Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ 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

Выполнила:

студент группы ИУ5-63Б

Латыпова К.Н.

18.04.2021

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Москва, 2020 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.

Набор данных:
https://www.kaggle.com/brsdincer/star-type-classification

Текст программы в экранные формы с примерами выполнения программы (ячейки ноутбука):

РК2

Импорт библиотек

B [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
sns.set(style="ticks")
%matplotlib inline
```

B [2]:
```
data = pd.read_csv('Stars.csv')
```

B [3]:
```
data.head()
```

Out[3]:

	Temperature	L	R	A_M	Color	Spectral_Class	Type
0	3068	0.002400	0.1700	16.12	Red	M	0
1	3042	0.000500	0.1542	16.60	Red	M	0
2	2600	0.000300	0.1020	18.70	Red	M	0
3	2800	0.000200	0.1600	16.65	Red	M	0
4	1939	0.000138	0.1030	20.06	Red	M	0

B [4]:
```
data.dtypes
```

Out[4]:
```
Temperature         int64
L                 float64
R                 float64
A_M               float64
Color              object
Spectral_Class     object
Type                int64
dtype: object
```

```
B [5]: data.drop(['Color','Spectral_Class','Color'], axis = 1, inplace = True)
```

```
B [6]: data.isnull().sum()
       # проверим есть ли пропущенные значения
```

```
Out[6]: Temperature    0
        L              0
        R              0
        A_M            0
        Type           0
        dtype: int64
```

```
B [7]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Temperature  240 non-null    int64
 1   L            240 non-null    float64
 2   R            240 non-null    float64
 3   A_M          240 non-null    float64
 4   Type         240 non-null    int64
dtypes: float64(3), int64(2)
memory usage: 9.5 KB
```

```
B [8]: data.head()
```

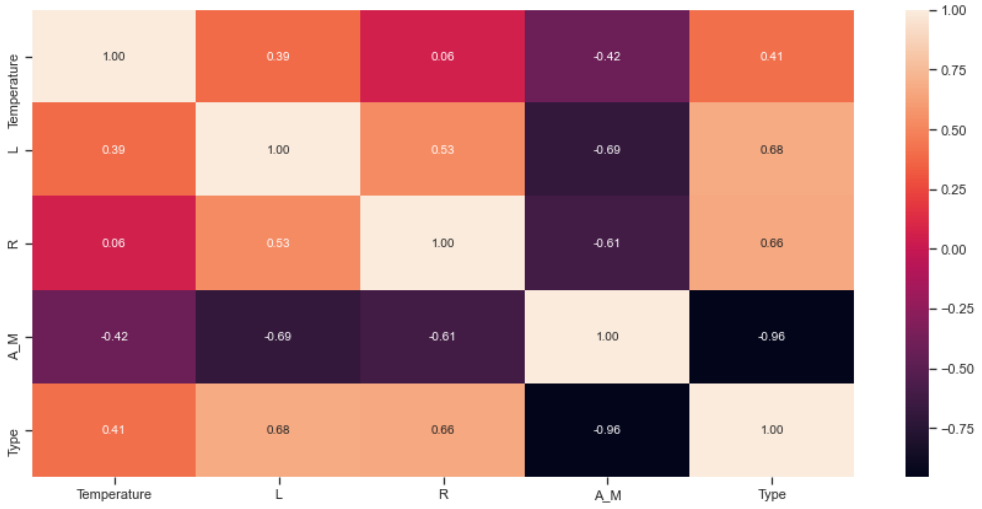

```
B [8]: data.head()
```


```
Out[8]:
```

	Temperature	L	R	A_M	Type
0	3068	0.002400	0.1700	16.12	0
1	3042	0.000500	0.1542	16.60	0
2	2600	0.000300	0.1020	18.70	0
3	2800	0.000200	0.1600	16.65	0
4	1939	0.000138	0.1030	20.06	0

```
B [9]: #Построим корреляционную матрицу
       fig, ax = plt.subplots(figsize=(15,7))
       sns.heatmap(data.corr(method='pearson'), ax=ax, annot=True, fmt='.2f')
```

```
Out[9]: <AxesSubplot:>
```




```
B [10]: data['L'] = data['L'].astype(int)
        X = data.drop(['L'], axis = 1)
        Y = data.L
        print('Входные данные:\n\n', X.head(), '\n\nВыходные данные:\n\n', Y.head())
```

```
Входные данные:

    Temperature       R    A_M  Type
0         3068  0.1700  16.12     0
1         3042  0.1542  16.60     0
2         2600  0.1020  18.70     0
3         2800  0.1600  16.65     0
4         1939  0.1030  20.06     0
```

Выходные данные:

```
 0    0
1    0
2    0
3    0
4    0
Name: L, dtype: int64
```

В [11]:
```
X_train,  X_test,  Y_train,  Y_test = train_test_split(X,  Y, random_state = 0, test_size = 0.1)
print('Входные параметры обучающей выборки:\n\n',X_train.head(), \
      '\n\nВходные параметры тестовой выборки:\n\n', X_test.head(), \
      '\n\nВыходные параметры обучающей выборки:\n\n', Y_train.head(), \
      '\n\nВыходные параметры тестовой выборки:\n\n', Y_test.head())
```

Входные параметры обучающей выборки:

	Temperature	R	A_M	Type
5	2840	0.110	16.980	0
22	7220	0.011	14.230	2
199	3463	0.675	14.776	1
97	7720	1.340	2.440	3
12	3134	0.196	13.210	1

Входные параметры тестовой выборки:

	Temperature	R	A_M	Type
109	33421	67.000	-5.79	4
71	3607	0.380	10.12	1
37	6380	0.980	2.93	3
74	3550	0.291	10.89	1
108	24345	57.000	-6.24	4

Выходные параметры обучающей выборки:

```
 5      0
22     0
199    0
97     7
12     0
Name: L, dtype: int64
```

Выходные параметры тестовой выборки:

```
 109    352000
71          0
37          1
74          0
108    142000
Name: L, dtype: int64
```

В [12]:
```
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.tree import export_graphviz
from sklearn import tree
import re
```

В [13]:
```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, Y)
from IPython.core.display import HTML
from sklearn.tree.export import export_text
tree_rules = export_text(clf, feature_names=list(X.columns))
HTML('<pre>' + tree_rules + '</pre>')
```

Out[13]:
```
|--- A_M <= 4.57
|   |--- R <= 1.23
|   |   |--- class: 1
|   |--- R >  1.23
|   |   |--- Temperature <= 3830.00
|   |   |   |--- Temperature <= 3766.00
|   |   |   |   |--- Temperature <= 3544.00
|   |   |   |   |   |--- A_M <= -6.71
|   |   |   |   |   |   |--- A_M <= -11.55
|   |   |   |   |   |   |   |--- class: 263000
|   |   |   |   |   |   |--- A_M >  -11.55
|   |   |   |   |   |   |   |--- Type <= 4.50
|   |   |   |   |   |   |   |   |--- class: 195000
|   |   |   |   |   |   |   |--- Type >  4.50
|   |   |   |   |   |   |   |   |--- R <= 1286.50
|   |   |   |   |   |   |   |   |   |--- class: 174000
```

B [14]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error,  median_absolute_error, r2_score
```

B [15]:
```
forest_1 = RandomForestRegressor(n_estimators=5, oob_score=True, random_state=10)
forest_1.fit(X, Y)
```

Out[15]:
```
RandomForestRegressor(n_estimators=5, oob_score=True, random_state=10)
```

B [16]:
```
Y_predict = forest_1.predict(X_test)
print('Средняя абсолютная ошибка:',   mean_absolute_error(Y_test, Y_predict))
print('Средняя квадратичная ошибка:', mean_squared_error(Y_test, Y_predict))
print('Median absolute error:',       median_absolute_error(Y_test, Y_predict))
print('Коэффициент детерминации:',    r2_score(Y_test, Y_predict))
```

```
Средняя абсолютная ошибка: 29120.574999999997
Средняя квадратичная ошибка: 5203187720.595
Median absolute error: 0.0
Коэффициент детерминации: 0.9230465292619281
```

B [17]:
```
plt.scatter(X_test.R, Y_test,    marker = 'o', label = 'Тестовая выборка')
plt.scatter(X_test.R, Y_predict, marker = '.', label = 'Предсказанные данные')
plt.legend(loc = 'lower right')
plt.xlabel('R')
plt.ylabel('L')
plt.show()
```