

Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»



Отчет
Рубежный контроль № 2
По курсу «Технологии машинного обучения»

ИСПОЛНИТЕЛЬ:

Группа ИУ5-65Б

Попов М.А.

"25" мая 2020 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

"__" _____ 2021 г.

Москва 2021

PK2

Попов Михаил Александрович, ИУ5-656

Вариант 17:

метод 1: метод опорных векторов

метод 2: градиентный бустинг

датасет: FIFA 2018 Statistics

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.

Решение

Импорт библиотек и загрузка данных

```
[57] import numpy as np
import pandas as pd
import seaborn as sns
from google.colab import files
import matplotlib.pyplot as plt
from IPython.display import Image
from io import StringIO
from typing import Tuple, Dict
from operator import itemgetter
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, median_absolute_error
from sklearn.svm import LinearSVR, SVR, SVC
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.impute import SimpleImputer, MissingIndicator
%matplotlib inline
sns.set(style="ticks")
```

```
[4] # загрузка данных
uploaded = files.upload()

Выборить файлы | FIFA 201...atistics.csv
• FIFA 2018 Statistics.csv(application/vnd.ms-excel) - 12557 bytes, last modified: 01.10.2019 - 100% done
Saving FIFA 2018 Statistics.csv to FIFA 2018 Statistics.csv
```

```
[5] data = pd.read_csv('FIFA 2018 Statistics.csv', sep=',')
```

Характеристики датасета

```
[77] # размер датасета
data.shape

(128, 27)
```

```
[78] # типы колонок
data.dtypes

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
```

```
[79] # первые 5 строк
data.head()
```

[79] # первые 5 строк
data.head()

	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
0	14-06-2018	Russia	Saudi Arabia	5	40	13	7	3	3	6	3	11	0	78	306	118	22	0	0	0	Ye
1	14-06-2018	Saudi Arabia	Russia	0	60	6	0	3	3	2	1	25	2	86	511	105	10	0	0	0	N
2	15-06-2018	Egypt	Uruguay	0	43	8	3	3	2	0	1	7	3	78	395	112	12	2	0	0	N
3	15-06-2018	Uruguay	Egypt	1	57	14	4	6	4	5	1	13	3	86	589	111	6	0	0	0	Ye
4	15-06-2018	Morocco	Iran	0	64	13	3	6	4	5	0	14	2	86	433	101	22	1	0	0	N

[80] # статистические характеристики признаков
data.describe()

	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
count	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000	128.000000
mean	1.320312	49.992188	12.593750	3.914062	5.273438	3.359375	4.718750	1.343750	14.890625	2.726562	82.554688	462.648438	106.664062	13.546875	1.695312
std	1.156519	10.444074	5.245827	2.234403	2.409675	2.403195	2.446072	1.193404	4.724262	2.049447	5.933766	151.186311	11.749537	4.619131	1.325454
min	0.000000	25.000000	3.000000	0.000000	1.000000	0.000000	0.000000	0.000000	5.000000	0.000000	67.000000	189.000000	80.000000	5.000000	0.000000
25%	0.000000	42.000000	9.000000	2.000000	4.000000	1.750000	3.000000	0.000000	11.000000	1.000000	79.000000	351.000000	101.000000	10.000000	1.000000
50%	1.000000	50.000000	12.000000	3.500000	5.000000	3.000000	5.000000	1.000000	15.000000	2.000000	83.000000	462.000000	104.500000	13.000000	2.000000
75%	2.000000	58.000000	15.000000	5.000000	7.000000	4.000000	6.000000	2.000000	18.000000	4.000000	87.000000	555.250000	109.000000	16.000000	2.000000
max	6.000000	75.000000	26.000000	12.000000	11.000000	10.000000	11.000000	5.000000	26.000000	9.000000	94.000000	1137.000000	148.000000	25.000000	6.000000

[81] # Количество пропусков в данных
data.isna().sum()

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

[82] # Количество уникальных значений для каждого признака
data.nunique()

Date	25
Team	32
Opponent	32
Goal Scored	7
Ball Possession %	43
Attempts	24
On-Target	12
Off-Target	11
Blocked	11
Corners	12
Offsides	6
Free Kicks	21
Saves	10
Pass Accuracy %	25
Passes	110
Distance Covered (Kms)	35
Fouls Committed	21
Yellow Card	7
Yellow & Red	2
Red	2
Man of the Match	2
1st Goal	56
Round	6
PSO	2
goals in PSO	4
Own goals	1
Own goal Time	11
dtype:	int64

▼ Заполнение пропусков данных

```
[83] # удалим признаки Own goals и Own goals time
# т.к. почти все значения этих признаков пусты
cols_to_drop = ['Own goals', 'Own goal Time']
data_new_1 = data.drop(cols_to_drop, axis=1)
(data.shape, data_new_1.shape)
```

((128, 27), (128, 25))

```
[84] # для признака 1st Goal заполним пустые значения 0
# что будет значить, что гола не было
data_new_2 = data_new_1.fillna(0)
data_new_2.head()
```

	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
0	14-06-2018	Russia	Saudi Arabia	5	40	13	7	3	3	6	3	11	0	78	306	118	22	0	0	0	Ye
1	14-06-2018	Saudi Arabia	Russia	0	60	6	0	3	3	2	1	25	2	86	511	105	10	0	0	0	N
2	15-06-2018	Egypt	Uruguay	0	43	8	3	3	2	0	1	7	3	78	395	112	12	2	0	0	N
3	15-06-2018	Uruguay	Egypt	1	57	14	4	6	4	5	1	13	3	86	589	111	6	0	0	0	Ye
4	15-06-2018	Morocco	Iran	0	64	13	3	6	4	5	0	14	2	86	433	101	22	1	0	0	N

```
[85] # пропусков не осталось
data_new_2.isna().sum()
```

```
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
dtype: int64
```

▼ Кодирование категориальных признаков

```
[87] # оценим важность признаков для целевого
dataLE = data_new_2.copy()
le = LabelEncoder()
col_obj = dataLE.dtypes[dataLE.dtypes==object].index.values.tolist()
for i in col_obj:
    dataLE[i] = le.fit_transform(dataLE[i])
(dataLE.corr()['Goal Scored']*100).sort_values(ascending=False)
```

```
Goal Scored      100.000000
Man of the Match  52.219294
On-Target        46.170195
1st Goal         27.826782
Attempts         14.491471
Pass Accuracy %  13.568842
Opponent         7.309972
Date             5.792964
Free Kicks       4.681488
Offsides         4.510519
Passes          4.397086
Corners          4.044615
Ball Possession % 3.475891
Fouls Committed  3.033115
Distance Covered (Kms) 1.435519
Round           0.202324
Goals in PSO     -1.120424
PSO              -1.575931
Off-Target       -2.037402
Yellow & Red     -3.503111
Yellow Card      -4.883817
Team             -8.352578
Blocked          -8.707206
Red              -8.971381
Saves           -11.089319
Name: Goal Scored, dtype: float64
```

```
[88] # по результатам корреляционного анализа удаляем столбцы,
# которые имеют меньшую значимость по отношению к целевому признаку
del_col = (dataE.corr()[['Goal Scored']]*100).sort_values(ascending=False)
del_col = del_data[(del_data < 10) & (del_data > -10) | (del_data.isnull())].index.values.tolist()
data_new_2.drop(columns=del_col, inplace=True)
dataE.drop(columns=del_col, inplace=True)
data_new_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128 entries, 0 to 127
Data columns (total 7 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   Goal Scored         128 non-null   int64
1   Attempts            128 non-null   int64
2   On-Target           128 non-null   int64
3   Saves               128 non-null   int64
4   Pass Accuracy %     128 non-null   int64
5   Man of the Match    128 non-null   object
6   1st Goal             128 non-null   float64
dtypes: float64(1), int64(5), object(1)
memory usage: 7.1+ KB
```

```
[89] # выполним one-hot encoding и масштабирование для применения в SVM
col_num = data_new_2.dtypes[data_new_2.dtypes!=object].index.values.tolist()
col_num.remove('Goal Scored')
se = StandardScaler()
data_new_2[col_num] = se.fit_transform(data_new_2[col_num])
data_new_2 = pd.get_dummies(data_new_2, drop_first=True)
```

```
[90] data_new_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128 entries, 0 to 127
Data columns (total 7 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   Goal Scored         128 non-null   int64
1   Attempts            128 non-null   float64
2   On-Target           128 non-null   float64
3   Saves               128 non-null   float64
4   Pass Accuracy %     128 non-null   float64
5   1st Goal             128 non-null   float64
6   Man of the Match Yes 128 non-null   uint8
dtypes: float64(5), int64(1), uint8(1)
memory usage: 6.2 KB
```

▼ Разделение выборки на обучающую и тестовую

```
[91] # разделим данные на целевой столбец и признаки
X = data_new_2.drop("Goal Scored", axis=1)
Y = data_new_2["Goal Scored"]
```

```
[92] # с использованием метода train_test_split разделим выборку на обучающую и тестовую
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=1)
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape

((96, 6), (32, 6), (96,), (32,))
```

▼ Метод опорных векторов

```
[93] svr = SVR(kernel='rbf')
svr.fit(X_train, Y_train)
print("r2_score:", r2_score(Y_test, svr.predict(X_test)))
print("mean_squared_error", mean_squared_error(Y_test, svr.predict(X_test)))
```

```
r2_score: 0.43056278181961705
mean_squared_error 0.43541927913597644
```

▼ Градиентный бустинг

```
[94] gbr = GradientBoostingRegressor()
gbr.fit(X_train, Y_train)
print("r2_score:", r2_score(Y_test, gbr.predict(X_test)))
print("mean_squared_error", mean_squared_error(Y_test, gbr.predict(X_test)))
```

```
r2_score: 0.2629006076992503
mean_squared_error 0.5636218986049678
```

▼ Итог

Метод опорных векторов показал себя лучше градиентного бустинга, однако коэффициент детерминации для обеих моделей получился меньше 50%, что говорит о плохом качестве моделей. Вероятно это связано со слабой связностью датасета.