Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчёт по рубежному контролю №2

«Технологии машинного обучения»

Вариант 12

Выполнила:

студентка группы ИУ5-63Б

Румак Д.П.

Преподаватель:

Гапанюк Ю. Е.

2023 г.

Задание:

Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). 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.

Группа	**Метод №1**	**Метод №2**
ИУ5-63Б, ИУ5Ц-83Б	Дерево решений	Случайный лес

https://www.kaggle.com/datasets/fivethirtyeight/fivethirtyeight-comic-characters-dataset?select=dc-wikia-data.csv

Page_id - Уникальный идентификатор страницы с этими персонажами в wikia

Name - Имя персонажа

Urlslug - Уникальный URL-адрес в wikia, который приведет вас к персонажу

ID - Статус личности персонажа (секретная личность, публичная личность, [только в Marvel: двойной личности нет])

ALIGN - Если персонаж хороший, плохой или нейтральный

EYE - Цвет глаз персонажа

HAIR - Цвет волос персонажа

SEX - Пол персонажа (например, мужчина, женщина и т.д.)

GSM - Если персонаж принадлежит к гендерному или сексуальному меньшинству (например, гомосексуальные персонажи, бисексуальные персонажи)

ALIVE - Если персонаж жив или умер

APPEARANCES - Количество появлений персонажа в комиксах (по состоянию на 2 сентября 2014 г.). С течением времени их количество будет становиться все более устаревшим.)

FIRST APPEARANCES - Месяц и год первого появления персонажа в комиксе, если таковой имеется

YEAR - Год первого появления персонажа в комиксе, если таковой имеется

Решение:

Загружаем датасет и подключаем необходимые библиотеки:

In [10]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
```

In [16]:
```
ds=pd.read_csv(r"C:\Users\Дарья\Desktop\6-семестр\ТМО\dc-wikia-data.csv")
data.shape
```

Out[16]: (6896, 13)

In [17]:
```
ds.head()
```

Out[17]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	GSM	ALIVE	APPE
0	1422	Batman (Bruce Wayne)	\wiki\Batman_(Bruce_Wayne)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	
1	23387	Superman (Clark Kent)	\wiki\Superman_(Clark_Kent)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	
2	1458	Green Lantern (Hal Jordan)	\wiki\Green_Lantern_(Hal_Jordan)	Secret Identity	Good Characters	Brown Eyes	Brown Hair	Male Characters	NaN	Living Characters	
3	1659	James Gordon (New Earth)	\wiki\James_Gordon_(New_Earth)	Public Identity	Good Characters	Brown Eyes	White Hair	Male Characters	NaN	Living Characters	
4	1576	Richard Grayson (New Earth)	\wiki\Richard_Grayson_(New_Earth)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	

Посчитаем количество пустых значений:

In [18]:
```
ds.isnull().sum()
```

Out[18]:
```
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
```

Проверка типов данных столбцов:

```
In [426]: 1 ds.dtypes

Out[426]: page_id              int64
          name                object
          urlslug             object
          ID                  object
          ALIGN               object
          EYE                 object
          HAIR                object
          SEX                 object
          GSM                 object
          ALIVE               object
          APPEARANCES        float64
          FIRST APPEARANCE    object
          YEAR               float64
          dtype: object
```

Столбцы, имеющие пропуски - **ID, ALIGN, EYE, HAIR, GSM, APPEARANCES, ALIVE, FIRST APPEARANCE, YEAR, SEX.**

Для начала удалим столбцы, которые не несут особо важной информации:

```
In [427]: 1 ds=ds.drop(['ID','urlslug','GSM','FIRST APPEARANCE','YEAR'],axis=1)
```

Теперь заполним пропущенные значения в столбцах **ALIGN, EYE, HAIR, APPEARANCES, ALIVE, SEX:**

ALIGN:

```
In [429]: 1 print(ds.ALIGN.value_counts(dropna=False))

          Bad Characters        2895
          Good Characters       2832
          NaN                    601
          Neutral Characters     565
          Reformed Criminals       3
          Name: ALIGN, dtype: int64

In [430]: 1 ds.ALIGN.fillna(value = "Neutral Characters", inplace = True)

In [431]: 1 print(ds.ALIGN.isna().sum())

          0
```

EYE:

In [432]:
```
print(ds.EYE.value_counts(dropna=False))
```

```
NaN                  3628
Blue Eyes            1102
Brown Eyes            879
Black Eyes            412
Green Eyes            291
Red Eyes              208
White Eyes            116
Yellow Eyes            86
Photocellular Eyes     48
Grey Eyes              40
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
```

In [433]:
```
eyes = ['Blue Eyes','Brown Eyes','Green Eyes','Red Eyes','Black Eyes']
eyes_after_ds = []
for i in ds.EYE.values:
    if i not in eyes:
        eyes_after_ds.append('Different Eyes')
    else:
        eyes_after_ds.append(i)
ds['EYE'] = eyes_after_ds
print(ds.EYE.value_counts(dropna=False))
```

```
Different Eyes    4004
Blue Eyes         1102
Brown Eyes         879
Black Eyes         412
Green Eyes         291
Red Eyes           208
Name: EYE, dtype: int64
```

SEX:

In [434]:
```
print(ds.SEX.value_counts(dropna=False))
```

```
Male Characters           4783
Female Characters         1967
NaN                        125
Genderless Characters       20
Transgender Characters       1
Name: SEX, dtype: int64
```

In [435]:
```
ds=ds[ds.SEX.isin(["Male Characters","Female Characters"])]
```

HAIR:

In [437]:
```
print(ds.HAIR.value_counts(dropna=False))
```

```
NaN                     2145
Black Hair              1566
Brown Hair              1145
Blond Hair               742
Red Hair                 461
White Hair               346
Grey Hair                156
Green Hair                42
Blue Hair                 39
Purple Hair               32
Strawberry Blond Hair     28
Orange Hair               20
Pink Hair                 11
Gold Hair                  5
Violet Hair                4
Silver Hair                3
Reddish Brown Hair         3
Platinum Blond Hair        2
Name: HAIR, dtype: int64
```

In [438]:
```
hair = ["Black Hair","Brown Hair","Blond Hair","Red Hair","Bald","No Hair",
hair_after_dc = []
for i in ds.HAIR.values:
    if i not in hair:
        hair_after_dc.append('Different Hair')
    else:
        hair_after_dc.append(i)
ds['HAIR'] = hair_after_dc
print(ds.HAIR.value_counts(dropna=False))
```

```
Different Hair          2306
Black Hair              1566
Brown Hair              1145
Blond Hair               742
Red Hair                 461
White Hair               346
Grey Hair                156
Strawberry Blond Hair     28
Name: HAIR, dtype: int64
```

Осталось еще два столбца с пропусками. Так как в столбце ALLIVE всего 2 пропуска, просто удалим строки, имеющие пропуски. Столбец APPEARANCE заполним медианным значением:

In [439]:
```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6750 entries, 0 to 6895
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   page_id      6750 non-null   int64
 1   name         6750 non-null   object
 2   ALIGN        6750 non-null   object
 3   EYE          6750 non-null   object
 4   HAIR         6750 non-null   object
 5   SEX          6750 non-null   object
 6   ALIVE        6748 non-null   object
 7   APPEARANCES  6407 non-null   float64
dtypes: float64(1), int64(1), object(6)
memory usage: 474.6+ KB
```

In [440]:
```
ds['APPEARANCES'].fillna(ds['APPEARANCES'].median(),inplace=True)
```

```
In [442]:
1 ds=ds.dropna()
```

```
In [443]:
1 ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6748 entries, 0 to 6895
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   page_id      6748 non-null   int64
 1   name         6748 non-null   object
 2   ALIGN        6748 non-null   object
 3   EYE          6748 non-null   object
 4   HAIR         6748 non-null   object
 5   SEX          6748 non-null   object
 6   ALIVE        6748 non-null   object
 7   APPEARANCES  6748 non-null   float64
dtypes: float64(1), int64(1), object(6)
memory usage: 474.5+ KB
```

Убедимся, что пропусков больше не осталось:

```
In [444]:
1 ds.isna().sum()
```

```
Out[444]: page_id        0
          name           0
          ALIGN          0
          EYE            0
          HAIR           0
          SEX            0
          ALIVE          0
          APPEARANCES    0
          dtype: int64
```

Кодирование категориальных признаков:

```
In [446]:
1  ds['name'] = pd.Categorical(ds['name'])
2  ds['name'] = ds['name'].cat.codes
3  ds['ALIGN'] = pd.Categorical(ds['ALIGN'])
4  ds['ALIGN'] = ds['ALIGN'].cat.codes
5  ds['EYE'] = pd.Categorical(ds['EYE'])
6  ds['EYE'] = ds['EYE'].cat.codes
7  ds['HAIR'] = pd.Categorical(ds['HAIR'])
8  ds['HAIR'] = ds['HAIR'].cat.codes
9  ds['SEX'] = pd.Categorical(ds['SEX'])
10 ds['SEX'] = ds['SEX'].cat.codes
11 ds['ALIVE'] = pd.Categorical(ds['ALIVE'])
12 ds['ALIVE'] = ds['ALIVE'].cat.codes
```

```
In [447]:
1 ds.dtypes
```

```
Out[447]: page_id          int64
          name             int16
          ALIGN             int8
          EYE               int8
          HAIR              int8
          SEX               int8
          ALIVE             int8
          APPEARANCES    float64
          dtype: object
```

Мы преобразуем категориальные признаки в числовые значения, используя метод кодирования категорий. Каждое уникальное значение категориального признака заменяется числом, начиная с 0.

Разделим датасет на обучающую и тестовую выборки с использованием функции train_test_split. X представляет набор признаков, а y – целевую переменную. Выделим для модели как можно больше учебных данных, однако оставим достаточную часть для проверки модели.

In [483]:
```
X = ds.drop(['ALIVE'], axis=1)
y = ds['ALIVE']
```

In [484]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Обучение моделей и оценка качества:

```
dt_model = tree.DecisionTreeRegressor()
dt_model.fit(X_train, y_train)
dt_preds = dt_model.predict(X_test)
```

```
dt_mae = mean_absolute_error(y_test, dt_preds)
dt_mse = mean_squared_error(y_test, dt_preds)
```

```
# Создаём модель леса из сотни деревьев
rf_model = RandomForestRegressor()
# Обучаем на тренировочных данных
rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)
```

```
rf_mae = mean_absolute_error(y_test, rf_preds)
rf_mse = mean_squared_error(y_test, rf_preds)
```

```
print("MAE дерева решений:", dt_mae)
print("MSE дерева решений:", dt_mse)

print("MAE случайного леса:", rf_mae)
print("MSE случайного леса:", rf_mse)
```

```
MAE дерева решений: 0.34271604938271605
MSE дерева решений: 0.34271604938271605
MAE случайного леса: 0.3566913580246913
MSE случайного леса: 0.1927774814814815
```

Итак, можем сказать, что модель случайного леса имеет результат лучше, чем модель дерева решений.

Мы создали модель дерева решений и модель случайного леса. Каждая модель обучается на обучающих данных (X_train и y_train), а затем используется для получения прогнозов на тестовых данных (X_test).

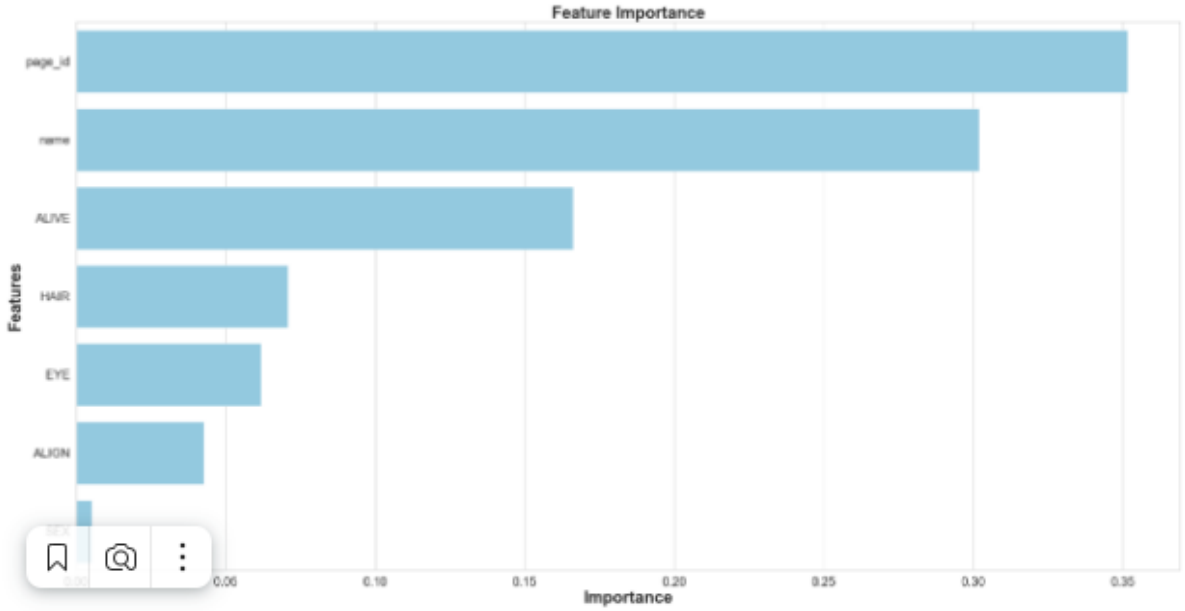
Для оценки качества прогнозов используются две метрики ошибки - средняя абсолютная ошибка (MAE) и средняя квадратичная ошибка (MSE). Для каждой модели рассчитываются значения этих метрик на тестовых данных.

Выводятся значения MAE и MSE для каждой модели. Обычно чем меньше значения MAE и MSE, тем лучше считается точность прогнозов.

Визуализация дерева решений:

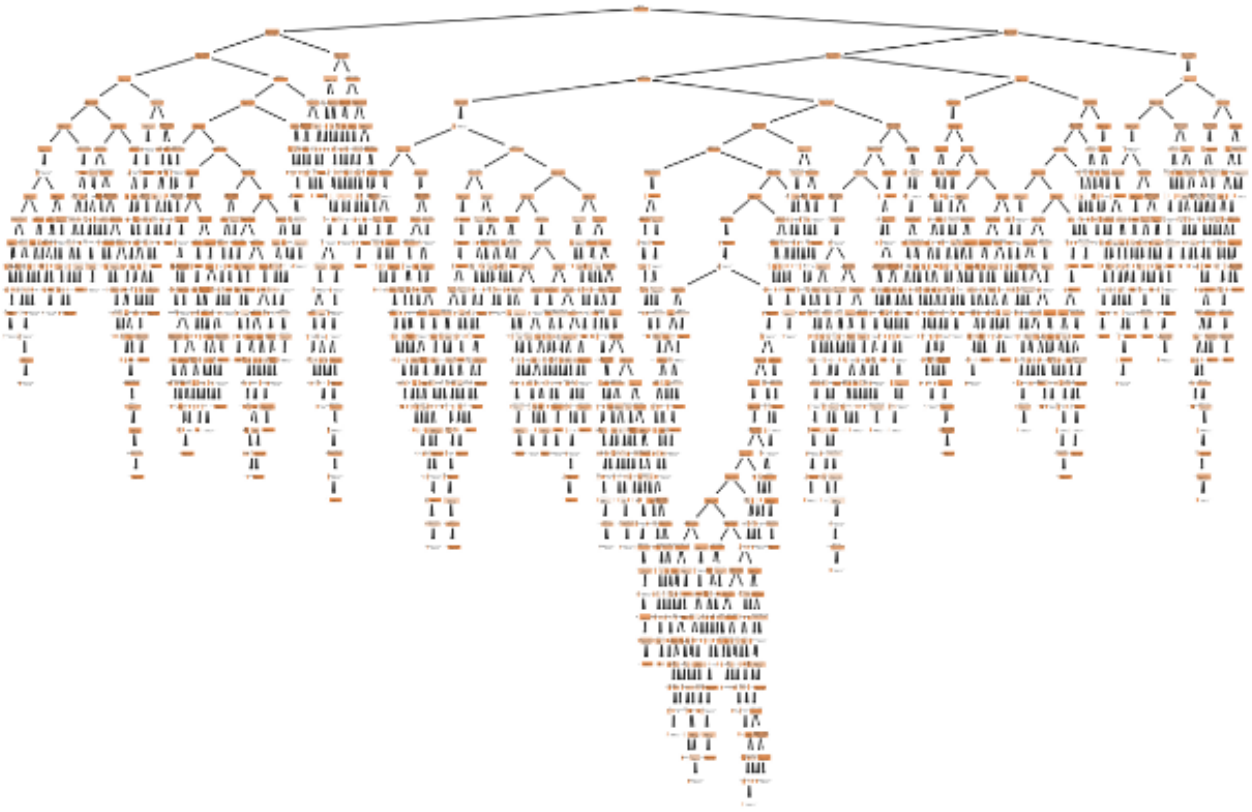
Визуализируем дерево решений с помощью функции plot_tree из библиотеки scikit-learn. Мы используем параметр filled=True, чтобы закрасить узлы дерева в соответствии с прогнозируемой переменной, и передаем имена признаков как параметр feature_names. Покажем важность признаков.

```
import seaborn as sns
feats = {}
for feature, importance in zip(ds.columns, dt_model.feature_importances_):
    feats[feature] = importance
importances = pd.DataFrame.from_dict(feats, orient='index').rename(columns={
importances = importances.sort_values(by='Gini-Importance', ascending=False)
importances = importances.reset_index()
importances = importances.rename(columns={'index': 'Features'})
sns.set(font_scale = 5)
sns.set(style="whitegrid", color_codes=True, font_scale = 1.7)
fig, ax = plt.subplots()
fig.set_size_inches(30,15)
sns.barplot(x=importances['Gini-Importance'], y=importances['Features'], dat
plt.xlabel('Importance', fontsize=25, weight = 'bold')
plt.ylabel('Features', fontsize=25, weight = 'bold')
plt.title('Feature Importance', fontsize=25, weight = 'bold')
display(plt.show())
display(importances)
```



	Features	Gini-Importance
0	page_id	0.351536
1	name	0.301854
2	ALIVE	0.166131
3	HAIR	0.070805
4	EYE	0.061863
5	ALIGN	0.042642
6	SEX	0.005168

```
plt.figure(figsize = (15,10))
tree.plot_tree(dt_model, feature_names = X.columns, filled = True)
plt.show()
```



Мы использовали две метрики - средняя абсолютная ошибка (MAE) и средняя квадратичная ошибка (MSE).

MAE - это средняя абсолютная разница между прогнозами и фактическими значениями. Чем ниже MAE, тем лучше модель соответствует набору данных. MSE - это средняя квадратичная ошибка между предсказаниями и фактическими значениями. Она показывает, насколько сильно отличаются прогнозы модели от

реальных значений. MSE чувствительнее к большим ошибкам, так как она возводит разницу в квадрат. Чем ниже MSE, тем лучше модель соответствует набору данных.