РК №2

Милевич Артём Андреевич ИУ5-62Б

Вариант 13

Задание. Для заданного набора данных (вариант 13) постройте модели классификации. Для построения моделей используйте методы опорных векторов и случайный лес. 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. Набор данных: https://www.kaggle.com/fivethirtyeight/fivethirtyeight-comic-characters-dataset (файл marvel-wikia-data.csv)

In [1]:
```
#импортируем нужные библиотеки
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import seaborn as sns
import scipy
import plotly
import missingno as msno
from numpy import nan
from sklearn.impute import SimpleImputer, MissingIndicator
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolut
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings('ignore')
```

Загрузим набор данных и выведем информацию о нем.

In [2]:
```
dataset = pd.read_csv('marvel-wikia-data.csv')
```

In [3]:
```
dataset.head(5)
```

Out[3]:

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

In [4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16376 entries, 0 to 16375
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   page_id           16376 non-null  int64  
 1   name              16376 non-null  object 
 2   urlslug           16376 non-null  object 
 3   ID                12606 non-null  object 
 4   ALIGN             13564 non-null  object 
 5   EYE               6609 non-null   object 
 6   HAIR              12112 non-null  object 
 7   SEX               15522 non-null  object 
 8   GSM               90 non-null     object 
 9   ALIVE             16373 non-null  object 
 10  APPEARANCES       15280 non-null  float64
 11  FIRST APPEARANCE  15561 non-null  object 
 12  Year              15561 non-null  float64
dtypes: float64(2), int64(1), object(10)
memory usage: 1.6+ MB
```

Подсчитаем количество и процент пропусков по столбцам.

In [5]:
```
for col in dataset.columns:
    pct_missing = np.mean(dataset[col].isnull())
    print('{}: {} - {}%'.format(col, dataset[col].isna().sum(), round(pct_missing*1
```

```
page_id: 0 - 0.0%
name: 0 - 0.0%
urlslug: 0 - 0.0%
ID: 3770 - 23.02%
ALIGN: 2812 - 17.17%
EYE: 9767 - 59.64%
HAIR: 4264 - 26.04%
SEX: 854 - 5.21%
GSM: 16286 - 99.45%
ALIVE: 3 - 0.02%
APPEARANCES: 1096 - 6.69%
FIRST APPEARANCE: 815 - 4.98%
Year: 815 - 4.98%
```

Обработка пропусков.

Столбец GSM невозможно восстановить из-за слишком большого процента пропусков, также он практически не влияет на определение признака при обучении модели, как и поле 'page_id', которое является первичным ключем, поэтому данные столбцы мы удалим.

In [6]:
```
dataset.drop(['GSM'], axis=1, inplace=True)
dataset.drop(['page_id'], axis=1, inplace=True)
```

Условия задачи позволяют сократить набор данных, поэтому лучшим решением будет оставить те данные, у которых одновременно присутствуют поля: 'EYE', 'ID', 'ALIGN', 'HAIR', 'SEX', 'FIRST APPEARANCE'.

В данном случае не получится заполнить чем-то данные поля (например наиболее часто встречающимся значением), т.к. это не статистические данные, а категориальные признаки, которые сильно влияют на шанс ошибки у модели.

Данные действия помогут минимизировать ошибки и улучшить качество данных для обучения модели.

In [7]:
```
dataset.dropna(subset=['EYE', 'ID', 'ALIGN', 'HAIR', 'SEX', 'FIRST APPEARANCE'], ax
```

In [8]:
```
for col in dataset.columns:
    pct_missing = np.mean(dataset[col].isnull())
    if pct_missing > 0:
        print('{}: {} - {}%'.format(col, dataset[col].isna().sum(), round(pct_missi
```

```
APPEARANCES: 193 - 4.2%
```

Оставшееся поле 'APPEARANCES' является количественным признаком, поэтому его можно заполнить наиболее часто встречающимся значением.

In [9]:
```
imputer = SimpleImputer(strategy='most_frequent', missing_values=nan)
imputer = imputer.fit(dataset[['APPEARANCES']])
dataset['APPEARANCES'] = imputer.transform(dataset[['APPEARANCES']])
```

In [10]:
```
dataset.isna().sum()
```

Out[10]:
```
name                0
urlslug             0
ID                  0
ALIGN               0
EYE                 0
HAIR                0
SEX                 0
ALIVE               0
APPEARANCES         0
FIRST APPEARANCE    0
Year                0
dtype: int64
```

In [11]:
```
dataset.shape
```

Out[11]:
```
(4595, 11)
```

Кодирование признаков и разделение выборки.

В качестве целевого признака возьмем столбец 'name'. Закодируем все не числовые столбцы в числовые, при помощи LabelEncoder.

In [12]:
```
le = LabelEncoder()
dataset['name']= le.fit_transform(dataset['name'])
dataset['urlslug']= le.fit_transform(dataset['urlslug'])
dataset['ID']= le.fit_transform(dataset['ID'])
dataset['ALIGN']= le.fit_transform(dataset['ALIGN'])
dataset['EYE']= le.fit_transform(dataset['EYE'])
dataset['HAIR']= le.fit_transform(dataset['HAIR'])
dataset['SEX']= le.fit_transform(dataset['SEX'])
dataset['ALIVE']= le.fit_transform(dataset['ALIVE'])
dataset['FIRST APPEARANCE']= le.fit_transform(dataset['FIRST APPEARANCE'])
```

In [13]:
```
X = dataset.drop(columns="name")
y = dataset["name"]
```

Обучение модели методом опорных векторов и оценка её качества.

В качестве метрик возьмём:

1. mean_squared_error - чтобы подчеркнуть ошибки
2. median_absolute_error - чтобы оценить качество модели с устойчивостью к выбросам
3. r2_score - чтобы точно и наглядно интерпретировать качество модели

In [14]:
```
# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

```
# Создание модели SVM
svm_model = SVC(kernel='linear')

# Обучение модели
svm_model.fit(X_train, y_train)

# Прогнозирование классов для тестовых данных
y_pred = svm_model.predict(X_test)
```

In [15]:
```
# Оценка точности модели
mse_svc = mean_squared_error(y_test, y_pred)
med_svc = median_absolute_error(y_test, y_pred)
r2_svc = r2_score(y_test, y_pred)
```

Обучение модели случайного леса и оценка её качества.

Метрики аналогичные.

In [16]:
```
# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Создание модели случайного леса
cf_model = RandomForestClassifier(n_estimators=100)

# Обучение модели
cf_model.fit(X_train, y_train)

# Прогнозирование классов для тестовых данных
y_pred = cf_model.predict(X_test)
```

In [17]:
```
# Оценка точности модели
mse_cf = mean_squared_error(y_test, y_pred)
med_cf = median_absolute_error(y_test, y_pred)
r2_cf = r2_score(y_test, y_pred)
```

Сравним качество 2-ух моделей.

In [18]:
```
print('--------------------------- mean_squared_error --------------------------')
print('SVM:          ', mse_svc)
print('RandomForest: ', mse_cf)
print('--------------------------- median_absolute_error ------------------------')
print('SVM:          ', med_svc)
print('RandomForest: ', med_cf)
print('-------------------------------- r2_score ----------------------------')
print('SVM:          ', r2_svc)
print('RandomForest: ', r2_cf)
```

```
--------------------------- mean_squared_error ---------------------------
SVM:           849.6050054406965
RandomForest:  869876.2916213275
--------------------------- median_absolute_error --------------------------
SVM:           13.0
RandomForest:  374.0
------------------------------- r2_score ----------------------------
SVM:           0.9995162959402933
RandomForest:  0.5047549261064876
```

Вывод.

Метод опорных векторов (Support Vector Machine, SVM) и случайный лес (Random Forest) являются двумя популярными методами машинного обучения, используемыми для задач классификации и регрессии. Оба метода имеют свои особенности и преимущества.

Модель по методу опорных векторов получилась очень точной, что показывает практически единичный коэффициент детерминации.

Модель опорных векторов оказалась более устойчивой к выбросам в данных, что показывает большая разница в метрике mean_squared_error, а также в 10 раз точнее модели случайного леса, что также показывает метрика median_absolute_error.

Такой высокой точности в методе опорных векторов удалось добиться из-за сильной корреляции в признаках выборки, а также из-за относительно малого объёма выборки. В свою очередь точность метода случайного леса получилась не такой большой, так как каждое дерево обучается независимо на случайной подвыборке данных и случайном подмножестве признаков, что могло привести к полученным результатам.