```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import seaborn as sns
import scipy.stats as scp
```

Задание 5_2

Задача (с использованием ПК):

А) Исследуйте зависимость цены квартиры (в тыс. у.е.) в г. Москве от общей площади (м2).
Б) Постройте 95%-й доверительный интервал для коэффициентов модели.
В) Проверьте значимость модели регрессии в целом и каждого коэффициента модели по отдельности.
Г) Сделайте выводы о качестве модели.
Д) Проверьте выполнение предпосылку о гомоскедастичности.
Е) При обнаружении гетероскедастичности:
1) оцените робастную ковариационную матрицу;
2) предложите пути устранения гетероскедастичности и постройте новые модели.

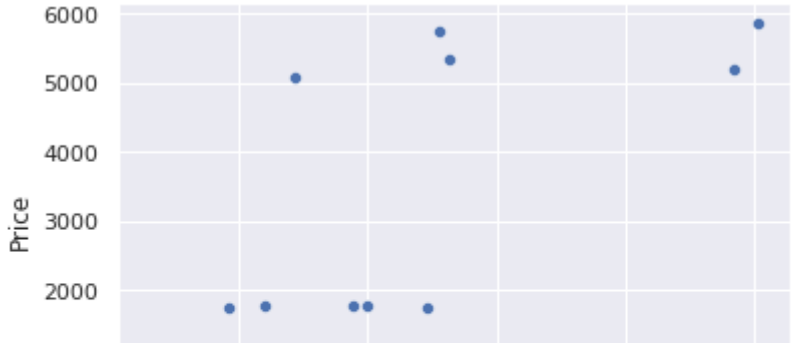
Файл с исходными данными задачи – task3.txt.

```
data = pd.read_csv('/content/task3.txt', sep = '\t')
x = data['Square']
y = data['Price']
data.head()
```

	Price	Square
0	5876	503
1	5743	256
2	5355	263
3	5202	484
4	5099	144

```
sns.set()
sns.scatterplot(x, y)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd46969c250>
```



A) Исследуйте зависимость цены квартиры (в тыс. у.е.) в г. Москве от общей площади (м2).

Square

```
import statsmodels.api as sm
results = sm.OLS.from_formula("Price ~ Square", data = data).fit()


print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.679
Model:                            OLS   Adj. R-squared:                  0.661
Method:                 Least Squares   F-statistic:                     38.08
Date:                Sat, 26 Nov 2022   Prob (F-statistic):           7.96e-06
Time:                        11:51:26   Log-Likelihood:                -169.58
No. Observations:                  20   AIC:                             343.2
Df Residuals:                      18   BIC:                             345.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     41.2723    425.487      0.097      0.924    -852.642     935.187
Square        12.6840      2.055      6.171      0.000       8.366      17.002
==============================================================================
Omnibus:                       12.728   Durbin-Watson:                   2.251
Prob(Omnibus):                  0.002   Jarque-Bera (JB):               10.469
Skew:                           1.596   Prob(JB):                      0.00533
Kurtosis:                       4.540   Cond. No.                         321.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
```

Считаем остатки

```
reg = LinearRegression().fit(x.values.reshape(-1,1), y.values.reshape(-1,1))
```

```
epsilons = y.values.reshape(-1,1) - reg.predict(x.values.reshape(-1, 1))
```

```
array([-3.41060513e-13])
```

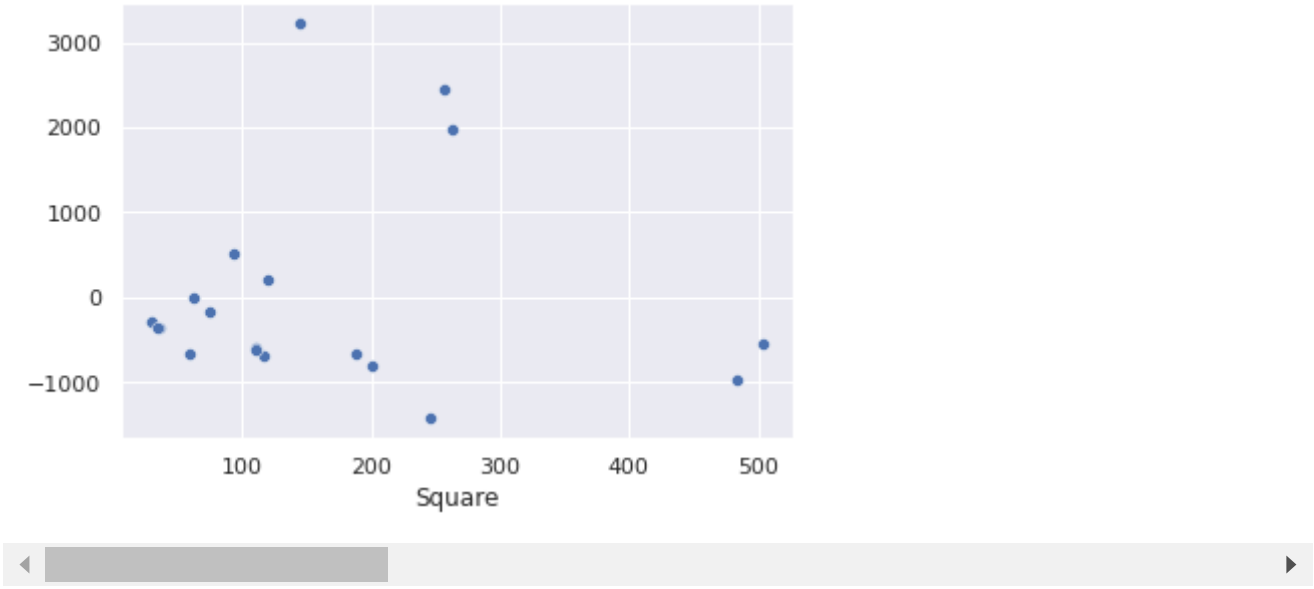
```
mean_e = epsilons.mean()
std_e = epsilons.std()

vec_w = (-1) * (epsilons - mean_e) / std_e
vec_w[:5]
```

```
array([[ 0.46819726],
       [-2.10741597],
       [-1.69806892],
       [ 0.83995307],
       [-2.7741756 ]])
```

```
sns.set()
sns.scatterplot(x, epsilons.reshape(1, -1)[0])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd46905d8d0>
```



тест ранговой корреляции Спирмена

```
rho, p = scp.spearmanr(x, abs(epsilons))

rho, p
```

```
(0.6759503677838935, 0.0010695333928390479)
```

тест Глейзера

```
result = [0, 1, 2, 3, 4, 5]
```

```
data_1 = data
data_1.columns = ['y', 'x']
model = ['x', 'np.sqrt(x)', 'np.abs(1/x)', 'np.abs(1 / np.sqrt(x))', 'np.sqrt(x ** 3)']

result[1] = sm.OLS.from_formula("np.abs(epsilons) ~ x", data = data_1).fit().tvalues['x']
result[2] = sm.OLS.from_formula("np.abs(epsilons) ~ np.sqrt(x)", data = data_1).fit().tval
result[3] = sm.OLS.from_formula("np.abs(epsilons) ~ np.abs(1/x)", data = data_1).fit().tva
result[4] = sm.OLS.from_formula("np.abs(epsilons) ~ np.abs(1 / np.sqrt(x))", data = data_1
result[5] = sm.OLS.from_formula("np.abs(epsilons) ~ np.sqrt(x ** 3)", data = data_1).fit()

for i in range(5):
  print(f'Model {model[i]} t-stats : {np.abs(result[i + 1])}')
print(scp.t.isf(0.05 / 2, 18))
```

```
Model x t-stats : 2.2366585900303493
Model np.sqrt(x) t-stats : 2.370204403323419
Model np.abs(1/x) t-stats : 1.5014660685170058
Model np.abs(1 / np.sqrt(x)) t-stats : 1.9661652840584585
Model np.sqrt(x ** 3) t-stats : 2.03966855587151
2.10092204024096
```

Из этого критерия следует, что гипотеза H0 отвергается, и остатки гетероскедастичны

```
print(results.get_robustcov_results(cov_type = "HC0").summary2())
```

```
                  Results: Ordinary least squares
=================================================================
Model:              OLS              Adj. R-squared:     0.661
Dependent Variable: Price            AIC:                343.1681
Date:               2022-11-26 13:49 BIC:                345.1596
No. Observations:   20               Log-Likelihood:     -169.58
Df Model:           1                F-statistic:        75.41
Df Residuals:       18               Prob (F-statistic): 7.47e-08
R-squared:          0.679            Scale:              1.5074e+06
-----------------------------------------------------------------
               Coef.   Std.Err.    t     P>|t|     [0.025    0.975]
-----------------------------------------------------------------
Intercept     41.2723  251.0755  0.1644  0.8713  -486.2176  568.7623
Square        12.6840    1.4606  8.6841  0.0000     9.6154   15.7526
-----------------------------------------------------------------
Omnibus:              12.728        Durbin-Watson:           2.251
Prob(Omnibus):        0.002         Jarque-Bera (JB):        10.469
Skew:                 1.596         Prob(JB):                0.005
Kurtosis:             4.540         Condition No.:           321
=================================================================
```

Метод Доступных Взвешенных Наименьших Квадратов

```
data_2 = data_1
```

```
data_2['x'] = 1 / data_2['x']
data_2['y'] = data_2['y'] * data_2['x']

fin_model = sm.OLS.from_formula("y ~ x", data = data_2)
result_dmnk = fin_model.fit()


sum_dm = result_dmnk.summary()
print(sum_dm)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.241
Model:                            OLS   Adj. R-squared:                  0.199
Method:                 Least Squares   F-statistic:                     5.709
Date:                Sat, 26 Nov 2022   Prob (F-statistic):             0.0280
Time:                        13:15:17   Log-Likelihood:                -66.700
No. Observations:                  20   AIC:                             137.4
Df Residuals:                      18   BIC:                             139.4
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     15.9805      2.539      6.293      0.000      10.645      21.316
x           -380.0875    159.070     -2.389      0.028    -714.281     -45.894
==============================================================================
Omnibus:                       16.216   Durbin-Watson:                   2.633
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               16.236
Skew:                           1.631   Prob(JB):                     0.000298
Kurtosis:                       5.973   Cond. No.                         99.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
```

```
y_pred = fin_model.predict(fin_model.fit().model.exog.reshape(2, 20))
y_pred
```

```
             2.0100e+02, 6.2000e+03, 2.0100e+02, 7.3200e+03, 2.0100e+02,
             6.1880e+03, 2.0100e+02, 7.0930e+03, 2.0100e+02, 1.2246e+04],
            [1.2100e+02, 1.3703e+04, 1.2100e+02, 7.8160e+03, 1.2100e+02,
             1.4303e+04, 1.2100e+02, 9.4840e+03, 1.2100e+02, 1.3344e+04,
             1.2100e+02, 3.8000e+03, 1.2100e+02, 4.4400e+03, 1.2100e+02,
             3.7880e+03, 1.2100e+02, 4.2930e+03, 1.2100e+02, 7.4460e+03],
            [1.8900e+02, 2.1183e+04, 1.8900e+02, 1.2100e+04, 1.8900e+02,
             2.2259e+04, 1.8900e+02, 1.4584e+04, 1.8900e+02, 2.0824e+04,
             1.8900e+02, 5.8400e+03, 1.8900e+02, 6.8880e+03, 1.8900e+02,
             5.8280e+03, 1.8900e+02, 6.6730e+03, 1.8900e+02, 1.1526e+04],
            [9.4000e+01, 1.0733e+04, 9.4000e+01, 6.1150e+03, 9.4000e+01,
             1.1144e+04, 9.4000e+01, 7.4590e+03, 9.4000e+01, 1.0374e+04,
             9.4000e+01, 2.9900e+03, 9.4000e+01, 3.4680e+03, 9.4000e+01,
             2.9780e+03, 9.4000e+01, 3.3480e+03, 9.4000e+01, 5.8260e+03],
            [2.4700e+02, 2.7563e+04, 2.4700e+02, 1.5754e+04, 2.4700e+02,
             2.9045e+04, 2.4700e+02, 1.8934e+04, 2.4700e+02, 2.7204e+04,
             2.4700e+02, 7.5800e+03, 2.4700e+02, 8.9760e+03, 2.4700e+02,
             7.5680e+03, 2.4700e+02, 8.7030e+03, 2.4700e+02, 1.5006e+04],
            [1.1100e+02, 1.2603e+04, 1.1100e+02, 7.1860e+03, 1.1100e+02,
```

```
 1.3133e+04, 1.1100e+02, 8.7340e+03, 1.1100e+02, 1.2244e+04,
 1.1100e+02, 3.5000e+03, 1.1100e+02, 4.0800e+03, 1.1100e+02,
 3.4880e+03, 1.1100e+02, 3.9430e+03, 1.1100e+02, 6.8460e+03],
[6.4000e+01, 7.4330e+03, 6.4000e+01, 4.2250e+03, 6.4000e+01,
 7.6340e+03, 6.4000e+01, 5.2090e+03, 6.4000e+01, 7.0740e+03,
 6.4000e+01, 2.0900e+03, 6.4000e+01, 2.3880e+03, 6.4000e+01,
 2.0780e+03, 6.4000e+01, 2.2980e+03, 6.4000e+01, 4.0260e+03],
[1.1800e+02, 1.3373e+04, 1.1800e+02, 7.6270e+03, 1.1800e+02,
 1.3952e+04, 1.1800e+02, 9.2590e+03, 1.1800e+02, 1.3014e+04,
 1.1800e+02, 3.7100e+03, 1.1800e+02, 4.3320e+03, 1.1800e+02,
 3.6980e+03, 1.1800e+02, 4.1880e+03, 1.1800e+02, 7.2660e+03],
[7.6000e+01, 8.7530e+03, 7.6000e+01, 4.9810e+03, 7.6000e+01,
 9.0380e+03, 7.6000e+01, 6.1090e+03, 7.6000e+01, 8.3940e+03,
 7.6000e+01, 2.4500e+03, 7.6000e+01, 2.8200e+03, 7.6000e+01,
 2.4380e+03, 7.6000e+01, 2.7180e+03, 7.6000e+01, 4.7460e+03],
[1.1100e+02, 1.2603e+04, 1.1100e+02, 7.1860e+03, 1.1100e+02,
 1.3133e+04, 1.1100e+02, 8.7340e+03, 1.1100e+02, 1.2244e+04,
 1.1100e+02, 3.5000e+03, 1.1100e+02, 4.0800e+03, 1.1100e+02,
 3.4880e+03, 1.1100e+02, 3.9430e+03, 1.1100e+02, 6.8460e+03],
[3.1000e+01, 3.8030e+03, 3.1000e+01, 2.1460e+03, 3.1000e+01,
 3.7730e+03, 3.1000e+01, 2.7340e+03, 3.1000e+01, 3.4440e+03,
 3.1000e+01, 1.1000e+03, 3.1000e+01, 1.2000e+03, 3.1000e+01,
 1.0880e+03, 3.1000e+01, 1.1430e+03, 3.1000e+01, 2.0460e+03],
[3.7000e+01, 4.4630e+03, 3.7000e+01, 2.5240e+03, 3.7000e+01,
 4.4750e+03, 3.7000e+01, 3.1840e+03, 3.7000e+01, 4.1040e+03,
 3.7000e+01, 1.2800e+03, 3.7000e+01, 1.4160e+03, 3.7000e+01,
 1.2680e+03, 3.7000e+01, 1.3530e+03, 3.7000e+01, 2.4060e+03],
[3.1000e+01, 3.8030e+03, 3.1000e+01, 2.1460e+03, 3.1000e+01,
 3.7730e+03, 3.1000e+01, 2.7340e+03, 3.1000e+01, 3.4440e+03,
 3.1000e+01, 1.1000e+03, 3.1000e+01, 1.2000e+03, 3.1000e+01,
 1.0880e+03, 3.1000e+01, 1.1430e+03, 3.1000e+01, 2.0460e+03],
[3.6000e+01, 4.3530e+03, 3.6000e+01, 2.4610e+03, 3.6000e+01,
 4.3580e+03, 3.6000e+01, 3.1090e+03, 3.6000e+01, 3.9940e+03,
 3.6000e+01, 1.2500e+03, 3.6000e+01, 1.3800e+03, 3.6000e+01,
 1.2380e+03, 3.6000e+01, 1.3180e+03, 3.6000e+01, 2.3460e+03],
[6.1000e+01, 7.1030e+03, 6.1000e+01, 4.0360e+03, 6.1000e+01,
 7.2830e+03, 6.1000e+01, 4.9840e+03, 6.1000e+01, 6.7440e+03,
 6.1000e+01, 2.0000e+03, 6.1000e+01, 2.2800e+03, 6.1000e+01,
 1.9880e+03, 6.1000e+01, 2.1930e+03, 6.1000e+01, 3.8460e+03]])
```

- 95% доверительный интервал :

[0.025	0.975]
10.645	21.316
-714.281	-45.894

```
scp.f.ppf(q = 1 - 0.05, dfn = 1, dfd = len(data_2) - 2)
```

```
4.413873419170566
```

Значимость параметров регресии:

b0 => p-value < 0.05, следовательно гипотеза H0 отвергается и параметр значим

b1 => p-value < 0.05, следовательно гипотеза H0 отвергается и параметр значим

Значимость модели регрессии:

F-stat = 5.709 > F-table = 4.413, следовательно гипотеза H0 отвергается и модель значима

Качество модели

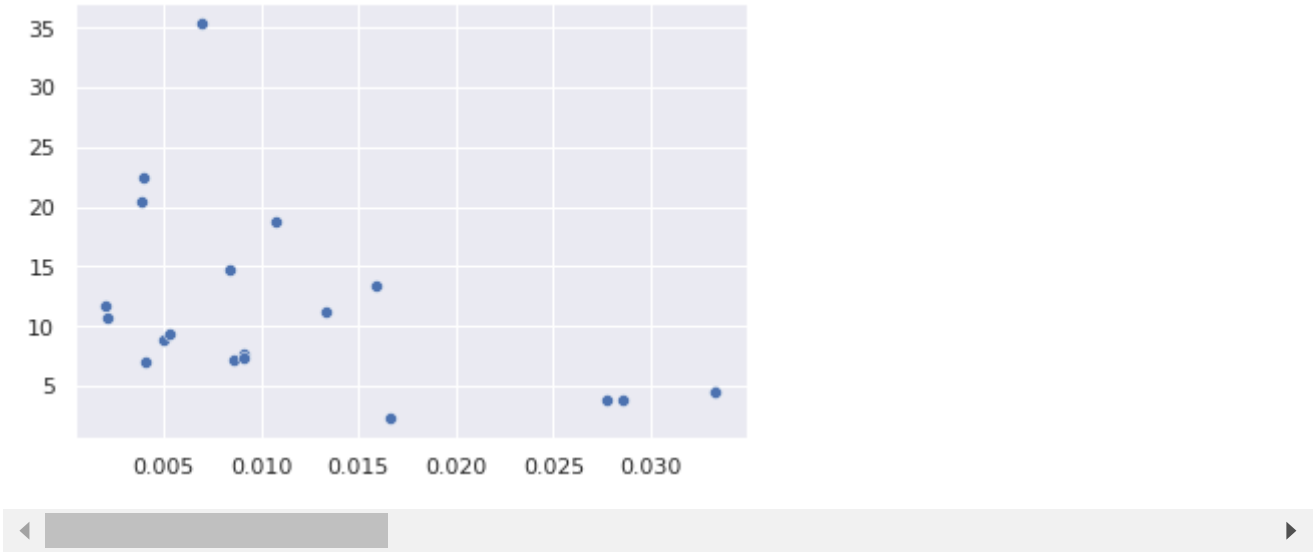
R^2 = 0.241

R^2(adj) = 0.199

Качество модели плохое - значение R^2 меньше 0.5, кроме того есть значительное различие между скорректированным коэффициентом и обычным из-за малого количества наблюдений

```
sns.set()
sns.scatterplot(data_2['x'].values, data_2['y'].values)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd469058cd0>
```



После преобразования нет нормального распределения

Тест Уайта

```
from statsmodels.stats.diagnostic import het_white

#perform White's test
white_test = het_white(fin_model.fit().resid, fin_model.fit().model.exog)
```

```
#define labels to use for output of White's test
labels = ['Test Statistic', 'Test Statistic p-value', 'F-Statistic', 'F-Test p-value']

#print results of White's test
print(dict(zip(labels, white_test))['Test Statistic p-value'])
```

```
0.08498413502113245
```

p-value = 0.085 > 0.05 следовательно гипотеза H0 принимается и модель показывает гомоскедастичность

После применения ДМНК качество и адекватность модели значительно понизилась, что свидетельствует о неадекватности метода решения. При этом метод робустной ковариационной матрицы, не уменьшает качество модели. Стоит выбрать его.

Задание 6_2

Задача (с использованием ПК):

А) Оцените коэффициенты регрессии на примере данных о динамике золотовалютных резервов РФ за период с 26.12.03 по 07.01.05.
X – время, отсчитываемое в днях от начального момента времени 26.12.03, а столбец Y – золотовалютные резервы (в млрд долл.).
Б) Постройте 90%-й доверительный интервал для коэффициентов модели.
В) Проверьте значимость модели регрессии в целом и каждого коэффициента модели по отдельности.
Г) Сделайте выводы о качестве модели.
Д) Проверьте выполнение предпосылки о гомоскедастичности и об отсутствии автокорреляции остатков.
Е) При обнаружении гетероскедастичности или автокорреляции:
1) оцените робастную ковариационную матрицу;
2) предложите пути устранения гетероскедастичности или автокорреляции и постройте новые модели.

Файл с исходными данными задачи – task1.txt.

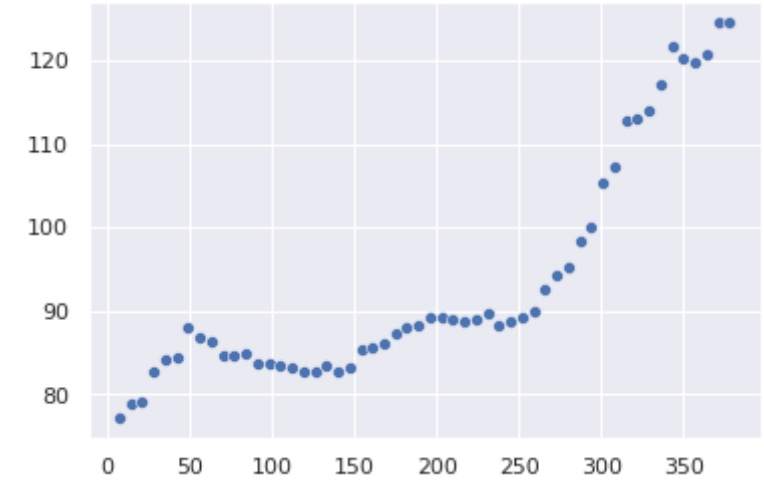
«Автокорреляция»

```
data = pd.read_csv('/content/task1.txt', sep = '\t', decimal = ',')
data.head()
```

	X	Y
0	7	77.1
1	14	78.9
2	21	79.1
3	28	82.7
4	35	84.1

```
sns.set()
sns.scatterplot(data['X'].values, data['Y'].values)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd4685f4290>
```




```
results = sm.OLS.from_formula("Y ~ X", data = data).fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.749
Model:                            OLS   Adj. R-squared:                  0.745
Method:                 Least Squares   F-statistic:                     155.6
Date:                Sat, 26 Nov 2022   Prob (F-statistic):           2.96e-17
Time:                        14:03:49   Log-Likelihood:                -179.59
No. Observations:                  54   AIC:                             363.2
Df Residuals:                      52   BIC:                             367.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     72.8507      1.893     38.483      0.000      69.052      76.649
X              0.1067      0.009     12.473      0.000       0.090       0.124
==============================================================================
Omnibus:                       17.247   Durbin-Watson:                   0.058
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                3.890
```

```
Skew:                          0.188   Prob(JB):                     0.143
Kurtosis:                      1.740   Cond. No.                      449.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
```

```
from statsmodels.stats.stattools import durbin_watson

#perform Durbin-Watson test
durbin_watson(model.resid)
```

```
#Тест Голфреда-Кванта
from statsmodels.stats import api
new_x = np.concatenate((np.ones((len(data['y']), 1)), x.values.reshape(-1, 1)), axis = 1)

api.het_goldfeldquandt(epsilons, new_x)
```

```
(0.016108527948533475, 0.9999978722994026, 'increasing')
```

```
vec_w = sorted(vec_w)
```

```
import scipy.stats as scp

new_w = [round(scp.norm.cdf(vec_w[i])[0], 3) for i in range(len(vec_w))]
new_w
```

```
[0.072,
 0.127,
 0.129,
 0.156,
 0.185,
 0.21,
 0.213,
 0.246,
 0.247,
 0.351,
 0.353,
 0.38,
```

```
        0.389,
        0.416,
        0.561,
        0.588,
        0.693,
        0.783,
        0.816,
        0.865,
        0.878,
        0.897,
        0.937,
        0.995]
```

```
i = np.linspace(0., 1., len(new_w))
k = 0


for cur in range(len(new_w) - 1):
  k += (((new_w[cur] > i[cur]) & (new_w[cur] < i[cur + 1])) == 0)

k, len(new_w)
```

```
     (17, 24)
```