Национальный исследовательский университет «Высшая школа экономики»

Факультет: Московский институт электроники и математики им.Тихонова

Образовательная программа: Прикладная математика

# Отчет по модульной работе №2
# по майнору «Прикладной статистический анализ»

Работу выполнила

студентка 3 курса

Беломытцева Алена

Владимировна

Москва, 2023г.

# Задание №1

По данным по Великобритании о потреблении цыплят (Y), среднедушевом доходе (X1), стоимости одного фунта цыплят (X2), стоимости одного фунта свинины (X3) и стоимости одного фунта говядины (X4).

***а)*** Необходимо построить, сравнить и проинтерпретировать уравнения регрессии вида:

Для этого все изначальные данные были прологарифмированы, так как далее будут применяться замены переменных на их логарифмы.

```
for column in df_log.columns:
  df_log[column] = np.log(df_log[column])
```

```
df_log
```

| | Y | X1 | X2 | X3 | X4 |
|---|---|---|---|---|---|
| 0 | 3.440418 | 6.200306 | 3.618993 | 4.001864 | 4.348987 |
| 1 | 3.505557 | 6.270232 | 3.640214 | 4.154185 | 4.384524 |
| 2 | 3.572346 | 6.328472 | 3.671225 | 4.245634 | 4.387014 |
| 3 | 3.594569 | 6.437111 | 3.632309 | 4.188138 | 4.429626 |
| 4 | 3.602777 | 6.501890 | 3.648057 | 4.166665 | 4.448516 |
| 5 | 3.648057 | 6.576191 | 3.691376 | 4.248495 | 4.540098 |
| 6 | 3.698830 | 6.644050 | 3.653252 | 4.293195 | 4.664382 |
| 7 | 3.696351 | 6.737323 | 3.683867 | 4.216562 | 4.652054 |
| 8 | 3.732896 | 6.815201 | 3.681351 | 4.370713 | 4.736198 |
| 9 | 3.698830 | 6.836367 | 3.953165 | 4.558079 | 4.821088 |
| 10 | 3.706228 | 6.929027 | 3.889777 | 4.545420 | 4.848900 |
| 11 | 3.691376 | 7.061249 | 4.065602 | 4.816241 | 4.962145 |
| 12 | 3.754199 | 7.207564 | 4.058717 | 4.866765 | 4.967032 |
| 13 | 3.786460 | 7.278905 | 4.034241 | 4.767289 | 4.935912 |
| 14 | 3.843744 | 7.362328 | 4.154185 | 4.874434 | 5.108971 |
| 15 | 3.923952 | 7.472558 | 4.120662 | 4.865995 | 5.314683 |
| 16 | 3.914021 | 7.597998 | 4.075841 | 4.852030 | 5.391808 |
| 17 | 3.945458 | 7.722279 | 4.195697 | 4.948760 | 5.400874 |
| 18 | 3.968403 | 7.815490 | 4.254193 | 5.125154 | 5.449320 |

1. функция спроса $\hat{y} = b_0 * x_2^{b_2}$

Для построения модели степенной регрессии необходимо изначально сделать замену переменных:

Пусть $u = ln\, y$ и $v = ln\, x_2$

Тогда уравнение примет вид: $\hat{u} = ln\, b_0 + b_2 * v$

После такой замены можно построить модель линейной регрессии:

```
m1=smf.ols('Y~X2', data=df_log)
fitted=m1.fit()
print(fitted.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.738
Model:                            OLS   Adj. R-squared:                  0.722
Method:                 Least Squares   F-statistic:                     47.86
Date:                Sat, 09 Dec 2023   Prob (F-statistic):           2.49e-06
Time:                        14:44:13   Log-Likelihood:                 22.538
No. Observations:                  19   AIC:                            -41.08
Df Residuals:                      17   BIC:                            -39.19
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      1.5777      0.311      5.081      0.000       0.923       2.233
X2             0.5527      0.080      6.918      0.000       0.384       0.721
==============================================================================
Omnibus:                        0.824   Durbin-Watson:                   0.780
Prob(Omnibus):                  0.662   Jarque-Bera (JB):                0.757
Skew:                          -0.228   Prob(JB):                        0.685
Kurtosis:                       2.135   Cond. No.                         71.8
==============================================================================

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

Итак, можно сделать обратную замену: тогда уравнение регрессии будет выглядеть:

$$\hat{y} = ln\, 1.577 * x_2^{0.5527} = 0.455 * x_2^{0.5527}$$

У этой модели неплохой $R^2 = 0,738$ - модель объясняет 73,8% дисперсии зависимой . Скорректированный коэффициент

детерминации - тот, в котором количество факторов не влияет на оценку равен 0,722.

Коэффициенты значимы, так как p-value меньше уровня значимости 0,05.

2. Функция потребления $\hat{y} = b_0 * x_1^{b_1}$

Для построения также нужно сделать замену переменных:

Пусть $u = ln\, y$ и $v = ln\, x_1$

Тогда уравнение примет вид: $\hat{u} = ln\, b_0 + b_1 * v$

После такой замены можно построить модель линейной регрессии:

```
m2=smf.ols('Y~X1', data=df_log)
fitted=m2.fit()
print(fitted.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.939
Model:                            OLS   Adj. R-squared:                  0.935
Method:                 Least Squares   F-statistic:                     259.8
Date:                Sat, 09 Dec 2023   Prob (F-statistic):           9.84e-12
Time:                        14:44:13   Log-Likelihood:                 36.323
No. Observations:                  19   AIC:                            -68.65
Df Residuals:                      17   BIC:                            -66.76
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      1.7460      0.123     14.204      0.000       1.487       2.005
X1             0.2849      0.018     16.119      0.000       0.248       0.322
==============================================================================
Omnibus:                        0.699   Durbin-Watson:                   0.753
Prob(Omnibus):                  0.705   Jarque-Bera (JB):                0.672
Skew:                          -0.381   Prob(JB):                        0.715
Kurtosis:                       2.484   Cond. No.                         101.
==============================================================================

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

У этой модели хороший $R^2 = 0,939$ - модель объясняет 93,9% дисперсии зависимой. Скорректированный коэффициент детерминации равен 0,935. Получилась хорошая модель регрессии. Произведем обратную замену и получим:

$$\hat{y} = ln\,1.746 * x_1^{0.2849} = 0.557 * x_2^{0.02849}$$

Коэффициенты значимы, так как p-value меньше уровня значимости 0,05.

3. Далее необходимо построить функцию спроса-потребления $\hat{y} = b_0 * x_1^{b_1} * x_2^{b_2}$

Для построения также нужно сделать замену переменных:

Пусть $u = ln\,y$ и $v_1 = ln\,x_1$ и $v_2 = ln\,x_2$

Тогда уравнение примет вид: $\hat{u} = ln\,b_0 + b_1 * v_1 + b_2 * v_2$

После такой замены можно построить модель линейной регрессии:

```
m3=smf.ols('Y~X1+X2', data=df_log)
fitted=m3.fit()
print(fitted.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.964
Model:                            OLS   Adj. R-squared:                  0.959
Method:                 Least Squares   F-statistic:                     213.7
Date:                Sat, 09 Dec 2023   Prob (F-statistic):           2.87e-12
Time:                        14:44:13   Log-Likelihood:                 41.376
No. Observations:                  19   AIC:                            -76.75
Df Residuals:                      16   BIC:                            -73.92
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.0185      0.127     15.937      0.000       1.750       2.287
X1             0.4161      0.042     10.011      0.000       0.328       0.504
X2            -0.3048      0.091     -3.352      0.004      -0.498      -0.112
==============================================================================
Omnibus:                        0.235   Durbin-Watson:                   1.566
Prob(Omnibus):                  0.889   Jarque-Bera (JB):                0.415
Skew:                           0.166   Prob(JB):                        0.812
Kurtosis:                       2.356   Cond. No.                         170.
==============================================================================

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

У этой модели хороший $R^2 = 0,964$ - модель объясняет 96,4% дисперсии зависимой. Скорректированный коэффициент детерминации равен 0,959. Получилась хорошая модель регрессии. Произведем обратную замену и получим:

$$\widehat{y} = ln\,2.0185 * x_1^{0.4161} * x_2^{0.0.091} = 0.702 * x_1^{0.4161} * x_2^{0.0.091}$$

Коэффициенты значимы, так как p-value меньше уровня значимости 0,05.

4. Функция спроса с учетом цены на товары-заменители $\widehat{y} = b_0 * x_2^{b_2} * x_3^{b_3} * x_4^{b_4}$

Для построения также нужно сделать замену переменных:

Пусть $u = ln\,y$ и $v_2 = ln\,x_2$ и $v_3 = ln\,x_3$ $v_4 = ln\,x_4$

Тогда уравнение примет вид: $\widehat{u} = ln\,b_0 + b_2 * v_2 + b_3 * v_3 + b_4 * v_4$

```
m4=smf.ols('Y~X2+X3+X4', data=df_log)
fitted=m4.fit()
print(fitted.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.946
Model:                            OLS   Adj. R-squared:                  0.935
Method:                 Least Squares   F-statistic:                     87.74
Date:                Sat, 09 Dec 2023   Prob (F-statistic):           9.73e-10
Time:                        14:44:13   Log-Likelihood:                 37.560
No. Observations:                  19   AIC:                            -67.12
Df Residuals:                      15   BIC:                            -63.34
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.3152      0.222     10.450      0.000       1.843       2.787
X2            -0.4875      0.211     -2.310      0.036      -0.937      -0.038
X3             0.2374      0.156      1.526      0.148      -0.094       0.569
X4             0.4601      0.075      6.112      0.000       0.300       0.620
==============================================================================
Omnibus:                        2.117   Durbin-Watson:                   1.235
Prob(Omnibus):                  0.347   Jarque-Bera (JB):                1.739
Skew:                          -0.641   Prob(JB):                        0.419
Kurtosis:                       2.255   Cond. No.                         288.
==============================================================================

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

У этой модели хороший $R^2 = 0,946$ - модель объясняет 94,6% дисперсии зависимой. Скорректированный коэффициент детерминации равен 0,935. Получилась хорошая модель регрессии. Произведем обратную замену и получим:

$$\hat{y} = ln\,2.3152 * x_2^{-0.4875} * x_3^{0.2374} * x_3^{0.4601} = 0.839 * x_2^{-0.4875} * x_3^{0.2374} * x$$

Коэффициенты $b_0$, $b_4$ значимы, так как p-value меньше уровня значимости 0,05. Но $b_2$, $b_3$ незначимы, так как p-значение больше 0,05.

***b)*** Применить тест Шапиро-Уилка для проверки нормальности распределения X1, X2, X3, X4.

Тест Шапиро-Уилка это тест на нормальность

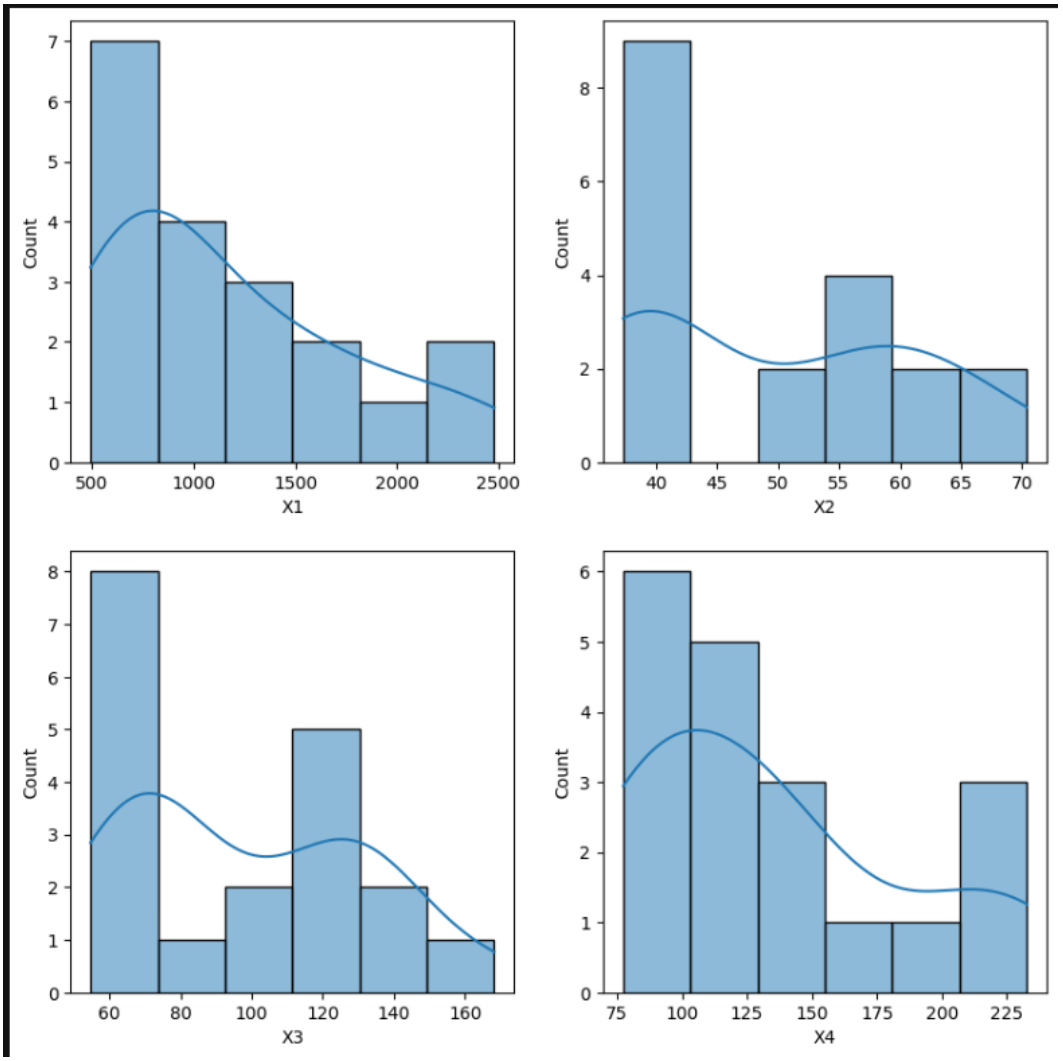
$H_0$: Выборка получена из нормального распределения.

$H_1$: Выборка получена из не нормального распределения.

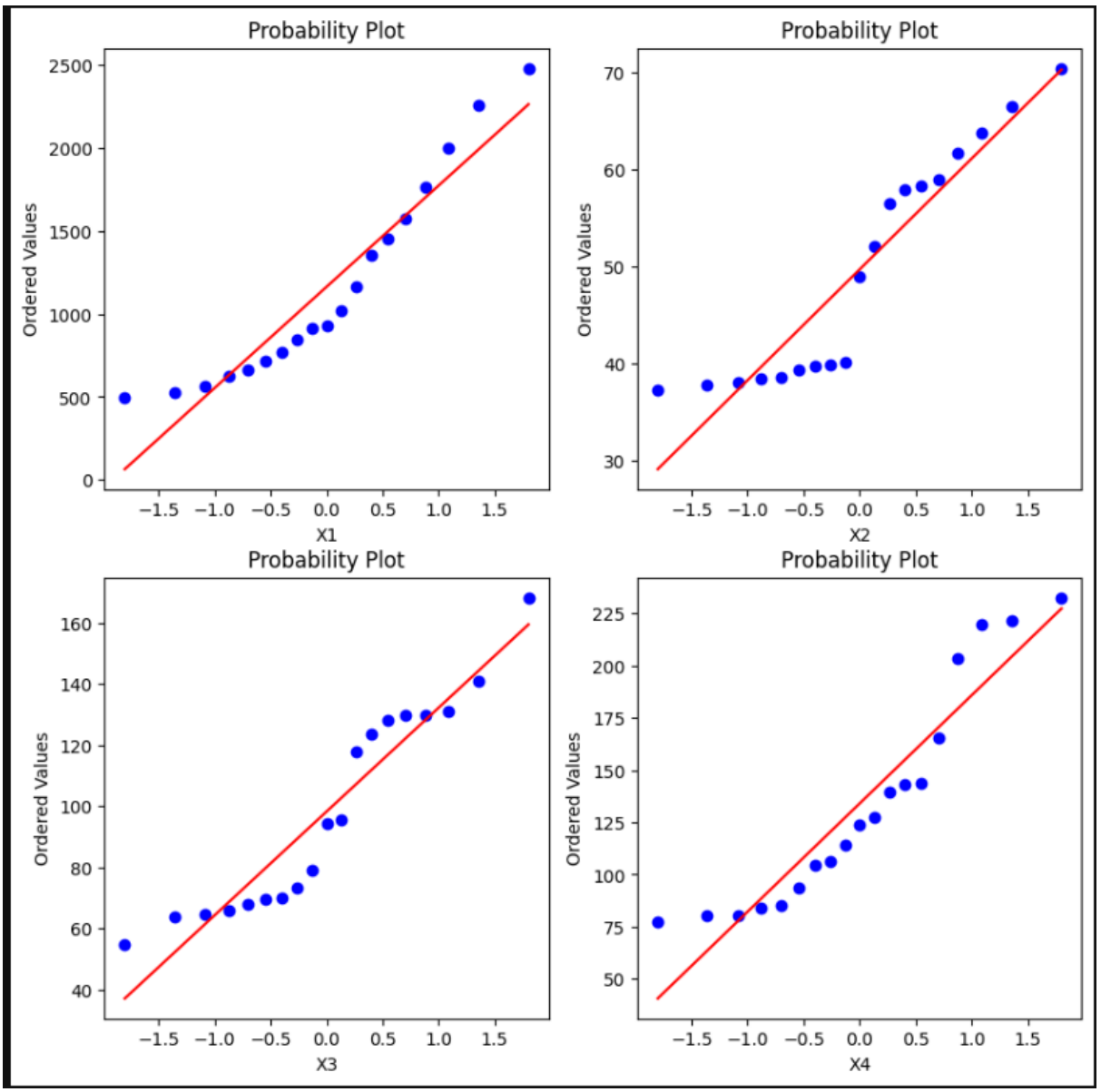
```
cols = data_1.columns
for i in range(2,6):
   print(cols[i],'       ', stats.shapiro(data_1.iloc[:, i]))
```

```
X1          ShapiroResult(statistic=0.896443247795105, pvalue=0.04197002574801445)
X2          ShapiroResult(statistic=0.8548540472984314, pvalue=0.008054640144109726)
X3          ShapiroResult(statistic=0.89530348777771, pvalue=0.04003702849149704)
X4          ShapiroResult(statistic=0.8776261210441589, pvalue=0.01952611841261387)
```

Так как p-value для всех переменных меньше критического значения в 0,05, то имеем основания для отвержения нулевой гипотезы. А значит, что все переменные не распределены нормально.
Посмотрим также на их гистограммы.




И график QQ (квантиль-квантиль)




На этом графике, чем больше распределение выборки соответствует нормальному распределению, тем синие точки больше лежат на красной прямой. По этим графикам видно, что распределения далеки от нормального.

***c)*** Для данных из пункта b, которые не прошли проверку, применить преобразования Бокса-Кокса.

Так как все переменные не прошли проверку на нормальность в предыдущем пункте, поправку Бокса-Кокса будем делать для всех переменных x.

```
best_lambda = np.zeros(4)
new_data_1 = data_1.drop('t',axis=1)
new_data_1 = new_data_1.drop('Y',axis=1)
```

```
i =0
for column in  new_data_1.columns:
  new_data_1[column], best_lambda[i] = boxcox(data_1[column])
  i+=1
```

```
cols = new_data_1.columns
for i in range(4):
   print(cols[i],'       ', stats.shapiro(new_data_1.iloc[:, i]))
```

```
X1         ShapiroResult(statistic=0.9598411917686462, pvalue=0.569256603717804)
X2         ShapiroResult(statistic=0.8407939672470093, pvalue=0.004768925718963146)
X3         ShapiroResult(statistic=0.9084967374801636, pvalue=0.06947548687458038)
X4         ShapiroResult(statistic=0.9304439425468445, pvalue=0.17648252844810486)
```

```
best_lambda
```

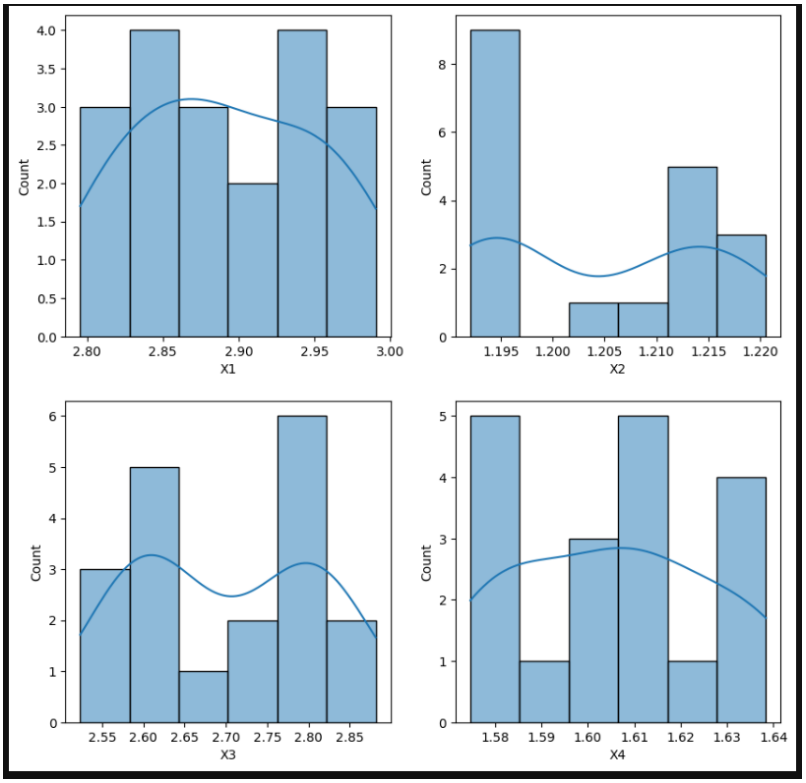
```
array([-0.30309361, -0.79097492, -0.25132616, -0.58520377])
```

После поправки Бокса-Кокса для всех переменных, кроме $x_2$, в тесте Шапиро-Уилка p-значение стало больше, чем 0,05, а значит не имеем оснований для отвержения нулевой гипотезы и говорим о нормальности распределения выборок $x_1$, $x_3$, $x_4$. Для выборки $x_2$ нулевая гипотеза все также отвергается, так как ее p-value меньше критического значения = 0,05.

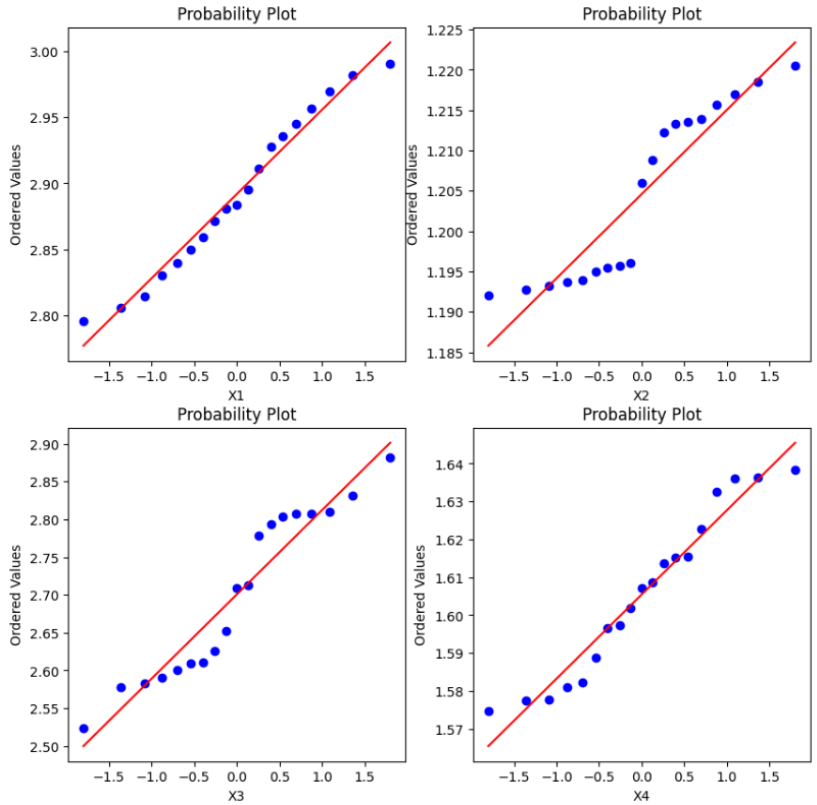
После применения поправки Бокса-Кокса данные выглядят так:

| | X1 | X2 | X3 | X4 |
|---|---|---|---|---|
| 0 | 2.795503 | 1.192043 | 2.523566 | 1.574717 |
| 1 | 2.806068 | 1.193245 | 2.578226 | 1.577477 |
| 2 | 2.814698 | 1.194966 | 2.610052 | 1.577668 |
| 3 | 2.830396 | 1.192800 | 2.590128 | 1.580898 |
| 4 | 2.839513 | 1.193684 | 2.582613 | 1.582304 |
| 5 | 2.849752 | 1.196062 | 2.611036 | 1.588905 |
| 6 | 2.858904 | 1.193974 | 2.626317 | 1.597316 |
| 7 | 2.871180 | 1.195655 | 2.600014 | 1.596509 |
| 8 | 2.881167 | 1.195519 | 2.652413 | 1.601905 |
| 9 | 2.883841 | 1.208818 | 2.713429 | 1.607086 |
| 10 | 2.895347 | 1.205967 | 2.709396 | 1.608728 |
| 11 | 2.911216 | 1.213536 | 2.792929 | 1.615145 |
| 12 | 2.928051 | 1.213259 | 2.807893 | 1.615413 |
| 13 | 2.935992 | 1.212262 | 2.778248 | 1.613696 |
| 14 | 2.945064 | 1.216968 | 2.810148 | 1.622857 |
| 15 | 2.956704 | 1.215698 | 2.807666 | 1.632605 |
| 16 | 2.969485 | 1.213945 | 2.803549 | 1.635968 |
| 17 | 2.981678 | 1.218496 | 2.831777 | 1.636353 |
| 18 | 2.990526 | 1.220565 | 2.881521 | 1.638379 |

И их гистограммы выглядят так:




QQ-plot:



На графике видно, что ближе всего к нормальному распределению выборка X1, X4, а точки в X2 и X3 не лежат прямой куда сильнее, чем должны.

***d)*** Применить критерий Манна-Уитни для оценки различий данных X3 и X4. Дать интерпретацию.

$H_0$: Различия между выборками незначительные.

$H_1$: Различия между X3 и X4 существенны.

```
stats.mannwhitneyu(data_1['X3'], data_1['X4'], alternative='two-sided')
```

```
MannwhitneyuResult(statistic=102.0, pvalue=0.022774865372337766)
```

Так как p-value меньше критического значения в 0,05, имеем основания для отвержения нулевой гипотезы. А значит принимается гипотеза $H_1$ о том, что между выборками X3 и X4 есть существенные различия.

***e)*** Применить критерий Флигнера-Клипера для X2 и X3. Применить критерий Стьюдента для этих данных с учетом полученных результатов. Привести интерпретацию.

Для начала применим критерий Флигнера-Клипера.

$H_0$: Гомогенность дисперсий: $\sigma_2 = \sigma_3$ .

$H_1$: Гомогенность дисперсий отсутствует: $\sigma_2 \neq \sigma_3$ .

```
stats.fligner( data_1['X2'], data_1['X3'])
```

```
FlignerResult(statistic=16.797459660607196, pvalue=4.158894013328987e-05)
```

Так как p-value меньше критического значения в 0,05, то имеем основания для отвержения нулевой гипотезы. Значит принимается альтернативная гипотеза и дисперсии у X2 и X3 не равны.

Далее критерий Стьюдента:

Применим критерий Стьюдента для проверки гипотезы о равенстве средних в переменных X2 и X3

$H_0$: Средние выборок равны: $\mu_2 = \mu_3$ .

$H_1$: Средние выборок не равны: $\mu_2 \neq \mu_3$ .

```
stats.ttest_rel( data_1['X2'], data_1['X3'])
```

```
TtestResult(statistic=-9.433823005758445, pvalue=2.1731208937339906e-08, df=18)
```

Так как p-value меньше критического значения в 0,05, то имеем основания для отвержения нулевой гипотезы. Значит принимается альтернативная гипотеза о неравенстве средних в выборках X2 и X3.

## Задание №2

***a)*** Найти степень полинома, при которой коэффициент детерминации оказывается близким к 1.

```
r_adj = np.zeros(30)
for i in range(30):
    polynom=PolynomialFeatures(degree=i+1)
    X_poly = polynom.fit_transform(X_2)
    reg=LinearRegression()
    reg.fit(X_poly,y_2)
    r_adj[i] = reg.score(X_poly,y_2)
```

```
r_adj
```

```
array([0.75650076, 0.91250161, 0.96065086, 0.96564433, 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 0.99999856, 0.99999888, 0.99999912, 0.9999993 ,
       0.9999938 , 0.99998887, 0.99999366, 0.99990648, 0.99995477])
```

Для этого построим матрицу, в которую будем добавлять значение коэффициента детерминации для каждого полинома.
Коэффициент детерминации становится равным единице при полиноме 5 степени.

***b)*** Показать, что результат достигнут за счет переобучения модели.
Для этого будет применена кросс-валидация. Выборка разделена на тренировочную - 67% от выборки и тренировочную - 33%. Обучим модель на полиноме 5 степени и проверим качество модели на тестовой выборке.

```
polynom=PolynomialFeatures(degree=5)
X_poly = polynom.fit_transform(X_2)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_poly, y_2, test_size=0.25, random_state=0)
```

```
reg_poly=LinearRegression()
reg_poly.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
reg_poly.score(X_train,y_train)
```

```
1.0
```

```
reg_poly.score(X_test,y_test)
```

```
-43106.41869105432
```

При обучении коэффициент детерминации все еще равен единице, а вот коэффициент детерминации при использовании тестовой выборки вообще отрицательный. Это говорит о том, что построенная модель вообще не объясняет результирующую переменную. И такой хороший коэффициент детерминации на тренировочной выборке был получен лишь благодаря переобучению.

***c)*** Выбрать оптимальную степень полинома, в том числе с содержательной точки зрения, предполагая, что фирмы можно считать однородными по прочим возможным показателям их деятельности.

Для этого построим полиномы степеней меньше пятой и посмотрим на MSE и коэффициенты детерминации получившихся моделей

```
polynom=PolynomialFeatures(degree=4)
X_poly = polynom.fit_transform(X_2)
X_train, X_test, y_train, y_test = train_test_split(X_poly, y_2, test_size=0.25, random_state=0)
reg_poly=LinearRegression()
reg_poly.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
reg_poly.score(X_train,y_train)
```

```
0.845518363012804
```

```
reg_poly.score(X_test,y_test)
```

```
-162.72586209154625
```

Для полинома 4 степени можно заметить, что модель все также переобучена. Ее MSE равен 4.97458485e+05 на тестовой выборке.

Для полинома 3 степени коэффициент детерминации стал равен 0,23 на тестовой выборке, что говорит о 23% объясняемых дисперсий - это значит что данные лучше описываются полиномом 3 степени, чем полиномом 4 степени.

```
polynom=PolynomialFeatures(degree=3)
X_poly = polynom.fit_transform(X_2)
X_train, X_test, y_train, y_test = train_test_split(X_poly, y_2, test_size=0.25, random_state=0)
reg_poly=LinearRegression()
reg_poly.fit(X_train,y_train)
reg_poly.score(X_train,y_train)
```

```
0.969758225661135
```

```
reg_poly.score(X_test,y_test)
```

```
0.2317917096180936
```

При этом MSE для полинома 3 степени равен 2.33409510e+03.

Проверим также полином 2 степени:

```
polynom=PolynomialFeatures(degree=2)
X_poly = polynom.fit_transform(X_2)
X_train, X_test, y_train, y_test = train_test_split(X_poly, y_2, test_size=0.25, random_state=0)
reg_poly=LinearRegression()
reg_poly.fit(X_train,y_train)
reg_poly.score(X_train,y_train)
```

```
0.9135799866595289
```

```
reg_poly.score(X_test,y_test)
```

```
0.8612810916833512
```

Для этого полинома коэффициент детерминации на тестовой выборке стал равен 0,86, что говорит 86% объясняемых дисперсий - хороший результат, модель не переобучена и действительно способна “предсказывать” некоторые данные по полиному. При этом ее MSE равен 421,47829, что сравнительно меньше, чем полиномы третьей степени.
Также ради интереса была рассмотрена линейная модель:

```
X_train, X_test, y_train, y_test = train_test_split(X_2, y_2, test_size=0.25, random_state=0)
reg_2=LinearRegression()
reg_2.fit(X_train,y_train)
reg_2.score(X_test,y_test)
```

```
0.6627814374785264
```

Ее коэффициент детерминации равен 0,66, что говорит о 66% объясняемых дисперсий. Этот показатель хуже, чем для полинома второй степени.

Таким образом, оптимальной моделью будет модель построенная на полиноме второй степени.

## Задание №3

Применить метод главных компонент для понижения размерности данных о результативности деятельности российских вузов. Данные в файле «задание3.xlsx»

***a)*** Определить минимальное количество компонент, которые необходимо использовать для сохранения 75% первоначальной информации.

Метод главных компонент применяется для уменьшения размерности данных и уменьшения в них шума - данных, не несущих значимой информации.

Для начала работы с методом главных компонент необходимо центрировать данные.  Можно также просто стандартизировать, что и было сделано.

Далее с помощью встроенного модуля были найдены главные компоненты для всех переменных (соответственно, на входе потребовалось 14 компонент). Для получившихся данных был посмотрен параметр explained_variance_ratio, который показывает долю объясняемой дисперсии каждой компонентой. Если посмотреть на кумулятивную сумму, можно сделать вывод о том, что при использовании первых пяти компонент количество сохраняемой информации будет равно 77,92865% от изначального количества. Из этого следует то, что можно использовать первые 5 компонент для сохранения более чем 75% данных.

```
pca=PCA(n_components=14)
pca_result=pca.fit_transform(df_std)
```

```
pca_result
```

```
array([[-1.49829115e+00, -3.15552644e-01,  4.40496262e-01, ...,
         1.64342630e-01,  1.62305155e-01, -4.10688129e-02],
       [-1.65456677e+00,  4.18533598e-01,  7.27593106e-01, ...,
        -1.29954182e-02,  1.89024657e-02,  1.62467510e-03],
       [-7.23180294e-02,  6.91172854e-01,  2.54190197e-01, ...,
        -1.53858814e-01, -6.10068702e-02, -5.04606217e-02],
       ...,
       [-1.70684630e+00,  2.25213157e+00, -1.72138949e+00, ...,
         1.17928713e-01,  4.16926365e-02,  1.15055572e-01],
       [ 9.69921300e-01, -1.09259771e+00, -1.02562833e+00, ...,
        -1.33841890e+00, -7.36880445e-01,  1.37991990e-01],
       [-1.28287332e+00, -6.61970714e-01, -2.52055565e+00, ...,
         3.19051846e-01, -6.08883375e-03,  4.92186600e-03]])
```

```
pca.explained_variance_ratio_
```

```
array([0.41343993, 0.14185423, 0.09350968, 0.0663221 , 0.06416057,
       0.04846162, 0.04009085, 0.03766761, 0.03471945, 0.02301361,
       0.02033594, 0.00919131, 0.00479933, 0.00243377])
```

```
np.cumsum(pca.explained_variance_ratio_)
```

```
array([0.41343993, 0.55529416, 0.64880383, 0.71512594, 0.7792865 ,
       0.82774813, 0.86783898, 0.90550658, 0.94022603, 0.96323964,
       0.98357558, 0.99276689, 0.99756623, 1.        ])
```

***b)*** Выписать формулы зависимости главных компонент из пункта а) от первоначальных данных.

```
loadings = pca.components_
num_pc = pca.n_features_
pc_list = ["PC"+str(i) for i in list(range(1, num_pc+1))]
loadings_df = pd.DataFrame.from_dict(dict(zip(pc_list, loadings)))
loadings_df['variable'] = df.columns.values
loadings_df = loadings_df.set_index('variable')
loadings_df
```

```
c:\users\a-bel\pycharmprojects\4\lib\site-packages\sklearn\utils\deprecation.py:101: FutureWarning: Attribute `n_features_` was deprecated in version 1.
2 and will be removed in 1.4. Use `n_features_in_` instead.
  warnings.warn(msg, category=FutureWarning)
```

| variable | PC1 | PC2 | PC3 | PC4 | PC5 |
|---|---|---|---|---|---|
| Ведущий вуз | 0.286077 | -0.024815 | 0.006826 | -0.394031 | -0.281518 |
| Средний балл ЕГЭ студентов, принятых по результатам ЕГЭ на обучение по очной форме по программам бакалавриата и специалитета за счет средств соответствующих бюджетов бюджетной системы РФ | 0.130998 | -0.286017 | -0.552065 | 0.089233 | -0.258878 |
| Количество цитирований публикаций, изданных за последние 5 лет, индексируемых в информационно-аналитической системе научного цитирования Web of Science в расчете на 100 НПР | 0.341195 | 0.220335 | -0.067550 | 0.257806 | 0.314041 |
| Количество цитирований публикаций, изданных за последние 5 лет, индексируемых в информационно-аналитической системе научного цитирования Scopus в расчете на 100 НПР | 0.328794 | 0.207432 | -0.083680 | 0.195291 | 0.339940 |
| Число публикаций организации, индексируемых в информационно-аналитической системе научного цитирования Web of Science, в расчете на 100 НПР | 0.377938 | 0.109494 | -0.071680 | 0.065173 | 0.059975 |
| Число публикаций организации, индексируемых в информационно-аналитической системе научного цитирования Scopus, в расчете на 100 НПР | 0.384608 | 0.084955 | -0.037909 | 0.002735 | 0.071934 |
| Доходы от НИОКР (за исключением средств бюджетов бюджетной системы Российской Федерации, государственных фондов поддержки науки) в расчете на одного НПР | 0.321761 | -0.136513 | 0.193202 | -0.383670 | 0.111599 |
| Удельный вес численности НПР без ученой степени – до 30 лет, кандидатов наук – до 35 лет, докторов наук – до 40 лет, в общей численности НПР | 0.202066 | -0.160758 | 0.355389 | 0.073756 | -0.512340 |
| Количество полученных грантов за отчетный год в расчете на 100 НПР | 0.155014 | 0.161343 | 0.346103 | 0.514355 | -0.402571 |
| Доходы образовательной организации из средств от приносящей доход деятельности в расчете на одного НПР | 0.182643 | -0.498700 | -0.209975 | 0.023377 | 0.092271 |
| Число статей, подготовленных совместно с зарубежными организациями | 0.280627 | 0.092188 | -0.186320 | 0.188699 | -0.183204 |
| Доля доходов вуза из внебюджетных источников | 0.098931 | -0.594256 | -0.012687 | 0.251930 | 0.061234 |
| Доля доходов вуза от научных исследований и разработок в общих доходах вуза | 0.307728 | 0.039309 | 0.170181 | -0.435781 | -0.021750 |
| Доля внебюджетных средств в доходах от научных исследований и разработок | 0.027116 | -0.352705 | 0.534504 | 0.145561 | 0.384015 |

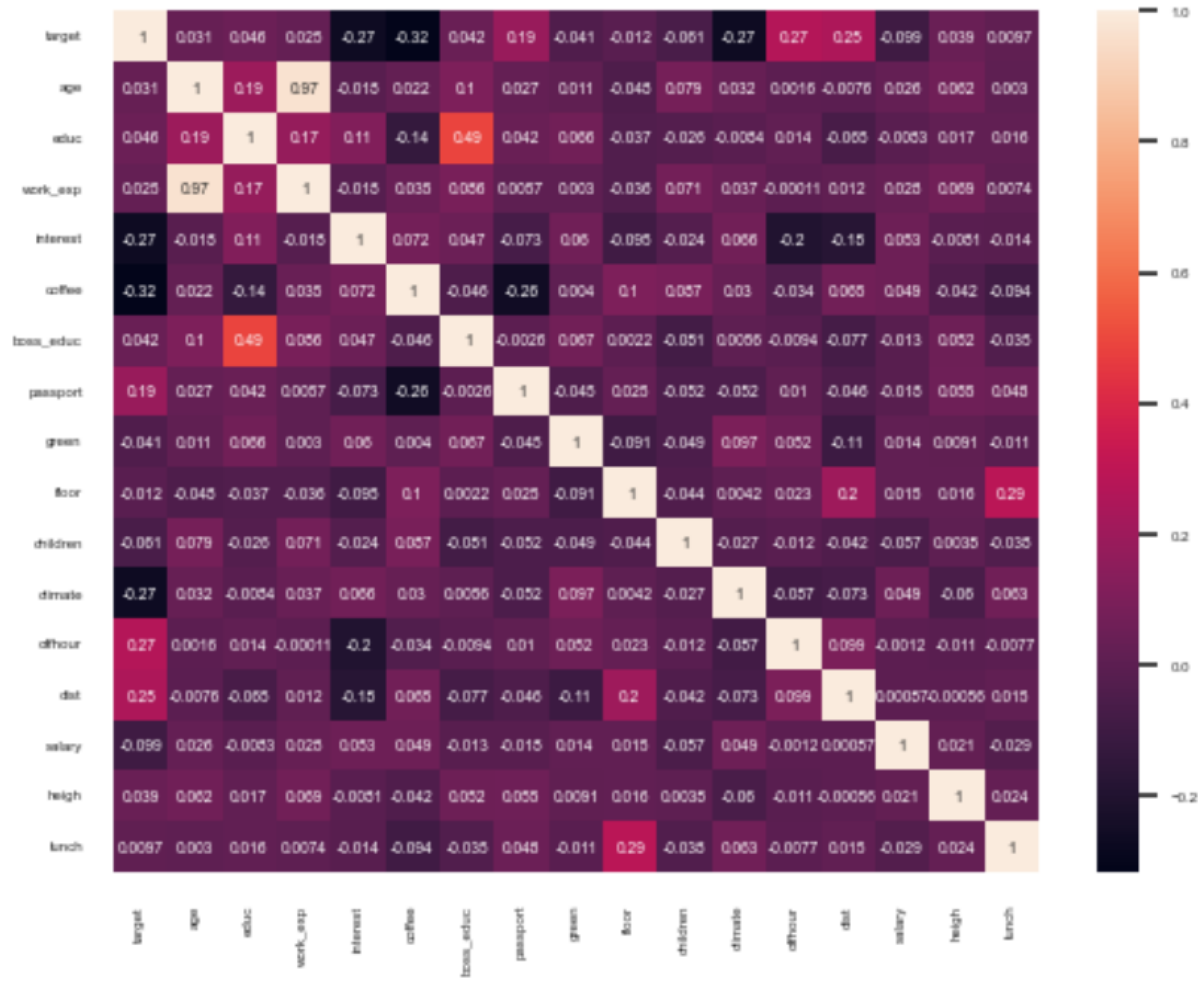
Дополнение: пункт а) был сделан двумя способами: с помощью sklearn и вручную с помощью нахождения ковариационных матриц, их собственных значений и векторов. Затем для каждого собственного значения были посчитаны отношение собственных значений  к сумме всех собственных значений, с помощью чего была получена доля объясняемая компонентами. Далее то же, что было сделано и с помощью модулей.

## Задание №4

По данным из файла «задание4.xls» обучить модель логистической регрессии с целью прогнозирования увольнения сотрудника.
***а)*** Отобрать переменные, которые будут включаться в модель. Обосновать сделанный выбор. Обучить модель логистической регрессии. Дать интерпретацию полученных результатов.

Для того, чтоб отобрать переменные, которые будут участвовать в логистической регрессии может быть использовано несколько подходов:

- можно составить корреляционную матрицу и выбрать те данные, которые хорошо коррелируют с результирующей переменной
- с помощью встроенных модулей выбрать переменные которые наиболее сильно влияют на результирующую переменную.




По матрице корреляций по всем данным видно, что все не очень хорошо коррелирует между собой, что вообще хорошо, но также и корреляции с результирующей переменной тоже не самые большие. Итак, наиболее сильно с переменной target коррелируют переменные  interest, coffee, passport,  climate, offhour, dist, salary.

| | features | coefficients |
|---|---|---|
| 12 | dist | 1.758746 |
| 11 | offhour | 0.525394 |
| 6 | passport | 0.364096 |
| 5 | boss_educ | 0.170270 |
| 2 | work_exp | 0.156533 |
| 0 | age | 0.146026 |
| 1 | educ | 0.056602 |
| 7 | green | 0.033212 |
| 14 | heigh | -0.073093 |
| 13 | salary | -0.164500 |
| 15 | lunch | -0.339667 |
| 9 | children | -0.341151 |
| 8 | floor | -0.447065 |
| 3 | interest | -0.774321 |
| 10 | climate | -0.797006 |
| 4 | coffee | -1.532943 |

По коэффициентам можно понять, что в модели наименьшие коэффициенты у параметров boss_educ, work_exp, age, educ, green, heigh, salary. Попробуем удалить эти переменные и сравним характеристиками изначальной модели, содержащей все переменные.

Изначальная модель:

```
Confusion Matrix :
 [[182  29]
 [ 60  54]]
Accuracy :  0.7261538461538461
Roc-auc score:
 0.6681217261162384
```

Модель после удаления boss_educ, lunch ,  work_exp, age, educ, green, heigh, salary, children.( осталось 7 переменных) и с нормировкой minmax

```
Confusion Matrix :
 [[188  23]
 [ 63  51]]
Accuracy :  0.7353846153846154
Roc-auc score:
 0.6691818408580693
```

Видно, что после удаления 9 переменных значение accuracy незначительно выросло, так же как и количество действительно положительных ответов, число действительно ложных же уменьшилось.

Попробуем удалить те же переменные, но стандартизируем, а не нормируем данные:

```
Confusion Matrix :
 [[186  25]
 [ 60  54]]
Accuracy :  0.7384615384615385
Roc-auc score:
 0.6776003991020204
```

Снова незначительное повышение accuracy.
Также были рассчитаны показатели для модели, у которой от первоначальных данных были удалены 'boss_educ', 'work_exp', 'age', 'green', 'educ', 'heigh', 'lunch и также стандартизированы данные.

```
Confusion Matrix :
 [[184  27]
 [ 58  56]]
Accuracy :  0.7384615384615385
Roc-auc score:
 0.6816329924336909
```

Значение ROC-AUC score больше у второй модели, а значит теоретически она должна быть лучше.

Таким образом далее будут использоваться переменные:

| | interest | coffee | passport | floor | children | climate | offhour | dist | salary |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 8 | 1.3 | 0 | 3 | 2 | 0 | 0 | 10.6 | 1 |
| **1** | 0 | 0.0 | 1 | 3 | 0 | 0 | 0 | 11.5 | 0 |
| **2** | 7 | 1.9 | 1 | 5 | 0 | 3 | 0 | 12.8 | 1 |
| **3** | 7 | 2.0 | 0 | 3 | 1 | 1 | 0 | 15.2 | 1 |
| **4** | 7 | 0.0 | 1 | 2 | 0 | 0 | 0 | 16.1 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **979** | 3 | 1.3 | 0 | 6 | 1 | 1 | 1 | 10.1 | 1 |
| **980** | 6 | 1.3 | 0 | 2 | 2 | 2 | 0 | 10.4 | 0 |
| **981** | 9 | 2.0 | 0 | 6 | 3 | 4 | 0 | 9.8 | 0 |
| **982** | 7 | 0.0 | 0 | 6 | 0 | 3 | 1 | 12.6 | 1 |
| **983** | 1 | 0.0 | 1 | 5 | 3 | 1 | 1 | 16.6 | 1 |

| | features | coefficients |
|---|---|---|
| 7 | dist | 0.321296 |
| 6 | offhour | 0.278032 |
| 2 | passport | 0.189365 |
| 8 | salary | -0.083857 |
| 4 | children | -0.105872 |
| 3 | floor | -0.133584 |
| 0 | interest | -0.234305 |
| 5 | climate | -0.284478 |
| 1 | coffee | -0.449660 |

***b)*** Привести матрицу ошибок, рассмотреть различные метрики качества модели.

```
Confusion Matrix:
 [[582  87]
 [143 172]]
Accuracy:  0.766260162601626
Precision:  0.6640926640926641
Recall: 0.546031746031746
F1: 0.5993031358885018
Roc-auc score:
 0.7079934514912093
```

Итак, по матрице ошибок
Истинно-положительных ответов - 582
Ложно-положительных ответов - 87
Ложно отрицательных ответов - 143
Истинно отрицательных - 172

Accuracy — доля правильных ответов алгоритма:

$$Accuracy = \frac{TP + TN}{TP+TN+FP+FN} = 0.7662$$

Модель с вероятностью в 76,6% предскажет правильный результат.

$$precision = \frac{TP}{TP+FP} = 0.664$$

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными

Recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

$$Recall = \frac{TP}{TP+NF} = 0.54$$

$$F_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

F-мера является объединением Precision и Recal, если она стремится к единице, то оба эти параметра также стремятся к единице, если же мера стремится к нулю, то одно из значений стремится к нулю. В этой модели F-мера равна 0,599.

AUC-ROC — площадь под кривой ошибок (Receiver Operating Characteristic curve )

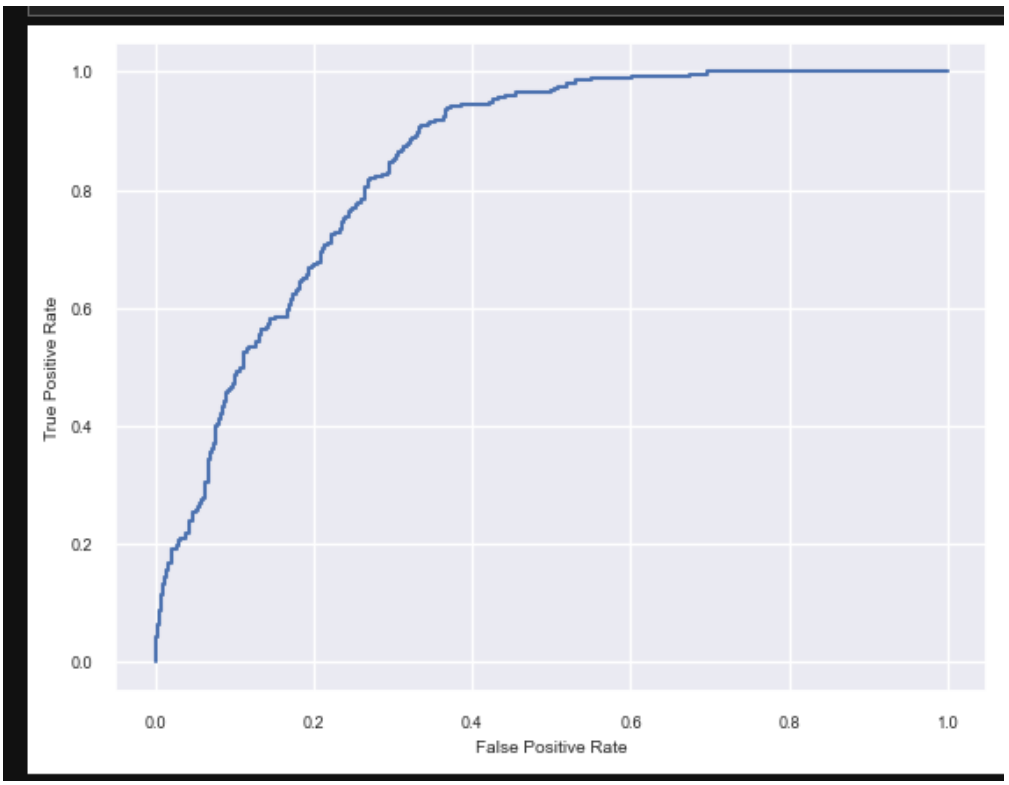
Данная кривая представляет из себя линию от (0, 0) до (1,1) в координатах True Positive Rate (TPR) и False Positive Rate (FPR):

$$TPR = \frac{TP}{TP+FN} \qquad FPR = \frac{FP}{FP+TN}$$

Площадь под кривой в данном случае показывает качество алгоритма (больше — лучше). У построенной модели показатель площади под графиком равен 0,707

(Наибольший из всех построенных до этого моделей).

***c)*** Построить ROC-кривую.


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate