Лабораторная работа №1 - построение и анализ регрессионных зависимостей

Датасет - Ирисы Фишера

1. Разведочный анализ данных

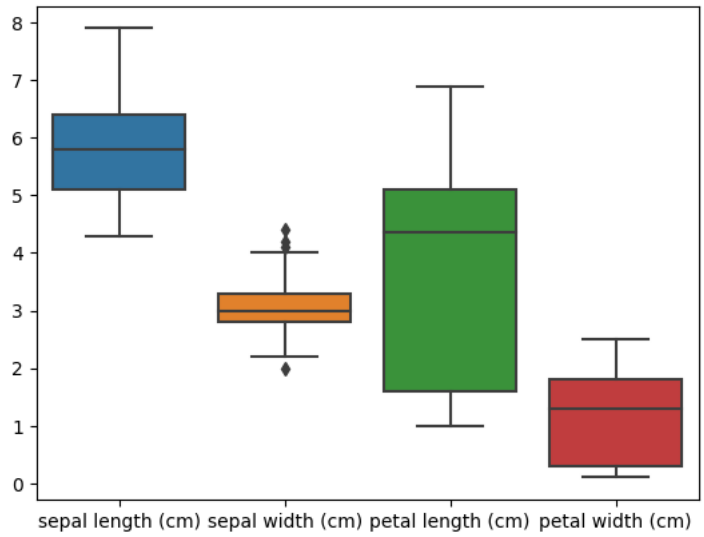
Подготовка датасета:

[ ] ↳ *Скрыто 2 ячейки.*

Диаграммы Тьюки для оцениваемых признаков:

```
sns.boxplot(data=iris_pd.drop('target', axis = 1))
```

```
<Axes: >
```



Диапазон изменения данных

```
print('Параметр\tРазмах')
print(np.ptp(iris_pd.drop('target', axis = 1), axis=0))
```

```
Параметр        Размах
sepal length (cm)    3.6
sepal width (cm)     2.4
petal length (cm)    5.9
petal width (cm)     2.4
dtype: float64
```

Для данного датасета:

1. Самый большой разброс имеет длина лепестка (petal length)
2. Ширина чашелистика (sepal width) имеет некоторое количество аномальных измерений

Рассчет матрицы парных корреляций:

[ ] ↳ *Скрыто 3 ячейки.*

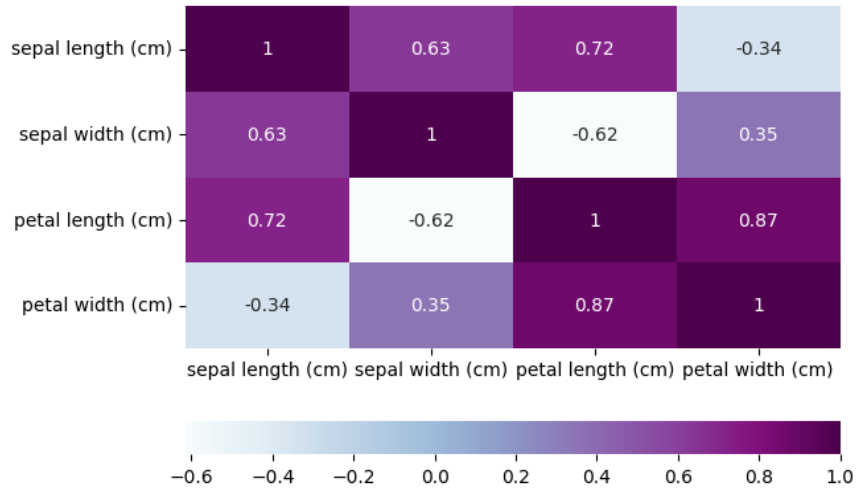
Рассчет матрицы частных корреляций:

```
partial_corr = iris_pd.drop('target', axis = 1).pcorr() #матрица частных корреляций
partial_corr
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
sepal length (cm)	1.000000	0.628571	0.719066	-0.339617
sepal width (cm)	0.628571	1.000000	-0.615292	0.352626
petal length				

```
sns.heatmap(
    partial_corr,
    cbar_kws = {'orientation' : 'horizontal'},
    cmap="BuPu",
    annot=True,
)
```

```
<Axes: >
```



Оценки корреляционной связи:

- Силная связь между длиной и шириной лепестка (petal length и petal width) = 0.87
- Сильная связь между длиной лепестка (petal lenght) и длиной чашелистика (sepal length) = 0.72
- Умеренная связь между шириной чашелистика (sepal width) и длиной чашелистика (sepal length) = 0.63
- Умеренная обратная связь между шириной чашелистика (sepal width) и длиной лепестка (petal length) = -0.62
- Слабая связь между шириной лепестка (petal width) и шириной чашелистика (sepal width) = 0.35
- Слабая обратная связь между шириной лепестка (petal width) и длиной чашелистика (sepal length) = -0.34

Для некотроых признаков связь остается достаточно сильной, даже не смотря на отсутствие влияния других переменных.

Критерий Колмогорова-Смирнова

```
for feature in iris_pd.drop('target', axis = 1):
    print('KS_test for:', feature)
    print(
        sp.stats.kstest(
            iris_pd[feature],
            'norm',
            args=(iris_pd[feature].mean(), iris_pd[feature].std(ddof = 1))
        ),
        '\n')
```

```
KS_test for: sepal length (cm)
KstestResult(statistic=0.08865361377316228, pvalue=0.17813737848592026, statistic_location=5.1, statistic_sign=1)

KS_test for: sepal width (cm)
KstestResult(statistic=0.10565879047721255, pvalue=0.06521857239192497, statistic_location=3.0, statistic_sign=1)

KS_test for: petal length (cm)
KstestResult(statistic=0.19815409613999851, pvalue=1.2284783647761751e-05, statistic_location=1.7, statistic_sign=1)

KS_test for: petal width (cm)
KstestResult(statistic=0.17283424907904044, pvalue=0.00021788859386874477, statistic_location=0.4, statistic_sign=1)
```

Критерий Колмогорова-Смирнова проверяет, подчиняется ли выборка нормальному закону распределения. Если p-значение выше уровня значимости = 0.05, то гипотеза о нормальности распределения выборки не отвергается. Таким образом, гипотеза о

нормальности распределения выборок:

- sepal length - не отвергается
- sepal width - не отвергается
- petal length - отвергается
- sepal width - отвергается

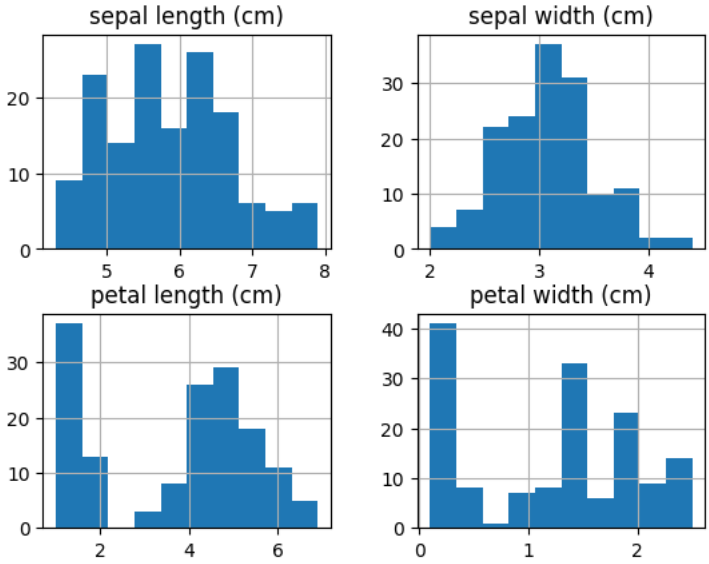
```
print(sp.stats.kstest(iris_pd['sepal length (cm)'], iris_pd['sepal width (cm)']))
print(sp.stats.kstest(iris_pd['sepal length (cm)'], iris_pd['petal length (cm)']))
print(sp.stats.kstest(iris_pd['sepal length (cm)'], iris_pd['petal width (cm)']))
print(sp.stats.kstest(iris_pd['sepal width (cm)'], iris_pd['petal length (cm)']))
print(sp.stats.kstest(iris_pd['sepal length (cm)'], iris_pd['petal width (cm)']))
print(sp.stats.kstest(iris_pd['petal length (cm)'], iris_pd['petal width (cm)']))
```

```
KstestResult(statistic=0.9933333333333333, pvalue=6.399337692587972e-87, statistic_location=4.2, statistic_sign=-1)
KstestResult(statistic=0.56, pvalue=5.290396633242105e-22, statistic_location=4.7, statistic_sign=-1)
KstestResult(statistic=1.0, pvalue=2.133112564195991e-89, statistic_location=2.5, statistic_sign=-1)
KstestResult(statistic=0.5733333333333334, pvalue=4.092283006961527e-23, statistic_location=3.8, statistic_sign=1)
KstestResult(statistic=1.0, pvalue=2.133112564195991e-89, statistic_location=2.5, statistic_sign=-1)
KstestResult(statistic=0.6666666666666666, pvalue=6.639808432803654e-32, statistic_location=2.5, statistic_sign=-1)
```

Проверка на однородность выборок показала, что все выборки взяты из разных распределений, так как все значения p-value < 0.05

```
iris_pd.drop('target', axis = 1).hist()
```

```
array([[<Axes: title={'center': 'sepal length (cm)'}>,
        <Axes: title={'center': 'sepal width (cm)'}>],
       [<Axes: title={'center': 'petal length (cm)'}>,
        <Axes: title={'center': 'petal width (cm)'}>]], dtype=object)
```



По построенным гистограммам видно, что выборки sepal length и sepal length подчиняются нормальному закону, а выборки petal length и petal length не подчиняются нормальному закону.

2. Построение регрессии

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error

X = iris_pd.drop(['petal length (cm)', 'target'], axis = 1)
y = iris_pd['petal length (cm)']
```

Возможные входные переменные:

- sepal length
- sepal width
- petal width

Выходные данные:

- petal length

Парная регрессия

```
for feature in X:
    regressor = LinearRegression().fit(np.array(X[feature]).reshape((-1,1)), y)

    y_pred = regressor.predict(np.array(X[feature]).reshape((-1,1)))

    score = r2_score(y, y_pred)
    print('Pair regression for', feature)
    print('koef determ =', score, '\n')
```

```
 Pair regression for sepal length (cm)
 koef determ = 0.759954645772515

 Pair regression for sepal width (cm)
 koef determ = 0.1835609229987637

 Pair regression for petal width (cm)
 koef determ = 0.9271098389904927
```

Самый большой коэффициент детерминации у petal width (determ = 0.927). Эти данные будут входными

```
regressor = LinearRegression().fit(np.array(X['petal width (cm)']).reshape((-1,1)), y)
y_pred_pair = regressor.predict(np.array(X['petal width (cm)']).reshape((-1,1)))
```

Множественный коэффициент корреляции

```
def Ry_score(X, y):
    R_det = np.linalg.det(pd.concat([X, y], axis = 1).corr())
    Ry = np.linalg.det(X.corr())

    return np.sqrt(1 - (R_det / Ry))

mnoz_corr = Ry_score(X, y)
```

Скорректированный коэффициент детерминации

```
def R2_corr(y_true, y_pred, K):
    S2_ost = (1 / (y_true.shape[0] - K)) * ((y_pred - y_true)**2).sum()
    S2_obsch = (1 / (y_true.shape[0] - 1)) * ((y_true - y_true.mean())**2).sum()

    return 1 - (S2_ost / S2_obsch)

det_corr_score = R2_corr(y, y_pred_pair, K=3)
```

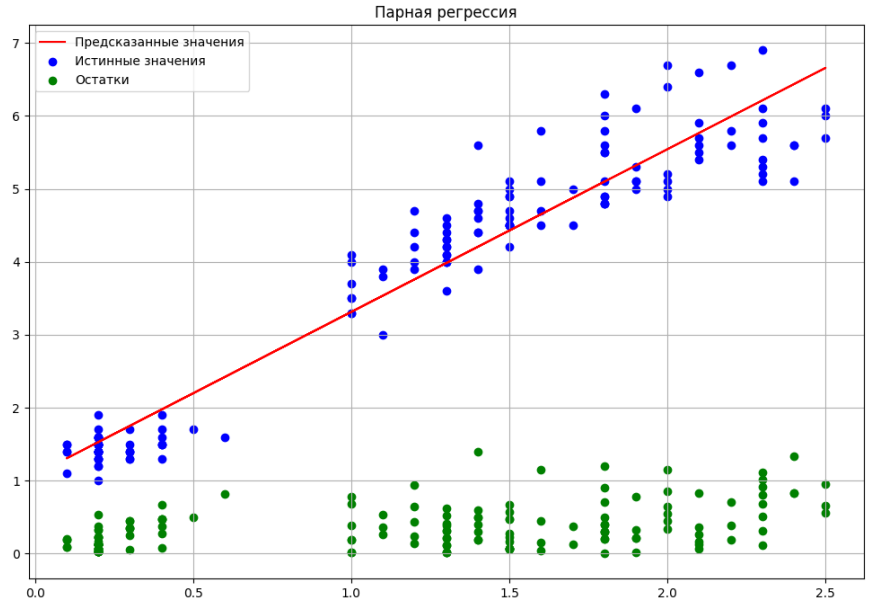
Вывод результатов

```
det_score = r2_score(y, y_pred_pair)
print('Коэффициент детерминации = ', round(det_score, 3))
print('Множественный коэффициент корреляции = ', round(mnoz_corr, 3))
print('Скорректированный коэффициент детерминации = ', round(det_corr_score, 3))
print('Стандартная ошибка = ', round(mean_absolute_error(y, y_pred_pair), 3))
```

```
 Коэффициент детерминации =  0.927
 Множественный коэффициент корреляции =  0.984
 Скорректированный коэффициент детерминации =  0.926
 Стандартная ошибка =  0.366
```

```
fig = plt.figure(figsize = (12, 8))
ax1 = fig.add_subplot(111)

ax1.grid()
ax1.plot(
    X['petal width (cm)'], y_pred_pair, label = 'Предсказанные значения', color = 'red'
)
ax1.scatter(
    X['petal width (cm)'], y, label = 'Истинные значения', color = 'blue', marker = 'o'
)
ax1.scatter(
    X['petal width (cm)'],
    abs(y - y_pred_pair),
    label = 'Остатки',
    color = 'green',
    marker = 'o'
)

plt.legend(loc='upper left')
plt.title('Парная регрессия')
```

```
Text(0.5, 1.0, 'Парная регрессия')
```



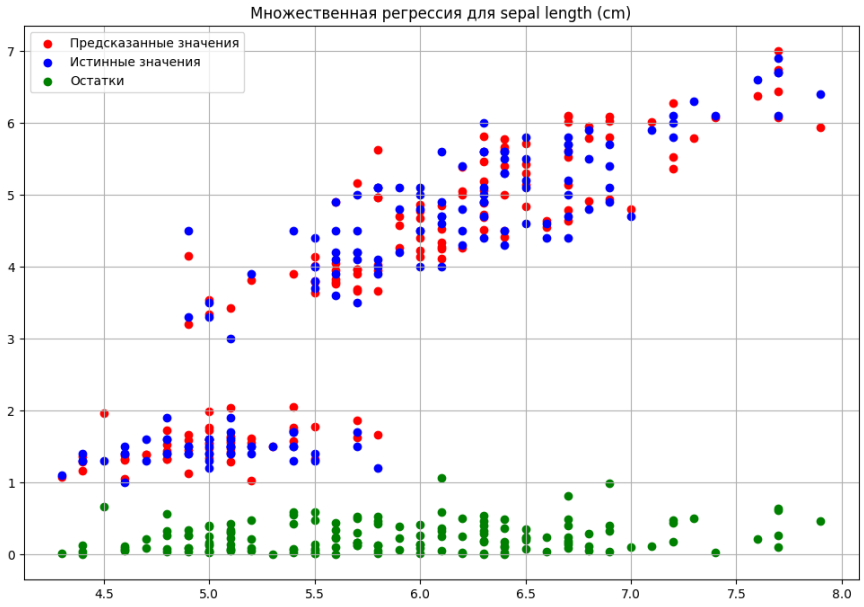
Множественная регрессия

```
regressor_mnoz = LinearRegression()
regressor_mnoz.fit(X, y)
y_pred = regressor_mnoz.predict(X)

det_score = r2_score(y, y_pred)
print('Коэффициент детерминации = ', round(det_score, 3))

mnoz_corr = Ry_score(X, y)
print('Множественный коэффициент корреляции = ', round(mnoz_corr, 3))

det_corr_score = R2_corr(y, y_pred, K=3)
print('Скорректированный коэффициент детерминации = ', round(det_corr_score, 3))

print('Стандартная ошибка = ', round(mean_absolute_error(y, y_pred), 3))
```

```
Коэффициент детерминации =  0.968
Множественный коэффициент корреляции =  0.984
Скорректированный коэффициент детерминации =  0.968
Стандартная ошибка =  0.24
```

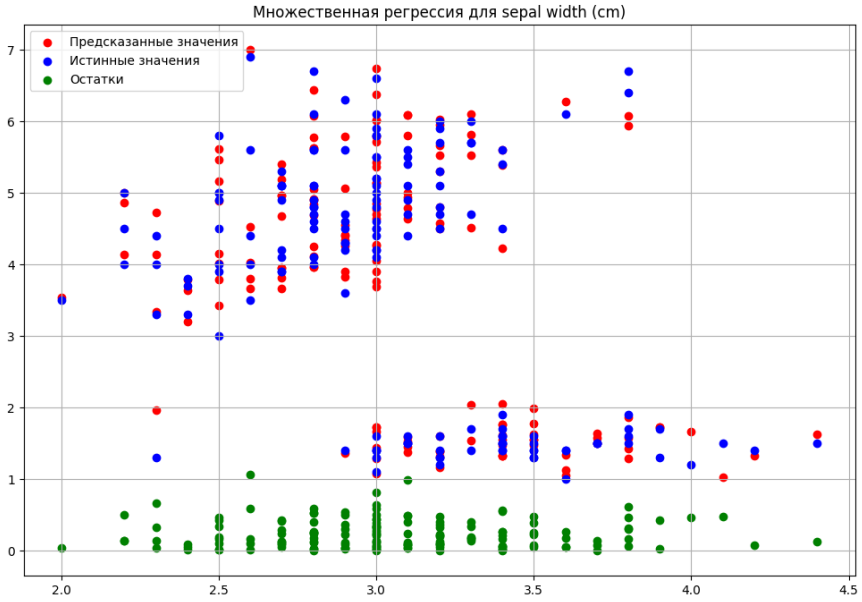
```
fig = plt.figure(figsize = (12, 8))
ax1 = fig.add_subplot(111)

ax1.grid()
ax1.scatter(X['sepal length (cm)'], y_pred, label='Предсказанные значения', color ='red')
ax1.scatter(X['sepal length (cm)'], y, marker="o", label='Истинные значения', color = 'blue')
ax1.scatter(X['sepal length (cm)'], abs(y - y_pred), label = 'Остатки', color = 'green', marker = 'o')

plt.legend(loc='upper left')
plt.title('Множественная регрессия для sepal length (cm)')
```

```
Text(0.5, 1.0, 'Множественная регрессия для sepal length (cm)')
```



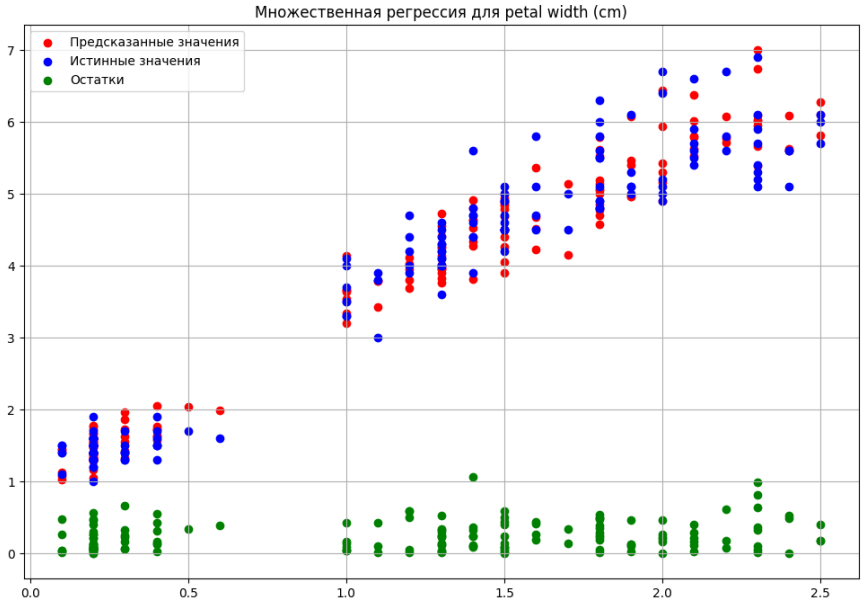

```
fig = plt.figure(figsize = (12, 8))
ax1 = fig.add_subplot(111)

ax1.grid()
ax1.scatter(X['sepal width (cm)'], y_pred, label='Предсказанные значения', color ='red')
ax1.scatter(X['sepal width (cm)'], y, marker="o", label='Истинные значения', color = 'blue')
ax1.scatter(X['sepal width (cm)'], abs(y - y_pred), label = 'Остатки', color = 'green', marker = 'o')

plt.legend(loc='upper left')
plt.title('Множественная регрессия для sepal width (cm)')
```

```
Text(0.5, 1.0, 'Множественная регрессия для sepal width (cm)')
```




```
fig = plt.figure(figsize = (12, 8))
ax1 = fig.add_subplot(111)

ax1.grid()
ax1.scatter(X['petal width (cm)'], y_pred, label='Предсказанные значения', color ='red')
ax1.scatter(X['petal width (cm)'], y, marker="o", label='Истинные значения', color = 'blue')
ax1.scatter(X['petal width (cm)'], abs(y - y_pred), label = 'Остатки', color = 'green', marker = 'o')

plt.legend(loc='upper left')
plt.title('Множественная регрессия для petal width (cm)')
```

```
Text(0.5, 1.0, 'Множественная регрессия для petal width (cm)')
```



Исходя из множетсвенного коэффициента корреляции, коэффициента детерминации и графиков остатков, можно сделать вывод, что множетсвенная регрессия - наиболее адекватная модель для данной задачи.

3. Критерий Дурбина-Уотсона

```
def DW_statistic(y_true, y_pred):
    eps = y_true - y_pred
    eps_sum = 0

    for i in range(1, len(eps)):
        eps_sum += ((eps[i] - eps[i-1]) ** 2)

    Q_ost = (eps ** 2).sum()

    return eps_sum / Q_ost

print(DW_statistic(y, y_pred))
```

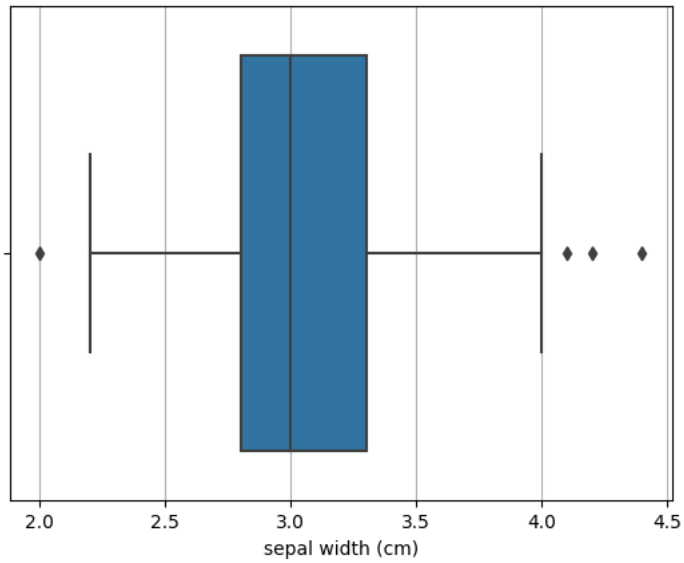
```
 1.782966528592163
```

Критерий Дурбина-Уотсона близок к 2. Это означает, что автокорреляция остатков отсутствует, они распределены случайно.

4. Анализ выбросов

```
plt.grid()
sns.boxplot(x = X['sepal width (cm)'])
```

```
<Axes: xlabel='sepal width (cm)'>
```



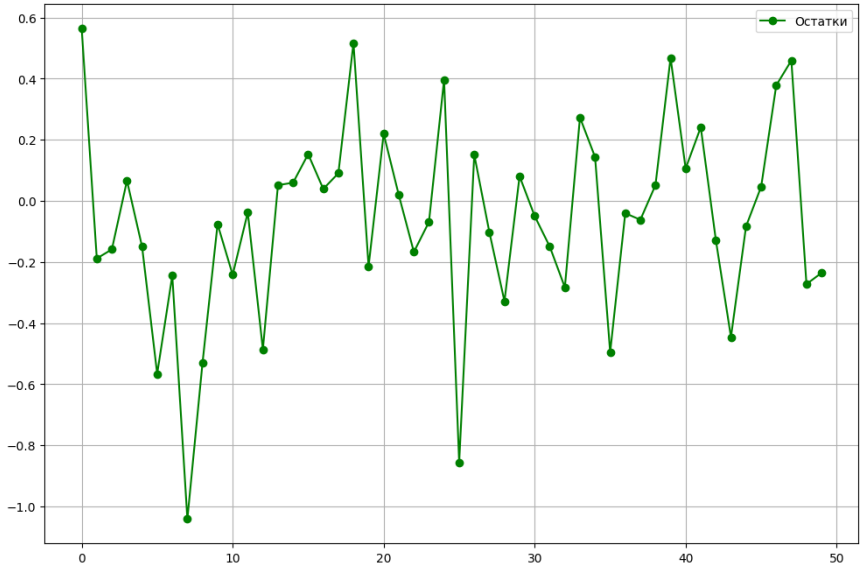
5. Предсказание зависимой переменной

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.33, random_state=42
)

X_train.shape, y_train.shape, X_test.shape, y_test.shape

regressor = LinearRegression().fit(X_train, y_train)
y_pred = regressor.predict(X_test)

det_score = r2_score(y_test, y_pred)
print('Коэффициент детерминации = ', round(det_score, 3))

mse = mean_squared_error(y_test, y_pred)
print('Средний квадрат ошибки = ', round(mse, 3))

det_corr_score = R2_corr(y_test, y_pred, K=3)
print('Скорректированный коэффициент детерминации = ', round(det_corr_score, 3))

print('Стандартная ошибка = ', round(mean_absolute_error(y_test, y_pred), 3))
```

```
Коэффициент детерминации =  0.967
Средний квадрат ошибки =  0.108
Скорректированный коэффициент детерминации =  0.966
Стандартная ошибка =  0.246
```

```
y_test = y_test.reset_index().drop('index', axis = 1)
y_test = y_test['petal length (cm)']
fig = plt.figure(figsize = (12, 8))
plt.grid()
plt.plot(y_test - y_pred, label = 'Остатки', marker = 'o', color = 'green')

plt.legend()
```

```
<matplotlib.legend.Legend at 0x7cca820ef8e0>
```



Мы провели предсказание зависмой переменной, разделив выборку на тестовую и обучающую. Еще раз вычислили уравнение множетсвенной регрессии на обучающей выборке, после чего рассчитали оценки необходимых метрик.

Набор данных "Рост-Вес-Возраст-Позиция"

1. Разведочный анализ данных

Подготовка датасета

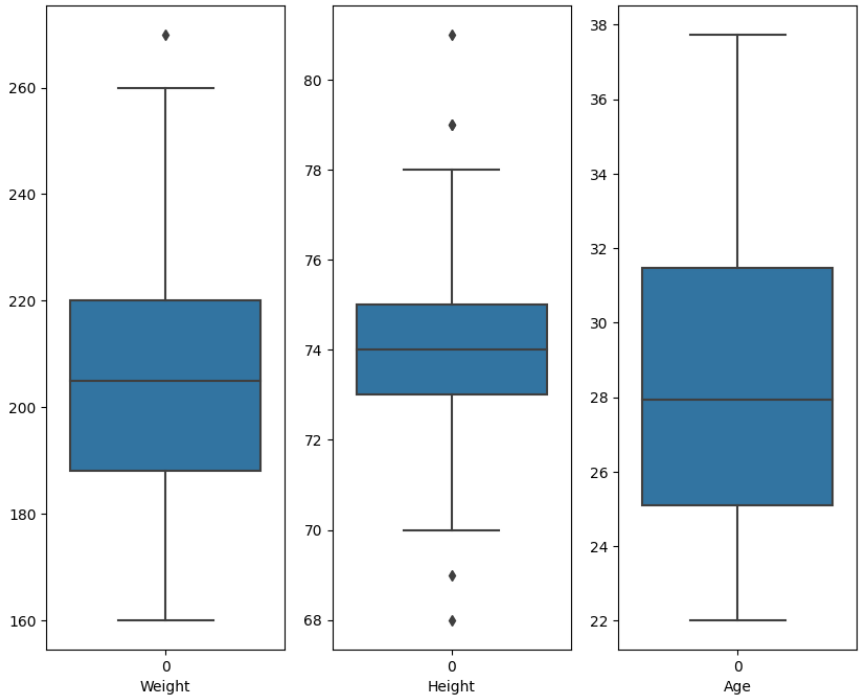
```
baseball_df = pd.read_csv('baseball.csv')
baseball_df.columns = ['Position', 'Height', 'Weight', 'Age']
baseball_df.head()
```

	Position	Height	Weight	Age
0	Catcher	74	180	22.99
1	Catcher	74	215	34.69
2	Catcher	72	210	30.78
3	Baseman	72	210	35.43
4	Baseman	73	188	35.71

Диаграммы Тьюки для оцениваемых признаков:

```
plt.figure(figsize=(10, 8))

ax1 = plt.subplot(1, 3, 1)
ax1.set_xlabel("Weight")
sns.boxplot(baseball_df["Weight"])

ax2 = plt.subplot(1, 3, 2)
ax2.set_xlabel("Height")
sns.boxplot(baseball_df["Height"])

ax3 = plt.subplot(1, 3, 3)
ax3.set_xlabel("Age")
sns.boxplot(baseball_df["Age"])
```

```
<Axes: xlabel='Age'>
```



Диапазон изменения данных

```
print('Параметр\tРазмах')
print(np.ptp(baseball_df.drop('Position', axis=1), axis=0))
```

```
Параметр        Размах
Height     13.00
Weight    110.00
Age        15.72
dtype: float64
```

Для данного датасета:

1. Самый большой разброс имеет вес (Weight)
2. Рост (Height) имеет некоторое количество аномальных измерени

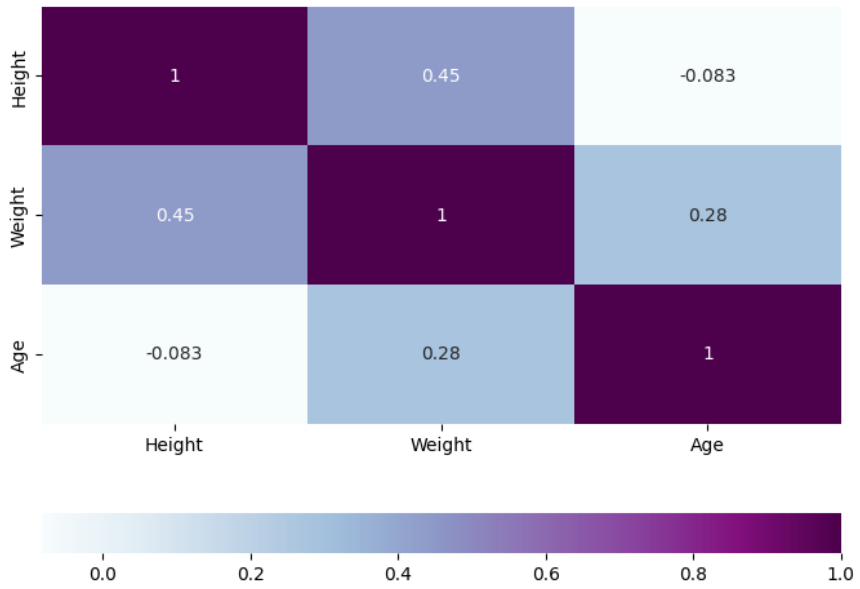
Рассчет матрицы парных корреляций

```
pearson_corr_b = baseball_df.drop(['Position'], axis = 1).corr()
pearson_corr_b
```

	Height	Weight	Age
Height	1.000000	0.445074	-0.082647
Weight	0.445074	1.000000	0.277046
Age	-0.082647	0.277046	1.000000

```
plt.figure(figsize=(8, 6))
sns.heatmap(
    pearson_corr_b,
    cbar_kws = {'orientation' : 'horizontal'},
    cmap="BuPu",
    annot=True,
)
```

```
<Axes: >
```



Оценки корреляционной связи:

- Слабая связь между весом (Weight) и ростом (Height) = 0.45
- Очень слабая связь между возрастом (Age) и весом (Weight) = 0.28
- Очень слабая связь между возрастом (Age) и весом (Height) = -0.08

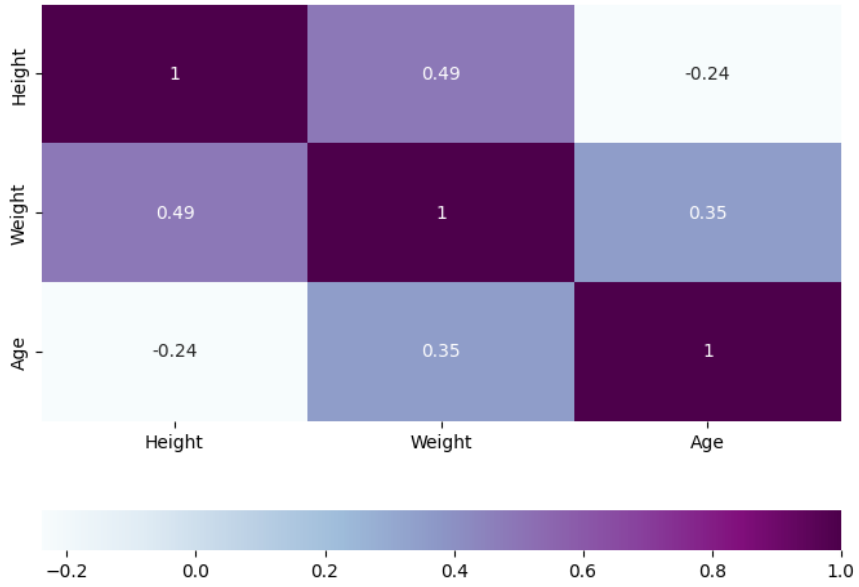
Рассчет матрицы частных корреляций

```
partial_corr_b = baseball_df.drop(['Position'], axis = 1).pcorr()
partial_corr_b
```

	Height	Weight	Age
Height	1.000000	0.488708	-0.239358
Weight	0.488708	1.000000	0.351658
Age	-0.239358	0.351658	1.000000

```
plt.figure(figsize=(8, 6))
sns.heatmap(
    partial_corr_b,
    cbar_kws = {'orientation' : 'horizontal'},
    cmap="BuPu",
    annot=True,
)
```

```
<Axes: >
```



Оценки корреляционной связи:

- Слабая связь между весом (Weight) и ростом (Height) = 0.49
- Слабая связь между возрастом (Age) и весом (Weight) = 0.35
- Слабая связь между возрастом (Age) и весом (Height) = -0.24

Для некотроых признаков связь остается достаточно сильной, даже не смотря на отсутствие влияния других переменных.

Критерий Колмогорова-Смирнова

```
for feature in baseball_df.drop(["Position"], axis = 1):
    print('KS_test for:', feature)
    print(
        sp.stats.kstest(
            baseball_df[feature],
            'norm',
            args=(baseball_df[feature].mean(), baseball_df[feature].std(ddof = 1))
        ),
        '\n'
    )
```

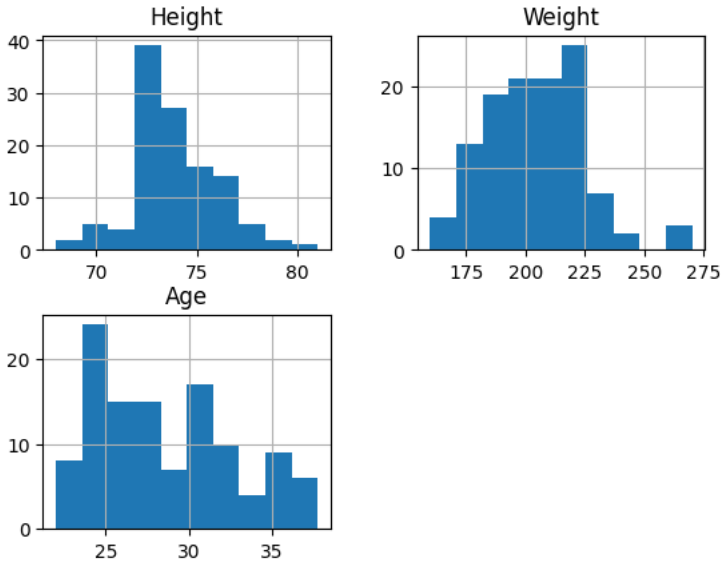
```
KS_test for: Height
KstestResult(statistic=0.1502796452577414, pvalue=0.009870680172291554, statistic_location=74, statistic_sign=1)

KS_test for: Weight
KstestResult(statistic=0.08115950732864483, pvalue=0.4130657079524148, statistic_location=180, statistic_sign=-1)

KS_test for: Age
KstestResult(statistic=0.09945220794562915, pvalue=0.19212487337202364, statistic_location=26.29, statistic_sign=1)
```

Если p-значение выше уровня значимости = 0.05, то гипотеза о нормальности распределения выборки не отвергается. Таким образом, гипотеза о нормальности распределения выборок:

- Height - отвергается
- Weight - не отвергается
- Age - не отвергается

```
baseball_df.hist()
```

```
array([[<Axes: title={'center': 'Height'}>,
        <Axes: title={'center': 'Weight'}>],
       [<Axes: title={'center': 'Age'}>, <Axes: >]], dtype=object)
```



По построенным гистограммам видно, что выборки Weight и Height подчиняются нормальному закону, а выборка Age не подчиняется нормальному закону.

2. Построение регрессии

Парная регрессия

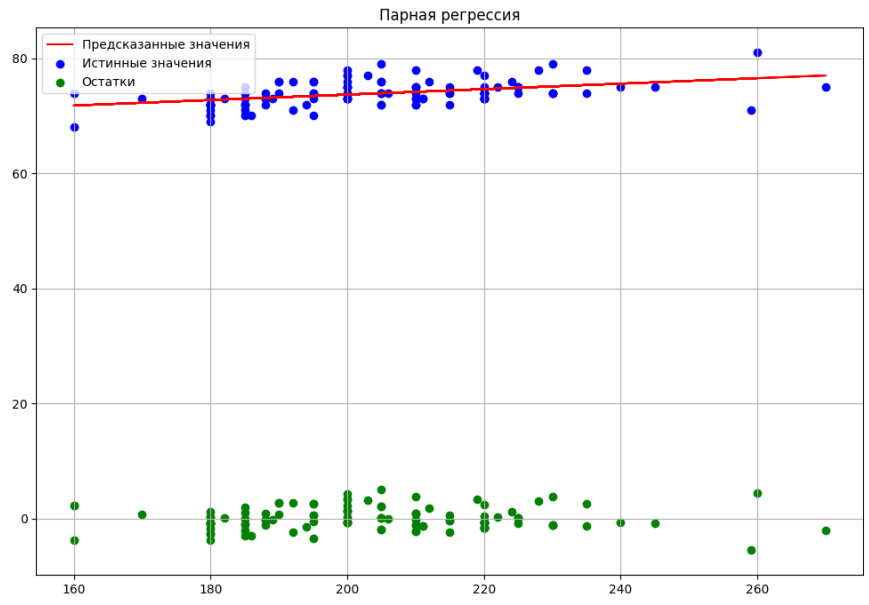
```
X = baseball_df.drop(['Height', "Position"], axis = 1)
y = baseball_df['Height']

for feature in X:
    regressor = LinearRegression().fit(np.array(X[feature]).reshape((-1,1)), y)

    y_pred_b = regressor.predict(np.array(X[feature]).reshape((-1,1)))

    score = r2_score(y, y_pred_b)
    print('Pair regression for', feature)
    print('R2 score =', score, '\n')
```

```
 Pair regression for Weight
 R2 score = 0.19809105652664694

 Pair regression for Age
 R2 score = 0.006830601377061041
```

```
regressor = LinearRegression().fit(np.array(X['Weight']).reshape((-1,1)), y)
y_pred_pair = regressor.predict(np.array(X['Weight']).reshape((-1,1)))

#Множественный коэффициент корреляции
def Ry_score(X, y):
    R_det = np.linalg.det(pd.concat([X, y], axis = 1).corr())
    Ry = np.linalg.det(X.corr())

    return np.sqrt(1 - (R_det / Ry))

#Скорректированный коэффициент детерминации
def R2_corr(y_true, y_pred, K):
    S2_ost = (1 / (y_true.shape[0] - K)) * ((y_pred - y_true)**2).sum()
    S2_obsch = (1 / (y_true.shape[0] - 1)) * ((y_true - y_true.mean())**2).sum()

    return 1 - (S2_ost / S2_obsch)

det_score = r2_score(y, y_pred_pair)
print('Коэффициент детерминации = ', round(det_score, 2))

mnoz_corr = Ry_score(X, y)
print('Множественный коэффициент корреляции = ', round(mnoz_corr, 2))

det_corr_score = R2_corr(y, y_pred_pair, K=2)
print('Скорректированный коэффициент детерминации = ', round(det_corr_score, 2))

print('Стандартная ошибка = ', round(mean_absolute_error(y, y_pred_pair), 2))
```

```
Коэффициент детерминации =  0.2
Множественный коэффициент корреляции =  0.49
Скорректированный коэффициент детерминации =  0.19
Стандартная ошибка =  1.51
```

```
fig = plt.figure(figsize = (12, 8))
ax1 = fig.add_subplot(111)

ax1.grid()
ax1.plot(X['Weight'], y_pred_pair, label='Предсказанные значения', color = 'red')
ax1.scatter(X['Weight'], y, marker="o", label='Истинные значения', color = 'blue' )
ax1.scatter(X['Weight'], y - y_pred_pair, marker="o", label='Остатки', color = 'green')

plt.legend(loc='upper left')
plt.title('Парная регрессия')

plt.show()
```



```
ost = y - y_pred_pair
```

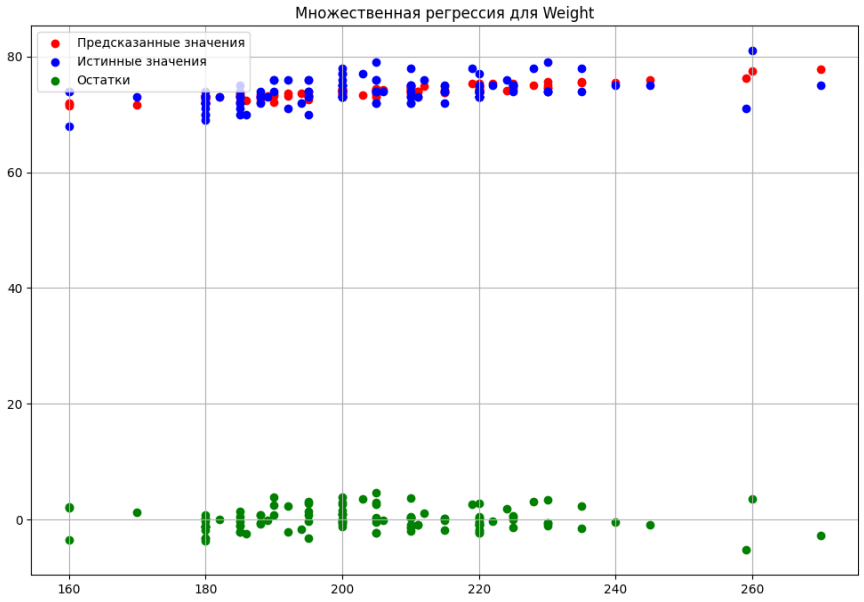
Множественная регрессия

```
regressor_mnoz = LinearRegression()
regressor_mnoz.fit(X, y)

y_pred = regressor_mnoz.predict(X)

det_score = r2_score(y, y_pred)
print('Коэффициент детерминации = ', round(det_score, 2))

mnoz_corr = Ry_score(X, y)
print('Множественный коэффициент корреляции = ', round(mnoz_corr, 2))

det_corr_score = R2_corr(y, y_pred, K=2)
print('Скорректированный коэффициент детерминации = ', round(det_corr_score, 2))

print('Стандартная ошибка = ', round(mean_absolute_error(y, y_pred), 2))
```

```
Коэффициент детерминации =  0.24
Множественный коэффициент корреляции =  0.49
Скорректированный коэффициент детерминации =  0.24
Стандартная ошибка =  1.46
```

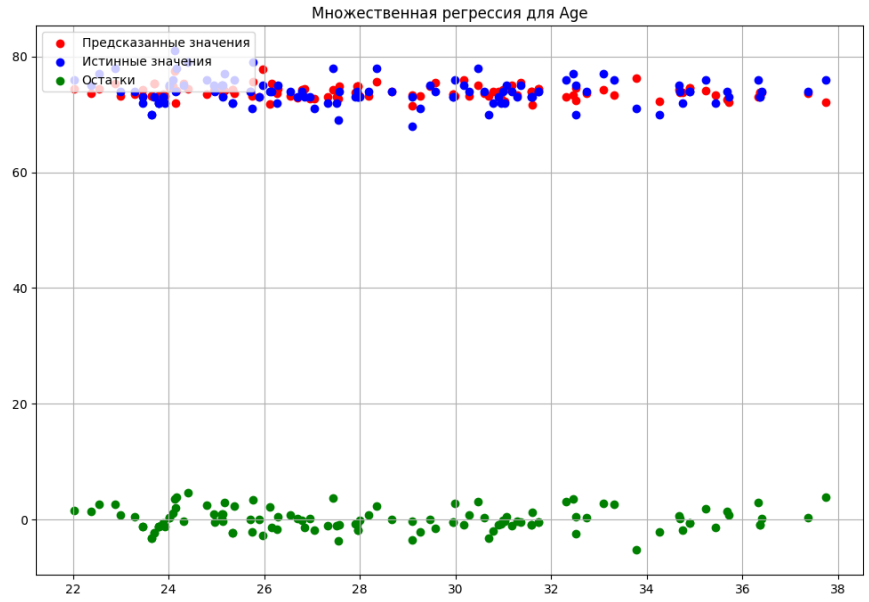
```
fig = plt.figure(figsize = (12, 8))
ax1 = fig.add_subplot(111)

ax1.grid()
ax1.scatter(X['Weight'], y_pred, label='Предсказанные значения', color = 'red')
ax1.scatter(X['Weight'], y, marker="o", label='Истинные значения', color = 'blue')
ax1.scatter(X['Weight'], y - y_pred, marker="o", label='Остатки', color = 'green')

plt.legend(loc='upper left')
plt.title('Множественная регрессия для Weight')
```

```
Text(0.5, 1.0, 'Множественная регрессия для Weight')
```



```
fig = plt.figure(figsize = (12, 8))
ax1 = fig.add_subplot(111)

ax1.grid()
ax1.scatter(X['Age'], y_pred, label='Предсказанные значения', color = 'red')
ax1.scatter(X['Age'], y, marker="o", label='Истинные значения', color = 'blue')
ax1.scatter(X['Age'], y - y_pred, marker="o", label='Остатки', color = 'green')

plt.legend(loc='upper left')
plt.title('Множественная регрессия для Age')
```

```
Text(0.5, 1.0, 'Множественная регрессия для Age')
```



3. Критерий Дурбина-Уотсона

```
def DW_statistic(y_true, y_pred):
    eps = y_true - y_pred
    eps_sum = 0

    for i in range(1, len(eps)):
        eps_sum += ((eps[i] - eps[i-1]) ** 2)

    Q_ost = (eps ** 2).sum()

    return eps_sum / Q_ost

DW_statistic(y, y_pred)
```

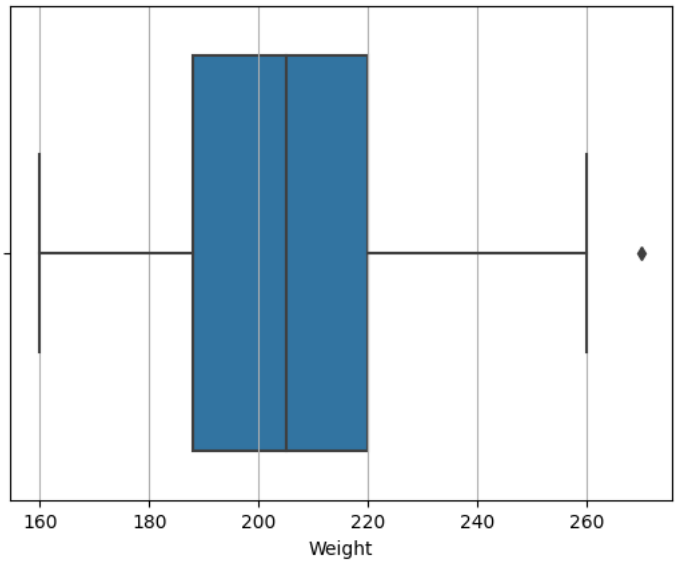
```
1.6136607977644188
```

Критерий Дурбина-Уотсона меньше граничного значения. Это означает, что автокорреляция остатков присутствует, они распределены не случайно.

4. Анализ выбросов

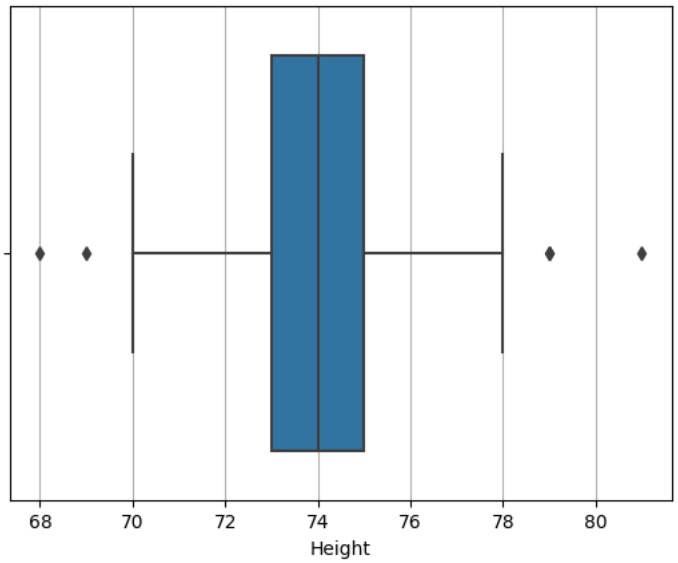
```
plt.grid()
sns.boxplot(x = baseball_df['Weight'])
```

```
<Axes: xlabel='Weight'>
```




```
plt.grid()
sns.boxplot(x = baseball_df['Height'])
```

```
<Axes: xlabel='Height'>
```



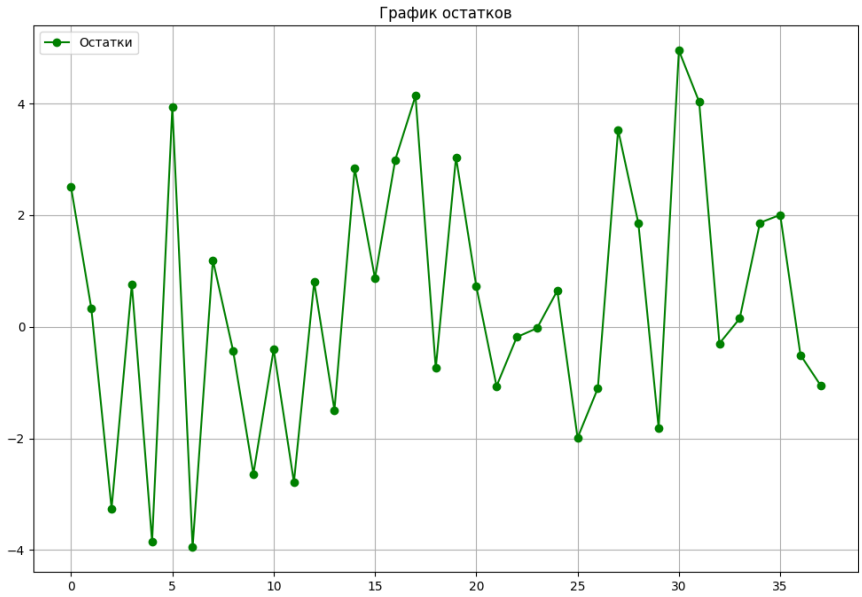
5. Предсказание зависимой переменной

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
regressor = LinearRegression().fit(X_train, y_train)
y_pred = regressor.predict(X_test)

det_score = r2_score(y_test, y_pred)
print('Коэффициент детерминации = ', round(det_score, 2))

mse = mean_squared_error(y_test, y_pred)
print('Средний квадрат ошибки = ', round(mse, 2))

det_corr_score = R2_corr(y_test, y_pred, K=2)
print('Скорректированный коэффициент детерминации = ', round(det_corr_score, 2))

print('Стандартная ошибка = ', round(mean_absolute_error(y_test, y_pred), 2))
```

```
Коэффициент детерминации =  0.26
Средний квадрат ошибки =  5.36
Скорректированный коэффициент детерминации =  0.24
Стандартная ошибка =  1.86
```

```
y_test = y_test.reset_index()
y_test = y_test['Height']

fig = plt.figure(figsize = (12, 8))
ax1 = fig.add_subplot(111)

ax1.grid()
ax1.plot(y_test - y_pred, marker="o", label='Остатки', color = 'green')

plt.legend(loc='upper left')
plt.title('График остатков')
```

```
Text(0.5, 1.0, 'График остатков')
```



6. Исключение выбросов

```
X.loc[X['Weight'] > 260, 'Weight']= X['Weight'].mean()

plt.grid()
sns.boxplot(x = X['Weight'])
```

```
<Axes: xlabel='Weight'>
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
X_train.shape, y_train.shape, X_test.shape, y_test.shape

regressor = LinearRegression().fit(X_train, y_train)
y_pred = regressor.predict(X_test)

det_score = r2_score(y_test, y_pred)
print('Коэффициент детерминации = ', round(det_score, 2))

mse = mean_squared_error(y_test, y_pred)
print('Средний квадрат ошибки = ', round(mse, 2))

det_corr_score = R2_corr(y_test, y_pred, K=2)
print('Скорректированный коэффициент детерминации = ', round(det_corr_score, 2))

print('Стандартная ошибка = ', round(mean_absolute_error(y_test, y_pred), 2))

y_test = y_test.reset_index()
y_test = y_test['Height']

fig = plt.figure(figsize = (12, 8))
ax1 = fig.add_subplot(111)

ax1.grid()
ax1.plot(y_test - y_pred, marker="o", label='Остатки', color = 'green')


plt.legend(loc='upper left')
plt.title('График остатков')
```

```
Коэффициент детерминации =  0.29
Средний квадрат ошибки =  5.16
Скорректированный коэффициент детерминации =  0.27
Стандартная ошибка =  1.85
Text(0.5, 1.0, 'График остатков')
```