Родионов Д.А.

ИУ5-65Б Вариант №12

Импортируем библиотеки:

```
import numpy as np
import pandas as pd
from·typing·import·Dict,·Tuple
from·scipy·import stats
from IPython.display import Image
from sklearn.datasets import load_iris, load_boston
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_erro
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.datasets import make_blobs, make_circles
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import AdaBoostClassifier
from sklearn import svm
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

```
from google.colab import drive
drive.mount("/content/gdrive")
data = pd.read_csv('/content/gdrive/My Drive/occupancy_data/dc-wikia-data.csv', sep=",")
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive
```

```
data
```

Обработка пропусков

```
data.keys().to_list()
```

```
['page_id',
 'name',
 'urlslug',
 'ID',
 'ALIGN',
 'EYE',
 'HAIR',
 'SEX',
 'GSM',
 'ALIVE',
 'APPEARANCES',
 'FIRST APPEARANCE',
 'YEAR']
```

```
data.isnull().sum()
```

```
page_id                0
name                   0
urlslug                0
ID                  2013
ALIGN                601
EYE                 3628
HAIR                2274
SEX                  125
GSM                 6832
```

```
ALIVE                 3
APPEARANCES         355
FIRST APPEARANCE     69
YEAR                 69
dtype: int64
```

```
data.shape
```

```
(6896, 13)
```

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

```
Всего строк: 6896
```

```
data = data.dropna(axis=0, how='any')
data.shape
```

```
(38, 13)
```

```
data.head()
```

Кодирование категориальных признаков

Удалим колонки, которые не влияют на целевой признак:

```
data = data.drop(columns='name')
data = data.drop(columns='urlslug')
data = data.drop(columns='FIRST APPEARANCE')
```

```
data.shape
```

```
(38, 10)
```

```
data.head()
```

```
data.dtypes
```

```
page_id          int64
ID              object
ALIGN           object
EYE             object
HAIR            object
SEX             object
GSM             object
ALIVE           object
APPEARANCES    float64
YEAR           float64
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
df_int = le.fit_transform(data['ID'])
data['ID'] = df_int
df_int = le.fit_transform(data['ALIGN'])
data['ALIGN'] = df_int
df_int = le.fit_transform(data['EYE'])
data['EYE'] = df_int
df_int = le.fit_transform(data['HAIR'])
data['HAIR'] = df_int
df_int = le.fit_transform(data['SEX'])
data['SEX'] = df_int
df_int = le.fit_transform(data['GSM'])
data['GSM'] = df_int
df_int = le.fit_transform(data['ALIVE'])
data['ALIVE'] = df_int
data.head()
```

```
sc1 = MinMaxScaler()
data['ID'] = sc1.fit_transform(data[['ID']])
data['ALIGN'] = sc1.fit_transform(data[['ALIGN']])
data['EYE'] = sc1.fit_transform(data[['EYE']])
data['HAIR'] = sc1.fit_transform(data[['HAIR']])
data['SEX'] = sc1.fit_transform(data[['SEX']])
data['GSM'] = sc1.fit_transform(data[['GSM']])
data['ALIVE'] = sc1.fit_transform(data[['ALIVE']])
data.head()
```

Разделение на обучающую и тестовую выборки

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

```
data_X_train.shape, data_y_train.shape
```

```
     ((30, 10), (30,))
```

```
data_X_test.shape, data_y_test.shape
```

```
     ((8, 10), (8,))
```

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

```
     array([0., 1.])
```

Метод опорных векторов

```
svr_1 = LinearSVC()
```

```
svr_1.fit(data_X_train, data_y_train)
```

```
    Liblinear failed to converge, increase the number of iterations.
    LinearSVC()
```

```
data_y_pred_1 = svr_1.predict(data_X_test)
```

```
accuracy_score(data_y_test, data_y_pred_1)
```

```
    0.875
```

```
f1_score(data_y_test, data_y_pred_1, average='micro')
```

```
    0.875
```

```
f1_score(data_y_test, data_y_pred_1, average='macro')
```

```
    0.4666666666666667
```

```
f1_score(data_y_test, data_y_pred_1, average='weighted')
```

```
    0.8166666666666667
```

```
svr_2 = LinearSVC(C=1.0, max_iter=10000)
svr_2.fit(data_X_train, data_y_train)
```

```
    Liblinear failed to converge, increase the number of iterations.
    LinearSVC(max_iter=10000)
```

```
data_y_pred_2 = svr_2.predict(data_X_test)
```

```
accuracy_score(data_y_test, data_y_pred_2)
```

```
    0.875
```

```
f1_score(data_y_test, data_y_pred_2, average='micro')
```

```
    0.875
```

```
f1_score(data_y_test, data_y_pred_2, average='macro')
```

```
    0.4666666666666667
```

```
f1_score(data_y_test, data_y_pred_2, average='weighted')
```

```
    0.8166666666666667
```

```
svr_3 = LinearSVC(C=1.0, penalty='l1', dual=False, max_iter=10000)
```

```
svr_3.fit(data_X_train, data_y_train)
```

```
    LinearSVC(dual=False, max_iter=10000, penalty='l1')
```

```
data_y_pred_3_0 = svr_3.predict(data_X_train)
accuracy_score(data_y_train, data_y_pred_3_0)
```

```
    1.0
```

```
data_y_pred_3 = svr_3.predict(data_X_test)
accuracy_score(data_y_test, data_y_pred_3)
```

```
    1.0
```

```
f1_score(data_y_test, data_y_pred_3, average='micro')
```

```
    1.0
```

```
f1_score(data_y_test, data_y_pred_3, average='macro')
```

```
    1.0
```

```
f1_score(data_y_test, data_y_pred_3, average='weighted')
```

```
    1.0
```

Градиентный бустинг

```
ab1 = AdaBoostClassifier()
ab1.fit(data_X_train, data_y_train)
```

```
    AdaBoostClassifier()
```

```
data_y_pred_1 = ab1.predict(data_X_test)
```

```
data_y_pred_1_0 = ab1.predict(data_X_train)
```

```
accuracy_score(data_y_train, data_y_pred_1_0)
```

```
    1.0
```

```
accuracy_score(data_y_test, data_y_pred_1)
```

```
    1.0
```

```
f1_score(data_y_test, data_y_pred_1, average='micro')
```

```
1.0
```

```
f1_score(data_y_test, data_y_pred_1, average='macro')
```

```
1.0
```

```
f1_score(data_y_test, data_y_pred_1, average='weighted')
```

```
1.0
```

Градиентный бустинг показал лучше качество, чем метод опорных векторов

✓ 0 сек. выполнено в 21:57 ● ✕