Московский государственный технический
университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по рубежному контролю №2

Вариант 11

Выполнил:
Студент группы ИУ5-63Б
Коноваликова Светлана
Руководители: Гапанюк Ю.Е.

Дата: 12.06.22

Москва, 2022 г.

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.

Метод 1 – дерево решений

Метод 2 – случайный лес

Датасет - https://www.kaggle.com/datasets/winterbreeze/fifa19eda

Выполнение:

Загрузка датасет

```
: import numpy as np
  import pandas as pd
  import seaborn as sns
  from sklearn.preprocessing import LabelEncoder
  from sklearn import tree
  from sklearn.model_selection import train_test_split
  from sklearn.tree import DecisionTreeRegressor
  from sklearn import metrics
  from sklearn.preprocessing import StandardScaler
  from sklearn.ensemble import RandomForestRegressor
  %matplotlib inline
  sns.set(style="ticks")
```

```
: data = pd.read_csv('C:/Users/Kotos/Desktop/fifa_eda.csv', sep=",")
  data = data.drop_duplicates()
```

```
: # Список колонок с типами данных
  data.dtypes
```

```
: ID                          int64
  Name                       object
  Age                         int64
  Nationality                object
  Overall                     int64
  Potential                   int64
  Club                       object
  Value                     float64
  Wage                      float64
  Preferred Foot             object
  International Reputation  float64
  Skill Moves               float64
  Position                   object
  Joined                      int64
  Contract Valid Until       object
  Height                    float64
  Weight                    float64
  Release Clause            float64
  dtype: object
```

Кодирование категориальных признаков

```
lename=LabelEncoder()
data['Name']=lename.fit_transform(data['Name'])
lenation=LabelEncoder()
data['Nationality']=lenation.fit_transform(data['Nationality'])
leclub=LabelEncoder()
data['Club']=leclub.fit_transform(data['Club'])
lefoot=LabelEncoder()
data['Preferred Foot']=lefoot.fit_transform(data['Preferred Foot'])
lepisit=LabelEncoder()
data['Position']=lepisit.fit_transform(data['Position'])
lecont=LabelEncoder()
data['Contract Valid Until']=lecont.fit_transform(data['Contract Valid Until'])
```

```
data.head()
```

	ID	Name	Age	Nationality	Overall	Potential	Club	Value	Wage	Preferred Foot	International Reputation	Skill Moves	Position	Joined
0	158023	9632	31	6	94	94	212	110500.0	565.0	0	5.0	4.0	21	2004
1	20801	3153	33	123	94	94	326	77000.0	405.0	1	5.0	5.0	26	2018
2	190871	12508	26	20	92	93	435	118500.0	290.0	1	5.0	5.0	14	2017
3	193080	4136	27	139	91	93	375	72000.0	260.0	1	4.0	1.0	5	2011
4	192985	8617	27	13	91	92	374	102000.0	355.0	1	4.0	4.0	19	2015

Удаление строк с пустыми значениями

```
# проверим есть ли пропущенные значения
data.isnull().sum()
```

```
ID                          0
Name                        0
Age                         0
Nationality                 0
Overall                     0
Potential                   0
Club                        0
Value                     252
Wage                        0
Preferred Foot              0
International Reputation   48
Skill Moves                48
Position                    0
Joined                      0
Contract Valid Until        0
Height                      0
Weight                      0
Release Clause              0
dtype: int64
```

```
# Удаление строк, содержащих пустые значения
data = data.dropna(axis=0, how='any')
```

Дерево решений

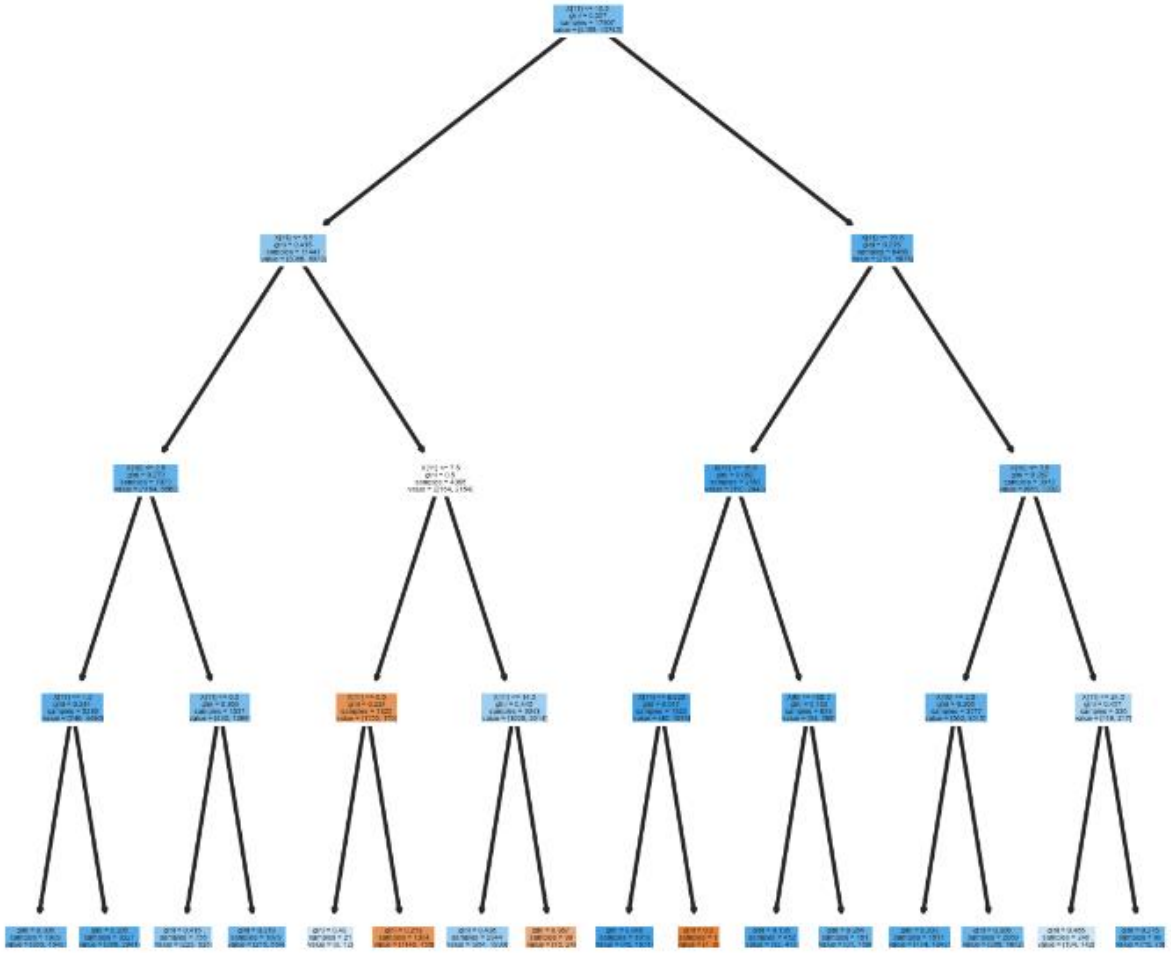
```
# Разделение признаков
x = data.drop('Preferred Foot', axis=1)
y = data['Preferred Foot']
```

```
# Построение модели
model=tree.DecisionTreeClassifier(max_depth=4, random_state=1)
model.fit(x,y)
```

```
DecisionTreeClassifier(max_depth=4, random_state=1)
```

```
# Визуализирование данных
from matplotlib import pyplot as plt
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (5,5), dpi=300)
tree.plot_tree(model, filled=True)
```

```
[Text(0.5, 0.9, 'X[11] <= 16.5\ngini = 0.357\nsamples = 17907\nvalue = [4159, 13748]'),
 Text(0.25, 0.7, 'X[11] <= 5.5\ngini = 0.415\nsamples = 11441\nvalue = [3368, 8073]'),
 Text(0.125, 0.5, 'X[10] <= 2.5\ngini = 0.279\nsamples = 7073\nvalue = [1184, 5889]'),
 Text(0.0625, 0.3, 'X[11] <= 1.5\ngini = 0.244\nsamples = 5236\nvalue = [746, 4490]'),
 Text(0.03125, 0.1, 'gini = 0.306\nsamples = 1909\nvalue = [360, 1549]'),
 Text(0.09375, 0.1, 'gini = 0.205\nsamples = 3327\nvalue = [386, 2941]'),
 Text(0.1875, 0.3, 'X[11] <= 0.5\ngini = 0.363\nsamples = 1837\nvalue = [438, 1399]'),
 Text(0.15625, 0.1, 'gini = 0.415\nsamples = 758\nvalue = [223, 535]'),
 Text(0.21875, 0.1, 'gini = 0.319\nsamples = 1079\nvalue = [215, 864]'),
 Text(0.375, 0.5, 'X[11] <= 7.5\ngini = 0.5\nsamples = 4368\nvalue = [2184, 2184]'),
 Text(0.3125, 0.3, 'X[11] <= 6.5\ngini = 0.224\nsamples = 1325\nvalue = [1155, 170]'),
 Text(0.28125, 0.1, 'gini = 0.49\nsamples = 21\nvalue = [9, 12]'),
 Text(0.34375, 0.1, 'gini = 0.213\nsamples = 1304\nvalue = [1146, 158]'),
 Text(0.4375, 0.3, 'X[11] <= 14.5\ngini = 0.448\nsamples = 3043\nvalue = [1029, 2014]'),
 Text(0.40625, 0.1, 'gini = 0.438\nsamples = 2944\nvalue = [954, 1990]'),
 Text(0.46875, 0.1, 'gini = 0.367\nsamples = 99\nvalue = [75, 24]'),
 Text(0.75, 0.7, 'X[11] <= 20.5\ngini = 0.215\nsamples = 6466\nvalue = [791, 5675]'),
 Text(0.625, 0.5, 'X[11] <= 18.5\ngini = 0.082\nsamples = 2553\nvalue = [110, 2443]'),
 Text(0.5625, 0.3, 'X[14] <= 6.625\ngini = 0.047\nsamples = 1920\nvalue = [46, 1874]'),
 Text(0.53125, 0.1, 'gini = 0.046\nsamples = 1919\nvalue = [45, 1874]'),
 Text(0.59375, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.6875, 0.3, 'X[6] <= 468.5\ngini = 0.182\nsamples = 633\nvalue = [64, 569]'),
 Text(0.65625, 0.1, 'gini = 0.135\nsamples = 452\nvalue = [33, 419]'),
 Text(0.71875, 0.1, 'gini = 0.284\nsamples = 181\nvalue = [31, 150]'),
 Text(0.875, 0.5, 'X[10] <= 3.5\ngini = 0.287\nsamples = 3913\nvalue = [681, 3232]'),
 Text(0.8125, 0.3, 'X[10] <= 2.5\ngini = 0.265\nsamples = 3577\nvalue = [562, 3015]'),
 Text(0.78125, 0.1, 'gini = 0.203\nsamples = 1517\nvalue = [174, 1343]'),
 Text(0.84375, 0.1, 'gini = 0.306\nsamples = 2060\nvalue = [388, 1672]'),
 Text(0.9375, 0.3, 'X[11] <= 24.5\ngini = 0.457\nsamples = 336\nvalue = [119, 217]'),
 Text(0.90625, 0.1, 'gini = 0.488\nsamples = 246\nvalue = [104, 142]'),
 Text(0.96875, 0.1, 'gini = 0.278\nsamples = 90\nvalue = [15, 75]')]
```



```
# Разделение выборки на обучающую и тестовую
x_train, x_test, y_train, y_test = train_test_split(
    x, y,  test_size=0.2, random_state=1)
```

```
dtr = DecisionTreeRegressor()
dtr.fit(x_train, y_train)
y_pred = dtr.predict(x_test)
```

```
print('Дерево решений')
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Дерево решений
Mean Absolute Error: 0.27917364600781686
Mean Squared Error: 0.27917364600781686
Root Mean Squared Error: 0.5283688541235345
```

Случайный лес

```
x_train, x_test, y_train, y_test = train_test_split(data.drop(['Preferred Foot'], axis=1),
                                                    data['Preferred Foot'], test_size=0.5, random_state
```

```
# Масштабирование
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

```
reg = RandomForestRegressor(n_estimators=20, random_state=0)
reg.fit(x_train, y_train)
y_pred = reg.predict(x_test)
```

```
print('Случайный лес')
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Случайный лес
Mean Absolute Error: 0.27931650658923385
Mean Squared Error: 0.14789423721241904
Root Mean Squared Error: 0.38457019802946124
```

Случайный лес оказался более качественной моделью.