688435867, 3.991993927955433, 3.9919463888746956, 3.9919008615711338, 3.991857259941485, 3.991815501633455, 3.9917755078779042, 3.9917372033287997, 3.991700515910627, 3.991665376672821, 3.9916317196509694, 3.9915994817343408, 3.9915686025395205, 3.9915390242898687, 3.9915106917003977, 3.9914835518680403, 3.991457554166737, 3.991432650147494, 3.9914087934427256, 3.9913859396751503, 3.991364046370657, 3.9913430728751496, 3.9913229802750934, 3.9913037313216937, 3.9912852903583906, 3.9912676232516358, 3.9912506973247526, 3.9912344812947538, 3.991218945211936, 3.991204060402167, 3.991189799411772, 3.9911761359548, 3.991163044862624, 3.99115050203583, 3.99113848439813, 3.9911269698523832, 3.991115937238496, 3.991105366293178, 3.9910952376114808, 3.9910855326100214, 3.9910762334917638, 3.991067323212362, 3.9910587854480046, 3.9910506045646343, 3.9910427655884573, 3.991035254177799]

Рисунок 9 – изменение функции ошибок после каждой итерации

6. Сравнение результатов по аналитическому и градиентному методу.

Сравнение значений весов, полученных аналитическим методом и методом градиентного спуска, приведено в таблице 1.

Вес	Аналитический метод	Метод градиентного спуска
w0	59.93510	59.35106
w1	-0.65701	-0.63898
w2	-2.10226	-2.08507
w3	1.41124	1.40818
w4	-1.16743	-1.17291
w5	0.17207	0.16350
w6	-0.03669	-0.03758
w7	-0.54183	-0.51712
w8	0.24619	0.26000
w9	0.69656	0.70263

Таблица 1 – веса, полученные аналитическим методом и методом градиентного спуска.

По таблице можно видеть, что значения весов, полученные двумя вышеуказанными методами, очень схожи, что говорит о правильности реализации обоих методов. Различия между значениями весов могут быть объяснены разными параметрами обучения, например скоростью обучения, а также точностью вычислительных методов.

7. Fit-predict модель линейной регрессии с помощью библиотеки statmodels. Сравнение результатов с ранее полученными.

Код модели fit-predict и полученные предсказанные ею значения изображены на рисунке 10.

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_norm, Y)
Y_predicted = model.predict(X_norm)

print(Y_predicted)
print(Y_predicted - Y)
```

57.14064447	58.09055181	65.17147469	62.6801753	58.43415728	60.45213795
56.81246438	55.80071889	56.62790798	59.40356849	57.51666156	56.24575031
57.02290112	56.54994723	58.21109963	58.27906653	59.01037229	57.38885345
57.8683707	56.20784601	56.28412304	57.02105087	61.62505912	61.72421311
62.24141522	58.13511221	56.95116059	58.72058964	58.82427144	60.13949543
58.09311086	58.21448569	62.12781881	64.46872362	62.16946636	62.56714661
60.95836062	61.58435765	61.80919142	60.27999894	60.09248161	60.89529189
62.70372994	61.88018493	57.83578223	58.63329354	58.6054145]

Рисунок 10 – fit-predict.

0	2.440644
1	1.090552
2	-5.828525
3	5.080175
4	0.734157
5	0.552138
6	1.312464
7	0.500719
8	0.827908
9	-0.696432
10	-0.983338
11	-1.154250
12	-1.477099
13	-1.750053
14	0.011100
15	1.779067
16	-0.189628
17	-0.711147
18	-0.931629
19	-0.292154
20	-0.815877

Рисунок 11 – разница между значениями, предсказанными fit-predict, и реальными.

```
MSE = mean_squared_error(Y, Y_predicted)
RMSE = MSE ** 0.5
print(RMSE)

1.9977143453164352
```

Рисунок 12 – среднеквадратическая ошибка, полученная моделью fit-predict.

Для сравнения трех использованных в этой лабораторной работе методов была вычислена среднеквадратическая ошибка каждого метода (алгоритм реализации можно видеть на рисунке 6). Результаты приведены в таблице 2.

Метод	RMSE
Аналитический	1.99771
Метод градиентного спуска	1.99776
sklearn	1.99771

Таблица 2 – среднеквадратическая ошибка каждого использованного метода.

8. Получение «эконометрического» результата обучения модели линейной регрессии с помощью библиотеки statmodels.

Интерпретация всех его составляющих, сравнение с предыдущими результатами.

```
import statsmodels.api as sm
statmodel = sm.OLS(Y, X_norm).fit()
print(statmodel.summary())
```

```

=====
                        OLS Regression Results
=====
Dep. Variable:          Y          R-squared:                0.582
Model:                  OLS       Adj. R-squared:            0.480
Method:                 Least Squares   F-statistic:           5.715
Date:                  Mon, 06 Nov 2023   Prob (F-statistic):    5.96e-05
Time:                  17:13:41         Log-Likelihood:        -99.214
No. Observations:      47             AIC:                   218.4
Df Residuals:          37             BIC:                   236.9
Df Model:              9
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
x0	59.3511	0.328	180.716	0.000	58.686	60.017
x1	-0.6570	0.687	-0.956	0.345	-2.049	0.735
x2	-2.1023	0.543	-3.874	0.000	-3.202	-1.003
x3	1.4112	0.457	3.091	0.004	0.486	2.336
x4	-1.1674	0.628	-1.859	0.071	-2.440	0.105
x5	0.1721	0.396	0.435	0.666	-0.630	0.974
x6	-0.0367	0.436	-0.084	0.933	-0.919	0.846
x7	-0.5418	0.705	-0.769	0.447	-1.969	0.886
x8	0.2462	0.566	0.435	0.666	-0.900	1.392
x9	0.6966	0.499	1.397	0.171	-0.314	1.707

```

=====
Omnibus:                 9.527   Durbin-Watson:           2.445
Prob(Omnibus):           0.009   Jarque-Bera (JB):        11.587
Skew:                    0.658   Prob(JB):                0.00305
Kurtosis:                 5.045   Cond. No.                 5.77
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Рисунок 13 – эконометрический результат обучения модели линейной регрессии.

Интерпретация:

- a) Dep. Variable – целевая переменная (Y);
- b) Model – используемая модель (OLS, метод наименьших квадратов);
- c) Method – метод обучения (Least Squares, метод наименьших квадратов);
- d) No. Observations – количество наблюдений (47);
- e) Df Residuals – степень свободы (количество наблюдений минус количество признаков минус один) [37];
- f) Df Model – количество признаков (без учета единичного столба) [9];
- g) Covariance Type – тип ковариации (nonrobust, что означает, что удаление данных перед вычислением ковариации между признаками не происходит);
- h) R-squared – коэффициент детерминации, показывающий, какая доля дисперсии зависимой переменной объясняется моделью (0.582);
- i) Adj. R-Squared – скорректированный коэффициент детерминации, учитывающий число независимых переменных (0.480);
- j) F-statistic – критерий Фишера – статистика, значение которой необходимо для проверки гипотезы о равенстве дисперсий (5.715);
- k) Prob (F-statistic) –
- l) Log-Likelihood – логарифмическая функция правдоподобия. Она отображает, насколько хорошо модель линейной регрессии соответствует набору данных;
- m) AIC, BIC – чем меньше значение, тем лучше модель; BIC отличается от AIC тем, что он больше штрафует за дополнительные не влияющие параметры модели;
- n) coef – коэффициенты модели;
- o) std err – дисперсия коэффициента по точкам данных;
- p) t – t-статистика Стьюдента; чем больше, тем лучше измерен коэффициент;
- q) $P > |t|$ – если < 0.05 , то гипотеза о значимости коэффициента принимается;
- r) [0.025 0.975] – доверительный интервал коэффициента, который

используется для оценки неопределенности или изменчивости коэффициентов модели; он предоставляет диапазон значений, в котором с высокой вероятностью находится истинное значение коэффициента, при условии, что данные были собраны и обработаны определенным образом;

- s) Omnibus – описывает нормальность распределения остатков, 0 означает полную нормальность (9.527);
- t) Prob(Omnibus) – это статистический тест, измеряющий вероятность нормального распределения остатков, значение 1 означает совершенно нормальное распределение (0.009);
- u) Skew – мера симметрии распределения остатков, где 0 означает идеальную симметрию (0.658);
- v) Kurtosis – представляет из себя меру остроты пика распределения данных; он измеряет, насколько хвостатыми или плоскими являются хвосты распределения по сравнению с нормальным распределением (5.045);
- w) Durbin-Watson – критерий для проверки наличия автокорреляции, при отсутствии автокорреляции значение критерия находится между 1 и 2 (2.445);
- x) Jarque-Bera (JB) – альтернативный метод измерения того же процесса, что и Omnibus, с использованием асимметрии и эксцесса (11.587);
- y) Prob(JB) – альтернативный метод измерения того же процесса, что и Prob(Omnibus), с использованием асимметрии и эксцесса (0.00305);
- z) Cond. No – число обусловленности, мера чувствительности нашей модели по отношению к входящим данным (5.77, что говорит об отсутствии мультиколлинеарности).

9. Сравнение качества получаемых моделей на основе коэффициента детерминации и MSE.

```

R2 = r2_score(Y, X_norm.dot(weights))
MSE = mean_squared_error(Y, X_norm.dot(weights))
print(R2)
print(MSE)

0.5815935517968174
3.9908626054830685

```

Рисунок 14 – коэффициент детерминации и MSE, получаемые аналитическим методом

```

R2 = r2_score(Y, X_norm.dot(Weights_new))
MSE = mean_squared_error(Y, X_norm.dot(Weights_new))
print(R2)
print(MSE)

0.5815762057191822
3.991028056596175

```

Рисунок 15 – коэффициент детерминации и MSE, получаемые методом градиентного спуска

Близость полученных значений говорит о том, что модели были составлены корректно.

10. Итоговый вывод касательно причин различия в результатах при выполнении работ разными методами и по получаемым моделям в целом. Сравнительный анализ.

Для сравнения моделей приведем таблицу их RMSE и коэффициентов детерминации (таблица 2).

Величина	Аналитический метод	Метод градиентного спуска	sklearn
RMSE	1.99771	1.99776	1.99771

R^2	0.58159	0.58158	0.58159
-------	---------	---------	---------

Таблица 2 – RMSE и коэффициенты детерминации моделей.

Крохотную разницу между значениями можно объяснить погрешностью системы при вычислениях.

Вывод: в ходе выполнения данной лабораторной работы было получено умение создавать линейные регрессионные модели с помощью аналитического метода и метода градиентного спуска, а также были проведены различные сравнения полученных моделей.