Мартынова П.В ИУ5-61Б Вариант №13

```
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")

data = pd.read_csv('marvel-wikia-data.csv', sep=",")

data.head()
```

	page_id	name	urlslug	ID
0	1678	Spider-Man (Peter Parker)	\/Spider-Man_(Peter_Parker)	Secret Identity
1	7139	Captain America (Steven Rogers)	\/Captain_America_(Steven_Rogers)	Public Identity
2	64786	Wolverine (James \"Logan\" Howlett)	\/Wolverine_(James_%22Logan%22_Howlett)	Public Identity
3	1868	Iron Man (Anthony \"Tony\" Stark)	\/Iron_Man_(Anthony_%22Tony%22_Stark)	Public Identity
4	2460	Thor (Thor Odinson)	\/Thor_(Thor_Odinson)	No Dual Identity

```
                ALIGN         EYE        HAIR              SEX  GSM  \
0     Good Characters  Hazel Eyes  Brown Hair  Male Characters  NaN
1     Good Characters   Blue Eyes  White Hair  Male Characters  NaN
2  Neutral Characters   Blue Eyes  Black Hair  Male Characters  NaN
3     Good Characters   Blue Eyes  Black Hair  Male Characters  NaN
4     Good Characters   Blue Eyes  Blond Hair  Male Characters  NaN

               ALIVE  APPEARANCES FIRST APPEARANCE    Year
0  Living Characters       4043.0           Aug-62  1962.0
1  Living Characters       3360.0           Mar-41  1941.0
2  Living Characters       3061.0           Oct-74  1974.0
3  Living Characters       2961.0           Mar-63  1963.0
4  Living Characters       2258.0           Nov-50  1950.0
```

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

```
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
```

```
data.shape
```

```
(16376, 13)
```

```
data.pop('GSM')
```

```
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
```

```
data.shape
```

```
(16376, 12)
```

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

```
(4402, 12)
```

```
data.head()
```

```
   page_id                                    name  \
0     1678            Spider-Man (Peter Parker)
1     7139      Captain America (Steven Rogers)
2    64786  Wolverine (James \"Logan\" Howlett)
3     1868    Iron Man (Anthony \"Tony\" Stark)
4     2460                  Thor (Thor Odinson)

                                  urlslug                ID  \
0              \/Spider-Man_(Peter_Parker)   Secret Identity
1        \/Captain_America_(Steven_Rogers)   Public Identity
2  \/Wolverine_(James_%22Logan%22_Howlett)   Public Identity
3    \/Iron_Man_(Anthony_%22Tony%22_Stark)   Public Identity
4                   \/Thor_(Thor_Odinson)  No Dual Identity

                ALIGN         EYE        HAIR              SEX  \
0     Good Characters  Hazel Eyes  Brown Hair  Male Characters
1     Good Characters   Blue Eyes  White Hair  Male Characters
2  Neutral Characters   Blue Eyes  Black Hair  Male Characters
3     Good Characters   Blue Eyes  Black Hair  Male Characters
4     Good Characters   Blue Eyes  Blond Hair  Male Characters

               ALIVE  APPEARANCES FIRST APPEARANCE    Year
0  Living Characters       4043.0           Aug-62  1962.0
1  Living Characters       3360.0           Mar-41  1941.0
2  Living Characters       3061.0           Oct-74  1974.0
3  Living Characters       2961.0           Mar-63  1963.0
4  Living Characters       2258.0           Nov-50  1950.0
```

Кодируем категориальные признаки

```
data.dtypes
```

```
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
```

```
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()
```

```
   page_id  name  urlslug  ID  ALIGN  EYE  HAIR  SEX  ALIVE
APPEARANCES  \
0     1678  3738     3738   3      1    8     5    3      1
4043.0
1     7139   624      624   2      1    3    20    3      1
3360.0
2    64786  4302     4302   2      2    3     2    3      1
3061.0
3     1868  1785     1785   2      1    3     2    3      1
2961.0
4     2460  3942     3942   1      1    3     3    3      1
2258.0

   FIRST APPEARANCE    Year
0                80  1962.0
1               425  1941.0
2               622  1974.0
3               434  1963.0
4               548  1950.0
```

Масштабируем числовые данные

```
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()
```

```
    page_id  name  urlslug  ID  ALIGN  EYE  HAIR  SEX  ALIVE
APPEARANCES  \
0  0.000866  3738     3738   3      1    8     5    3      1
1.000000
1  0.008108   624      624   2      1    3    20    3      1
0.831024
2  0.084554  4302     4302   2      2    3     2    3      1
0.757051
3  0.001118  1785     1785   2      1    3     2    3      1
0.732311
4  0.001903  3942     3942   1      1    3     3    3      1
0.558387

   FIRST APPEARANCE      Year
0                80  0.310811
1               425  0.027027
2               622  0.472973
3               434  0.324324
4               548  0.148649
```

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

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

```
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
```

```
((3521, 12), (3521,))
```

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

```
((881, 12), (881,))
```

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

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

Логистическая регрессия

```
model = LogisticRegression()
model.fit(data_X_train, data_y_train)
```

```
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:763: ConvergenceWarning: lbfgs failed to converge
(status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
data_y_pred = model.predict(data_X_test)
accuracy_score(data_y_test, data_y_pred)
```

```
1.0
```

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

```
1.0
```

Случайный лес

```
model_2 = RandomForestClassifier()
model_2.fit(data_X_train, data_y_train)
```

```
RandomForestClassifier()
```

```
data_y_pred = model_2.predict(data_X_test)
accuracy_score(data_y_test, data_y_pred)
```

```
1.0
```

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

```
1.0
```