

ДИСЦИПЛИНА «Технологии и методы программирования»

ИНСТИТУТ «Институт кибербезопасности и цифровых технологий»

КАФЕДРА КБ-2 «Информационно-аналитические системы кибербезопасности»

ВИД УЧЕБНОГО МАТЕРИАЛА Практическое занятие

ПРЕПОДАВАТЕЛЬ Новиков Евгений Иванович

СЕМЕСТР 5 семестр 2024-2025 учебный год

## **ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. ТЕХНОЛОГИИ РЕГРЕССИОННОГО АНАЛИЗА ДАННЫХ В PYTHON. ЧАСТЬ 1**

План практического занятия:

1. Корреляция между признаками.
2. Оценивание параметров модели.
3. Оценивание качества модели.

✓ ТЕХНОЛОГИИ РЕГРЕССИОННОГО АНАЛИЗА ДАННЫХ. ЧАСТЬ 1

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import warnings
5 warnings.filterwarnings('ignore')

1 np.random.seed(10)
2 Рост = np.random.normal(176, 11, 200)
3 Bec = (Рост*0.7-50) + 5 * np.random.randn(200)
4 Bec[:5], Рост[:5]
```

```
➔ (array([84.11890357, 84.72136747, 56.17665292, 73.93744017, 72.33191056]),
   array([190.64745155, 183.86806872, 159.00059679, 175.90777765,
         182.83469571]))
```

```
1 dfp = pd.DataFrame()
2 dfp['Рост'] = Рост
3 dfp['Bec'] = Bec
4 dfp.head()
```

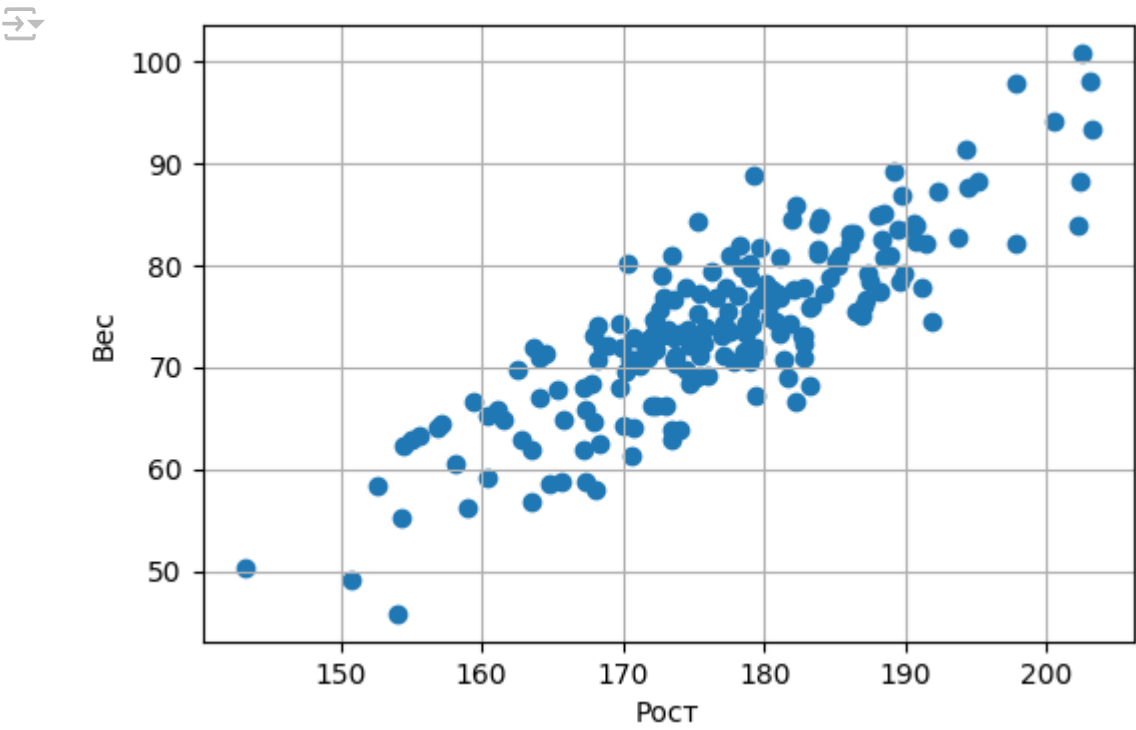
➔

	Рост	Вес
0	190.647452	84.118904
1	183.868069	84.721367
2	159.000597	56.176653
3	175.907778	73.937440
4	182.834696	72.331911

Далее:

☒ [Посмотреть рекомендованные графики](#)

```
1 plt.figure(figsize = (6, 4))
2 plt.scatter (dfp['Рост'], dfp['Bec'])
3 plt.xlabel('Рост')
4 plt.ylabel('Bec')
5 plt.grid()
```



✓ 1. Корреляция между признаками

```
1 from scipy.stats import pearsonr, spearmanr, kendalltau
2 print('Коэффициент корреляции Пирсона:', pearsonr(dfp['Вес'], dfp['Рост'])[0],
3       'p_уровень:', pearsonr(dfp['Вес'], dfp['Рост'])[1])
4 print('Коэффициент корреляции Спирмена:', spearmanr(dfp['Вес'], dfp['Рост'])[0],
5       'p_уровень:', spearmanr(dfp['Вес'], dfp['Рост'])[1])
```

➡ Коэффициент корреляции Пирсона: 0.8468719024245387 p\_уровень: 3.3270835644504402e-56  
Коэффициент корреляции Спирмена: 0.8160294007350185 p\_уровень: 5.052764388947029e-49

✓ 2. Оценивание параметров модели

```
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(dfp['Рост'], dfp['Вес'],
3                                                    test_size = 0.3,
4                                                    random_state = 42)
```

```
1 dfp_train = pd.DataFrame({'Рост' : X_train, 'Вес' : y_train}).reset_index(drop = True)
2 dfp_train.head()
```

➡

	Рост	Вес
0	170.335991	80.240026
1	176.985464	73.164502
2	169.957601	71.927586
3	165.384278	67.781340
4	178.145146	76.946711

⌄

Далее: [🔘 Посмотреть рекомендованные графики](#)

```
1 dfp_test = pd.DataFrame({'Рост' : X_test, 'Вес' : y_test}).reset_index(drop = True)
2 dfp_test.head()
```

➡

	Рост	Вес
0	186.100961	82.055368
1	180.896514	77.156620
2	173.059511	66.118076
3	186.367504	83.128378
4	194.223883	91.355371

⌄

Далее: [🔘 Посмотреть рекомендованные графики](#)

✓ Метод наименьших квадратов

✓ С использованием аналитических выражений

$$b_1 = \frac{\overline{x \cdot y} - \bar{y} \cdot \bar{x}}{\overline{x^2} - \bar{x}^2}$$
$$b_0 = \bar{y} - b_1 \cdot \bar{x}$$

```
1 b1_p = ((dfp_train['Рост'] * dfp_train['Вес']).mean() \
2         - dfp_train['Вес'].mean() * dfp_train['Рост'].mean()) \
3         / ((dfp_train['Рост']**2).mean() - (dfp_train['Рост'].mean())**2)
4 b1_p
```

➡ 0.7070887137532831

```
1 b0_p = dfp_train['Вес'].mean() - b1_p * dfp_train['Рост'].mean()
2 b0_p
```

 -51.19555975346388


```
1 dfp_train['Bec_мод'] = b0_p + b1_p * dfp_train['Рост']
2 dfp_train.head()
```





	Рост	Вес	Вес_мод	
0	170.335991	80.240026	69.247097	
1	176.985464	73.164502	73.948864	
2	169.957601	71.927586	68.979542	
3	165.384278	67.781340	65.745796	
4	178.145146	76.946711	74.768862	

Далее: [Посмотреть рекомендованные графики](#)

```
1 dfp_test['Bec_мод'] = b0_p + b1_p * dfp_test['Рост']
2 dfp_test.head()
```




	Рост	Вес	Вес_мод	
0	186.100961	82.055368	80.394329	
1	180.896514	77.156620	76.714323	
2	173.059511	66.118076	71.172867	
3	186.367504	83.128378	80.582799	
4	194.223883	91.355371	86.137956	

Далее: [Посмотреть рекомендованные графики](#)

▼ Sklearn


```
1 from sklearn.linear_model import LinearRegression
2 model_p = LinearRegression()
3 model_p.fit(dfp_train[['Рост']], dfp_train['Bec'])
```



▼ LinearRegression

LinearRegression()

```
1 b1_p = model_p.coef_
2 b0_p = model_p.intercept_
3 b1_p, b0_p
```

 (array([0.70708871]), -51.19555975346903)

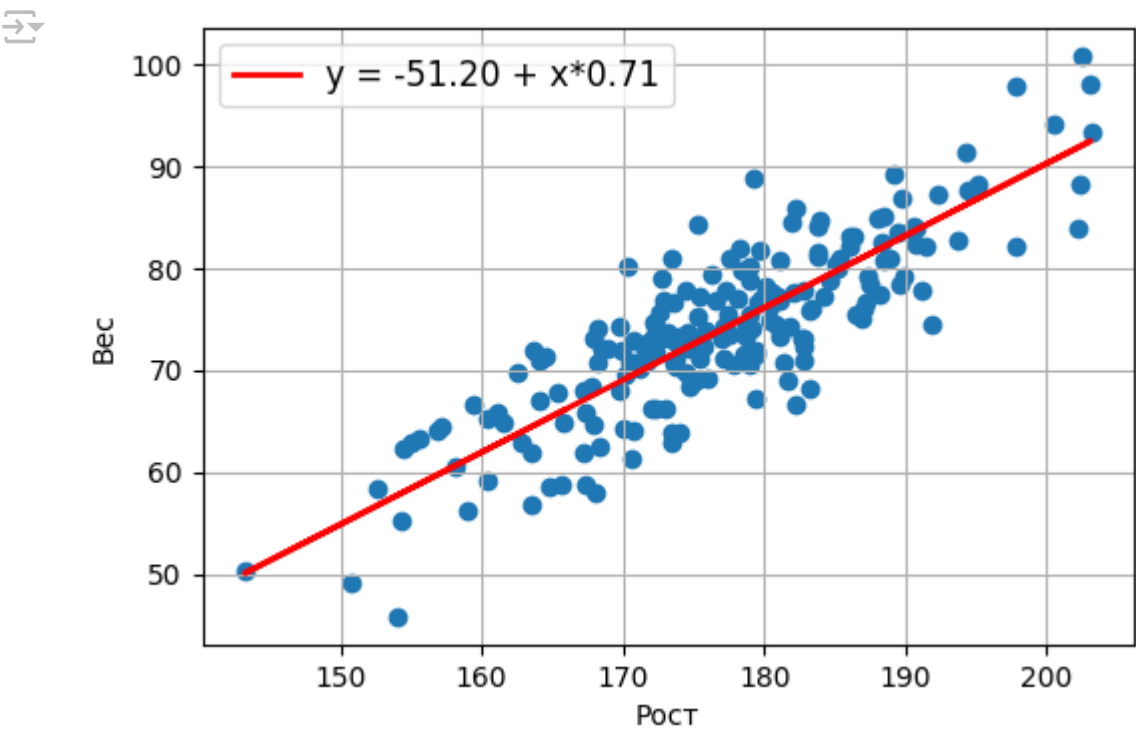
```
1 model_p.predict(dfp_train[['Рост']])[:5]
```

 array([69.24709692, 73.94886415, 68.97954162, 65.74579639, 74.76886244])

```
1 model_p.predict(dfp_test[['Рост']])[:5]
```

 array([80.39432915, 76.71432347, 71.17286723, 80.58279929, 86.13795576])

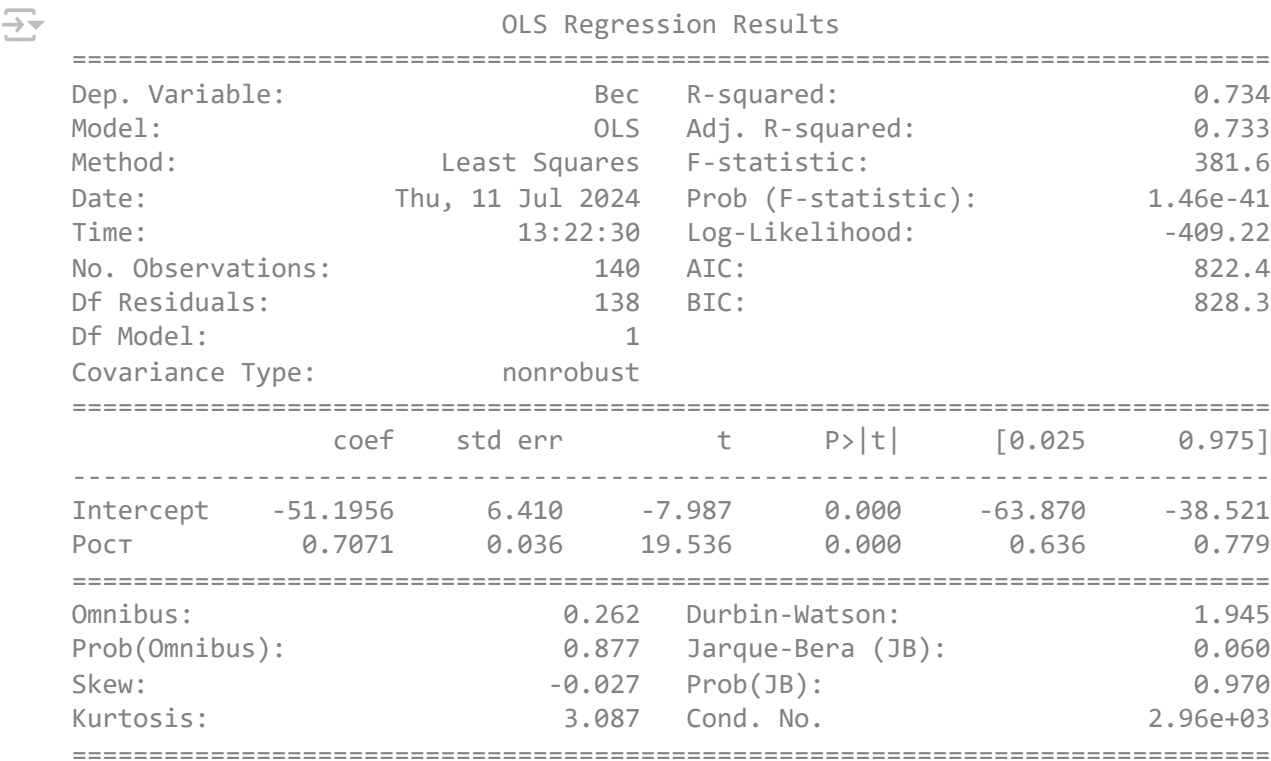
```
1 plt.figure(figsize = (6, 4))
2 plt.scatter (dfp['Рост'], dfp['Bec'])
3 plt.plot(dfp_train['Рост'], dfp_train['Bec_мод'],
4         linewidth = 2, c = 'r',
5         label = f'y = {b0_p:.2f} + x*{b1_p[0]:.2f}')
6 plt.xlabel('Рост')
7 plt.ylabel('Bec')
8 plt.legend(prop = {'size': 12})
9 plt.grid()
```



Statsmodels

```
1 from statsmodels.formula.api import ols
2 model = ols('Bec ~ Почт', data = dfp_train)
3 model = model.fit()
```

```
1 print(model.summary())
```



Notes:

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

[2] The condition number is large, 2.96e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Метод градиентного спуска

```
1 from sklearn.linear_model import SGDRegressor
2 SGDRegressor = SGDRegressor()
3 SGDRegressor = SGDRegressor.fit(dfp_train[['Почт']], dfp_train[['Bec']])
4 SGDRegressor.coef_, SGDRegressor.intercept_
```

(array([1.00060366e+11]), array([-5.89415077e+09]))

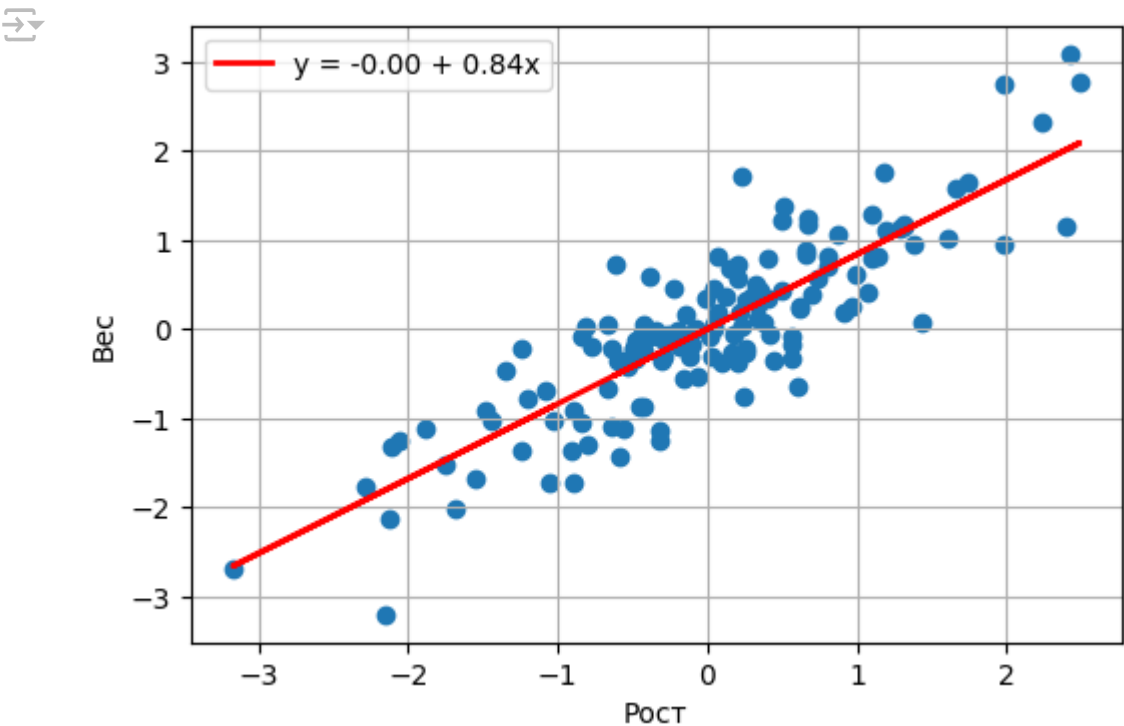
```
1 from sklearn.preprocessing import StandardScaler
2 scaler = StandardScaler()
3 x_st = scaler.fit_transform(dfp_train[['Почт']])
4 y_st = scaler.fit_transform(dfp_train[['Bec']])
```

```
1 SGDRegressor = SGDRegressor.fit(x_st, y_st)
2 SGDRegressor.coef_, SGDRegressor.intercept_
```

```
(array([0.83852923]), array([-0.00046092]))
```

```
1 b1_st = SGDRegressor.coef_[0]
2 b0_st = SGDRegressor.intercept_[0]
3 y_st_mod = b0_st + x_st * b1_st

1 plt.figure(figsize = (6, 4))
2 plt.scatter(x_st, y_st)
3 plt.plot(x_st, y_st_mod, linewidth = 2, c = 'r',
4          label = f'y = {b0_st:.2f} + {b1_st:.2f}x')
5 plt.xlabel('Рост')
6 plt.ylabel('Bec')
7 plt.legend(prop = {'size': 10})
8 plt.grid()
```



$$b_1 = b_1^* \cdot \frac{s_y}{s_x}$$

$$b_0 = \bar{y} - b_1 \cdot \bar{x}$$

```
1 mx = dfp_train['Рост'].mean()
2 my = dfp_train['Bec'].mean()
3 sx = dfp_train['Рост'].std()
4 sy = dfp_train['Bec'].std()
5 v1 = SGDRegressor.coef_ * (sy / sx)
6 v0 = my - v1 * mx
7 v1, v0
```

```
(array([0.69185495]), array([-48.50240988]))
```

3. Оценивание качества модели


```
1 from sklearn.metrics import mean_squared_error
2 print(mean_squared_error(dfp_train['Bec'],
3                           dfp_train['Bec_мод']))
4 print(mean_squared_error(dfp_test['Bec'],
5                           dfp_test['Bec_мод']))
```

```
20.248915359016657
24.150302056267968
```

```
1 from sklearn.metrics import r2_score
2 print(r2_score(dfp_train['Bec'], dfp_train['Bec_мод']))
3 print(model_p.score(dfp_train[['Рост']], dfp_train['Bec']))
```

```
0.7344362665024011
0.7344362665024011
```

```
1 print(r2_score(dfp_test['Bec'], dfp_test['Bec_мод']))
2 print(model_p.score(dfp_test[['Поcт']], dfp_test['Bec']))
```

 0.6725128806818603  
0.672512880681854

```
1 def r_squared(x, y, y_mod):
2     r2 = 1 - ((y - y_mod)**2).sum() / ((y - y.mean())**2).sum()
3     n, m = x.shape
4     r2_adj = 1 - (1 - r2)*((n-1)/(n-m-1))
5     return r2, r2_adj
```

```
1 r_squared(dfp_train[['Поcт']], dfp_train['Bec'], dfp_train['Bec_мод'])
```

 (0.7344362665024011, 0.7325118916219837)

```
1 r_squared(dfp_test[['Поcт']], dfp_test['Bec'], dfp_test['Bec_мод'])
```

 (0.6725128806818603, 0.6668665510384442)