**ИУ5-64 Сорокин Андрей**

**Вариант 19**

**Задание:** для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

**Методы:** линейная/логистическая регрессия, градиентный бустинг.

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn import preprocessing
from sklearn import svm
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error
from xgboost import XGBRegressor
```

In [9]:
```
data.keys().to_list()
```

Out[9]:
```
['Date',
 'Team',
 'Opponent',
 'Goal Scored',
 'Ball Possession %',
 'Attempts',
 'On-Target',
 'Off-Target',
 'Blocked',
 'Corners',
 'Offsides',
 'Free Kicks',
 'Saves',
 'Pass Accuracy %',
 'Passes',
 'Distance Covered (Kms)',
 'Fouls Committed',
 'Yellow Card',
 'Yellow & Red',
 'Red',
 'Man of the Match',
 '1st Goal',
 'Round',
 'PSO',
 'Goals in PSO',
 'Own goals',
 'Own goal Time']
```

Подсчет пропусков

```
data.isna().sum()
```

```
Out[10]: Date                        0
         Team                        0
         Opponent                    0
         Goal Scored                 0
         Ball Possession %           0
         Attempts                    0
         On-Target                   0
         Off-Target                  0
         Blocked                     0
         Corners                     0
         Offsides                    0
         Free Kicks                  0
         Saves                       0
         Pass Accuracy %             0
         Passes                      0
         Distance Covered (Kms)      0
         Fouls Committed             0
         Yellow Card                 0
         Yellow & Red                0
         Red                         0
         Man of the Match            0
         1st Goal                   34
         Round                       0
         PSO                         0
         Goals in PSO                0
         Own goals                 116
         Own goal Time             116
         dtype: int64
```

Заполнение пропусков

```
data['1st Goal'] = data['1st Goal'].fillna(data['1st Goal'].mean())
data['Own goals'] = data['Own goals'].fillna(data['Own goals'].mean())
data['Own goal Time'] = data['Own goal Time'].fillna(data['Own goal Time'].mean())
```

Out[31]:

| | Goal Scored | Free Kicks | Goals in PSO | Date | Team | Opponent | Man of the Match | Round | PSO |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 5 | 11 | 0 | 14-06-2018 | Russia | Saudi Arabia | Yes | Group Stage | No |
| 1 | 0 | 25 | 0 | 14-06-2018 | Saudi Arabia | Russia | No | Group Stage | No |
| 2 | 0 | 7 | 0 | 15-06-2018 | Egypt | Uruguay | No | Group Stage | No |
| 3 | 1 | 13 | 0 | 15-06-2018 | Uruguay | Egypt | Yes | Group Stage | No |
| 4 | 0 | 14 | 0 | 15-06-2018 | Morocco | Iran | No | Group Stage | No |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 123 | 1 | 24 | 0 | 11-07-2018 | England | Croatia | No | Semi- Finals | No |
| 124 | 2 | 5 | 0 | 14-07-2018 | Belgium | England | Yes | 3rd Place | No |
| 125 | 0 | 12 | 0 | 14-07-2018 | England | Belgium | No | 3rd Place | No |
| 126 | 4 | 14 | 0 | 15-07-2018 | France | Croatia | Yes | Final | No |
| 127 | 2 | 15 | 0 | 15-07-2018 | Croatia | France | No | Final | No |

```
In [32]: #Кодирование категориальных признаков
data_1["Date"].value_counts()
data_1["Date"] = data_1["Date"].astype('category')

data_1["Team"] = data_1["Team"].astype('category')
data_1["Opponent"] = data_1["Opponent"].astype('category')
data_1["Man of the Match"] = data_1["Man of the Match"].astype('category')
data_1["Round"] = data_1["Round"].astype('category')
data_1["PSO"] = data_1["PSO"].astype('category')

#Назначить закодированную переменную новому столбцу с помощью метода доступа
data_1["Date_cat"] = data_1["Date"].cat.codes
data_1["Team_cat"] = data_1["Team"].cat.codes
data_1["Opponent_cat"] = data_1["Opponent"].cat.codes
data_1["Man of the Match_cat"] = data_1["Man of the Match"].cat.codes
data_1["Round_cat"] = data_1["Round"].cat.codes
data_1["PSO_cat"] = data_1["PSO"].cat.codes
```

Out[32]:

| | Goal Scored | Free Kicks | Goals in PSO | Date_cat | Team_cat | Opponent_cat | Man of the Match_cat | Round_cat | PSO_cat |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 5 | 11 | 0 | 7 | 23 | 24 | 1 | 2 | 0 |
| 1 | 0 | 25 | 0 | 7 | 24 | 23 | 0 | 2 | 0 |
| 2 | 0 | 7 | 0 | 9 | 8 | 31 | 0 | 2 | 0 |
| 3 | 1 | 13 | 0 | 9 | 31 | 8 | 1 | 2 | 0 |
| 4 | 0 | 14 | 0 | 9 | 17 | 13 | 0 | 2 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 123 | 1 | 24 | 0 | 6 | 9 | 6 | 0 | 5 | 0 |
| 124 | 2 | 5 | 0 | 8 | 2 | 9 | 1 | 0 | 0 |
| 125 | 0 | 12 | 0 | 8 | 9 | 2 | 0 | 0 | 0 |
| 126 | 4 | 14 | 0 | 10 | 10 | 5 | 1 | 1 | 0 |
| 127 | 2 | 16 | 0 | 10 | 5 | 10 | 0 | 1 | 0 |

128 rows × 9 columns

In [33]:
```
#разделение выборки
from sklearn.model_selection import train_test_split
y = data_1['Free Kicks']
X = data_1.drop('Free Kicks', axis=1)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
x_train
```

Out[33]:

| | Goal Scored | Goals in PSO | Date_cat | Team_cat | Opponent_cat | Man of the Match_cat | Round_cat | PSO_cat |
|---|---|---|---|---|---|---|---|---|
| 94 | 0 | 0 | 23 | 9 | 2 | 0 | 2 | 0 |
| 108 | 1 | 0 | 2 | 28 | 29 | 1 | 4 | 0 |
| 47 | 0 | 0 | 17 | 5 | 3 | 0 | 2 | 0 |
| 83 | 3 | 0 | 22 | 28 | 16 | 1 | 2 | 0 |
| 27 | 2 | 0 | 13 | 9 | 30 | 1 | 2 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 56 | 2 | 0 | 18 | 11 | 28 | 1 | 2 | 0 |
| 3 | 1 | 0 | 9 | 31 | 8 | 1 | 2 | 0 |
| 121 | 0 | 0 | 5 | 2 | 10 | 0 | 5 | 0 |
| 24 | 3 | 0 | 13 | 2 | 19 | 1 | 2 | 0 |
| 106 | 3 | 0 | 1 | 2 | 14 | 1 | 4 | 0 |

89 rows × 8 columns

In [34]:
```
y_train
```

Out[34]:
```
94     15
108    13
47     14
83     16
27     16
       ..
56     15
3      13
121     7
24     21
106    10
Name: Free Kicks, Length: 89, dtype: int64
```

In [35]:
```
#Логистическая регрессия
from sklearn.linear_model import LogisticRegression
```

In [47]:
```
def print_metrics(y_test, y_pred):
    print(f"R^2: {r2_score(y_test, y_pred)}")
    print(f"MSE: {mean_squared_error(y_test, y_pred)}")
    print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
```

In [41]:
```
import warnings
warnings.filterwarnings('ignore')
model_logistic = LogisticRegression()
model_logistic.fit(x_train, y_train)
```

Out[41]:
```
LogisticRegression()
```

In [43]:
```
targ_logistic = model_logistic.predict(x_test)
```

In [44]:
```
mae = mean_absolute_error(y_test,targ_logistic)
mape = mean_absolute_percentage_error(y_test,targ_logistic)
mse = mean_squared_error(y_test,targ_logistic)
print('MAE:' + str(round(mae,3)) + ' MAPE:' + str(round(mape,3)) + ' MSE:' + str(round(mse,3)))
```

```
MAE:5.564 MAPE:0.43 MSE:45.513
```

In [49]:
```
#Градиентный бустинг
XGB_model = XGBRegressor()
mape = -cross_val_score(XGB_model,x_train,y_train,cv=4,scoring='neg_mean_absolute_percentage_error').mean()
mae = -cross_val_score(XGB_model,x_train,y_train,cv=4,scoring='neg_mean_absolute_error').mean()
mse = -cross_val_score(XGB_model,x_train,y_train,cv=4,scoring='neg_mean_squared_error').mean()
print('SVM Errors')
print('MAE:' + str(round(mae,3)) + ' MAPE:' + str(round(mape,3)) + ' MSE:' + str(round(mse,3)))
```

```
SVM Errors
MAE:4.361 MAPE:0.355 MSE:30.424
```

In [50]:
```
XGB_model.fit(x_train,y_train)
mae = mean_absolute_error(y_test,XGB_model.predict(x_test))
mape = mean_absolute_percentage_error(y_test,XGB_model.predict(x_test))
mse = mean_squared_error(y_test,XGB_model.predict(x_test))
print('MAE:' + str(round(mae,3)) + ' MAPE:' + str(round(mape,3)) + ' MSE:' + str(round(mse,3)))
```

```
MAE:5.732 MAPE:0.416 MSE:45.084
```

Вывод: модель градиентный бустинг показала себя луче, чем логическая регрессия.