

Зубарева Антонина ИУ5-65Б Вариант 7

Тема: Методы построения моделей машинного обучения.

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 - Метод опорных векторов Метод 2 - Градиентный бустинг

Ввод [29]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn import preprocessing
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_lo
from sklearn.metrics import roc_curve, roc_auc_score

from sklearn.ensemble import GradientBoostingRegressor
from sklearn.impute import SimpleImputer
```

Ввод [2]:

```
data = pd.read_csv('./datasets/Admission_Predict_Ver1.1.csv', sep=",")
TARGET_COL_NAME = 'Chance of Admit '
TARGET_IS_NUMERIC = data[TARGET_COL_NAME].dtype != 'O'
TARGET_IS_NUMERIC
```

Out[2]:

True

Ввод [3]:

```
data
```

Out[3]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 9 columns

Обработка пропусков

Ввод [7]:

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

Out[7]:

```
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
```

Пропусков нет

Кодирование строковых признаков (LabelEncoding)

Ввод [8]:

```
not_number_cols = data.select_dtypes(include=['object'])
number_cols = data.select_dtypes(exclude=['object'])
```

Ввод [9]:

```
le = preprocessing.LabelEncoder()

for col_name in not_number_cols:
    data[col_name] = le.fit_transform(data[col_name])

data
```

Out[9]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 9 columns

Масштабируем числовые данные

Масштабирование признаков важно для работы машины опорных векторов. Если признаки представлены в различных масштабах, то это может отрицательно повлиять на сходимость градиентного спуска.

Ввод [10]:

```
scaler = preprocessing.MinMaxScaler()

number_fields_source = number_cols.loc[:, number_cols.columns!=TARGET_COL_NAME] if T

for col_name in number_fields_source:
    data[col_name] = scaler.fit_transform(data[[col_name]])

data
```

Out[10]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	0.000000	0.94	0.928571	0.75	0.875	0.875	0.913462	1.0	0.92
1	0.002004	0.68	0.535714	0.75	0.750	0.875	0.663462	1.0	0.76
2	0.004008	0.52	0.428571	0.50	0.500	0.625	0.384615	1.0	0.72
3	0.006012	0.64	0.642857	0.50	0.625	0.375	0.599359	1.0	0.80
4	0.008016	0.48	0.392857	0.25	0.250	0.500	0.451923	0.0	0.65
...
495	0.991984	0.84	0.571429	1.00	0.875	0.750	0.711538	1.0	0.87
496	0.993988	0.94	0.892857	1.00	1.000	1.000	0.983974	1.0	0.96
497	0.995992	0.80	1.000000	1.00	0.875	1.000	0.884615	1.0	0.93
498	0.997996	0.44	0.392857	0.75	0.750	1.000	0.522436	0.0	0.73
499	1.000000	0.74	0.750000	0.75	0.875	0.875	0.717949	0.0	0.84

500 rows × 9 columns

Разделяем на обучающую и тестовую выборки

Ввод [11]:

```
target = data[TARGET_COL_NAME]
data_X_train, data_X_test, data_y_train, data_y_test = train_test_split(
    data, target, test_size=0.2, random_state=1)
```

Ввод [12]:

```
data_X_train.shape, data_X_test.shape
```

Out[12]:

```
((400, 9), (100, 9))
```

Ввод [13]:

```
data_y_train.shape, data_y_test.shape
```

Out[13]:

```
((400,), (100,))
```

Ввод [14]:

```
np.unique(target)
```

Out[14]:

```
array([0.34, 0.36, 0.37, 0.38, 0.39, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47,
       0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58,
       0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69,
       0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ,
       0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 , 0.91,
       0.92, 0.93, 0.94, 0.95, 0.96, 0.97])
```

Метод 1 - Метод опорных векторов

Ввод [18]:

```
svr_1 = svm.SVR()
svr_1.fit(data_X_train, data_y_train)
```

Out[18]:

```
SVR()
```

Ввод [20]:

```
data_y_pred_1 = svr_1.predict(data_X_test)
mean_absolute_error(data_y_test, data_y_pred_1), mean_squared_error(data_y_test, data_y_pred_1)
```

Out[20]:

```
(0.045166233388262735, 0.0029216901491620384)
```

Ввод [36]:

```
median_absolute_error(data_y_test, data_y_pred_1)
```

Out[36]:

```
0.0055470085470084585
```

Ввод [37]:

```
r2_score(data_y_test, data_y_pred_1)
```

Out[37]:

```
0.996508402169581
```

Метод 2 - Градиентный бустинг

Ввод [41]:

```
gbr = GradientBoostingRegressor()  
gbr.fit(data_X_train, data_y_train)  
data_y_pred_1_0 = gbr.predict(data_X_test)  
mean_absolute_error(data_y_test, data_y_pred_1_0), mean_squared_error(data_y_test, c
```

Out[41]:

```
(0.00027525181235648846, 2.0736465275015988e-06)
```

Ввод [38]:

```
median_absolute_error(data_y_test, data_y_pred_1_0)
```

Out[38]:

```
1.9169502148141948e-05
```

Ввод [40]:

```
r2_score(data_y_test, data_y_pred_1_0)
```

Out[40]:

```
0.9998942478613917
```

Выводы

Исходя из полученных выше метрик можно сделать вывод, что в данном случае использование градиентного бустинга дает наилучший результат (mean_squared_error, mean_absolute_error, median_absolute_error ближе к нулю, чем при использовании метода опорных векторов, а R2 коэффициент выше и ближе к 100%)

Ввод []: