Нырков Илья ИУ5-62Б

Вариант 14

Датасет: https://www.kaggle.com/datasets/noriuk/us-education-datasets-unification-project?resource=download

набор данных о студентах американских образовательных учреждениях

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.

Задание по группам

Гру
ИУ5

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn import svm
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_absolute_error,
mean_absolute_percentage_error, mean_squared_error
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix, classification_report
```

```
data = pd.read_csv("drive/MyDrive/Colab Notebooks/states_all.csv",
sep=",")
```

```
data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1715 entries, 0 to 1714
Data columns (total 25 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   PRIMARY_KEY                   1715 non-null   object 
 1   STATE                         1715 non-null   object 
 2   YEAR                          1715 non-null   int64  
 3   ENROLL                        1224 non-null   float64
 4   TOTAL_REVENUE                 1275 non-null   float64
 5   FEDERAL_REVENUE               1275 non-null   float64
 6   STATE_REVENUE                 1275 non-null   float64
 7   LOCAL_REVENUE                 1275 non-null   float64
 8   TOTAL_EXPENDITURE             1275 non-null   float64
 9   INSTRUCTION_EXPENDITURE       1275 non-null   float64
 10  SUPPORT_SERVICES_EXPENDITURE  1275 non-null   float64
 11  OTHER_EXPENDITURE             1224 non-null   float64
 12  CAPITAL_OUTLAY_EXPENDITURE    1275 non-null   float64
 13  GRADES_PK_G                   1542 non-null   float64
 14  GRADES_KG_G                   1632 non-null   float64
 15  GRADES_4_G                    1632 non-null   float64
 16  GRADES_8_G                    1632 non-null   float64
 17  GRADES_12_G                   1632 non-null   float64
 18  GRADES_1_8_G                  1020 non-null   float64
 19  GRADES_9_12_G                 1071 non-null   float64
 20  GRADES_ALL_G                  1632 non-null   float64
 21  AVG_MATH_4_SCORE              565 non-null    float64
 22  AVG_MATH_8_SCORE              602 non-null    float64
 23  AVG_READING_4_SCORE           650 non-null    float64
 24  AVG_READING_8_SCORE           562 non-null    float64
dtypes: float64(22), int64(1), object(2)
memory usage: 335.1+ KB

       PRIMARY_KEY       STATE  YEAR  ENROLL  TOTAL_REVENUE
FEDERAL_REVENUE  \
0     1992_ALABAMA     ALABAMA  1992     NaN      2678885.0
304177.0
1      1992_ALASKA      ALASKA  1992     NaN      1049591.0
106780.0
2     1992_ARIZONA     ARIZONA  1992     NaN      3258079.0
297888.0
3    1992_ARKANSAS    ARKANSAS  1992     NaN      1711959.0
178571.0
4  1992_CALIFORNIA  CALIFORNIA  1992     NaN     26260025.0
2072470.0

   STATE_REVENUE  LOCAL_REVENUE  TOTAL_EXPENDITURE
```

```
INSTRUCTION_EXPENDITURE  \
0      1659028.0       715680.0          2653798.0
1481703.0
1       720711.0       222100.0           972488.0
498362.0
2      1369815.0      1590376.0          3401580.0
1435908.0
3       958785.0       574603.0          1743022.0
964323.0
4     16546514.0      7641041.0         27138832.0
14358922.0

   ...  GRADES_4_G  GRADES_8_G  GRADES_12_G  GRADES_1_8_G
GRADES_9_12_G  \
0  ...     57948.0     58025.0      41167.0           NaN
NaN
1  ...      9748.0      8789.0       6714.0           NaN
NaN
2  ...     55433.0     49081.0      37410.0           NaN
NaN
3  ...     34632.0     36011.0      27651.0           NaN
NaN
4  ...    418418.0    363296.0     270675.0           NaN
NaN

   GRADES_ALL_G  AVG_MATH_4_SCORE  AVG_MATH_8_SCORE
AVG_READING_4_SCORE  \
0      731634.0             208.0             252.0
207.0
1      122487.0               NaN               NaN
NaN
2      673477.0             215.0             265.0
209.0
3      441490.0             210.0             256.0
211.0
4     5254844.0             208.0             261.0
202.0

   AVG_READING_8_SCORE
0                  NaN
1                  NaN
2                  NaN
3                  NaN
4                  NaN

[5 rows x 25 columns]
```

```
(data.isnull() | data.empty | data.isna()).sum()
```

```
PRIMARY_KEY                        0
STATE                              0
YEAR                               0
ENROLL                           491
TOTAL_REVENUE                    440
FEDERAL_REVENUE                  440
STATE_REVENUE                    440
LOCAL_REVENUE                    440
TOTAL_EXPENDITURE                440
INSTRUCTION_EXPENDITURE          440
SUPPORT_SERVICES_EXPENDITURE     440
OTHER_EXPENDITURE                491
CAPITAL_OUTLAY_EXPENDITURE       440
GRADES_PK_G                      173
GRADES_KG_G                       83
GRADES_4_G                        83
GRADES_8_G                        83
GRADES_12_G                       83
GRADES_1_8_G                     695
GRADES_9_12_G                    644
GRADES_ALL_G                      83
AVG_MATH_4_SCORE                1150
AVG_MATH_8_SCORE                1113
AVG_READING_4_SCORE             1065
AVG_READING_8_SCORE             1153
dtype: int64
```

##Процент пропусков в каждом признаке

```
target_column = 'AVG_READING_4_SCORE'
data = data[(data[target_column].isna() == False)]


(data.isnull() | data.empty | data.isna()).sum()
```

```
PRIMARY_KEY                       0
STATE                             0
YEAR                              0
ENROLL                          169
TOTAL_REVENUE                   127
FEDERAL_REVENUE                 127
STATE_REVENUE                   127
LOCAL_REVENUE                   127
TOTAL_EXPENDITURE               127
INSTRUCTION_EXPENDITURE         127
SUPPORT_SERVICES_EXPENDITURE    127
OTHER_EXPENDITURE               169
CAPITAL_OUTLAY_EXPENDITURE      127
GRADES_PK_G                      81
GRADES_KG_G                      76
GRADES_4_G                       76
```

```
GRADES_8_G                      76
GRADES_12_G                     76
GRADES_1_8_G                   209
GRADES_9_12_G                  158
GRADES_ALL_G                    76
AVG_MATH_4_SCORE               129
AVG_MATH_8_SCORE               129
AVG_READING_4_SCORE              0
AVG_READING_8_SCORE             88
dtype: int64
```

```
row_count = data.shape[0]
for key, elem in (data.isnull() | data.empty |
data.isna()).sum().items():
  print(key, "{:.2f}".format(elem / row_count * 100) + "%" )
```

```
PRIMARY_KEY 0.00%
STATE 0.00%
YEAR 0.00%
ENROLL 26.00%
TOTAL_REVENUE 19.54%
FEDERAL_REVENUE 19.54%
STATE_REVENUE 19.54%
LOCAL_REVENUE 19.54%
TOTAL_EXPENDITURE 19.54%
INSTRUCTION_EXPENDITURE 19.54%
SUPPORT_SERVICES_EXPENDITURE 19.54%
OTHER_EXPENDITURE 26.00%
CAPITAL_OUTLAY_EXPENDITURE 19.54%
GRADES_PK_G 12.46%
GRADES_KG_G 11.69%
GRADES_4_G 11.69%
GRADES_8_G 11.69%
GRADES_12_G 11.69%
GRADES_1_8_G 32.15%
GRADES_9_12_G 24.31%
GRADES_ALL_G 11.69%
AVG_MATH_4_SCORE 19.85%
AVG_MATH_8_SCORE 19.85%
AVG_READING_4_SCORE 0.00%
AVG_READING_8_SCORE 13.54%
```

###Заполнять пропуски в признаках, с процентом пустых значений более 30 имеет мало смысла, поэтому составим список признаков, которые нужно удалить.

```
row_count = data.shape[0]
columns_to_remove = []
for key, elem in (data.isnull() | data.empty |
data.isna()).sum().items():
  if elem / row_count >= 0.3:
```

```
    columns_to_remove.append(key)
columns_to_remove
```

```
['GRADES_1_8_G']
```

```
data_new = data.drop(columns_to_remove, axis=1)
data_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 650 entries, 0 to 1714
Data columns (total 24 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   PRIMARY_KEY                   650 non-null    object
 1   STATE                         650 non-null    object
 2   YEAR                          650 non-null    int64
 3   ENROLL                        481 non-null    float64
 4   TOTAL_REVENUE                 523 non-null    float64
 5   FEDERAL_REVENUE               523 non-null    float64
 6   STATE_REVENUE                 523 non-null    float64
 7   LOCAL_REVENUE                 523 non-null    float64
 8   TOTAL_EXPENDITURE             523 non-null    float64
 9   INSTRUCTION_EXPENDITURE       523 non-null    float64
 10  SUPPORT_SERVICES_EXPENDITURE  523 non-null    float64
 11  OTHER_EXPENDITURE             481 non-null    float64
 12  CAPITAL_OUTLAY_EXPENDITURE    523 non-null    float64
 13  GRADES_PK_G                   569 non-null    float64
 14  GRADES_KG_G                   574 non-null    float64
 15  GRADES_4_G                    574 non-null    float64
 16  GRADES_8_G                    574 non-null    float64
 17  GRADES_12_G                   574 non-null    float64
 18  GRADES_9_12_G                 492 non-null    float64
 19  GRADES_ALL_G                  574 non-null    float64
 20  AVG_MATH_4_SCORE              521 non-null    float64
 21  AVG_MATH_8_SCORE              521 non-null    float64
 22  AVG_READING_4_SCORE           650 non-null    float64
 23  AVG_READING_8_SCORE           562 non-null    float64
dtypes: float64(21), int64(1), object(2)
memory usage: 127.0+ KB
```

Обработка пропусков данных

Посмотрим средние значения в каждом столбце с пропусками

```
for column in data_new:
  if (data_new[column].isnull() | data_new[column].empty |
data_new[column].isna()).sum() != 0:
    if data_new[column].dtype != 'object':
      print(column + ":", data_new[column].describe()[['mean']][0])
    else:
      print(column + ":", data_new[column].describe()[['top']][0])
```

```
ENROLL: 934450.4885654886
TOTAL_REVENUE: 9610588.969407266
FEDERAL_REVENUE: 840781.4225621414
STATE_REVENUE: 4467784.89292543
LOCAL_REVENUE: 4302022.653919694
TOTAL_EXPENDITURE: 9730734.829827916
INSTRUCTION_EXPENDITURE: 5039264.764818355
SUPPORT_SERVICES_EXPENDITURE: 2834123.634799235
OTHER_EXPENDITURE: 461512.7006237006
CAPITAL_OUTLAY_EXPENDITURE: 954404.3078393881
GRADES_PK_G: 21156.2460456942
GRADES_KG_G: 71505.45296167248
GRADES_4_G: 72367.06794425087
GRADES_8_G: 72312.85365853658
GRADES_12_G: 62889.35017421603
GRADES_9_12_G: 286875.8699186992
GRADES_ALL_G: 961249.6393728222
AVG_MATH_4_SCORE: 237.39539347408828
AVG_MATH_8_SCORE: 279.9846449136276
AVG_READING_8_SCORE: 263.55871886120997
```

Заполним пропуски

```
for column in data_new:
  if (data_new[column].isnull() | data_new[column].empty |
data_new[column].isna()).sum() != 0:
    if data_new[column].dtype != 'object' and column != target_column:
      data_new[column].fillna(data_new[column].median(), inplace=True)
```

```
(data_new.isnull() | data_new.empty | data_new.isna()).sum()
```

```
PRIMARY_KEY                     0
STATE                           0
YEAR                            0
ENROLL                          0
TOTAL_REVENUE                   0
FEDERAL_REVENUE                 0
STATE_REVENUE                   0
LOCAL_REVENUE                   0
TOTAL_EXPENDITURE               0
INSTRUCTION_EXPENDITURE         0
SUPPORT_SERVICES_EXPENDITURE    0
OTHER_EXPENDITURE               0
CAPITAL_OUTLAY_EXPENDITURE      0
GRADES_PK_G                     0
GRADES_KG_G                     0
GRADES_4_G                      0
GRADES_8_G                      0
GRADES_12_G                     0
GRADES_9_12_G                   0
GRADES_ALL_G                    0
```

```
AVG_MATH_4_SCORE                0
AVG_MATH_8_SCORE                0
AVG_READING_4_SCORE             0
AVG_READING_8_SCORE             0
dtype: int64
```

```python
data_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 650 entries, 0 to 1714
Data columns (total 24 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   PRIMARY_KEY                   650 non-null    object
 1   STATE                         650 non-null    object
 2   YEAR                          650 non-null    int64
 3   ENROLL                        650 non-null    float64
 4   TOTAL_REVENUE                 650 non-null    float64
 5   FEDERAL_REVENUE               650 non-null    float64
 6   STATE_REVENUE                 650 non-null    float64
 7   LOCAL_REVENUE                 650 non-null    float64
 8   TOTAL_EXPENDITURE             650 non-null    float64
 9   INSTRUCTION_EXPENDITURE       650 non-null    float64
 10  SUPPORT_SERVICES_EXPENDITURE  650 non-null    float64
 11  OTHER_EXPENDITURE             650 non-null    float64
 12  CAPITAL_OUTLAY_EXPENDITURE    650 non-null    float64
 13  GRADES_PK_G                   650 non-null    float64
 14  GRADES_KG_G                   650 non-null    float64
 15  GRADES_4_G                    650 non-null    float64
 16  GRADES_8_G                    650 non-null    float64
 17  GRADES_12_G                   650 non-null    float64
 18  GRADES_9_12_G                 650 non-null    float64
 19  GRADES_ALL_G                  650 non-null    float64
 20  AVG_MATH_4_SCORE              650 non-null    float64
 21  AVG_MATH_8_SCORE              650 non-null    float64
 22  AVG_READING_4_SCORE           650 non-null    float64
 23  AVG_READING_8_SCORE           650 non-null    float64
dtypes: float64(21), int64(1), object(2)
memory usage: 127.0+ KB
```

```python
category_cols = ['PRIMARY_KEY', 'STATE']
print("Количество уникальных значений\n")
for col in category_cols:
    print(f'{col}: {data_new[col].unique().size}')
```

```
Количество уникальных значений

PRIMARY_KEY: 650
STATE: 53
```

```python
data_new = data_new.drop(category_cols, axis=1)
```

```
X_data = data_new[['YEAR',
 'ENROLL',
 'FEDERAL_REVENUE',
 'STATE_REVENUE',
 'LOCAL_REVENUE',
 'INSTRUCTION_EXPENDITURE',
 'SUPPORT_SERVICES_EXPENDITURE',
 'OTHER_EXPENDITURE',
 'CAPITAL_OUTLAY_EXPENDITURE',
 'GRADES_ALL_G']]
Y_data = data_new[target_column].to_list()
```

```
X_data['YEAR'] = data_new['YEAR'] - data_new['YEAR'].mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```
X_data = preprocessing.normalize(X_data,axis = 0)
```

```
X_data.shape
```

```
(650, 10)
```

```
X_train, X_test, y_train, y_test = train_test_split(
     X_data, Y_data, test_size=0.3, random_state=42)
```

```
data_new.isnull().sum()
```

```
YEAR                            0
ENROLL                          0
TOTAL_REVENUE                   0
FEDERAL_REVENUE                 0
STATE_REVENUE                   0
LOCAL_REVENUE                   0
TOTAL_EXPENDITURE               0
INSTRUCTION_EXPENDITURE         0
SUPPORT_SERVICES_EXPENDITURE    0
OTHER_EXPENDITURE               0
CAPITAL_OUTLAY_EXPENDITURE      0
GRADES_PK_G                     0
GRADES_KG_G                     0
GRADES_4_G                      0
GRADES_8_G                      0
GRADES_12_G                     0
GRADES_9_12_G                   0
```

```
GRADES_ALL_G                0
AVG_MATH_4_SCORE            0
AVG_MATH_8_SCORE            0
AVG_READING_4_SCORE         0
AVG_READING_8_SCORE         0
dtype: int64
```

SVC

Выбрал MAE, MSE и MAPE для оценки качества, так как это самые показательные метрики для регрессии

```
svm_model = SVC()
mape = -
cross_val_score(svm_model,X_train,y_train,cv=4,scoring='neg_mean_absol
ute_percentage_error').mean()
mae = -
cross_val_score(svm_model,X_train,y_train,cv=4,scoring='neg_mean_absol
ute_error').mean()
mse = -
cross_val_score(svm_model,X_train,y_train,cv=4,scoring='neg_mean_squar
ed_error').mean()
print('SVM Errors')
print('MAE:' + str(round(mae,3)) + ' MAPE:' + str(round(mape,3)) + '
MSE:' + str(round(mse,3)))
```

```
The least populated class in y has only 1 members, which is less than
n_splits=4.
The least populated class in y has only 1 members, which is less than
n_splits=4.
The least populated class in y has only 1 members, which is less than
n_splits=4.
```

```
SVM Errors
MAE:5.89 MAPE:0.028 MSE:65.313
```

```
svm_model.fit(X_train,y_train)
mae = mean_absolute_error(y_test,svm_model.predict(X_test))
mape =
mean_absolute_percentage_error(y_test,svm_model.predict(X_test))
mse = mean_squared_error(y_test,svm_model.predict(X_test))
print('MAE:' + str(round(mae,3)) + ' MAPE:' + str(round(mape,3)) + '
MSE:' + str(round(mse,3)))
```

```
MAE:5.462 MAPE:0.026 MSE:63.851
```

RandomForestClassifier

```
rfc_model = RandomForestClassifier()
mape = -
cross_val_score(rfc_model,X_train,y_train,cv=2,scoring='neg_mean_absol
```

```
ute_percentage_error').mean()
mae = -
cross_val_score(rfc_model,X_train,y_train,cv=2,scoring='neg_mean_absol
ute_error').mean()
mse = -
cross_val_score(rfc_model,X_train,y_train,cv=2,scoring='neg_mean_squar
ed_error').mean()
print('SVM Errors')
print('MAE:' + str(round(mae,3)) + ' MAPE:' + str(round(mape,3)) + '
MSE:' + str(round(mse,3)))
```

```
The least populated class in y has only 1 members, which is less than
n_splits=2.
The least populated class in y has only 1 members, which is less than
n_splits=2.
The least populated class in y has only 1 members, which is less than
n_splits=2.
```

```
SVM Errors
MAE:4.264 MAPE:0.019 MSE:37.903
```

```
rfc_model.fit(X_train,y_train)
mae = mean_absolute_error(y_test,rfc_model.predict(X_test))
mape =
mean_absolute_percentage_error(y_test,rfc_model.predict(X_test))
mse = mean_squared_error(y_test,rfc_model.predict(X_test))
print('MAE:' + str(round(mae,3)) + ' MAPE:' + str(round(mape,3)) + '
MSE:' + str(round(mse,3)))
```

```
MAE:3.851 MAPE:0.018 MSE:34.333
```

Можно сделать вывод о том, что обе модели очень хорошо показали себя на тестовых и тренировочных данных. Даже при условии что значений было немного.