Рубежный контроль №2

Покшубина Софья, ИУ5-61Б

Вариант 12

Задание. Для заданного набора данных построить модели логистической регрессии и случайного леса. Оценить 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.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score
```

EDA

```
df = pd.read_csv("dataset.csv")
```

```
df.head()
```

```
   page_id                          name
urlslug  \
0     1422         Batman (Bruce Wayne)
\/wiki\/Batman_(Bruce_Wayne)
1    23387        Superman (Clark Kent)
\/wiki\/Superman_(Clark_Kent)
2     1458   Green Lantern (Hal Jordan)
\/wiki\/Green_Lantern_(Hal_Jordan)
3     1659     James Gordon (New Earth)
\/wiki\/James_Gordon_(New_Earth)
4     1576  Richard Grayson (New Earth)
\/wiki\/Richard_Grayson_(New_Earth)

                ID            ALIGN         EYE        HAIR
SEX  \
0  Secret Identity  Good Characters   Blue Eyes  Black Hair  Male
Characters
1  Secret Identity  Good Characters   Blue Eyes  Black Hair  Male
Characters
2  Secret Identity  Good Characters  Brown Eyes  Brown Hair  Male
```

```
Characters
3  Public Identity  Good Characters  Brown Eyes  White Hair  Male
Characters
4  Secret Identity  Good Characters   Blue Eyes  Black Hair  Male
Characters
```

	GSM	ALIVE	APPEARANCES	FIRST APPEARANCE	YEAR
0	NaN	Living Characters	3093.0	1939, May	1939.0
1	NaN	Living Characters	2496.0	1986, October	1986.0
2	NaN	Living Characters	1565.0	1959, October	1959.0
3	NaN	Living Characters	1316.0	1987, February	1987.0
4	NaN	Living Characters	1237.0	1940, April	1940.0

Столбцы *page_id*, *name*, *urlslug*, *first appearance* и *year* не имеют никакой информативности для построения моделей машинного обучения, поэтому удалим их.

```
df = df.drop(columns = ['page_id', 'name', 'urlslug', 'FIRST
APPEARANCE', 'YEAR'], axis = 1)
```

Приведем названия колонок к нижнему регистру.

```
df.columns = df.columns.str.lower()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6896 entries, 0 to 6895
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id           4883 non-null   object
 1   align        6295 non-null   object
 2   eye          3268 non-null   object
 3   hair         4622 non-null   object
 4   sex          6771 non-null   object
 5   gsm          64 non-null     object
 6   alive        6893 non-null   object
 7   appearances  6541 non-null   float64
dtypes: float64(1), object(7)
memory usage: 431.1+ KB
```

Проверим наличие дубликатов.

```
sum(df.duplicated(df.columns))
```

```
2301
```

Удалим все дубликаты и произведем проверку.

```
df = df.drop_duplicates(df.columns, keep = 'last')
sum(df.duplicated(df.columns))
```

```
0
```

```
df.shape
```

```
(4595, 8)
```

```
df.isnull().sum()
```

```
id             1054
align           441
eye            1703
hair            986
sex              85
gsm            4531
alive             3
appearances     196
dtype: int64
```

Пропусков в столбце alive мало, поэтому можем их удалить.

```
df.dropna(subset=['alive'], inplace=True)
```

Узнаем количество уникальных значений в каждом столбце.

```
df.nunique()
```

```
id               3
align            4
eye             17
hair            17
sex              4
gsm              2
alive            2
appearances    282
dtype: int64
```

```
df.gsm.value_counts(dropna=False)
```

```
NaN                     4529
Homosexual Characters     53
Bisexual Characters       10
Name: gsm, dtype: int64
```

В данной колонке много пустых значений, поэтому можем ее удалить.

```
df = df.drop(columns = ['gsm'], axis = 1)
```

Рассмотрим подробнее столбцы id, align, sex, alive.

```
df['id'].value_counts(dropna=False)
```

```
Public Identity     1877
Secret Identity     1654
NaN                 1053
```

```
Identity Unknown       8
Name: id, dtype: int64
```

Заменим пропуски значением Identity Unknown.

```
df['id'].fillna(value = "Identity Unknown", inplace = True)
```

```
df['align'].value_counts(dropna=False)
```

```
Good Characters       2047
Bad Characters        1600
Neutral Characters     502
NaN                    440
Reformed Criminals       3
Name: align, dtype: int64
```

Заменим пропуски значением Neutral Characters.

```
df['align'].fillna(value = "Neutral Characters", inplace = True)
```

```
df['sex'].value_counts(dropna=False)
```

```
Male Characters           2913
Female Characters         1575
NaN                         84
Genderless Characters       19
Transgender Characters       1
Name: sex, dtype: int64
```

Заполним пропуски значением Genderless Characters.

```
df['sex'].fillna(value = "Genderless Characters", inplace = True)
```

Рассмотрим детальнее столбец eye.

```
df['eye'].value_counts(dropna=False)
```

```
NaN                   1700
Blue Eyes             1003
Brown Eyes             704
Black Eyes             364
Green Eyes             277
Red Eyes               191
White Eyes             110
Yellow Eyes             83
Grey Eyes               39
Photocellular Eyes      35
Hazel Eyes              23
Purple Eyes             14
Violet Eyes             12
Orange Eyes             10
Gold Eyes                9
Auburn Hair              7
```

```
Pink Eyes                6
Amber Eyes               5
Name: eye, dtype: int64
```

В данном столбце много разных данных, поэтому разделим их на более крупные категории.

```
eyes = ['Blue Eyes','Brown Eyes', 'Black Eyes', 'Green Eyes','Red
Eyes']
eyes_new = []
for i in df.eye.values:
    if i not in eyes:
        eyes_new.append('Other color')
    else:
        eyes_new.append(i)
df['eye'] = eyes_new
```

```
df['eye'].value_counts(dropna=False)
```

```
Other color    2053
Blue Eyes      1003
Brown Eyes      704
Black Eyes      364
Green Eyes      277
Red Eyes        191
Name: eye, dtype: int64
```

Со столбцом hair поступим так же.

```
df['hair'].value_counts(dropna=False)
```

```
Black Hair               1141
NaN                       984
Brown Hair                829
Blond Hair                603
Red Hair                  409
White Hair                302
Grey Hair                 137
Green Hair                 41
Blue Hair                  38
Purple Hair                32
Strawberry Blond Hair      28
Orange Hair                20
Pink Hair                  11
Gold Hair                   5
Violet Hair                 4
Reddish Brown Hair          3
Silver Hair                 3
Platinum Blond Hair         2
Name: hair, dtype: int64
```

```
hair = ['Black Hair','Brown Hair', 'Blond Hair', 'Red Hair','White
Hair']
hair_new = []
for i in df.hair.values:
    if i not in hair:
        hair_new.append('Other color')
    else:
        hair_new.append(i)
df['hair'] = hair_new
```

```
df['hair'].value_counts(dropna=False)
```

```
Other color    1308
Black Hair     1141
Brown Hair      829
Blond Hair      603
Red Hair        409
White Hair      302
Name: hair, dtype: int64
```

Пропуски в столбце appearances заполним медианным значением.

```
df['appearances'] =
df['appearances'].fillna(df['appearances'].median())
```

```
df.isnull().sum()
```

```
id             0
align          0
eye            0
hair           0
sex            0
alive          0
appearances    0
dtype: int64
```

Кодирование категориальных признаков

Теперь закодируем категориальные признаки с помощью Label Encoder.

```
le = LabelEncoder()
df['id'] = le.fit_transform(df['id'])
df['align'] = le.fit_transform(df['align'])
df['eye'] = le.fit_transform(df['eye'])
df['hair'] = le.fit_transform(df['hair'])
df['sex'] = le.fit_transform(df['sex'])
df['alive'] = le.fit_transform(df['alive'])
```

```
df.head()
```

```
   id  align  eye  hair  sex  alive  appearances
0   2      1    1     0    2      1       3093.0
```

```
1   2      1    1     0    2      1       2496.0
2   2      1    2     2    2      1       1565.0
3   1      1    2     5    2      1       1316.0
4   2      1    1     0    2      1       1237.0
```

Разделение выборки

Разделим выборку на обучающую и тестовую.
Целевым признаком выберем столбец alive (жив герой или нет).

```
y = df['alive']
x = df.drop(['alive'], axis = 1)

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(x)

x_train, x_test, y_train, y_test = train_test_split(scaled_data, y,
test_size = 0.2, random_state = 0)
```

Метрики

```
def print_metrics(test, prediction):
    print("Accuracy:", accuracy_score(test, prediction))
    print("Precision:", precision_score(test, prediction))
```

Логистическая регрессия

```
lr = LogisticRegression()
lr_prediction = lr.fit(x_train, y_train).predict(x_test)
print_metrics(y_test, lr_prediction)

Accuracy: 0.719260065288357
Precision: 0.719260065288357
```

По значению метрик можно сказать, что модель приблизительно на 72% идентифицирует как сам объект, так и его класс.

Случайный лес

```
rf = RandomForestClassifier()
rf_prediction = rf.fit(x_train, y_train).predict(x_test)
print_metrics(y_test, rf_prediction)

Accuracy: 0.6082698585418934
Precision: 0.7009345794392523
```

В данном случае можно сделать вывод о том, что модель правильно классифицирует 60% объектов и при этом в 70% случаев верно определяет класс объекта.