

Обработка и анализ фильмов

Импортируем библиотеки для работы с данными

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

import warnings
warnings.filterwarnings("ignore")
```

Импортируем данные и выведем некоторые строки

```
df = pd.read_csv("The 500 best films according to Kinopoisk.csv")
df.head()
```

| | Название | Год | Рейтинг | Длительность | \ |
|---|-------------------|------|---------|--------------|---|
| 0 | 1+1 | 2011 | 8.4 | 112 | |
| 1 | Интерстеллар | 2014 | 8.3 | 169 | |
| 2 | Побег из Шоушенка | 1994 | 8.2 | 142 | |
| 3 | Остров проклятых | 2009 | 8.1 | 138 | |
| 4 | Зеленая миля | 1999 | 8.1 | 189 | |

| | Ссылка на фильм | \ |
|---|---|---|
| 0 | https://www.kinopoisk.ru/film/535341/ | |
| 1 | https://www.kinopoisk.ru/film/258687/ | |
| 2 | https://www.kinopoisk.ru/film/326/ | |
| 3 | https://www.kinopoisk.ru/film/397667/ | |
| 4 | https://www.kinopoisk.ru/film/435/ | |

| | Описание |
|---|---|
| 0 | Пострадав в результате несчастного случая, бог... |
| 1 | Когда засуха, пыльные бури и вымирание растени... |
| 2 | Бухгалтер Энди Дюфрейн обвинён в убийстве собс... |
| 3 | Два американских судебных пристава отправляютс... |
| 4 | Пол Эджкомб – начальник блока смертников в тю... |

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 6 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Название        500 non-null    object
1   Год              500 non-null    int64
2   Рейтинг         500 non-null    float64
```

```
3    Длительность      500 non-null    int64
4    Ссылка на фильм   500 non-null    object
5    Описание          500 non-null    object
dtypes: float64(1), int64(2), object(3)
memory usage: 23.6+ KB
```

Предварительная обработка данных

Очистим данных для их последующей векторизации:

- Приведение к нижнему регистру
- Очистка от латинских символов
- Очистка от пунктуации и спец символов
- Очистка от цифр
- Очистка от лишних пробеллов
- Очистка от стоп слов
- Tokenизация
- Лемматизация

Импортируем библиотеки для предобработка данных

```
import re
import string

import nltk
import pymorphy3
import wordcloud

stopwords = nltk.corpus.stopwords.words("russian") + [
    "который", "весь", "всё", "это", "свой", "мочь", "история",
    "год", "человек", "самый", "день", "молодой", "хороший"
]

punctuation = string.punctuation + "_-"
morph = pymorphy3.MorphAnalyzer()
```

Создадим функции для удобной обработки текста

```
def remove_punctuations(text: str) -> str:
    return "".join([char for char in text if char not in punctuation])

def remove_digits(text: str) -> str:
    return "".join([char for char in text if not char.isdigit()])

def remove_latinic(text: str) -> str:
    return re.sub("[a-z]", "", text)

def remove_multiple_spaces(text: str) -> str:
```

```

    return re.sub("\s+", " ", text)

def tokenize(text: str) -> list:
    return re.split("\W+", text)

def lemmatize(tokenized_text: list) -> list:
    return [morph.parse(word)[0].normal_form for word in
tokenized_text]

def remove_stop_words(tokenized_text: list) -> list:
    return [word for word in tokenized_text if word not in stopwords]

def join_to_string(tokenized_text: list) -> str:
    return " ".join(tokenized_text)

```

Поочерёдно применим все эти функции

```

df["Предобработанный текст"] = df["Описание"].apply(lambda x:
remove_punctuations(x.lower()))
df.head()

```

| | Название | Год | Рейтинг | Длительность | \ |
|---|-------------------|------|---------|--------------|---|
| 0 | 1+1 | 2011 | 8.4 | 112 | |
| 1 | Интерстеллар | 2014 | 8.3 | 169 | |
| 2 | Побег из Шоушенка | 1994 | 8.2 | 142 | |
| 3 | Остров проклятых | 2009 | 8.1 | 138 | |
| 4 | Зеленая миля | 1999 | 8.1 | 189 | |

| | Ссылка на фильм | \ |
|---|---|---|
| 0 | https://www.kinopoisk.ru/film/535341/ | |
| 1 | https://www.kinopoisk.ru/film/258687/ | |
| 2 | https://www.kinopoisk.ru/film/326/ | |
| 3 | https://www.kinopoisk.ru/film/397667/ | |
| 4 | https://www.kinopoisk.ru/film/435/ | |

| | Описание | \ |
|---|---|---|
| 0 | Пострадав в результате несчастного случая, бог... | |
| 1 | Когда засуха, пыльные бури и вымирание растени... | |
| 2 | Бухгалтер Энди Дюфрейн обвинён в убийстве собс... | |
| 3 | Два американских судебных пристава отправляютс... | |
| 4 | Пол Эджкомб – начальник блока смертников в тюр... | |

| | Предобработанный текст |
|---|---|
| 0 | пострадав в результате несчастного случая бога... |
| 1 | когда засуха пыльные бури и вымирание растений... |
| 2 | бухгалтер энди дюфрейн обвинён в убийстве собс... |
| 3 | два американских судебных пристава отправляютс... |
| 4 | пол эджкомб начальник блока смертников в тюрь... |

```
df["Предобработанный текст"] = df["Предобработанный
текст"].apply(lambda x: remove_digits(x))
df.head()
```

| | Название | Год | Рейтинг | Длительность \ |
|---|-------------------|------|---------|----------------|
| 0 | 1+1 | 2011 | 8.4 | 112 |
| 1 | Интерстеллар | 2014 | 8.3 | 169 |
| 2 | Побег из Шоушенка | 1994 | 8.2 | 142 |
| 3 | Остров проклятых | 2009 | 8.1 | 138 |
| 4 | Зеленая миля | 1999 | 8.1 | 189 |

| | Ссылка на фильм \ |
|---|---|
| 0 | https://www.kinopoisk.ru/film/535341/ |
| 1 | https://www.kinopoisk.ru/film/258687/ |
| 2 | https://www.kinopoisk.ru/film/326/ |
| 3 | https://www.kinopoisk.ru/film/397667/ |
| 4 | https://www.kinopoisk.ru/film/435/ |

| | Описание \ |
|---|---|
| 0 | Пострадав в результате несчастного случая, бог... |
| 1 | Когда засуха, пыльные бури и вымирание растени... |
| 2 | Бухгалтер Энди Дюфрейн обвинён в убийстве собс... |
| 3 | Два американских судебных пристава отправляютс... |
| 4 | Пол Эджкомб – начальник блока смертников в тюр... |

| | Предобработанный текст |
|---|---|
| 0 | пострадав в результате несчастного случая бога... |
| 1 | когда засуха пыльные бури и вымирание растений... |
| 2 | бухгалтер энди дюфрейн обвинён в убийстве собс... |
| 3 | два американских судебных пристава отправляютс... |
| 4 | пол эджкомб начальник блока смертников в тюрь... |

```
df["Предобработанный текст"] = df["Предобработанный
текст"].apply(lambda x: remove_latinic(x))
df.head()
```

| | Название | Год | Рейтинг | Длительность \ |
|---|-------------------|------|---------|----------------|
| 0 | 1+1 | 2011 | 8.4 | 112 |
| 1 | Интерстеллар | 2014 | 8.3 | 169 |
| 2 | Побег из Шоушенка | 1994 | 8.2 | 142 |
| 3 | Остров проклятых | 2009 | 8.1 | 138 |
| 4 | Зеленая миля | 1999 | 8.1 | 189 |

| | Ссылка на фильм \ |
|---|---|
| 0 | https://www.kinopoisk.ru/film/535341/ |
| 1 | https://www.kinopoisk.ru/film/258687/ |
| 2 | https://www.kinopoisk.ru/film/326/ |
| 3 | https://www.kinopoisk.ru/film/397667/ |
| 4 | https://www.kinopoisk.ru/film/435/ |

| | Описание \ |
|---|---|
| 0 | Пострадав в результате несчастного случая, бог... |
| 1 | Когда засуха, пыльные бури и вымирание растени... |
| 2 | Бухгалтер Энди Дюфрейн обвинён в убийстве собс... |
| 3 | Два американских судебных пристава отправляютс... |
| 4 | Пол Эджкомб – начальник блока смертников в тюр... |

| | Предобработанный текст |
|---|---|
| 0 | пострадав в результате несчастного случая бога... |
| 1 | когда засуха пыльные бури и вымирание растений... |
| 2 | бухгалтер энди дюфрейн обвинён в убийстве собс... |
| 3 | два американских судебных пристава отправляютс... |
| 4 | пол эджкомб начальник блока смертников в тюрь... |

```
df["Предобработанный текст"] = df["Предобработанный
текст"].apply(lambda x: remove_multiple_spaces(x))
df.head()
```

| | Название | Год | Рейтинг | Длительность \ |
|---|-------------------|------|---------|----------------|
| 0 | 1+1 | 2011 | 8.4 | 112 |
| 1 | Интерстеллар | 2014 | 8.3 | 169 |
| 2 | Побег из Шоушенка | 1994 | 8.2 | 142 |
| 3 | Остров проклятых | 2009 | 8.1 | 138 |
| 4 | Зеленая миля | 1999 | 8.1 | 189 |

| | Ссылка на фильм \ |
|---|---|
| 0 | https://www.kinopoisk.ru/film/535341/ |
| 1 | https://www.kinopoisk.ru/film/258687/ |
| 2 | https://www.kinopoisk.ru/film/326/ |
| 3 | https://www.kinopoisk.ru/film/397667/ |
| 4 | https://www.kinopoisk.ru/film/435/ |

| | Описание \ |
|---|---|
| 0 | Пострадав в результате несчастного случая, бог... |
| 1 | Когда засуха, пыльные бури и вымирание растени... |
| 2 | Бухгалтер Энди Дюфрейн обвинён в убийстве собс... |
| 3 | Два американских судебных пристава отправляютс... |
| 4 | Пол Эджкомб – начальник блока смертников в тюр... |

| | Предобработанный текст |
|---|---|
| 0 | пострадав в результате несчастного случая бога... |
| 1 | когда засуха пыльные бури и вымирание растений... |
| 2 | бухгалтер энди дюфрейн обвинён в убийстве собс... |
| 3 | два американских судебных пристава отправляютс... |
| 4 | пол эджкомб начальник блока смертников в тюрьм... |

```
df["Предобработанный текст"] = df["Предобработанный
текст"].apply(lambda x: tokenize(x))
df.head()
```

| | Название | Год | Рейтинг | Длительность | \ |
|---|-------------------|------|---------|--------------|---|
| 0 | 1+1 | 2011 | 8.4 | 112 | |
| 1 | Интерстеллар | 2014 | 8.3 | 169 | |
| 2 | Побег из Шоушенка | 1994 | 8.2 | 142 | |
| 3 | Остров проклятых | 2009 | 8.1 | 138 | |
| 4 | Зеленая миля | 1999 | 8.1 | 189 | |

| | Ссылка на фильм | \ |
|---|---|---|
| 0 | https://www.kinopoisk.ru/film/535341/ | |
| 1 | https://www.kinopoisk.ru/film/258687/ | |
| 2 | https://www.kinopoisk.ru/film/326/ | |
| 3 | https://www.kinopoisk.ru/film/397667/ | |
| 4 | https://www.kinopoisk.ru/film/435/ | |

| | Описание | \ |
|---|---|---|
| 0 | Пострадав в результате несчастного случая, бог... | |
| 1 | Когда засуха, пыльные бури и вымирание растени... | |
| 2 | Бухгалтер Энди Дюфрейн обвинён в убийстве собс... | |
| 3 | Два американских судебных пристава отправляютс... | |
| 4 | Пол Эджкомб – начальник блока смертников в тюр... | |

| | Предобработанный текст |
|---|---|
| 0 | [пострадав, в, результате, несчастного, случая... |
| 1 | [когда, засуха, пыльные, бури, и, вымирание, р... |
| 2 | [бухгалтер, энди, дюфрейн, обвинён, в, убийств... |
| 3 | [два, американских, судебных, пристава, отправ... |
| 4 | [пол, эджкомб, начальник, блока, смертников, в... |

```
%%time
df["Предобработанный текст"] = df["Предобработанный
текст"].apply(lambda x: lemmatize(x))
df.head()
```

```
CPU times: total: 2.75 s
Wall time: 2.79 s
```

| | Название | Год | Рейтинг | Длительность | \ |
|---|-------------------|------|---------|--------------|---|
| 0 | 1+1 | 2011 | 8.4 | 112 | |
| 1 | Интерстеллар | 2014 | 8.3 | 169 | |
| 2 | Побег из Шоушенка | 1994 | 8.2 | 142 | |
| 3 | Остров проклятых | 2009 | 8.1 | 138 | |
| 4 | Зеленая миля | 1999 | 8.1 | 189 | |

| | Ссылка на фильм | \ |
|---|---|---|
| 0 | https://www.kinopoisk.ru/film/535341/ | |
| 1 | https://www.kinopoisk.ru/film/258687/ | |
| 2 | https://www.kinopoisk.ru/film/326/ | |
| 3 | https://www.kinopoisk.ru/film/397667/ | |
| 4 | https://www.kinopoisk.ru/film/435/ | |

| | Описание \ |
|---|---|
| 0 | Пострадав в результате несчастного случая, бог... |
| 1 | Когда засуха, пыльные бури и вымирание растени... |
| 2 | Бухгалтер Энди Дюфрейн обвинён в убийстве собс... |
| 3 | Два американских судебных пристава отправляютс... |
| 4 | Пол Эджкомб – начальник блока смертников в тю... |

| | Предобработанный текст |
|---|--|
| 0 | [пострадать, в, результат, несчастный, случай, ... |
| 1 | [когда, засуха, пыльный, буря, и, вымирание, р... |
| 2 | [бухгалтер, энди, дюфрейн, обвинить, в, убийст... |
| 3 | [два, американский, судебный, пристав, отправл... |
| 4 | [пол, эджкомба, начальник, блок, смертник, в, ... |

```
df["Предобработанный текст"] = df["Предобработанный
текст"].apply(lambda x: remove_stop_words(x))
df.head()
```

| | Название | Год | Рейтинг | Длительность \ |
|---|-------------------|------|---------|----------------|
| 0 | 1+1 | 2011 | 8.4 | 112 |
| 1 | Интерстеллар | 2014 | 8.3 | 169 |
| 2 | Побег из Шоушенка | 1994 | 8.2 | 142 |
| 3 | Остров проклятых | 2009 | 8.1 | 138 |
| 4 | Зеленая миля | 1999 | 8.1 | 189 |

| | Ссылка на фильм \ |
|---|---|
| 0 | https://www.kinopoisk.ru/film/535341/ |
| 1 | https://www.kinopoisk.ru/film/258687/ |
| 2 | https://www.kinopoisk.ru/film/326/ |
| 3 | https://www.kinopoisk.ru/film/397667/ |
| 4 | https://www.kinopoisk.ru/film/435/ |

| | Описание \ |
|---|---|
| 0 | Пострадав в результате несчастного случая, бог... |
| 1 | Когда засуха, пыльные бури и вымирание растени... |
| 2 | Бухгалтер Энди Дюфрейн обвинён в убийстве собс... |
| 3 | Два американских судебных пристава отправляютс... |
| 4 | Пол Эджкомб – начальник блока смертников в тю... |

| | Предобработанный текст |
|---|--|
| 0 | [пострадать, результат, несчастный, случай, бо... |
| 1 | [засуха, пыльный, буря, вымирание, растение, п... |
| 2 | [бухгалтер, энди, дюфрейн, обвинить, убийство, ... |
| 3 | [американский, судебный, пристав, отправляться... |
| 4 | [пол, эджкомба, начальник, блок, смертник, тю... |

```
%time
df["Предобработанный текст"] = df["Предобработанный
текст"].apply(lambda x: join_to_string(x))
df.head()
```

CPU times: total: 0 ns
Wall time: 996 µs

| | Название | Год | Рейтинг | Длительность | \ |
|---|-------------------|------|---------|--------------|---|
| 0 | 1+1 | 2011 | 8.4 | 112 | |
| 1 | Интерстеллар | 2014 | 8.3 | 169 | |
| 2 | Побег из Шоушенка | 1994 | 8.2 | 142 | |
| 3 | Остров проклятых | 2009 | 8.1 | 138 | |
| 4 | Зеленая миля | 1999 | 8.1 | 189 | |

| | Ссылка на фильм | \ |
|---|---|---|
| 0 | https://www.kinopoisk.ru/film/535341/ | |
| 1 | https://www.kinopoisk.ru/film/258687/ | |
| 2 | https://www.kinopoisk.ru/film/326/ | |
| 3 | https://www.kinopoisk.ru/film/397667/ | |
| 4 | https://www.kinopoisk.ru/film/435/ | |

| | Описание | \ |
|---|---|---|
| 0 | Пострадав в результате несчастного случая, бог... | |
| 1 | Когда засуха, пыльные бури и вымирание растени... | |
| 2 | Бухгалтер Энди Дюфрейн обвинён в убийстве собс... | |
| 3 | Два американских судебных пристава отправляютс... | |
| 4 | Пол Эджкомб – начальник блока смертников в тю... | |

| | Предобработанный текст |
|---|---|
| 0 | пострадать результат несчастный случай богатый... |
| 1 | засуха пыльный буря вымирание растение приводи... |
| 2 | бухгалтер энди дюфрейн обвинить убийство собст... |
| 3 | американский судебный пристав отправляться ост... |
| 4 | пол эджкомба начальник блок смертник тюрьма хо... |

Теперь, после предварительной обработки текста можно отобразить облако слов и расширить список стоп слов.

```
%%time
from wordcloud import WordCloud

text = " ".join(word for word in df["Предобработанный текст"])

plt.figure(figsize=(8, 8))
plt.imshow(WordCloud(background_color="black", width=2000,
height=2000, random_state=42).generate(text))
plt.show()
```



```

%%time
from sklearn.feature_extraction.text import TfidfVectorizer

# Создаём и обучаем модель
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(df["Предобработанный
текст"])

CPU times: total: 15.6 ms
Wall time: 29.1 ms

tfidf_matrix.shape

(500, 6213)

tfidf_vectorizer.get_feature_names_out()[:100]

array(['аарон', 'аббат', 'абдулл', 'абрамс', 'абсолютно',
      'абсолютный',
      'абсурдный', 'авантюрист', 'авария', 'август',
      'авиаконструктор',
      'авиалиния', 'авиация', 'аврелий', 'автобанда',
      'автобиография',
      'автобус', 'автобусный', 'автограф', 'автокатастрофа',
      'автоконструктор', 'автомат', 'автомеханик', 'автомобиль',
      'автомобильный', 'автопокрышка', 'автор', 'авторитет',
      'авторитетный', 'агаемнон', 'агент', 'агентство',
      'агрессивный',
      'ад', 'адалин', 'адам', 'адвокат', 'адил', 'адольф', 'адрес',
      'адриан', 'адриана', 'азиатский', 'азкабан', 'аиша', 'айова',
      'айрис', 'айсберг', 'ак', 'академия', 'акитаин', 'аккуратно',
      'акт', 'активистка', 'активно', 'активный', 'актёр', 'акция',
      'алан', 'албанский', 'алгоритм', 'алекс', 'александр',
      'алексей',
      'алеш', 'алиби', 'алкоголь', 'алмаата', 'алмаз', 'алонзый',
      'алтарь', 'альгрена', 'альма', 'альтернативный', 'альфред',
      'аманда', 'амано', 'амели', 'америка', 'американец',
      'американка',
      'американский', 'амидала', 'амидале', 'амнезия',
      'амнистировать',
      'аналогичный', 'анархист', 'ангел', 'английский', 'англичанин',
      'англия', 'андерсон', 'андре', 'андрей', 'андроид', 'анна',
      'антарктика', 'антикварный', 'антикриминальный'], dtype=object)

```

Кластеризация

Теперь, так как текст векторизирован, можно кластеризовать данные

Для кластеризации будем использовать модель KMeans. Алгоритм k-means используют для группировки объектов в кластеры на основе их схожести. В основе работы k-means лежит принцип минимизации расстояния между объектами внутри одного кластера.

Основные преимущества k-means это простота и быстрота реализации, а так же эффективность при работе с большими наборами данных

Для начала уменьшим размерность матрицы. Это упростит обучение моделей путём уменьшения вычислительной сложности

```
from sklearn.decomposition import NMF
from sklearn.cluster import KMeans

nmf_model = NMF(n_components=5, random_state=0)
W = nmf_model.fit_transform(tfidf_matrix)

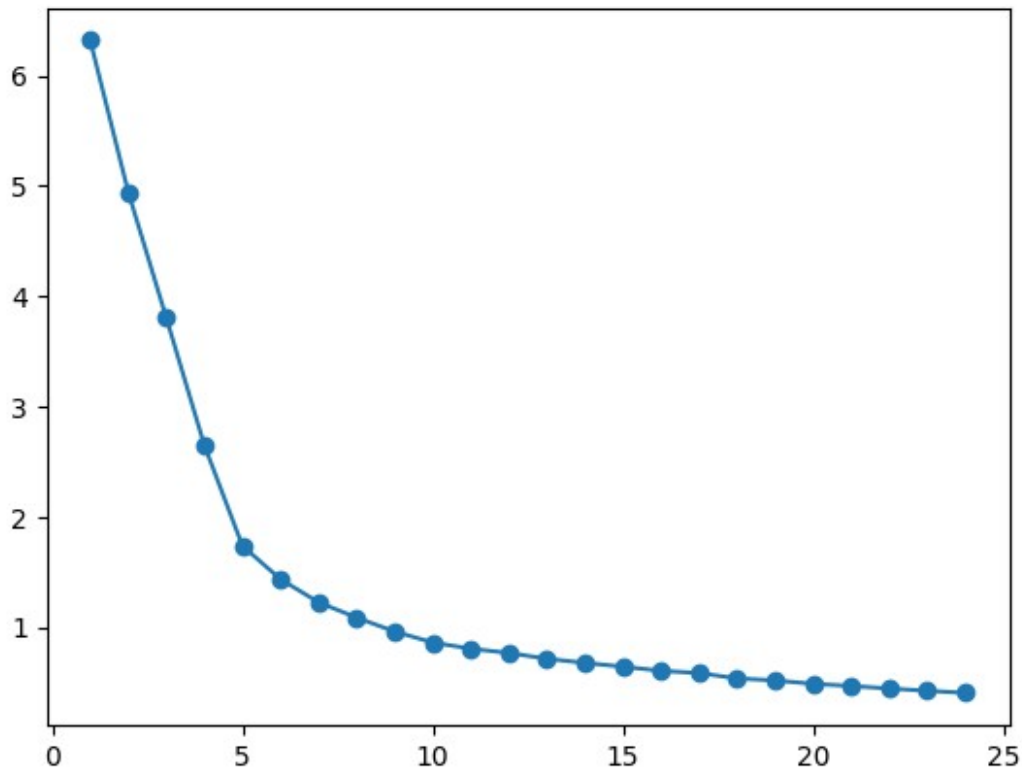
%%time

cluster_range = range(1, 25)
wcss = []

# Обучаем модели несколько раз
for i in cluster_range:
    # Создаём и обучаем модель
    kmeans = KMeans(n_clusters=i, n_init=10, init="k-means++",
                    random_state=42).fit(W)
    wcss.append(kmeans.inertia_)

CPU times: total: 5.8 s
Wall time: 1.48 s

plt.plot(cluster_range, wcss, marker="o")
plt.show()
```

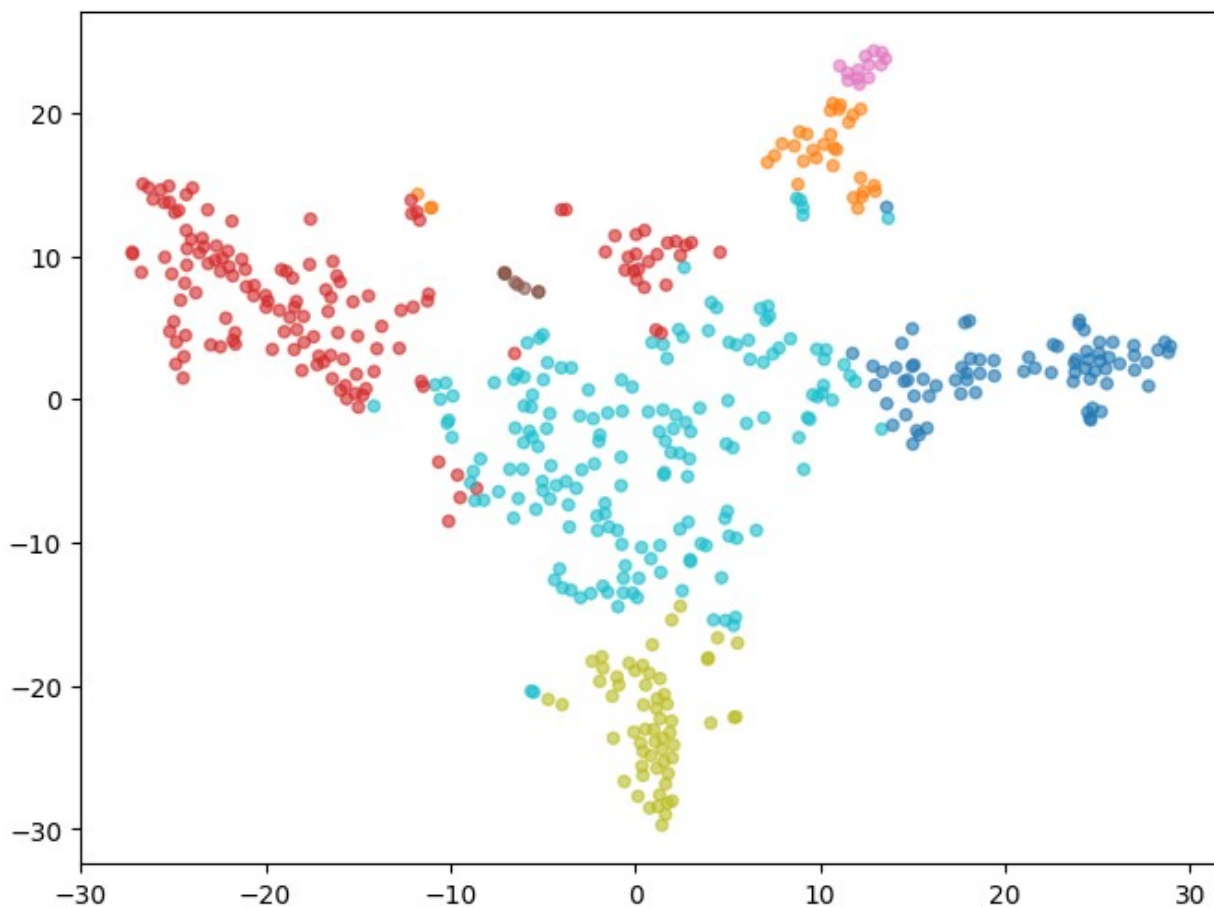


```
kmeans = KMeans(n_clusters=7, n_init=10, random_state=42)
labels = kmeans.fit_predict(W)
centers = kmeans.cluster_centers_
```

Визуализируем данные, каждый цвет прикреплен к кластеру. Так же отобразим центры кластеров красными крестиками

```
from sklearn.manifold import TSNE
x_tsne = TSNE(n_components=2, random_state=42).fit_transform(W)

plt.figure(figsize=(8, 6))
plt.scatter(x_tsne[:, 0], x_tsne[:, 1], c=labels, s=20, cmap="tab10",
            alpha=0.6)
plt.show()
```



По данному графику распределения видно, что 1 кластер расположен на большом удалении и содержит малое количество записей. Добавим кластеры в исходный набор данных

Добавим информацию о принадлежности к кластеру в набор данных

```
df["cluster"] = labels
df.sample()
```

| | Название | Год | Рейтинг | \ |
|-----|---|---|---------|---|
| 487 | Индиана Джонс: В поисках утраченного ковчега | 1981 | 6.9 | |
| | Длительность | Ссылка на фильм | | \ |
| 487 | 115 | https://www.kinopoisk.ru/film/339/ | | |
| | Описание | | | \ |
| 487 | Известный археолог и специалист по оккультным ... | | | |
| | Предобработанный текст | cluster | | |
| 487 | известный археолог специалист оккультный наука... | 5 | | |

Выведем распределение кластеров

```
df["cluster"].value_counts()

cluster
6      169
2      142
0       76
5       60
1       32
4       13
3        8
Name: count, dtype: int64
```

Теперь выведем несколько записей для каждого кластера

```
pd.set_option("display.max_colwidth", None)

for i in range(7):
    print(f"cluster: {i}: ")
    print(f"{df[df['cluster'] == i].sample(2)[['Название',
'Описание']].values}\n")

cluster: 0:
[['Планета Ка-Пэкс'
  'В Манхэттенский психиатрический институт привозят странного
человека в черных очках. Он зовет себя Протом и утверждает, что его
родина — далекая планета Ка-Пэкс, откуда он мгновенно перенесся на
Землю в луче света. Несмотря на все усилия, опытному доктору Пауэллу
не удастся разгадать загадку таинственного пациента, который охотно и
весьма убедительно доказывает всем свое внеземное происхождение и
заранее назначает дату своего возвращения на Ка-Пэкс.']]
['Звёздные войны: Эпизод 3 — Месть ситхов'
  'Идёт третий год Войн клонов. Галактическая Республика, некогда
бывшая спокойным и гармоничным государством, превратилась в поле битвы
между армиями клонов, возглавляемых канцлером Палпатином, и армадами
дроидов, которых ведёт граф Дуку, тёмный лорд ситхов. Республика
медленно погружается во тьму. Лишь рыцари-джедаи, защитники мира и
справедливости, могут противостоять злу, которое вскоре поглотит
галактику. Но настоящая битва идёт в душе у молодого рыцаря-джедая
Энакина, который разрывается между долгом джедая и любовью к своей
жене, сенатору Падме Амидале. И от того, какое чувство в нём победит,
зависит будущее всего мира.']]

cluster: 1:
[['Омерзительная восьмерка'
  'США после Гражданской войны. Легендарный охотник за головами Джон
Рут по кличке Вешатель конвоирует заключенную. По пути к ним
прибываются еще несколько путешественников. Снежная буря вынуждает
компанию искать укрытие в лавке на отшибе, где уже расположилось
весьма пёстрое общество: генерал конфедератов, мексиканец, ковбой... И
один из них — не тот, за кого себя выдает.']]
```

['Рок-волна'

'История о диджеях британского пиратского радиошоу 60-х. Сюжет фильма рассказывает правдивую историю британской пиратской радиостанции, которая вещала с корабля в северном море, в то время как BBC выдавала в эфир два часа поп-музыки в неделю.']]

cluster: 2:

[['Капитан Фантастик'

'Бен живёт в лесу с шестью своими детьми. Они говорят на нескольких языках, разбираются в квантовой физике, философии и литературе, а вместо Рождества отмечают день рождения Ноама Хомского. Они умеют охотиться, способны разделить добычу и приготовить её на костре, а также находятся в прекрасной физической форме. Когда их мать, лежащая в больнице, кончает с собой, всему семейству приходится отправиться в большой мир на её похороны.']]

['Чудо'

'С одной стороны мальчик Август Пулман такой же как и другие мальчишки его возраста - любит ходить на дни рождения к друзьям, играть в компьютерные игры, фанатеет от «Звездных войн», играет со своей собакой, ссорится и мирится со старшей сестрой. А с другой - он совсем не такой как другие мальчишки его возраста. Во-первых, Август никогда не ходил в обычную школу - с первого класса с ним дома занималась мама. Во-вторых, Август перенес 27 операций. Из-за очень редкой, но иногда встречающейся генетической ошибки у Августа нет лица. И вот такой мальчик должен пойти в школу. В первый раз. К обычным детям.']]

cluster: 3:

[['Гарри Поттер и узник Азкабана'

'Гарри, Рон и Гермиона возвращаются на третий курс школы чародейства и волшебства Хогвартс. На этот раз они должны раскрыть тайну узника, сбежавшего из тюрьмы Азкабан, чье пребывание на воле создает для Гарри смертельную опасность.']]

['Гарри Поттер и Дары Смерти: Часть II'

'Битва между добрыми и злыми силами мира волшебников перерастает во всеобщую войну. Ставки ещё никогда не были так высоки, а поиск убежища – столь сложен. И, быть может, именно Гарри Поттеру придется пожертвовать всем в финальном сражении с Волан-де-Мортом.']]

cluster: 4:

[['Назад в будущее 2'

'Продолжение фантастической истории о приключениях американского подростка во времени. На этот раз с помощью модернизированной Доком машины времени Марти из 80-х попадает в будущее.']]

['Бесславные ублюдки'

'Вторая мировая война. В оккупированной немцами Франции группа американских солдат-евреев наводит страх на нацистов, жестоко убивая и скальпируя солдат.']]

cluster: 5:


```

[['Васаби'
 'Исполняя последнюю волю бывшей любовницы, французский полицейский
 Юбер Фиорентини отправляется в Японию, где ему приходится встретиться
 со старым другом и дочерью, о существовании которой он даже не
 подозревал. А также получить наследство в 200 млн. долларов и
 познакомиться с якудза, которые охотятся за этими деньгами.']]
['Законопослушный гражданин'
 'Семью Клайда Шелтона убивают двое грабителей. Однако они не
 получают по заслугам – прокурор идет на сделку с преступниками. Тогда
 Клайд решает по-своему отомстить убийцам, а также представителям
 власти и юстиции США.']]

cluster: 6:
[['Кентавр'
 'Таксист Саша предпочитает работать по ночам, когда нет пробок и,
 как он считает, люди раскрываются с другой стороны. Очередная его
 клиентка – эскортница Лиза – проникается к парню симпатией и просит
 покатать её по делам. Ночь только начинается, и, кажется, обоим есть,
 что скрывать.']]
['День сурка'
 'Телевизионный комментатор Фил Коннорс каждый год приезжает в
 маленький городок в штате Пенсильвания на празднование Дня сурка. Но
 на этот раз веселье рискует зайти слишком далеко. Время сыграло с ним
 злую шутку: оно взяло да и остановилось.']]

```

Тематическое моделирование

Импортируем библиотеки для тематического моделирования

```

import gensim
import pyLDAvis.gensim_models as gensimvis
import pyLDAvis

```

Для подбора оптимального количества тем построим график зависимости когерентности от количества тем

```

%%time
df["Токенизированный предобработанный текст"] = df["Предобработанный
текст"].apply(lambda x: tokenize(x))

gensim_dictionary = gensim.corpora.Dictionary(df["Токенизированный
предобработанный текст"])
gensim_dictionary.filter_extremes(no_above=0.1, no_below=5)
gensim_dictionary.compactify()

corpus = [gensim_dictionary.doc2bow(text) for text in
df["Токенизированный предобработанный текст"]]
topics_range = range(3, 25, 1)

```



```

coherence_values = []
for num_topics in topics_range:
    lda_model = gensim.models.LdaMulticore(
        corpus=corpus,
        num_topics=num_topics,
        id2word=gensim_dictionary,
        passes=10,
        random_state=42)

    coherence_model = gensim.models.CoherenceModel(
        model=lda_model,
        texts=df["Токенизированный предобработанный текст"],
        dictionary=gensim_dictionary,
        coherence="c_v")

    coherence_values.append(coherence_model.get_coherence())

CPU times: total: 1min 32s
Wall time: 5min 27s

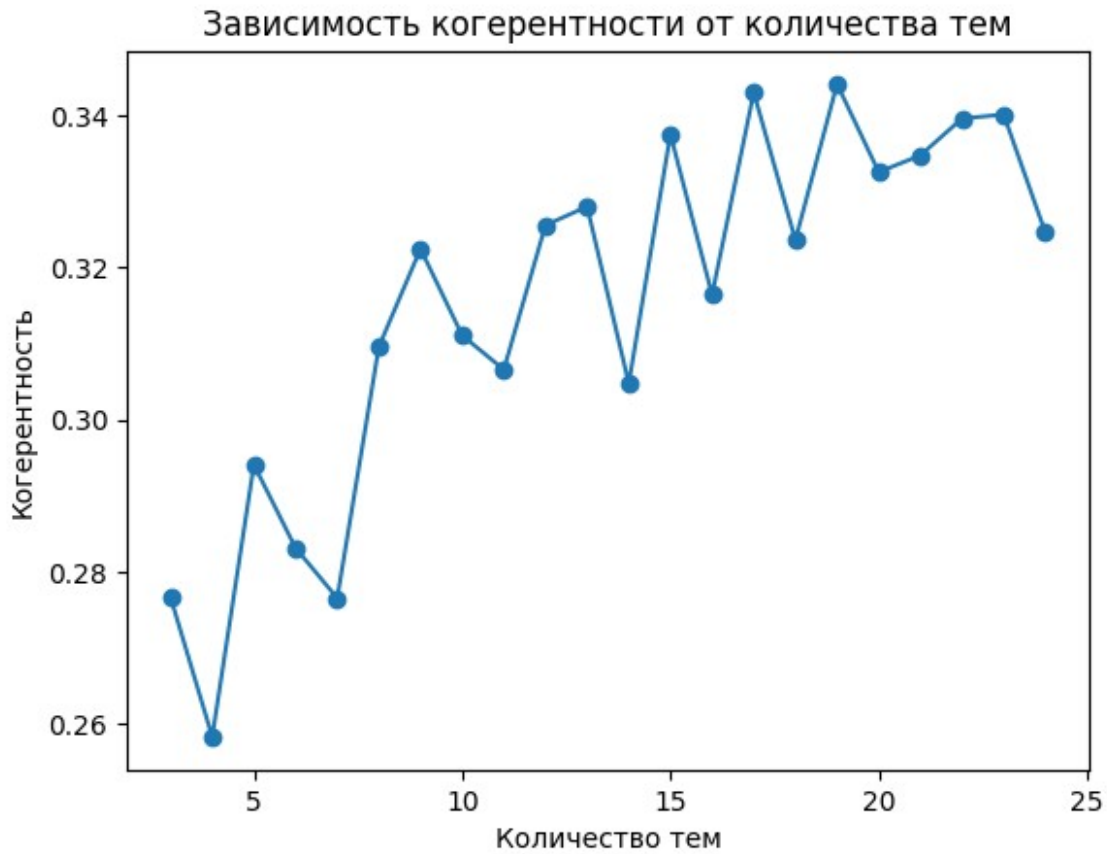
```

Выведем график на основе подсчитанных значений когерентности

```

plt.plot(topics_range, coherence_values, marker="o")
plt.xlabel("Количество тем")
plt.ylabel("Когерентность")
plt.title("Зависимость когерентности от количества тем")
plt.show()

```



На данном графике видны пики, однако при большом количестве темы получаются слишком "узкими"

Воспользуемся NMF (Non-negative Matrix Factorization) для определения скрытых тем

```
for i, topic in enumerate(nmf_model.components_):  
    print(f"Topic {i}: {",  
".join([tfidf_vectorizer.get_feature_names_out()[i] for i in  
topic.argsort()[:-11:-1]])}")
```

Topic 0: жизнь, друг, дом, девушка, однажды, жить, стать, любовь, отец, ребёнок

Topic 1: война, мировой, второй, время, американский, фильм, машина, событие, первый, рассказывать

Topic 2: гарри, школа, хогвартс, поттер, чародейство, волшебство, курс, волшебник, возвращаться, опасность

Topic 3: мир, земля, сила, планета, бой, вынудить, остаться, живой, человечество, уничтожить

Topic 4: полиция, полицейский, банда, убийца, преступление, город, опасный, убить, схватка, убийство

Визуализируем данную информацию

```
lda = gensim.models.LdaMulticore(corpus, num_topics=5,
id2word=gensim_dictionary, passes=10, random_state=42)
vis_data = gensimvis.prepare(lda, corpus, gensim_dictionary)
pyLDAvis.display(vis_data)
```

<IPython.core.display.HTML object>

Выведем наиболее часто встречающиеся слова для каждой темы, что бы понять, какие фильмы находятся в каждой теме

```
for i in range(7):
    gensim_dictionary =
gensim.corpora.Dictionary(df.loc[df["cluster"].isin([i]),
"Токенизированный предобработанный текст"])
    doc_count = len(df.loc[df["cluster"].isin([i]), "Токенизированный
предобработанный текст"])

    if doc_count >= 100:
        gensim_dictionary.filter_extremes(no_above=0.1, no_below=5)
    elif doc_count >= 50:
        gensim_dictionary.filter_extremes(no_above=0.3, no_below=3)
    else:
        gensim_dictionary.filter_extremes(no_above=0.5, no_below=2)

    gensim_dictionary.compactify()

    corpus = [gensim_dictionary.doc2bow(text) for text in
df.loc[df["cluster"].isin([i]), "Токенизированный предобработанный
текст"]]
    lda = gensim.models.LdaMulticore(corpus, num_topics=5,
id2word=gensim_dictionary, passes=10, random_state=42)

    print(f"\nТемы кластера {i}:")

    for j in lda.print_topics():
        print(j)
```

Темы кластера 0:

```
(0, '0.054*"планета" + 0.034*"земля" + 0.027*"живой" +
0.023*"путешествие" + 0.023*"остаться" + 0.021*"космический" +
0.018*"готовый" + 0.017*"новый" + 0.016*"должный" + 0.016*"вынудить"')
(1, '0.038*"земля" + 0.027*"сила" + 0.027*"пытаться" + 0.025*"команда"
+ 0.023*"большой" + 0.022*"бой" + 0.018*"вести" + 0.018*"новый" +
0.018*"победа" + 0.015*"невероятный"')
(2, '0.027*"планета" + 0.027*"найти" + 0.027*"человечество" +
0.027*"сила" + 0.027*"смерть" + 0.027*"невероятный" + 0.026*"странный"
+ 0.018*"мальчик" + 0.018*"остаться" + 0.018*"пытаться"')
(3, '0.034*"вынудить" + 0.030*"решить" + 0.030*"сила" + 0.025*" " +
0.024*"смерть" + 0.024*"битва" + 0.024*"бой" + 0.019*"пока" +
```

0.018*"любовь" + 0.018*"предстоять"')
(4, '0.043*"гном" + 0.031*"бильбо" + 0.031*"опасный" +
0.025*"путешествие" + 0.025*"хоббит" + 0.025*"сила" +
0.025*"волшебник" + 0.019*"оказаться" + 0.019*"герой" + 0.019*"воин"')

Темы кластера 1:

(0, '0.111*"время" + 0.041*"разный" + 0.032*"ход" + 0.032*"век" +
0.022*"событие" + 0.022*"самолёт" + 0.022*"единственный" +
0.022*"оказываться" + 0.022*"настоящий" + 0.022*"машина"')
(1, '0.060*"война" + 0.038*"отправляться" + 0.030*"британский" +
0.030*"второй" + 0.030*"мировой" + 0.030*"время" + 0.030*"капитан" +
0.029*" " + 0.029*"подросток" + 0.029*"великий"')
(2, '0.106*"война" + 0.046*"чей" + 0.046*"становиться" +
0.046*"свидетель" + 0.025*"рассказывать" + 0.025*"друг" +
0.025*"фильм" + 0.025*"эпоха" + 0.025*"бесценный" + 0.025*"европа"')
(3, '0.080*"война" + 0.042*"статья" + 0.042*"фильм" + 0.042*"пока" +
0.042*"произойти" + 0.037*"событие" + 0.023*"рассказывать" +
0.023*"служить" + 0.023*"наступление" + 0.023*"прийтись"')
(4, '0.058*"фильм" + 0.040*"жизнь" + 0.040*"век" + 0.040*"время" +
0.040*"женщина" + 0.040*"действие" + 0.040*"искусство" +
0.022*"машина" + 0.022*"событие" + 0.022*" "')

Темы кластера 2:

(0, '0.037*"встречать" + 0.029*"сын" + 0.028*"знать" +
0.028*"проводить" + 0.028*"возвращаться" + 0.026*"женщина" +
0.024*"летний" + 0.024*"брат" + 0.024*"отправляться" +
0.024*"делать"')
(1, '0.036*"мечтать" + 0.032*"деньга" + 0.026*"маленький" +
0.023*"узнавать" + 0.023*"муж" + 0.023*"начинать" + 0.023*" " +
0.020*"мать" + 0.019*"желание" + 0.019*"большой"')
(2, '0.044*"ещё" + 0.030*"парень" + 0.029*"мама" + 0.026*"летний" +
0.026*"имя" + 0.026*"оказываться" + 0.026*"бывший" + 0.021*"место" +
0.021*"хотеть" + 0.021*"остаться"')
(3, '0.037*"любой" + 0.035*"роман" + 0.033*"парень" +
0.027*"отправляться" + 0.027*"дочь" + 0.027*"семейство" + 0.025*"мать"
+ 0.024*"изменить" + 0.022*"мужчина" + 0.022*"мир"')
(4, '0.037*"мир" + 0.036*"первый" + 0.034*"начинать" + 0.026*"мечта" +
0.026*"путь" + 0.025*"век" + 0.024*"оказываться" + 0.024*"мальчик" +
0.021*"игра" + 0.021*"последний"')

Темы кластера 3:

(0, '0.043*"однако" + 0.043*"схватка" + 0.043*"добрый" + 0.043*"друг"
+ 0.043*"тайна" + 0.043*"смертельный" + 0.043*"воландеморт" +
0.043*"назвать" + 0.043*"чародейство" + 0.043*"волшебство"')
(1, '0.095*"добрый" + 0.095*"схватка" + 0.095*"друг" +
0.095*"воландеморт" + 0.095*"смертельный" + 0.095*"зло" +
0.095*"ждать" + 0.095*"жизнь" + 0.016*"мир" + 0.016*"убежище"')
(2, '0.043*"однако" + 0.043*"схватка" + 0.043*"добрый" + 0.043*"друг"
+ 0.043*"тайна" + 0.043*"смертельный" + 0.043*"назвать" +
0.043*"воландеморт" + 0.043*"волшебство" + 0.043*"чародейство"')

(3, '0.151*"жизнь" + 0.151*"ждать" + 0.082*"однако" + 0.082*"тайна" + 0.082*"назвать" + 0.082*"друг" + 0.082*"схватка" + 0.082*"зло" + 0.014*"воландеморт" + 0.014*"мир"')
(4, '0.072*"курс" + 0.072*"возвращаться" + 0.072*"волшебство" + 0.072*"чародейство" + 0.072*"опасность" + 0.072*"мир" + 0.072*"воландеморт" + 0.049*"гермиона" + 0.049*"рон" + 0.049*"большой"')

Темы кластера 4:

(0, '0.059*"ребёнок" + 0.059*"дружба" + 0.059*"пытаться" + 0.059*"американский" + 0.059*"жизнь" + 0.059*"спасти" + 0.059*"фильм" + 0.059*"машина" + 0.059*"затем" + 0.059*"жить"')
(1, '0.193*"страх" + 0.106*"дружба" + 0.105*"жить" + 0.105*"любовь" + 0.105*"потерять" + 0.105*"убивать" + 0.104*"американский" + 0.018*"ребёнок" + 0.018*"пытаться" + 0.018*"жизнь"')
(2, '0.235*"американский" + 0.128*"убивать" + 0.128*"первый" + 0.128*"затем" + 0.127*"жить" + 0.021*"ребёнок" + 0.021*"дружба" + 0.021*"пытаться" + 0.021*"жизнь" + 0.021*"фильм"')
(3, '0.262*"американский" + 0.260*"машина" + 0.143*"жизнь" + 0.025*"пытаться" + 0.024*"ребёнок" + 0.024*"дружба" + 0.024*"фильм" + 0.024*"спасти" + 0.024*"затем" + 0.024*"жить"')
(4, '0.153*"жизнь" + 0.153*"фильм" + 0.117*"событие" + 0.080*"судьба" + 0.080*"спасти" + 0.080*"ребёнок" + 0.080*"пытаться" + 0.044*"затем" + 0.044*"потерять" + 0.044*"любовь"')

Темы кластера 5:

(0, '0.056*"полицейский" + 0.036*"город" + 0.026*"преступление" + 0.026*"закон" + 0.026*"порядок" + 0.021*"убийство" + 0.021*"становиться" + 0.021*"лосанджелес" + 0.021*"брайан" + 0.020*"жизнь"')
(1, '0.036*"банда" + 0.033*"опасный" + 0.026*"бандит" + 0.026*"парень" + 0.020*"однако" + 0.020*"схватка" + 0.020*""" + 0.020*"расследование" + 0.019*"двое" + 0.017*"наркотик"')
(2, '0.033*"известный" + 0.028*"оказаться" + 0.028*"банда" + 0.028*"помощь" + 0.027*"холмс" + 0.027*"шерлок" + 0.025*"друг" + 0.025*"схватка" + 0.025*"доктор" + 0.023*"бандит"')
(3, '0.050*"задание" + 0.032*"герой" + 0.032*"пытаться" + 0.031*"попасть" + 0.031*"также" + 0.030*"опасный" + 0.029*"агент" + 0.029*"получить" + 0.022*"преступник" + 0.022*"жизнь"')
(4, '0.041*"убийца" + 0.024*"время" + 0.024*"помочь" + 0.024*"летний" + 0.024*"мориарти" + 0.024*"профессор" + 0.018*"преступление" + 0.018*"свидетель" + 0.018*"очередной" + 0.018*"босс"')

Темы кластера 6:

(0, '0.037*"вместе" + 0.035*"становиться" + 0.033*"пытаться" + 0.032*"жена" + 0.030*"статья" + 0.026*"знаменитый" + 0.024*"начинать" + 0.023*"имя" + 0.023*"школа" + 0.023*"""')
(1, '0.052*"оказаться" + 0.038*"каждый" + 0.030*"никто" + 0.030*"дать" + 0.028*"новый" + 0.028*"местный" + 0.027*"получить" + 0.024*"первый" + 0.024*"друг" + 0.024*"бандит"')

```
(2, '0.059*"найти" + 0.055*"преступник" + 0.046*"дело" +
0.032*"работа" + 0.032*"никто" + 0.031*"франция" + 0.029*"последний" +
0.028*"высокий" + 0.024*"прийтись" + 0.024*"хотеть"')
(3, '0.039*"путь" + 0.038*"бывший" + 0.036*"время" + 0.030*" " +
0.025*"игра" + 0.025*"работа" + 0.024*"старший" + 0.024*"команда" +
0.024*"решать" + 0.020*"однако"')
(4, '0.060*"должный" + 0.037*"семья" + 0.036*"ночь" +
0.029*"очередной" + 0.029*"русский" + 0.027*"пытаться" +
0.025*"отправляться" + 0.025*"дело" + 0.021*"школа" + 0.021*"поиск"')
```

Проанализировав полученный результат, по полученным словам можно выделить следующие темы:

- "Городской криминал / Драма"
- "Дружба / Любовь / Испытания"
- "Космические путешествия / Приключения / Судьба"
- "Магия / Волшебство / Эпическая борьба"
- "Война / Мужская доблесть"
- "Любовь / Семейные отношения"
- "Современная жизнь / Преступность / Социальные драмы"

Добавим в набор данных информацию о принадлежности к теме

```
df["cluster name"] = df["cluster"].map({
    0: "Городской криминал / Драма",
    1: "Дружба / Любовь / Испытания",
    2: "Космические путешествия / Приключения / Судьба",
    3: "Магия / Волшебство / Эпическая борьба",
    4: "Война / Мужская доблесть",
    5: "Любовь / Семейные отношения",
    6: "Современная жизнь / Преступность / Социальные драмы",
})

df["cluster name"].value_counts()

cluster name
Современная жизнь / Преступность / Социальные драмы    169
Космические путешествия / Приключения / Судьба          142
Городской криминал / Драма                               76
Любовь / Семейные отношения                              60
Дружба / Любовь / Испытания                             32
Война / Мужская доблесть                                13
Магия / Волшебство / Эпическая борьба                   8
Name: count, dtype: int64

df.to_csv("The 500 best films according to Kinopoisk (cluster).csv",
index=False)

pd.set_option("display.max_colwidth", 50)
```

```
df = pd.read_csv("The 500 best films according to Kinopoisk
(cluster).csv")
df.sample(1)
```

| | Название | Год | Рейтинг | Длительность | Ссылка на фильм \ |
|----|----------|------|---------|--------------|---|
| 34 | Шрек | 2001 | 7.7 | 90 | https://www.kinopoisk.ru/film/430/ |

| | Описание \ |
|----|---|
| 34 | Жил да был в сказочном государстве большой зел... |

| | Предобработанный текст | cluster \ |
|----|---|-----------|
| 34 | жить сказочный государство большой зелёный вел... | 6 |

| | Токенизированный предобработанный текст \ |
|----|---|
| 34 | ['жить', 'сказочный', 'государство', 'большой'... |

| | cluster name |
|----|---|
| 34 | Современная жизнь / Преступность / Социальные ... |

Классификация

Для классификации нам нужно разделить набор данных на обучающую и тестовую выборку. Воспользуемся методом `train_test_split`

```
from sklearn.model_selection import train_test_split

# Оставим 66% для обучающей выборки и 33% для тестовой выборки
X_train, X_test, y_train, y_test = train_test_split(W, df["cluster"],
test_size=0.33, random_state=42)

print(X_train.shape)
print(X_test.shape)

(335, 5)
(165, 5)
```

Для кластеризации возьмём 3 модели:

KNeighbors это алгоритм классификации, который предполагает, что расположенные близко друг к другу объекты принадлежат к одному классу. Данный алгоритм популярен и подходит для нашей задачи, возьмём его для сравнения результатов

Random Forest это множество решающих деревьев, это позволяет снизить переобучения и повысить точность в сравнении с одним деревом.

DecisionTreeClassifier - это класс, способный выполнять многоклассовую классификацию на наборе данных.

Импортируем модели

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

Далее обучим каждую модель

```
%%time
rfc = RandomForestClassifier(random_state=42)
rfc.fit(X_train, y_train);
rfc_predict = rfc.predict(X_test)

CPU times: total: 141 ms
Wall time: 154 ms

%%time
dtc = DecisionTreeClassifier(random_state=42)
dtc.fit(X_train, y_train);
dtc_predict = dtc.predict(X_test)

CPU times: total: 0 ns
Wall time: 1.99 ms

%%time
knn = KNeighborsClassifier()
knn.fit(X_train, y_train);
knn_predict = knn.predict(X_test)

CPU times: total: 31.2 ms
Wall time: 14 ms
```

Модели обучены, теперь сравним их метрики

Для оценки качества моделей необходимо понимать метрики оценки моделей.

Precision и Recall можно описать следующим образом:

- Precision - "Сколько выбранных элементов являются релевантными"
- Recall - "Сколько релевантных элементов выбранно"

Precision демонстрирует способность алгоритма отличать один класс от других классов, а Recall демонстрирует способность алгоритма обнаруживать данный класс.

Precision и recall не зависят, в отличие от accuracy, от соотношения классов и потому применимы в условиях несбалансированных выборок.

F1-score — это гармоническое среднее между Precision и Recall, которое используется для оценки баланса между точностью и полнотой модели. Он особенно полезен в задачах с несбалансированными классами.

Поскольку классы несбалансированными, для оценки качества обучения, как основную метрику, будем использовать F1-score.


```
from sklearn.metrics import classification_report
print(classification_report(y_test, rfc_predict))
```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 1.00 | 1.00 | 27 |
| 1 | 1.00 | 1.00 | 1.00 | 9 |
| 2 | 0.94 | 0.92 | 0.93 | 52 |
| 3 | 0.00 | 0.00 | 0.00 | 2 |
| 4 | 1.00 | 1.00 | 1.00 | 4 |
| 5 | 1.00 | 0.93 | 0.97 | 15 |
| 6 | 0.88 | 0.95 | 0.91 | 56 |
| accuracy | | | 0.94 | 165 |
| macro avg | 0.83 | 0.83 | 0.83 | 165 |
| weighted avg | 0.93 | 0.94 | 0.93 | 165 |

```
print(classification_report(y_test, dtc_predict))
```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.93 | 1.00 | 0.96 | 27 |
| 1 | 0.90 | 1.00 | 0.95 | 9 |
| 2 | 0.92 | 0.88 | 0.90 | 52 |
| 3 | 1.00 | 0.50 | 0.67 | 2 |
| 4 | 1.00 | 1.00 | 1.00 | 4 |
| 5 | 1.00 | 0.80 | 0.89 | 15 |
| 6 | 0.88 | 0.93 | 0.90 | 56 |
| accuracy | | | 0.92 | 165 |
| macro avg | 0.95 | 0.87 | 0.90 | 165 |
| weighted avg | 0.92 | 0.92 | 0.91 | 165 |

```
print(classification_report(y_test, knn_predict))
```

| | precision | recall | f1-score | support |
|-----------|-----------|--------|----------|---------|
| 0 | 1.00 | 1.00 | 1.00 | 27 |
| 1 | 1.00 | 1.00 | 1.00 | 9 |
| 2 | 0.94 | 0.94 | 0.94 | 52 |
| 3 | 1.00 | 1.00 | 1.00 | 2 |
| 4 | 1.00 | 1.00 | 1.00 | 4 |
| 5 | 1.00 | 0.87 | 0.93 | 15 |
| 6 | 0.91 | 0.95 | 0.93 | 56 |
| accuracy | | | 0.95 | 165 |
| macro avg | 0.98 | 0.97 | 0.97 | 165 |

| | | | | |
|--------------|------|------|------|-----|
| weighted avg | 0.95 | 0.95 | 0.95 | 165 |
|--------------|------|------|------|-----|

Показатели очень хорошие, лучший F1-score у KNeighborsClassifier. Сохраним эту модель для последующего предсказания

```
from joblib import dump

dump(knn, "KNeighborsClassifier.joblib")
dump(nmf_model, "nmf_model.joblib")
dump(tfidf_vectorizer, "tfidf_vectorizer.joblib")

['tfidf_vectorizer.joblib']
```

Результат проделанной работы

Была проведена предварительная обработка данных

- Приведение к нижнему регистру
- Очистка от латинских символов
- Очистка от пунктуации и спец символов
- Очистка от цифр
- Очистка от лишних пробелов
- Очистка от стоп слов
- Токенизация
- Лемматизация

Отбор признаков Для кластеризации использована модель KMeans, так как она проста и быстра в реализации, а так же эффективна при работе с большими наборами данных

Для кластеризации взяты модели KNeighborsClassifier, DecisionTreeClassifier и RandomForestClassifier.

- KNeighbors это алгоритм классификации, который предполагает, что расположенные близко друг к другу объекты принадлежат к одному классу. Данный алгоритм популярен и подходит для нашей задачи, возьмём его для сравнения результатов
- Random Forest это множество решающих деревьев, это позволяет снизить переобучения и повысить точность в сравнении с одним деревом.
- DecisionTreeClassifier - это класс, способный выполнять многоклассовую классификацию на наборе данных.

Для оценки качества обучения использована метрика F1-score.

Каждый кластер был поименован:

- "Городской криминал / Драма"
- "Дружба / Любовь / Испытания"

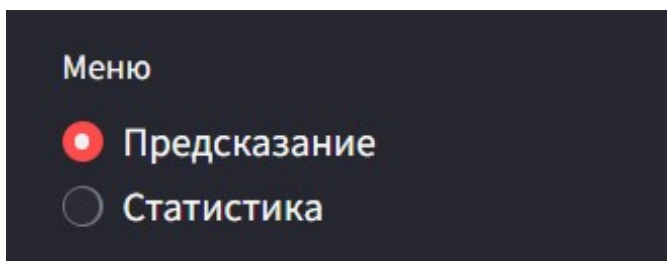
- "Космические путешествия / Приключения / Судьба"
- "Магия / Волшебство / Эпическая борьба"
- "Война / Мужская доблесть"
- "Любовь / Семейные отношения"
- "Современная жизнь / Преступность / Социальные драмы"

Выведена информация о количестве фильмов для каждого кластера

Web приложение

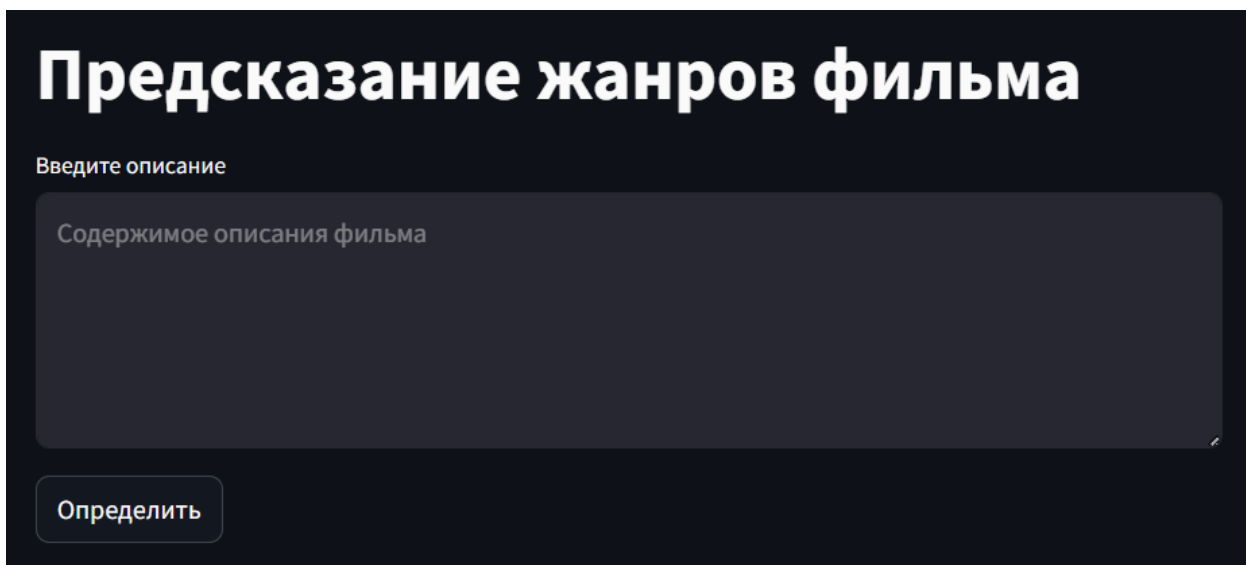
Для предсказания жанров фильмов создано web приложение на основе библиотек fastapi и streamlit

Приложение имеет меню навигации в левой части экрана



Предсказание

На основной вкладке "Предсказание" имеется форма ввода текста с кнопкой определения

A dark-themed form titled "Предсказание жанров фильма" (Prediction of movie genres). Below the title is a label "Введите описание" (Enter description) and a large text input area with a placeholder "Содержимое описания фильма" (Content of the movie description). At the bottom left of the form is a button labeled "Определить" (Determine).

По нажатию кнопки отправляется http запрос к API, в результате которого пользователь получает распределение вероятностей жанров

Предсказание жанров фильма

Введите описание

Миллиардер-изобретатель Тони Старк попадает в плен к Афганским террористам, которые пытаются заставить его создать оружие массового поражения. В тайне от своих захватчиков Старк конструирует высокотехнологичную киберброню, которая помогает ему сбежать. Однако по возвращении в США он узнаёт, что в совете директоров его фирмы плетётся заговор, чреватый страшными последствиями. Используя своё последнее изобретение, Старк пытается решить проблемы своей компании радикально...

Определить

Современная жизнь / Преступность / Социальные драмы - 1.000 %

Городской криминал / Драма - 0.000 %

Дружба / Любовь / Испытания - 0.000 %

Космические путешествия / Приключения / Судьба - 0.000 %

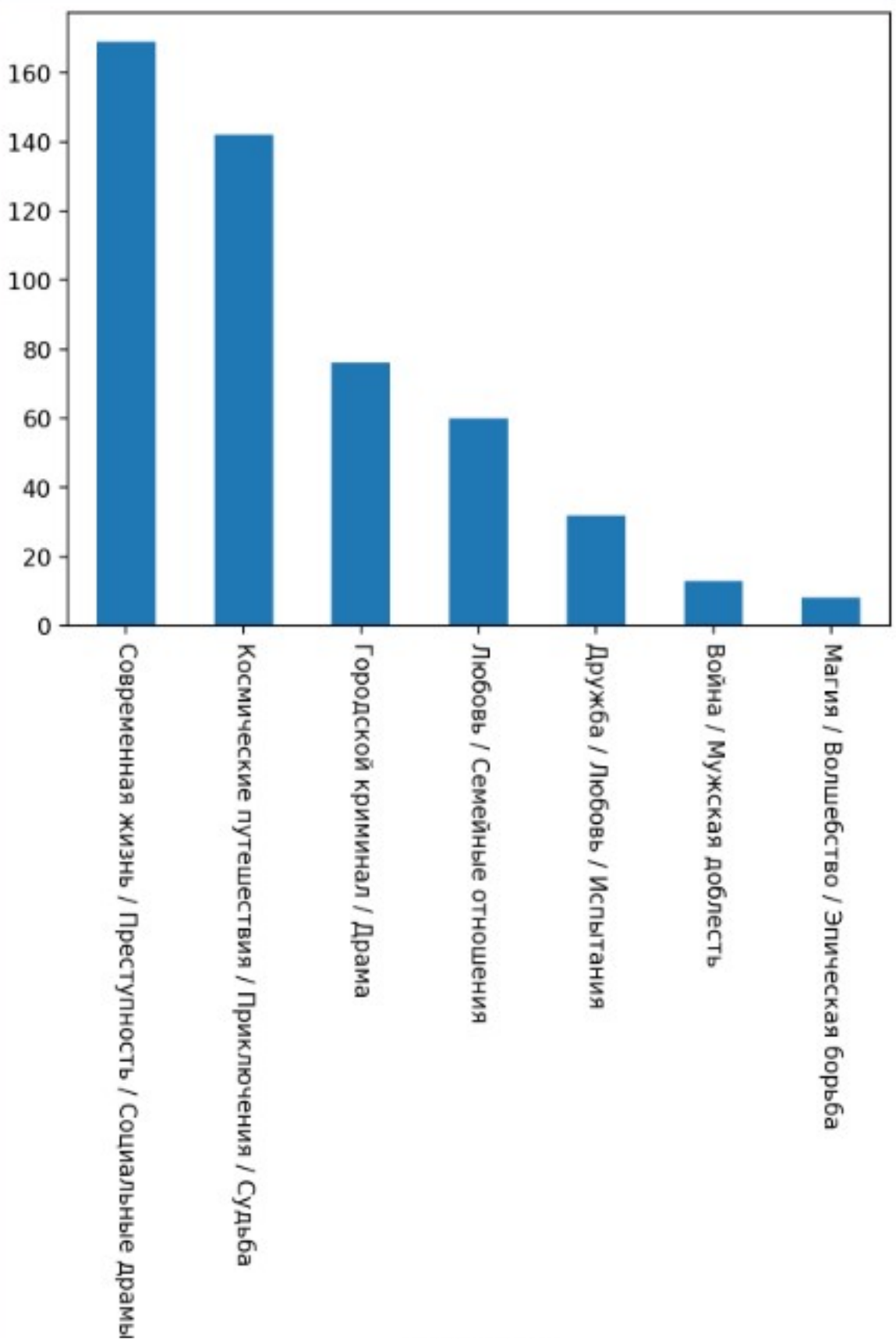
Магия / Волшебство / Эпическая борьба - 0.000 %

Война / Мужская доблесть - 0.000 %

Любовь / Семейные отношения - 0.000 %

Статистика

На вкладке "Статистика" имеется информация о наборе данных, сформированном в результате проделанной работы



Код API:

```
import re
import string

import fastapi
import joblib
import nltk
import pymorphy3

app = fastapi.FastAPI()

def preprocess_text(text: str) -> str:
    punctuation = set(string.punctuation) - { "(", ")" }
    stopwords = nltk.corpus.stopwords.words("russian") + [
        "который", "весь", "всё", "это", "свой", "мочь", "история",
        "год", "человек", "самый", "день", "молодой", "хороший"
    ]

    morph = pymorphy3.MorphAnalyzer()

    text = "".join([char for char in text if char not in
punctuation])
    text = "".join([char for char in text if not char.isdigit()])
    text = re.sub(r"[a-z]", "", text)
    text = re.sub(r"\s+", " ", text.strip())
    tokenized_text = re.split(r"\W+", text)
    tokenized_text = [morph.parse(word)[0].normal_form for word in
tokenized_text]
    tokenized_text = [word for word in tokenized_text if word not in
stopwords]
    return " ".join(tokenized_text)

def predict_cluster(text):
    with open('KNeighborsClassifier.joblib', 'rb') as file:
        model = joblib.load(file)

    with open('tfidf_vectorizer.joblib', 'rb') as file:
        vectorizer = joblib.load(file)

    with open('nmf_model.joblib', 'rb') as file:
        nmf_model = joblib.load(file)

    tfidf_matrix = vectorizer.transform([preprocess_text(text)])

    W = nmf_model.transform(tfidf_matrix)

    prediction = model.predict(W)
    probabilities = model.predict_proba(W)[0]

    mapping = {
```

```

0: "Городской криминал / Драма",
1: "Дружба / Любовь / Испытания",
2: "Космические путешествия / Приключения / Судьба",
3: "Магия / Волшебство / Эпическая борьба",
4: "Война / Мужская доблесть",
5: "Любовь / Семейные отношения",
6: "Современная жизнь / Преступность / Социальные драмы",
}

probabilities_dict = { mapping[i]: float(probabilities[i]) for i
in range(len(probabilities)) }

return {
    "probabilities" : probabilities_dict
}

@app.post("/predict")
def predict_class(text: str):
    return predict_cluster(text)

```

Код View:

```

import streamlit as st
import requests

import pandas as pd
import matplotlib.pyplot as plt

def request_predict(text: str):
    return requests.post("http://127.0.0.1:8002/predict",
params={"text": text})

def main():
    page = st.sidebar.radio("Меню", ["Предсказание", "Статистика"])

    if page == "Предсказание":
        st.title("Предсказание жанров фильма")
        input_text = st.text_area("Введите описание", height=150,
placeholder="Содержимое описания фильма")

        if st.button("Определить"):
            if input_text.strip():
                try:
                    response = request_predict(input_text)
                    response.raise_for_status()
                    result = response.json()

                    probabilities = result["probabilities"]
                    sorted_probabilities =
sorted(probabilities.items(), key=lambda x: x[1], reverse=True)

```

```

        try:
            for comment_nature, probability in
sorted_probabilities:
                st.write(f"{comment_nature} -
{probability:.3f} %")

        except Exception as e:
            st.error(f"Ошибка обработки вероятностей:
{e}")

        except requests.exceptions.RequestException as e:
            st.error(f"Ошибка соединения с API: {e}")
    else:
        st.warning("Пожалуйста, введите текст")

    else:
        df = pd.read_csv("The 500 best films according to Kinopoisk
(cluster).csv", keep_default_na=False)
        total_len = len(df)

        st.markdown(f"Общая длина набора данных: {total_len}")
        st.markdown(f"")

        st.markdown(f"Распределение фильмов:")
        counts = df["cluster name"].value_counts()

        fig, ax = plt.subplots()
        counts.plot(kind="bar", ax=ax)
        ax.set_xticklabels(counts.index, rotation=-90)
        st.pyplot(fig)

if __name__ == "__main__":
    main()

```