


Сомов К. А. ИУ5-61Б

```
import kagglehub
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

path = kagglehub.dataset_download("fivethirtyeight/fivethirtyeight-comic-charact
data = pd.read_csv(f"{path}/marvel-wikia-data.csv")
```

data.head()



	page_id	name	urlslug	ID	ALIGN	
0	1678	Spider-Man (Peter Parker)	VSpider-Man_(Peter_Parker)	Secret Identity	Good Characters	Hi
1	7139	Captain America (Steven Rogers)	VCaptain_America_(Steven_Rogers)	Public Identity	Good Characters	E
2	64786	Wolverine (James "Logan" Howlett)	VWolverine_(James_%22Logan%22_Howlett)	Public Identity	Neutral Characters	E

Далее:

[🔗 Посмотреть рекомендованные графики](#)

[New interactive sheet](#)

```
data.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
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score, f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

# 1. Удалим ненужные столбцы
data = data.drop(columns=["page_id", "name", "urlslug", "FIRST APPEARANCE", "GS

# 2. Удалим строки с пустым ALIVE
data = data.dropna(subset=["ALIVE"])

# 3. Заполним пропуски
for col in ["ID", "ALIGN", "EYE", "HAIR", "SEX"]:
    data[col] = data[col].fillna("Unknown")

data["APPEARANCES"] = data["APPEARANCES"].fillna(data["APPEARANCES"].median())
data["Year"] = data["Year"].fillna(data["Year"].median())

# 4. Кодировем ALIVE
data["ALIVE"] = data["ALIVE"].map({"Living Characters": 1, "Deceased Characters

# 5. Разделим признаки и целевую переменную
X = data.drop(columns=["ALIVE"])
y = data["ALIVE"]

# One-hot encoding категориальных признаков
X = pd.get_dummies(X)

# 6. Разделим на train и test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random
```

```
# Логистическая регрессия
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_test)

# Случайный лес
rf_model = RandomForestClassifier(random_state=13)
rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)

# Оценка качества
print("Logistic Regression:")
print("Accuracy:", accuracy_score(y_test, lr_preds))
print("F1 Score:", f1_score(y_test, lr_preds))

print("\nRandom Forest:")
print("Accuracy:", accuracy_score(y_test, rf_preds))
print("F1 Score:", f1_score(y_test, rf_preds))
```

➡ /usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:4
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(

Logistic Regression:

Accuracy: 0.7648854961832061

F1 Score: 0.8667820069204152

Random Forest:

Accuracy: 0.736793893129771

F1 Score: 0.8401335311572701

Оценка качества моделей

Для оценки качества построенных моделей использовались следующие метрики:

- **Accuracy (точность классификации)** — доля правильно предсказанных примеров среди всех.
- **F1-мера** — гармоническое среднее между полнотой и точностью. Она особенно полезна при несбалансированных классах, так как учитывает как ложные положительные, так и ложные отрицательные срабатывания.

Метрики выбраны потому, что:

- **Accuracy** даёт общую оценку качества классификации.
- **F1-мера** показывает, насколько хорошо модель справляется с балансом между ложными срабатываниями и пропущенными положительными примерами — это особенно важно при возможной дисбалансировке классов (например, если "мертвых" персонажей меньше, чем "живых").

Результаты:

Модель	Accuracy	F1 Score
Логистическая регрессия	0.765	0.867
Случайный лес	0.737	0.840

Выводы

- **Логистическая регрессия показала более высокое качество классификации**, чем случайный лес, как по метрике accuracy, так и по F1-мере.
- Несмотря на то, что случайный лес — более гибкая и мощная модель, **в данном случае она уступает простой линейной модели**. Возможные причины:
 - переобучение случайного леса на неочищенных или шумных признаках;
 - линейная разделимость классов в выбранном признаковом пространстве;
 - высокая доля пропущенных значений в изначальных данных, которую линейная модель "переживает" лучше.

