



**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)

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:

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from sklearn import tree
5 from sklearn.tree import plot_tree
6 from sklearn.ensemble import RandomForestClassifier
7 from sklearn.metrics import accuracy_score, precision_score
8 import matplotlib.pyplot as plt
9 import seaborn as sns
10 import os
```

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

(6896, 13)

```
1 ds.head()
```

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

Посчитаем количество пустых значений:

```
1 ds.isnull().sum()
```

```
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:

```
1 ds.dtypes
```

| | |
|------------------|---------|
| 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:

```
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:

```
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
```

```
1 ds.ALIGN.fillna(value = "Neutral Characters", inplace = True)
```

```
1 print(ds.ALIGN.isna().sum())
```

```
0
```

EYE:

```
1 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
```

```
1 eyes = ['Blue Eyes', 'Brown Eyes', 'Green Eyes', 'Red Eyes', 'Black Eyes']
2 eyes_after_ds = []
3 for i in ds.EYE.values:
4     if i not in eyes:
5         eyes_after_ds.append('Different Eyes')
6     else:
7         eyes_after_ds.append(i)
8 ds['EYE'] = eyes_after_ds
9 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:

```
1 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
```

```
1 ds=ds[ds.SEX.isin(["Male Characters", "Female Characters"])]
```

HAIR:

```
1 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
```

```
1 hair = ["Black Hair", "Brown Hair", "Blond Hair", "Red Hair", "Bald", "No Hair",
2 hair_after_dc = []
3 for i in ds.HAIR.values:
4     if i not in hair:
5         hair_after_dc.append('Different Hair')
6     else:
7         hair_after_dc.append(i)
8 ds['HAIR'] = hair_after_dc
9 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 пропуска. Столбец APPEARANCE заполним медианным значением:

```
1 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
```

```
1 ds['APPEARANCES'].fillna(ds['APPEARANCES'].median(),inplace=True)
```

```
1 ds=ds.dropna()
```

```
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:

```
1 ds.isna().sum()
```

```
page_id      0
name          0
ALIGN         0
EYE           0
HAIR          0
SEX           0
ALIVE         0
APPEARANCES  0
dtype: int64
```

Закодируем категориальные признаки с помощью LabelEncoder:

```
1 from sklearn import preprocessing
2 le = preprocessing.LabelEncoder()
3 ds['name'] = le.fit_transform(ds['name'])
4 ds['ALIGN'] = le.fit_transform(ds['ALIGN'])
5 ds['EYE'] = le.fit_transform(ds['EYE'])
6 ds['HAIR'] = le.fit_transform(ds['HAIR'])
7 ds['SEX'] = le.fit_transform(ds['SEX'])
8 ds['ALIVE'] = le.fit_transform(ds['ALIVE'])
```

```
1 ds.dtypes
```

```
page_id      int64
name          int32
ALIGN         int32
EYE           int32
HAIR          int32
SEX           int32
ALIVE         int32
APPEARANCES  float64
dtype: object
```

Разделим датасет на обучающую и тестовую выборки с использованием функции `train_test_split`. В качестве целевого признака берем столбец `ALIVE`. Выделим для

модели как можно больше учебных данных, однако оставим достаточную часть для проверки модели. Отмасштабируем данные.

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

```
1 scaler = preprocessing.MinMaxScaler()
2 scaled_data = scaler.fit_transform(X)
```

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Дерево решений

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

```
1 dt_model = tree.DecisionTreeClassifier()
2 dt_model.fit(X_train, y_train)
3 dt_preds = dt_model.predict(X_test)
```

```
1 print_metrics(y_test,dt_preds)
```

```
Accuracy: 0.6553086419753087
Precision: 0.7669074195666448
```

Можно сделать вывод о том, что модель правильно классифицирует порядка 65% объектов и при этом в 77% случаев верно определяет класс объекта.

Случайный лес

```
1 # Создаём модель леса из сотни деревьев
2 rf_model = RandomForestClassifier()
3 # Обучаем на тренировочных данных
4 rf_model.fit(X_train, y_train)
5 rf_preds = rf_model.predict(X_test)
```

```
1 print_metrics(y_test,rf_preds)
```

```
Accuracy: 0.7362962962962963
Precision: 0.7572406529752501
```

Здесь можем сказать, что модель приблизительно одинаково идентифицирует как сам объект, так и его класс.

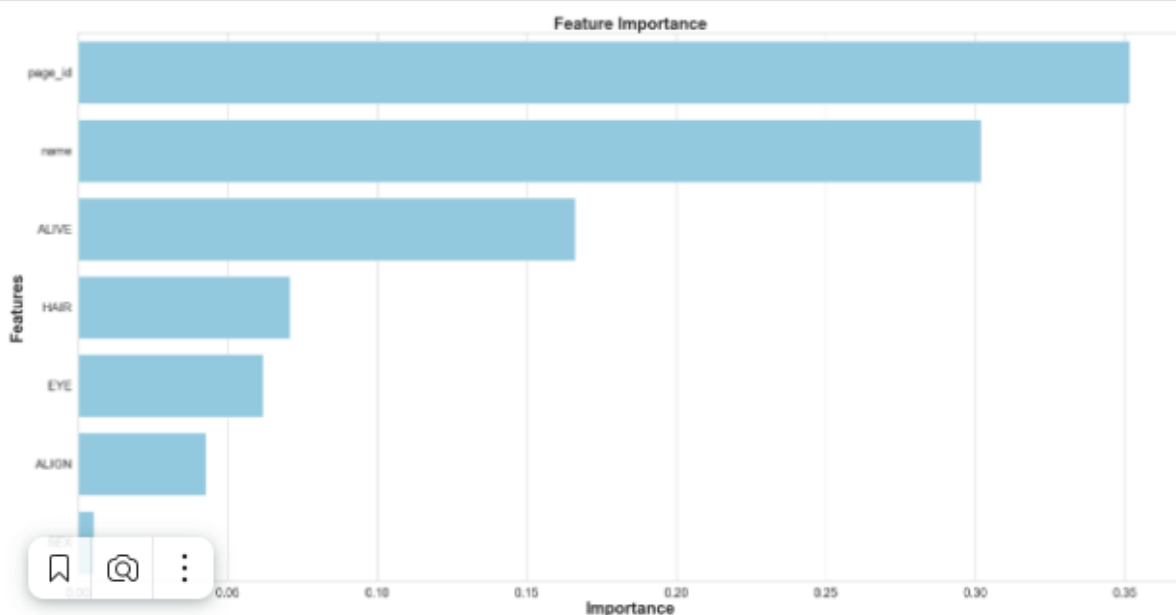
Итак, можем сказать, что модель случайного леса имеет результат лучше, чем модель дерева решений.

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}).

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.

```
1 import seaborn as sns
2 feats = {}
3 for feature, importance in zip(ds.columns, dt_model.feature_importances_):
4     feats[feature] = importance
5 importances = pd.DataFrame.from_dict(feats, orient='index').rename(columns={
6 importances = importances.sort_values(by='Gini-Importance', ascending=False)
7 importances = importances.reset_index()
8 importances = importances.rename(columns={'index': 'Features'})
9 sns.set(font_scale = 5)
10 sns.set(style="whitegrid", color_codes=True, font_scale = 1.7)
11 fig, ax = plt.subplots()
12 fig.set_size_inches(30,15)
13 sns.barplot(x=importances['Gini-Importance'], y=importances['Features'], data=
14 plt.xlabel('Importance', fontsize=25, weight = 'bold')
15 plt.ylabel('Features', fontsize=25, weight = 'bold')
16 plt.title('Feature Importance', fontsize=25, weight = 'bold')
17 display(plt.show())
18 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 |

```

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

```

