РК №2

Зелинский Даниил Михайлович

ИУ5-61Б

вариант №6

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: Линейная регрессия

Метод №2: Случайный лес

Набор данных: https://www.kaggle.com/mohansacharya/graduate-admissions (файл Admission_Predict.csv)

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.ensemble import RandomForestClassifier,
RandomForestRegressor
from sklearn.metrics import accuracy_score, f1_score
from sklearn.preprocessing import MinMaxScaler
from typing import Dict
from sklearn.tree import export_text
from IPython.display import Image
from io import StringIO
from sklearn.tree import DecisionTreeClassifier,
DecisionTreeRegressor, export_graphviz
import pydotplus
from IPython.core.display import HTML
import statsmodels.formula.api as smf
```

```
import warnings
warnings.filterwarnings('ignore')
```

Вывод информации о датасете

```
df = pd.read_csv(r'Admission_Predict.csv')
df.head()
```

```
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR
CGPA  \
0           1        337          118                  4  4.5   4.5
9.65
1           2        324          107                  4  4.0   4.5
8.87
2           3        316          104                  3  3.0   3.5
8.00
3           4        322          110                  3  3.5   2.5
8.67
4           5        314          103                  2  2.0   3.0
8.21

   Research  Chance of Admit
0         1              0.92
1         1              0.76
2         1              0.72
3         1              0.80
4         0              0.65
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         400 non-null    int64
 1   GRE Score          400 non-null    int64
 2   TOEFL Score        400 non-null    int64
 3   University Rating  400 non-null    int64
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64
 8   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 28.2 KB
```

Поиск дубликатов и пропусков

```
df.isna().sum()
```

```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

Разделение на объекты-признаки и целевой признак

```
df_x = pd.DataFrame(df.iloc[:, :-1].values,
                    columns=df.columns[:-1]
                   )
df_y = pd.DataFrame(df.iloc[:, -1].values, columns=['Chance of
Admit'])
```

```
df_x
```

```
     Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR
CGPA  \
0           1.0      337.0        118.0                4.0  4.5   4.5
9.65
1           2.0      324.0        107.0                4.0  4.0   4.5
8.87
2           3.0      316.0        104.0                3.0  3.0   3.5
8.00
3           4.0      322.0        110.0                3.0  3.5   2.5
8.67
4           5.0      314.0        103.0                2.0  2.0   3.0
8.21
..          ...        ...          ...                ...  ...   ...
...
395       396.0      324.0        110.0                3.0  3.5   3.5
9.04
396       397.0      325.0        107.0                3.0  3.0   3.5
9.11
397       398.0      330.0        116.0                4.0  5.0   4.5
9.45
398       399.0      312.0        103.0                3.0  3.5   4.0
8.78
399       400.0      333.0        117.0                4.0  5.0   4.0
9.66

     Research
0         1.0
```

```
1         1.0
2         1.0
3         1.0
4         0.0
..        ...
395       1.0
396       1.0
397       1.0
398       0.0
399       1.0

[400 rows x 8 columns]
```

```
df_y
```

```
     Chance of Admit
0               0.92
1               0.76
2               0.72
3               0.80
4               0.65
..               ...
395             0.82
396             0.84
397             0.91
398             0.67
399             0.95

[400 rows x 1 columns]
```

```
_df=pd.concat([df_x, df_y.reindex(df_x.index)], axis=1)
```

```
_df.head()
```

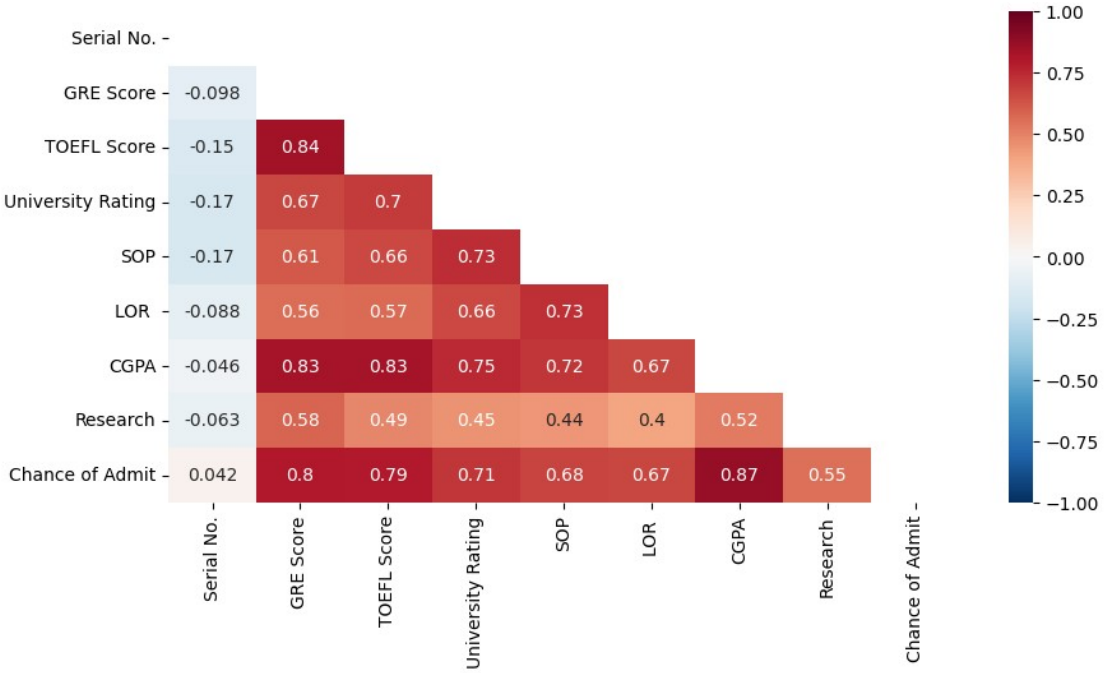
```
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR
CGPA  \
0         1.0      337.0        118.0                4.0  4.5   4.5
9.65
1         2.0      324.0        107.0                4.0  4.0   4.5
8.87
2         3.0      316.0        104.0                3.0  3.0   3.5
8.00
3         4.0      322.0        110.0                3.0  3.5   2.5
8.67
4         5.0      314.0        103.0                2.0  2.0   3.0
8.21

   Research  Chance of Admit
0       1.0             0.92
1       1.0             0.76
2       1.0             0.72
```

```
3       1.0              0.80
4       0.0              0.65
```

Корреляционная матрица

```
plt.figure(figsize=(10,5))
m=np.triu(np.ones_like(_df.corr(), dtype=bool))
sns.heatmap(_df.corr(), mask=m, annot=True, vmin=-1.0, vmax=1.0,
center=0, cmap='RdBu_r')
```

```
<AxesSubplot: >
```



Разделение на обучающую и тестовую выборки

```
X_train, X_test, y_train, y_test = train_test_split(df_x, df_y,
test_size = 0.2, random_state = 1)
```

1. Линейная регрессия

```
lin_r = LinearRegression()

lin_r.fit(X_train, y_train)

LinearRegression()

from sklearn import metrics
import math

y_pred = lin_r.predict(X_test)
mae = metrics.mean_absolute_error(y_test,y_pred)
mse = metrics.mean_squared_error(y_test,y_pred)
```

```
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {math.sqrt(mse)}")
print('Root Mean Squared Error (RMSE):',
metrics.mean_squared_error(y_test, y_pred, squared=False))
print('Mean Absolute Percentage Error (MAPE):',
metrics.mean_absolute_percentage_error(y_test, y_pred))
print('Explained Variance Score:',
metrics.explained_variance_score(y_test, y_pred))
print('Max Error:', metrics.max_error(y_test, y_pred))
```

```
Mean Absolute Error (MAE): 0.04435721469748688
Mean Squared Error (MSE): 0.06312385134060096
Root Mean Squared Error (RMSE): 0.06312385134060096
Mean Absolute Percentage Error (MAPE): 0.0754661301174471
Explained Variance Score: 0.8371928064177185
Max Error: 0.21728058308873588
```

2. Случайный лес

```
r_forest = RandomForestRegressor(n_estimators=20, oob_score=True,
random_state=10)
r_forest.fit(X_train, y_train)
```

```
RandomForestRegressor(n_estimators=20, oob_score=True,
random_state=10)
```

```
tree = export_text(r_forest[0], feature_names=list(X_train.columns))
HTML('<pre>' + tree + '</pre>')
```

```
<IPython.core.display.HTML object>
```

```
# Out-of-bag error, возвращаемый классификатором
r_forest.oob_score_, 1-r_forest.oob_score_
```

```
(0.7720339900076307, 0.22796600999236927)
```

```
from sklearn import metrics
import math

y_pred = r_forest.predict(X_test)
mae = metrics.mean_absolute_error(y_test,y_pred)
mse = metrics.mean_squared_error(y_test,y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {math.sqrt(mse)}")
print('Root Mean Squared Error (RMSE):',
metrics.mean_squared_error(y_test, y_pred, squared=False))
print('Mean Absolute Percentage Error (MAPE):',
metrics.mean_absolute_percentage_error(y_test, y_pred))
print('Explained Variance Score:',
```

```
metrics.explained_variance_score(y_test, y_pred))
print('Max Error:', metrics.max_error(y_test, y_pred))
```

```
Mean Absolute Error (MAE): 0.04351875000000001
Mean Squared Error (MSE): 0.06477767265810033
Root Mean Squared Error (RMSE): 0.06477767265810033
Mean Absolute Percentage Error (MAPE): 0.07333529988892434
Explained Variance Score: 0.8214936955969505
Max Error: 0.23500000000000004
```

Вывод

Сравнивая полученные модели по трём метрикам: MAE, MSE, RMSE, MAPE Explained Variance Score и Max Error, можно прийти к выводу, что обе модели производят предсказание результатов с приблизительно одинаковой точностью.