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»

Отчёт по рубежному контролю №2

Вариант 11

Выполнила:
Студентка группы ИУ5-65Б
Е. И. Мащенко

2021

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.

Метод №1: Метод опорных векторов.
Метод №2: Градиентный бустинг.

Текст программы

Рубежный контроль №2

Мащенко Е.И.

ИУ5-65Б

метод №1 - Метод опорных векторов

метод №2 - Градиентный бустинг

Для рубежного контроля №2 взят следующий датасет: https://www.kaggle.com/brsdincer/star-type-classification . Будем решать задачу регрессии: type - целевой признак.

Импорт библиотек

In [76]:
```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, median_absolute_error, r2_score
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingRegressor
```

Подготовка

In [110…
```
# отбираем 500 строк из всего датасета
data = pd.read_csv('Stars.csv', nrows=500)
```

In [111…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 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   Color           240 non-null    object
 5   Spectral_Class  240 non-null    object
 6   Type            240 non-null    int64 
dtypes: float64(3), int64(2), object(2)
memory usage: 13.2+ KB
```

In [112…
```
data.head()
```

Out[112…

	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

In [114…
```
# Оцениваем баланс классов целевого признака
data['Type'].value_counts()/data['Type'].shape[0]*100
```

Out[114…
```
5    16.666667
4    16.666667
3    16.666667
2    16.666667
1    16.666667
0    16.666667
Name: Type, dtype: float64
```

In [115…
```
# Проверяем процент пропусков в данных для всех колонок
(data.isnull().sum()/data.shape[0]*100).sort_values(ascending=False)
```

Out[115…
```
Type              0.0
Spectral_Class    0.0
Color             0.0
A_M               0.0
R                 0.0
L                 0.0
Temperature       0.0
dtype: float64
```

In [116…
```
# Проверяем категориальные признаки на уникальность
col_obj = data.dtypes[data.dtypes==object].index.values.tolist()
for i in enumerate(col_obj):
    uniq_obj = data[i[1]].unique()
    print(f'{i[0]+1}. {i[1]}: {uniq_obj} | КОЛ-ВО: {len(uniq_obj)}')
```

```
1. Color: ['Red' 'Blue White' 'White' 'Yellowish White' 'Blue white'
 'Pale yellow orange' 'Blue' 'Blue-white' 'Whitish' 'yellow-white'
 'Orange' 'White-Yellow' 'white' 'yellowish' 'Yellowish' 'Orange-Red'
 'Blue-White'] | КОЛ-ВО: 17
2. Spectral_Class: ['M' 'B' 'A' 'F' 'O' 'K' 'G'] | КОЛ-ВО: 7
```

In [117…
```
# Оцениваем важность признаков для целевого
dataLE = data.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()['Type']*100).sort_values(ascending=False)
```

Out[117…
```
Type              100.000000
L                  67.684495
R                  66.097527
Temperature        41.112879
Spectral_Class     -4.913076
Color             -30.499279
A_M               -95.527558
Name: Type, dtype: float64
```

In [119…
```
#По результатам корреляционного анализа удаляем столбцы, которые имеют меньшую значимость по отношению к целевому признаку
del_data = (dataLE.corr()['Type']*100).sort_values(ascending=False)
del_col = del_data[(del_data < 10) & (del_data > -10) | (del_data.isnull())].index.values.tolist()
data.drop(columns=del_col, inplace=True)
dataLE.drop(columns=del_col, inplace=True)
```

In [120…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 6 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   Color        240 non-null    object
 5   Type         240 non-null    int64
dtypes: float64(3), int64(2), object(1)
memory usage: 11.4+ KB
```

In [121…
```
# Выполняем one-hot encoding и масштабирование для применения в SVM
col_num = data.dtypes[data.dtypes!=object].index.values.tolist()
col_num.remove('Type')
se = StandardScaler()
data[col_num] = se.fit_transform(data[col_num])
data = pd.get_dummies(data, drop_first=True)
```

In [122…
```
TEST_SIZE = 0.3
RANDOM_STATE = 0
```

In [123…
```
data_X = data.drop(columns='Type')
data_Y = data['Type']
```

In [124…
```
data_X_train, data_X_test, data_y_train, data_y_test = train_test_split(data_X, data_y, \
                                                                        test_size = TEST_SIZE, \
                                                                        random_state= RANDOM_STATE)
```

In [125…
```
def print_metrics(X_train, Y_train, X_test, Y_test, clf):
    clf.fit(X_train, Y_train)
    target = clf.predict(X_test)
    print(f'Средняя абсолютная ошибка: {mean_absolute_error(Y_test, target)}')
    print(f'Коэффициент детерминации: {r2_score(Y_test, target)}')
```

Метод опорных векторов

In [126…
```
print_metrics(data_X_train, data_y_train, data_X_test, data_y_test, SVC(random_state=RANDOM_STATE))
```

```
Средняя абсолютная ошибка: 0.041666666666666664
Коэффициент детерминации: 0.985065339141257
```

Градиентный бустинг

In [127…
```
print_metrics(data_X_train, data_y_train, data_X_test, data_y_test, GradientBoostingRegressor(random_state=RANDOM_STATE))
```

```
Средняя абсолютная ошибка: 0.015657536165973273
Коэффициент детерминации: 0.9937047298132438
```

Вывод

В данной работе для оценки моделей были использованы следующие метрики, подходящие для задачи регрессии:

1) Mean absolute error - средняя абсолютная ошибка 2) Метрика R2 или коэффициент детерминации

По результатам оценивания можно сделать следующий вывод: модель GradientBoostingRegressor по обеим метрикам обладает немного большей предсказательной способностью, чем Support Vector Machine.