

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.

- Для студентов групп ИУ5-61Б, ИУ5-62Б, ИУ5-63Б, ИУ5-64Б, ИУ5-65Б, РТ5-61Б номер варианта = номер в списке группы.

Группа	Метод №1	Метод №2
ИУ5-64Б, ИУ5Ц-84Б	Линейная/логистическая регрессия	Градиентный бустинг

Используемый набор данных: [Predict FIFA 2018 Man of the Match | Kaggle](https://www.kaggle.com/mathan/fifa-2018-match-statistics)

Результат:

The screenshot shows a Jupyter Notebook titled "RK-2_Svetasheva_Yuliya_IU5-64B". The first cell contains the task description in Russian, which matches the text in the first block of the document. The second cell contains the following Python code:

```
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 [8]: data = pd.read_csv("FIFA 2018 Statistics.csv")
data
```

Out[8]:

	Date	Team	Opponent	Goal Scored	Possession %	Attempts	On-Target	Off-Target	Blocked	Corners	...	Yellow Card	Yellow & Red	Red	Man of the Match	1st Goal	Round	PSO	Goals in PSO	€
0	14-06-2018	Russia	Saudi Arabia	5	40	13	7	3	3	6	...	0	0	0	Yes	12.0	Group Stage	No	0	
1	14-06-2018	Saudi Arabia	Russia	0	60	6	0	3	3	2	...	0	0	0	No	NaN	Group Stage	No	0	
2	15-06-2018	Egypt	Uruguay	0	43	8	3	3	2	0	...	2	0	0	No	NaN	Group Stage	No	0	
3	15-06-2018	Uruguay	Egypt	1	57	14	4	6	4	5	...	0	0	0	Yes	89.0	Group Stage	No	0	
4	15-06-2018	Morocco	Iran	0	64	13	3	6	4	5	...	1	0	0	No	NaN	Group Stage	No	0	
...
123	11-07-2018	England	Croatia	1	46	11	1	6	4	4	...	1	0	0	No	5.0	Semi-Finals	No	0	
124	14-07-2018	Belgium	England	2	43	12	4	3	5	4	...	1	0	0	Yes	4.0	3rd Place	No	0	
125	14-07-2018	England	Belgium	0	57	15	5	7	3	5	...	2	0	0	No	NaN	3rd Place	No	0	
126	15-07-2018	France	Croatia	4	39	8	6	1	1	2	...	2	0	0	Yes	18.0	Final	No	0	
127	15-07-2018	Croatia	France	2	61	15	3	8	4	6	...	1	0	0	No	28.0	Final	No	0	

128 rows × 27 columns

```
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']
```

```
In [10]: #Подсчет пропусков
data.isna().sum()
```

```
File Edit View Insert Cell Kernel Widgets Help
```

```

In [10]: #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

```

```

In [13]: #заполним пропуски
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())

In [14]: #Проверяем, что пропуски заполнились
data.isna().sum()

```

```
File Edit View Insert Cell Kernel Widgets Help
```

```

Out[14]: 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                    0
Round                       0
PSO                         0
Goals in PSO                0
Own goals                   0
Own goal Time               0
dtype: int64

In [15]: #Определим типы столбцов
data.dtypes

```

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

In [15]: #Определим типы столбцов
data.dtypes

Out[15]:
Date object
Team object
Opponent object
Goal Scored int64
Ball Possession % int64
Attempts int64
On-Target int64
Off-Target int64
Blocked int64
Corners int64
Offsides int64
Free Kicks int64
Saves int64
Pass Accuracy % int64
Passes int64
Distance Covered (Kms) int64
Fouls Committed int64
Yellow Card int64
Yellow & Red int64
Red int64
Man of the Match object
1st Goal float64
Round object
PSO object
Goals in PSO int64
Own goals float64
Own goal Time float64
dtype: object

In [31]: #кодирование категориальными признаками, далее будем работать только с этими столбцами
data_1 = data[['Goal Scored', 'Free Kicks', 'Goals in PSO', 'Date', 'Team', 'Opponent', 'Man of the Match', 'Round', 'PSO']].copy()
data_1

File Edit View Insert Cell Kernel Widgets Help

In [31]:

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

128 rows x 9 columns

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 новой 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

```
File Edit View Insert Cell Kernel Widgets Help
data_cat = data_1.drop(['Date', 'Team', 'Opponent', 'Man of the Match', 'Round', 'PSO'], axis=1, inplace=True)
data_1
```

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
1	0	25	0	7	24	23		0	2
2	0	7	0	9	8	31		0	2
3	1	13	0	9	31	8		1	2
4	0	14	0	9	17	13		0	2
...
123	1	24	0	6	9	6		0	5
124	2	5	0	8	2	9		1	0
125	0	12	0	8	9	2		0	0
126	4	14	0	10	10	6		1	1
127	2	15	0	10	6	10		0	1

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
```

```
File Edit View Insert Cell Kernel Widgets Help
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
108	1	0	2	28	29		1	4
47	0	0	17	5	3		0	2
83	3	0	22	28	16		1	2
27	2	0	13	9	30		1	2
...
56	2	0	18	11	28		1	2
3	1	0	9	31	8		1	2
121	0	0	5	2	10		0	5
24	3	0	13	2	19		1	2
106	3	0	1	2	14		1	4

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
```

```
In [62]: #Сравнение моделей
          Видим, что модель Градиентный бустинг показал себя лучше, чем логистическая регрессия
```