Задача.

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 (https://www.kaggle.com/brsdincer/star-type-classification)

Импорт библиотек, загрузка данных ¶

Ввод [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
import matplotlib_inline
import matplotlib.pyplot as plt
from IPython.display import Image
from io import StringIO
import graphviz
import pydotplus
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error, r2_score
from sklearn.ensemble import RandomForestRegressor
%matplotlib inline
%matplotlib inline
sns.set(style="ticks")
from IPython.display import set_matplotlib_formats
matplotlib_inline.backend_inline.set_matplotlib_formats("retina")
```

Ввод [2]:
```
data = pd.read_csv('Stars.csv', sep=",")
```

Основные характеристики датасета

Ввод [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

Получим размер датасета

Ввод [4]:
```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
total_count = data.shape[1]
print('Всего колонок: {}'.format(total_count))
```

```
Всего строк: 240
Всего колонок: 7
```

Отобразим список столбцов с их типами данных

Ввод [5]:
```
data.dtypes
```

Out[5]:
```
Temperature         int64
L                 float64
R                 float64
A_M               float64
Color              object
Spectral_Class     object
Type                int64
dtype: object
```

Проверим количество пропущенных значений в каждом столбце

Ввод [6]:
```
for col_empty in data.columns:
    empty_count = data[data[col_empty].isnull()].shape[0]
    print('{} - {}'.format(col_empty, empty_count))
```

```
Temperature - 0
L - 0
R - 0
A_M - 0
Color - 0
Spectral_Class - 0
Type - 0
```

Количество пропущенных значений означает, что все ячейки в этих столбцах заполнены

Преобразование категориальных признаков

Закодируем цвета и спектральные классы числовыми значениями (label encoding)

Ввод [7]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

Создадим новый датафрейм, содержащий только столбцы с типом данных object

Ввод [8]:
```
obj_data = data.select_dtypes(include=['object']).copy()
```

Ввод [9]:
```
obj_data.head()
```

Out[9]:

	Color	Spectral_Class
0	Red	M
1	Red	M
2	Red	M
3	Red	M
4	Red	M

Ввод [10]:
```
data["Spectral_Class"].value_counts()
```

Out[10]:
```
Spectral_Class
M    111
B     46
O     40
A     19
F     17
K      6
G      1
Name: count, dtype: int64
```

Ввод [11]:
```
data["Color"].value_counts()
```

Out[11]:
```
Color
Red                   112
Blue                   56
Blue-white             26
Blue White             10
yellow-white            8
White                   7
Blue white              4
white                   3
Yellowish White         3
yellowish               2
Whitish                 2
Orange                  2
White-Yellow            1
Pale yellow orange      1
Yellowish               1
Orange-Red              1
Blue-White              1
Name: count, dtype: int64
```

Ввод [12]:
```
data["Color"] = data["Color"].astype('category')
data["Spectral_Class"] = data["Spectral_Class"].astype('category')
```

Ввод [13]:
```
1 data.dtypes
```

Out[13]:
```
Temperature          int64
L                  float64
R                  float64
A_M                float64
Color             category
Spectral_Class    category
Type                 int64
dtype: object
```

Ввод [14]:
```
1 data["Color_cat"] = data["Color"].cat.codes
2 data["Spectral_Class_cat"] = data["Spectral_Class"].cat.codes
3 data.head()
```

Out[14]:

	Temperature	L	R	A_M	Color	Spectral_Class	Type	Color_cat	Spectral_Class_cat
0	3068	0.002400	0.1700	16.12	Red	M	0	8	5
1	3042	0.000500	0.1542	16.60	Red	M	0	8	5
2	2600	0.000300	0.1020	18.70	Red	M	0	8	5
3	2800	0.000200	0.1600	16.65	Red	M	0	8	5
4	1939	0.000138	0.1030	20.06	Red	M	0	8	5

Ввод [15]:
```
1 data = data.drop(columns='Color')
2 data = data.drop(columns='Spectral_Class')
```

Ввод [16]:
```
1 data.head()
```

Out[16]:

	Temperature	L	R	A_M	Type	Color_cat	Spectral_Class_cat
0	3068	0.002400	0.1700	16.12	0	8	5
1	3042	0.000500	0.1542	16.60	0	8	5
2	2600	0.000300	0.1020	18.70	0	8	5
3	2800	0.000200	0.1600	16.65	0	8	5
4	1939	0.000138	0.1030	20.06	0	8	5

Ввод [17]:
```
1 data.dtypes
```

Out[17]:
```
Temperature            int64
L                    float64
R                    float64
A_M                  float64
Type                   int64
Color_cat               int8
Spectral_Class_cat      int8
dtype: object
```

Масштабирование данных

Ввод [18]:
```
1 from sklearn.preprocessing import MinMaxScaler
```

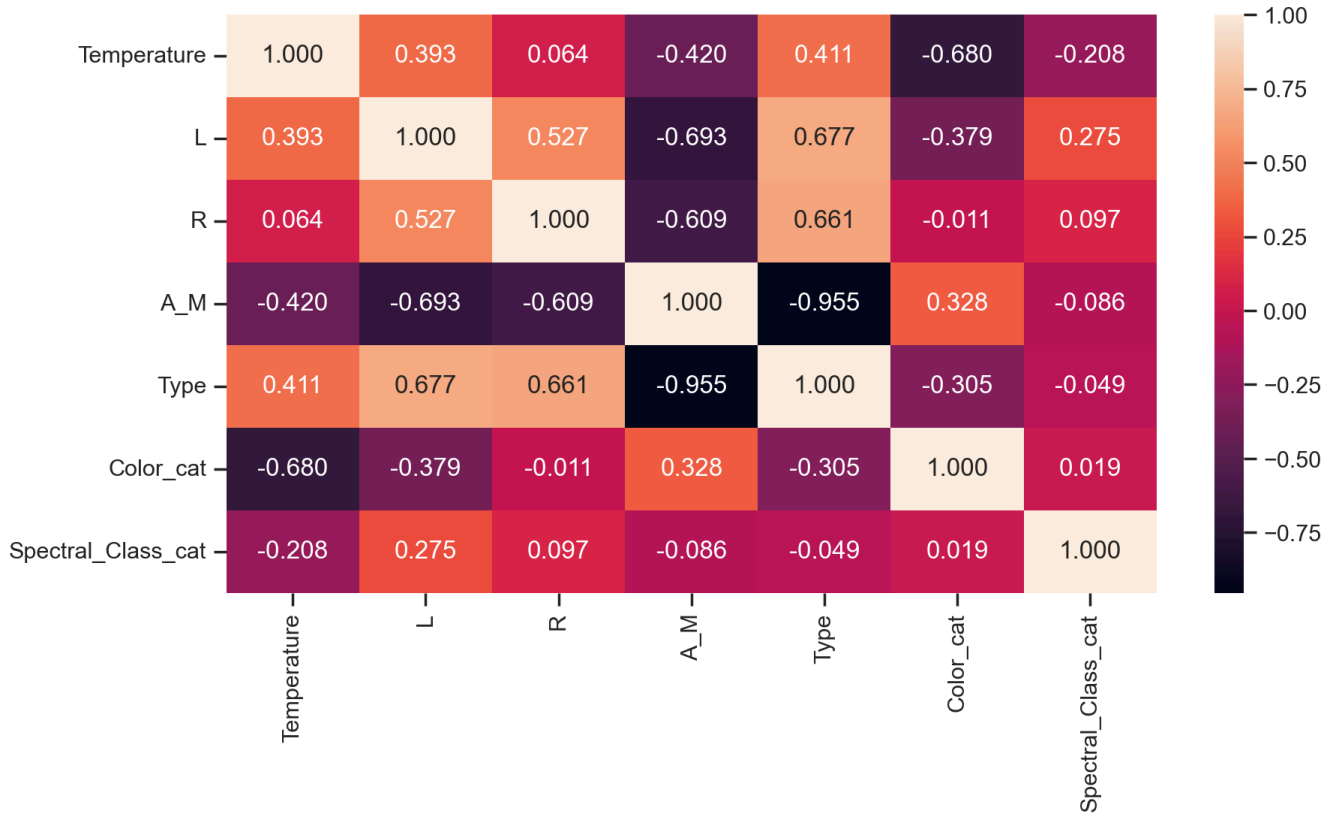
Ввод [19]:
```
1 sc1 = MinMaxScaler()
2 sc1_data = sc1.fit_transform(data)
3 sc1_data
```

Out[19]:
```
array([[2.96629095e-02, 2.73127546e-09, 8.29359490e-05, ...,
        0.00000000e+00, 5.00000000e-01, 8.33333333e-01],
       [2.89797956e-02, 4.94455040e-10, 7.48271124e-05, ...,
        0.00000000e+00, 5.00000000e-01, 8.33333333e-01],
       [1.73668585e-02, 2.59000259e-10, 4.80371586e-05, ...,
        0.00000000e+00, 5.00000000e-01, 8.33333333e-01],
       ...,
       [1.81025196e-01, 6.32776483e-01, 7.30304200e-01, ...,
        1.00000000e+00, 5.62500000e-01, 0.00000000e+00],
       [1.91692283e-01, 4.76725295e-01, 5.70693556e-01, ...,
        1.00000000e+00, 5.62500000e-01, 0.00000000e+00],
       [9.44352487e-01, 3.47181606e-01, 9.15062503e-01, ...,
        1.00000000e+00, 0.00000000e+00, 1.00000000e+00]])
```

Построим матрицу корреляции

Ввод [20]:
```
ig, ax = plt.subplots(figsize=(10,5))
sns.heatmap(data.corr(method='pearson'), ax=ax, annot=True, fmt='.3f')
```

Out[20]: <Axes: >



Выполнение прогноза целевого признака

Давайте выполним прогноз значения целевого признака L

Разделим выборку на обучающую и тестовую

Ввод [21]:
```
X = data.drop(columns='L')
Y = data['L']
```

Пример входных данных:

Ввод [22]:
```
X.head()
```

Out[22]:

	Temperature	R	A_M	Type	Color_cat	Spectral_Class_cat
0	3068	0.1700	16.12	0	8	5
1	3042	0.1542	16.60	0	8	5
2	2600	0.1020	18.70	0	8	5
3	2800	0.1600	16.65	0	8	5
4	1939	0.1030	20.06	0	8	5

Пример выходных данных:

Ввод [23]:
```
Y.head()
```

Out[23]:
```
0    0.002400
1    0.000500
2    0.000300
3    0.000200
4    0.000138
Name: L, dtype: float64
```

Ввод [24]:
```
X_train,  X_test,  Y_train,  Y_test = train_test_split(X,  Y, random_state = 2023, test_size = 0.1)
```

Характеристики обучающей выборки

Ввод [25]:
```
1 X_train.head()
```

Out[25]:

	Temperature	R	A_M	Type	Color_cat	Spectral_Class_cat
192	2994	0.28000	13.45	1	8	5
123	3146	0.09320	16.92	0	8	5
140	13420	0.00981	13.67	2	1	1
8	2650	0.11000	17.45	0	8	5
30	39000	10.60000	-4.70	3	0	6

Выходные параметры обучающей выборки

Ввод [26]:
```
1 Y_train.head()
```

Out[26]:
```
192         0.00720
123         0.00015
140         0.00059
8           0.00069
30     204000.00000
Name: L, dtype: float64
```

Выходные параметры тестовой выборки

Ввод [27]:
```
1 Y_test.head()
```

Out[27]:
```
42     150000.000000
205         0.001560
4           0.000138
120         0.000430
74          0.004000
Name: L, dtype: float64
```

Модель линейной регрессии

Метрики:
MSE - подчеркнуть большие ошибки
Median Absolute Error - оценить качество модели с устойчивостью к выбросам
R2 - точно и наглядно интерпретировать качество модели

Ввод [28]:
```
1 reg = LinearRegression().fit(X_train, Y_train)
```

Ввод [29]:
```
1 y_pred_test_reg = reg.predict(X_test)
2 y_pred_train_reg = reg.predict(X_train)
3 mse_reg = mean_squared_error(Y_train, y_pred_train_reg), mean_squared_error(Y_test, y_pred_test_reg)
4 mse_reg
```

Out[29]:
```
(13495193454.33473, 3325123742.294558)
```

Ввод [30]:
```
1 med_reg = median_absolute_error(Y_train, y_pred_train_reg), median_absolute_error(Y_test, y_pred_test_reg)
2 med_reg
```

Out[30]:
```
(36697.58446980561, 35483.77026547555)
```

Ввод [31]:
```
1 r2_reg = r2_score(Y_train, y_pred_train_reg), r2_score(Y_test, y_pred_test_reg)
2 r2_reg
```

Out[31]:
```
(0.6052590715631365, 0.6474841820679054)
```

Случайный лес

Ввод [32]:
```
1 rf = RandomForestRegressor(n_estimators=5, oob_score=True, random_state=1).fit(X_train, Y_train)
```

```
/Users/egorklesha/anaconda3/lib/python3.11/site-packages/sklearn/ensemble/_forest.py:615: UserWarning: Some i
nputs do not have OOB scores. This probably means too few trees were used to compute any reliable OOB estimat
es.
  warn(
```

Ввод [33]:
```
1 # Out-of-bag error
2 rf.oob_score_, 1-rf.oob_score_
```

Out[33]:
```
(0.36201812102914765, 0.6379818789708523)
```

Ввод [34]:
```
y_pred_test_rf = rf.predict(X_test)
y_pred_train_rf = rf.predict(X_train)
mse_rf = mean_squared_error(Y_train, y_pred_train_rf), mean_squared_error(Y_test, y_pred_test_rf)
mse_rf
```

Out[34]: (3062614212.827282, 10915695442.398329)

Ввод [35]:
```
med_rf = median_absolute_error(Y_train, y_pred_train_rf), median_absolute_error(Y_test, y_pred_test_rf)
med_rf
```

Out[35]: (0.003031, 0.004769000000000001)

Ввод [36]:
```
r2_rf = r2_score(Y_train, y_pred_train_rf), r2_score(Y_test, y_pred_test_rf)
r2_rf
```

Out[36]: (0.910417054641995, -0.15723672422468637)

Сравнение моделей

Ввод [37]:
```
print('MSE')
print('LinearRegression: ', mse_reg)
print('RandomForest:     ', mse_rf)
```

```
MSE
LinearRegression:  (13495193454.33473, 3325123742.294558)
RandomForest:      (3062614212.827282, 10915695442.398329)
```

Ввод [38]:
```
print('MedAE')
print('LinearRegression: ', med_reg)
print('RandomForest:     ', med_rf)
```

```
MedAE
LinearRegression:  (36697.58446980561, 35483.77026547555)
RandomForest:      (0.003031, 0.004769000000000001)
```

Ввод [39]:
```
print('R2')
print('LinearRegression: ', r2_reg)
print('RandomForest:     ', r2_rf)
```

```
R2
LinearRegression:  (0.6052590715631365, 0.6474841820679054)
RandomForest:      (0.910417054641995, -0.15723672422468637)
```

Вывод

На основании сравнения трех метрик - MSE, MedAE и R2, можно сделать вывод о том, что модель RandomForest демонстрирует более высокое качество по сравнению с моделью LinearRegression. Это указывает на большую эффективность RandomForest в прогнозировании. Метрика R2, отражающая степень соответствия модели данным, также подтверждает превосходство модели RandomForest.