Кузьмин Роман ИУ5-64Б Вариант №12

[*]:
```
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_error, median_absolute_error, r2_score
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.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
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")
```

[2]:
```
data = pd.read_csv('marvel-wikia-data.csv', sep=",")
```

[3]:
```
data.head()
```

[3]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	GSM	ALIVE	APPEARANCES	FIRST APPEARANCE	Year
0	1678	Spider-Man (Peter Parker)	\/Spider-Man_(Peter_Parker)	Secret Identity	Good Characters	Hazel Eyes	Brown Hair	Male Characters	NaN	Living Characters	4043.0	Aug-62	1962.0
1	7139	Captain America (Steven Rogers)	\/Captain_America_(Steven_Rogers)	Public Identity	Good Characters	Blue Eyes	White Hair	Male Characters	NaN	Living Characters	3360.0	Mar-41	1941.0
2	64786	Wolverine (James \"Logan\" Howlett)	\/Wolverine_(James_%22Logan%22_Howlett)	Public Identity	Neutral Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	3061.0	Oct-74	1974.0
3	1868	Iron Man (Anthony \"Tony\" Stark)	\/Iron_Man_(Anthony_%22Tony%22_Stark)	Public Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	2961.0	Mar-63	1963.0
4	2460	Thor (Thor Odinson)	\/Thor_(Thor_Odinson)	No Dual Identity	Good Characters	Blue Eyes	Blond Hair	Male Characters	NaN	Living Characters	2258.0	Nov-50	1950.0

[4]:
```
data.isnull().sum()
```

[4]:
```
page_id                0
name                   0
urlslug                0
ID                  3770
ALIGN               2812
EYE                 9767
HAIR                4264
SEX                  854
GSM                16286
ALIVE                  3
APPEARANCES         1096
FIRST APPEARANCE     815
Year                 815
dtype: int64
```

[5]:
```
data.shape
```

[5]:
```
(16376, 13)
```

[6]:
```
data.pop('GSM')
```

[6]:
```
0        NaN
1        NaN
2        NaN
3        NaN
4        NaN
        ...
16371    NaN
16372    NaN
16373    NaN
16374    NaN
16375    NaN
Name: GSM, Length: 16376, dtype: object
```

[7]:
```
data.shape
```

[7]:
```
(16376, 12)
```

[8]:
```
data = data.dropna(axis=0, how='any')
data.shape
```

[8]:
```
(4402, 12)
```

[9]:
```
data.head()
```

[9]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	ALIVE	APPEARANCES	FIRST APPEARANCE	Year
0	1678	Spider-Man (Peter Parker)	\/Spider-Man_(Peter_Parker)	Secret Identity	Good Characters	Hazel Eyes	Brown Hair	Male Characters	Living Characters	4043.0	Aug-62	1962.0
1	7139	Captain America (Steven Rogers)	\/Captain_America_(Steven_Rogers)	Public Identity	Good Characters	Blue Eyes	White Hair	Male Characters	Living Characters	3360.0	Mar-41	1941.0
2	64786	Wolverine (James \"Logan\" Howlett)	\/Wolverine_(James_%22Logan%22_Howlett)	Public Identity	Neutral Characters	Blue Eyes	Black Hair	Male Characters	Living Characters	3061.0	Oct-74	1974.0
3	1868	Iron Man (Anthony \"Tony\" Stark)	\/Iron_Man_(Anthony_%22Tony%22_Stark)	Public Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	Living Characters	2961.0	Mar-63	1963.0
4	2460	Thor (Thor Odinson)	\/Thor_(Thor_Odinson)	No Dual Identity	Good Characters	Blue Eyes	Blond Hair	Male Characters	Living Characters	2258.0	Nov-50	1950.0

Кодируем категориальные признаки

[10]:
```
data.dtypes
```

[10]:
```
page_id               int64
name                 object
urlslug              object
ID                   object
ALIGN                object
EYE                  object
HAIR                 object
SEX                  object
ALIVE                object
APPEARANCES         float64
FIRST APPEARANCE     object
Year                float64
dtype: object
```

[11]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
df_int = le.fit_transform(data['name'])
data['name'] = df_int
df_int = le.fit_transform(data['urlslug'])
data['urlslug'] = df_int
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['ALIVE'])
data['ALIVE'] = df_int
df_int = le.fit_transform(data['FIRST APPEARANCE'])
data['FIRST APPEARANCE'] = df_int
data.head()
```

[11]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	ALIVE	APPEARANCES	FIRST APPEARANCE	Year
0	1678	3738	3738	3	1	8	5	3	1	4043.0	80	1962.0
1	7139	624	624	2	1	3	20	3	1	3360.0	425	1941.0
2	64786	4302	4302	2	2	3	2	3	1	3061.0	622	1974.0
3	1868	1785	1785	2	1	3	2	3	1	2961.0	434	1963.0
4	2460	3942	3942	1	1	3	3	3	1	2258.0	548	1950.0

Масштабируем числовые данные

[12]:
```
sc1 = MinMaxScaler()
data['page_id'] = sc1.fit_transform(data[['page_id']])
data['APPEARANCES'] = sc1.fit_transform(data[['APPEARANCES']])
data['Year'] = sc1.fit_transform(data[['Year']])
data.head()
```

[12]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	ALIVE	APPEARANCES	FIRST APPEARANCE	Year
0	0.000866	3738	3738	3	1	8	5	3	1	1.000000	80	0.310811
1	0.008108	624	624	2	1	3	20	3	1	0.831024	425	0.027027
2	0.084554	4302	4302	2	2	3	2	3	1	0.757051	622	0.472973
3	0.001118	1785	1785	2	1	3	2	3	1	0.732311	434	0.324324
4	0.001903	3942	3942	1	1	3	3	3	1	0.558387	548	0.148649

[13]:
```
data['ALIVE'].unique()
```

[13]:
```
array([1, 0])
```

Разделение на выборки

```
[14]: 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)
```

```
[15]: data_X_train.shape, data_y_train.shape
```

```
[15]: ((3521, 12), (3521,))
```

```
[16]: data_X_test.shape, data_y_test.shape
```

```
[16]: ((881, 12), (881,))
```

```
[17]: np.unique(target)
```

```
[17]: array([0, 1])
```

Логистическая регрессия

```
[ ]: model = LogisticRegression()
model.fit(data_X_train, data_y_train)
```

```
[21]: data_y_pred = model.predict(data_X_test)
accuracy_score(data_y_test, data_y_pred)
```

```
[21]: 1.0
```

```
[22]: f1_score(data_y_test, data_y_pred, average='micro')
```

```
[22]: 1.0
```

Градиентный бустинг

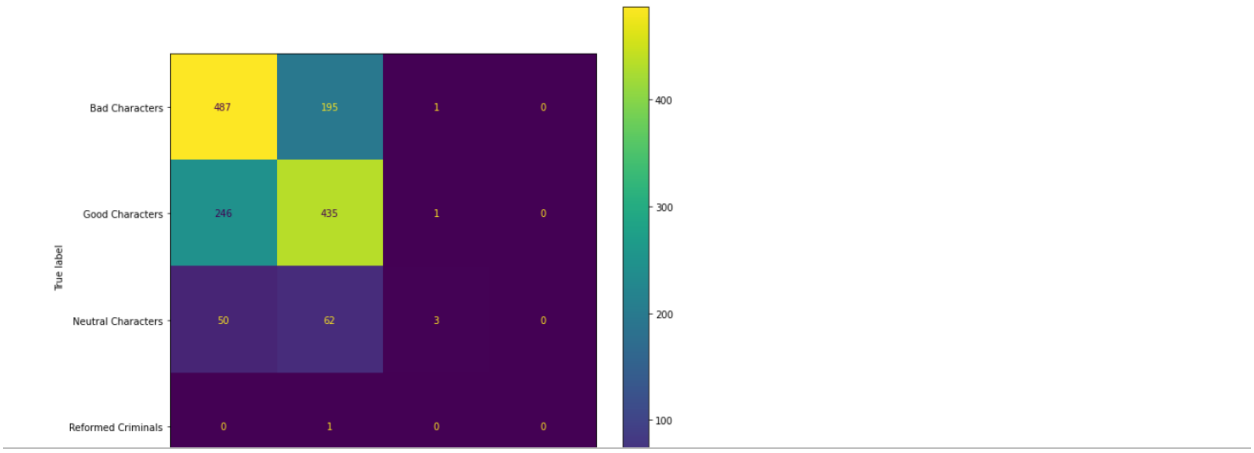
```
[22]: xgb_model = xgb.XGBClassifier()
xgb_params = {'learning_rate': [0.01, 0.05, 0.1, 0.12, 0.15, 0.2, 0.3, 0.5, 0.7],
              'max_depth': [5, 6, 7, 8],
              'min_child_weight': [2, 3, 5, 11],
              'n_estimators': [1, 2, 5, 10, 12, 15, 20]}

xgb_search = GridSearchCV(estimator=xgb_model, param_grid=xgb_params, cv=5, n_jobs=4, scoring='accuracy')
xgb_search.fit(train_X, train_y)
xgb_search.best_params_, xgb_search.best_score_
```

```
c:\Users\danib\Documents\pyton\myvenv\lib\site-packages\sklearn\model_selection\_split.py:676: UserWarning: The least populated class in y has only 2 members, which is less t
han n_splits=5.
  warnings.warn(
```

```
[22]: ({'learning_rate': 0.5,
  'max_depth': 5,
  'min_child_weight': 2,
  'n_estimators': 20},
 0.619091448028456)
```

```
[23]: xgb_class: xgb.XGBClassifier = xgb_search.best_estimator_
xgb_class.fit(train_X, train_y)
xgb_pred = xgb_class.predict(test_X)
fig, ax = plt.subplots(figsize=(10, 10))
ConfusionMatrixDisplay.from_predictions(test_y, xgb_pred, ax=ax, display_labels=le.classes_)
accuracy_score(xgb_pred, test_y)
```

```
[23]: 0.62457798784605
```



Градиентный бустинг получился менее точным по сравнению с логистической регрессией, но он хотя бы выглядит реалистично (accuracy не 1).