|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Изображение выглядит как текст  Автоматически созданное описание | | | | | | |
|  | Министерство науки и высшего образования Российской Федерации | | | | | | |
|  | Федеральное государственное бюджетное образовательное учреждение  высшего образования  «МИРЭА – Российский технологический университет»  РТУ МИРЭА | | | | | | |
| **Институт** | | ИКБ | | | | |  |
|  | | | | |  | |  |
| **Специальность (направление):** | | | | Технологии разработки программного обеспечения | | |  |
|  | | | | | |  |  |
| **Кафедра:** | | КБ-3 «Безопасность программных решений» | | | | |  |
|  | | | | | |  |  |
| **Дисциплина:** | | | «Методы искусственного интеллекта» | | | |  |
|  |  |  |  |  |  |  |  |

**Практическая работа 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Студент: |  | |  | 28.09.2024 |  | A.   M. Ищенко |
|  | | *подпись* |  | *Дата* |  | *инициалы и фамилия* |
|  |  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа: | БСБО-01-22 |  | Шифр: | 22Б0661 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель: |  |  |  |  |  |
|  | *подпись* |  | *дата* |  | *инициалы и фамилия* |

**Москва 2024 г.**

**Задание**

Выбран датасет: [16000+ Movies 1910-2024 (Metacritic) (kaggle.com)](https://www.kaggle.com/datasets/kashifsahil/16000-movies-1910-2024-metacritic)

**1. Выбрать и обосновать метрику для измерения качества**

В рамках данного пункта необходимо подобрать наиболее релевантную метрику или набор метрик для вашей задачи, написав краткое обоснование   
(1–2 предложения).

Критерии оценки:

* Предложена метрика
* Присутствует корректное обоснование

Для данного датасета будет применена задача бинарной классификации (получится ли фильм хорошим или нет) с метриками Accuracy и ROC AUC.. Для предсказания подобной переменной подходит такая модель, как логистическая регрессия. Оценка рейтинга будет зависеть от таких параметров, как жанр фильма, его продолжительность, количество голосов и другие характеристики.

**2. Провести EDA и предобработку данных.**

В рамках данного пункта необходимо провести предварительный разведочный анализ своего набора данных, проанализировать признаки и провести целесообразную предварительную обработку.

Критерии оценки:

* Рассмотрены базовые характеристики/статистики для набора данных
* Реализованы наиболее релевантные визуализации (не менее одной визуализации с применением plotly)
* По ходу работы присутствуют комментарии (вы, как эксперт в предметной области, можете проинтерпретировать полученные результаты анализа, сделав выводы о том, каким образом лучше работать с набором данных в будущем, какие могут возникнуть проблемы, какую предобработку целесообразно сделать и тп)
* Проведен анализ лучших практик работы с этим датасетом или с похожими данными (указать ссылки на эти работы)

Для начала импортируем все библиотеки и датасет:

import pandas as pd

import matplotlib.pyplot as plt

import plotly.express as plx

import seaborn as sns

from sklearn import preprocessing

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix, accuracy\_score, roc\_auc\_score, roc\_curve, auc

path = "16k\_Movies.csv"

df = pd.read\_csv(path)

df.head()

Вывод датасета:



Выведем количество строк и количество столбцов датасета:

df.shape



Из этого следует, что датасет состоит из 16290 строк и 10 столбцов.

Описание столбцов:

Unnamed: идентификатор

Title: название произведения

Release Date: официальная дата релиза фильма

Description: Краткий синопсис или критический обзор фильма

Rating: Средний балл на Metacritic, позволяющий судить о восприятии фильма критиками

Number of Persons Voted: Количество людей, оценивших фильм на Metacritic

Directed by: Режиссер(ы) фильма

Written by: Сценарист(ы), ответственный(ые) за сценарий фильма

Duration: Длительность фильма

Genres: Жанр(ы), к которому относится фильм, например драма, комедия или боевик

Получение информации о датасете:

#Удалим первый столбец без имени

df.drop(columns = 'Unnamed: 0', inplace=True)

print()

# покажет информацию о размерности данных и как данные индексируются, количество not-a-number элементов

print('Размерность, нумерация и не числовые значения набора данных (df.info()):\n',df.info())

print()

# показывает статистики: count,mean, std, min, 25%-50%-75% percentile, max

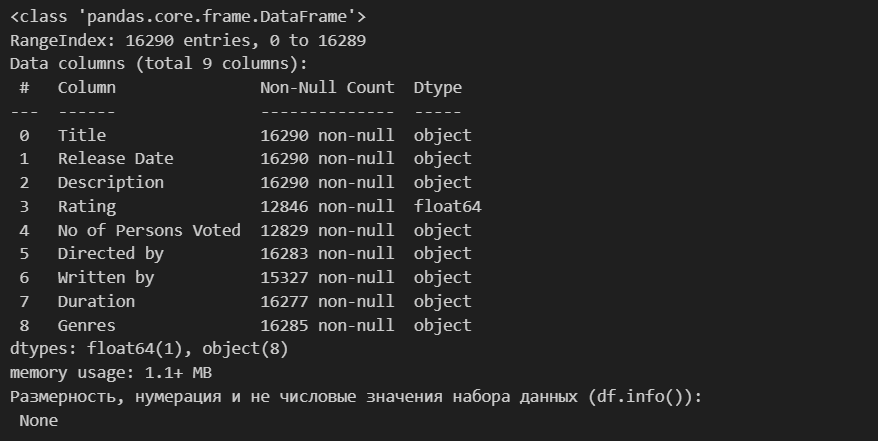
print('Статистики набора данных (df.describe()):\n',df.describe())

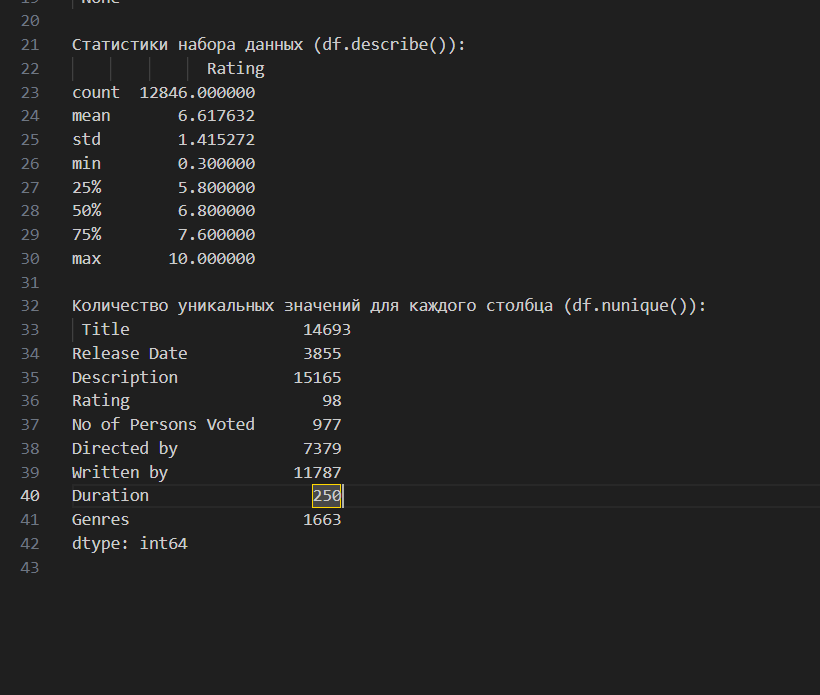
print()

# количество уникальных значений для каждого столбца.

print('Количество уникальных значений для каждого столбца (df.nunique()):\n', df.nunique())

Вывод:

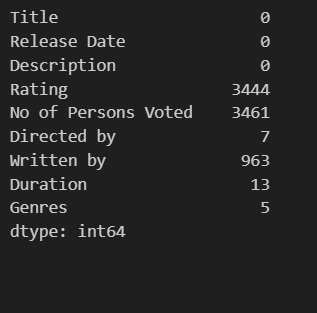




Теперь проверим датасет на наличие пустых значений:

# Проверим на пустые значения и посчитаем их для каждого столбца

df.isna().sum()



В столбцах присутствуют пустые значения:

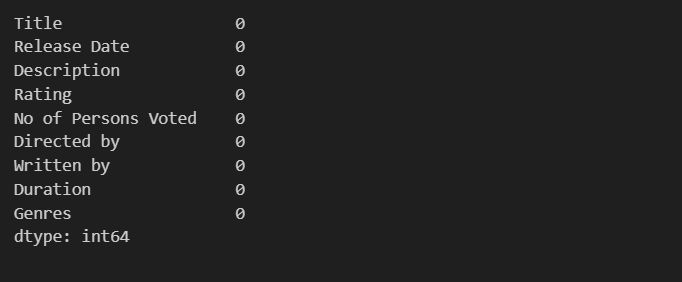
* Rating не был проставлен рейтинг
* No of Persons Voted есть фильмы, которые не оценивали
* Directed by отсутствует информация о режиссере
* Written by отсутствует информация о сценаристе
* Duration отсутствует информация о длительности
* Genres отсутствует информация о жанре

Удаление пустых значений в последних четырех столбцах незначительно скажется на результате.

# Удаление пустых значений

df.dropna(inplace=True)

df.isna().sum()



Визуализируем распределение по всем столбцам при помощи гистограмм, кроме столбца с навыками в текстовом виде:

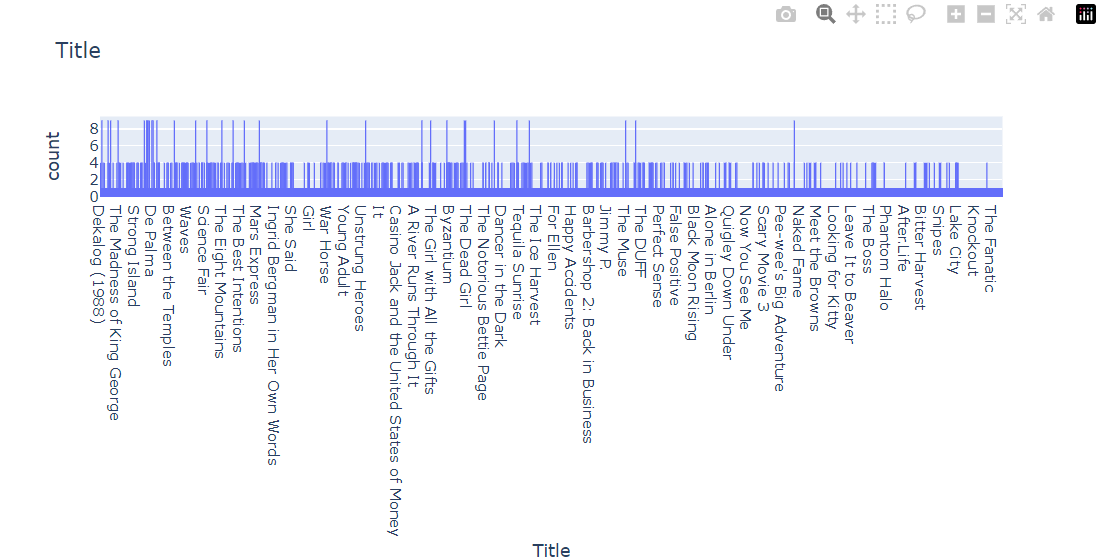
columns\_for\_figs = list(df.columns)

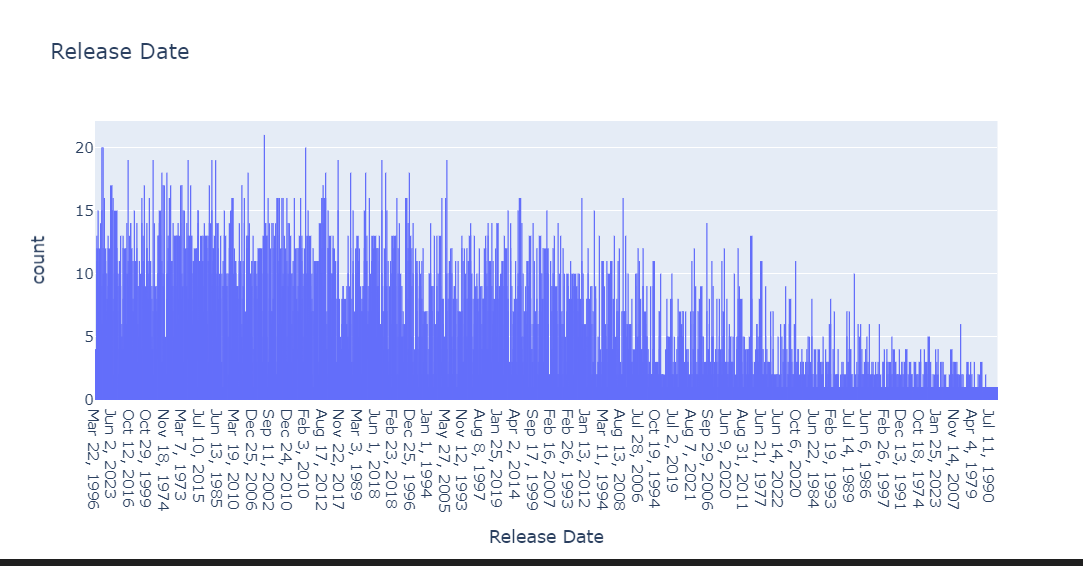
for col in columns\_for\_figs:

    if col!='Rating':

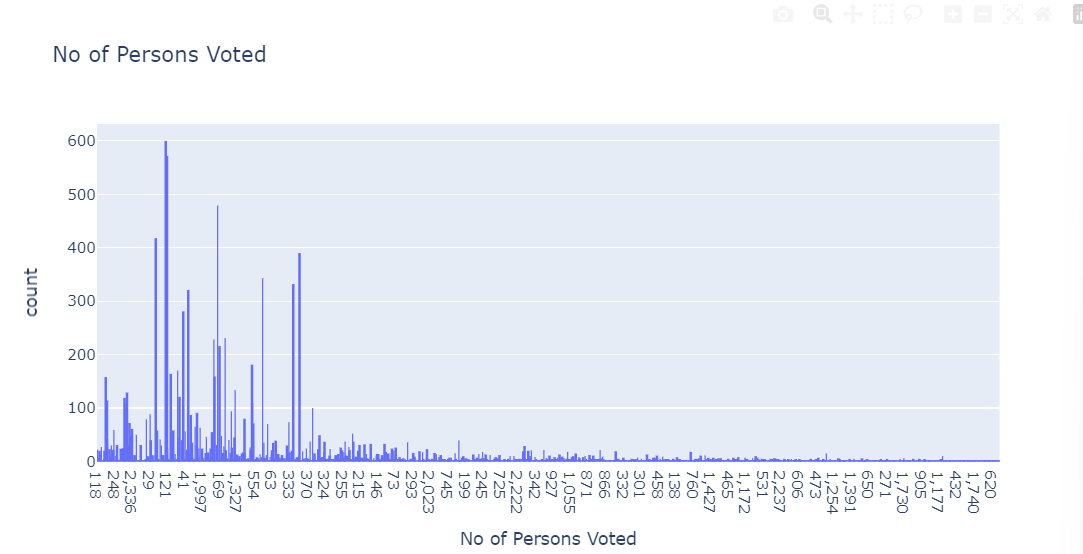
        fig = plx.histogram(df,x=col,title = col)

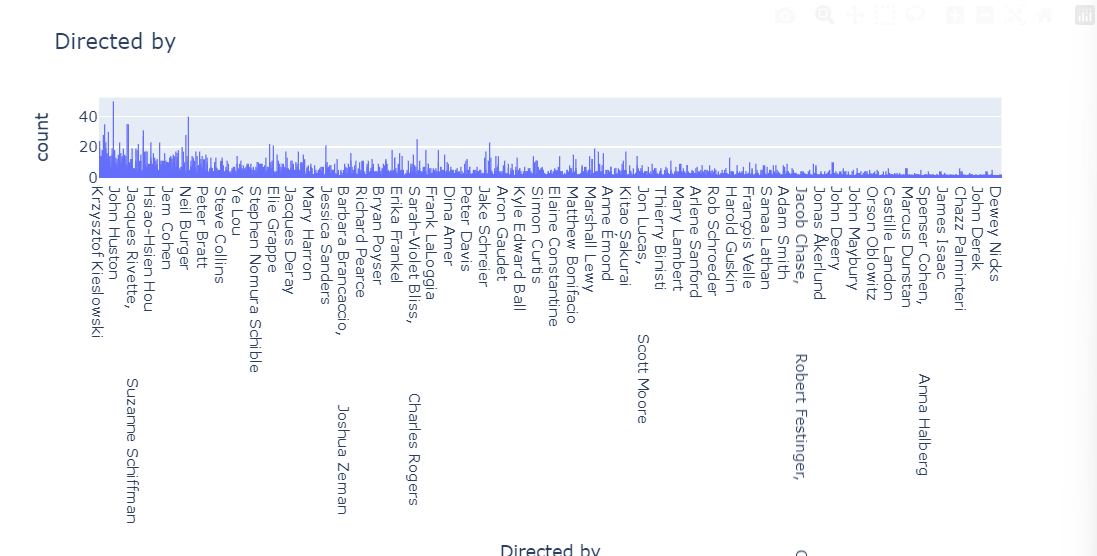
        fig.show()



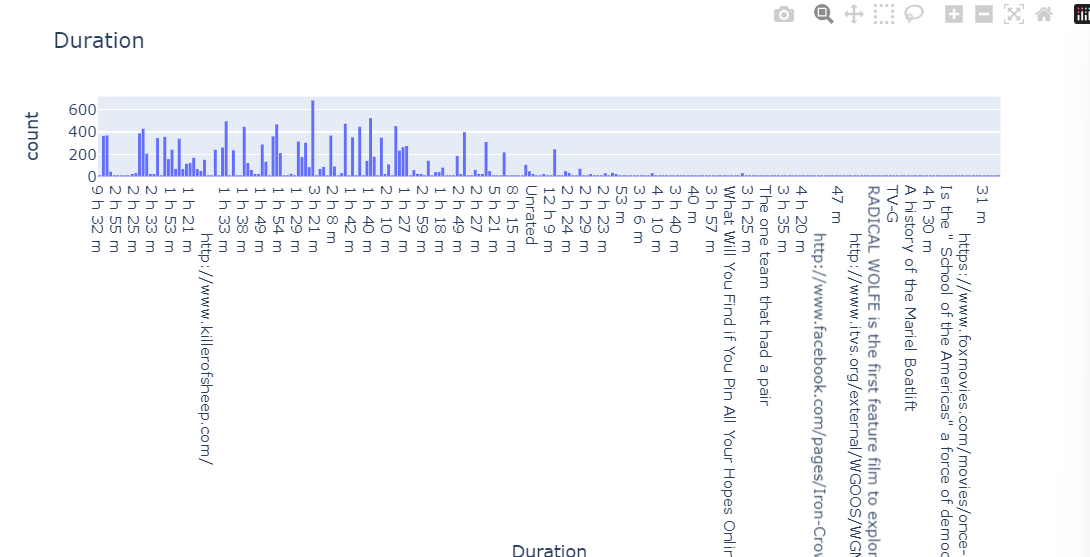














Проанализировав все гистограммы, были выявлены аномалии в столбце Duration, а именно вместо самой длительности присутствуют ссылки и прочие значения. Исправим это (переведем значения \* h \* m в минуты и отбросим оставшиеся строки):

import re

def convert\_to\_minutes(duration):

    if isinstance(duration, str):  # Проверяем, что значение строка

        # Ищем часы и минуты в строке

        match = re.match(r'(?:(\d+)\s\*h)?\s\*(?:(\d+)\s\*m)?', duration)

        if match:

            hours = int(match.group(1)) if match.group(1) else 0

            minutes = int(match.group(2)) if match.group(2) else 0

            total\_minutes = hours \* 60 + minutes

            return total\_minutes if total\_minutes > 0 else None  # Возвращаем None, если 0 минут

    return None  # Возвращаем None для значений, которые не являются строками

# Применим функцию к столбцу Duration

df['Duration'] = df['Duration'].apply(convert\_to\_minutes)

# Проверим результат

print("Первые строки после преобразования в минуты:\n", df['Duration'].head())

# Проверим, есть ли пропущенные значения после преобразования

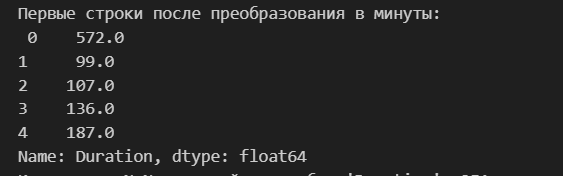
print("Количество NaN значений в столбце 'Duration':", df['Duration'].isna().sum())

# Удалим строки с NaN, если они присутствуют

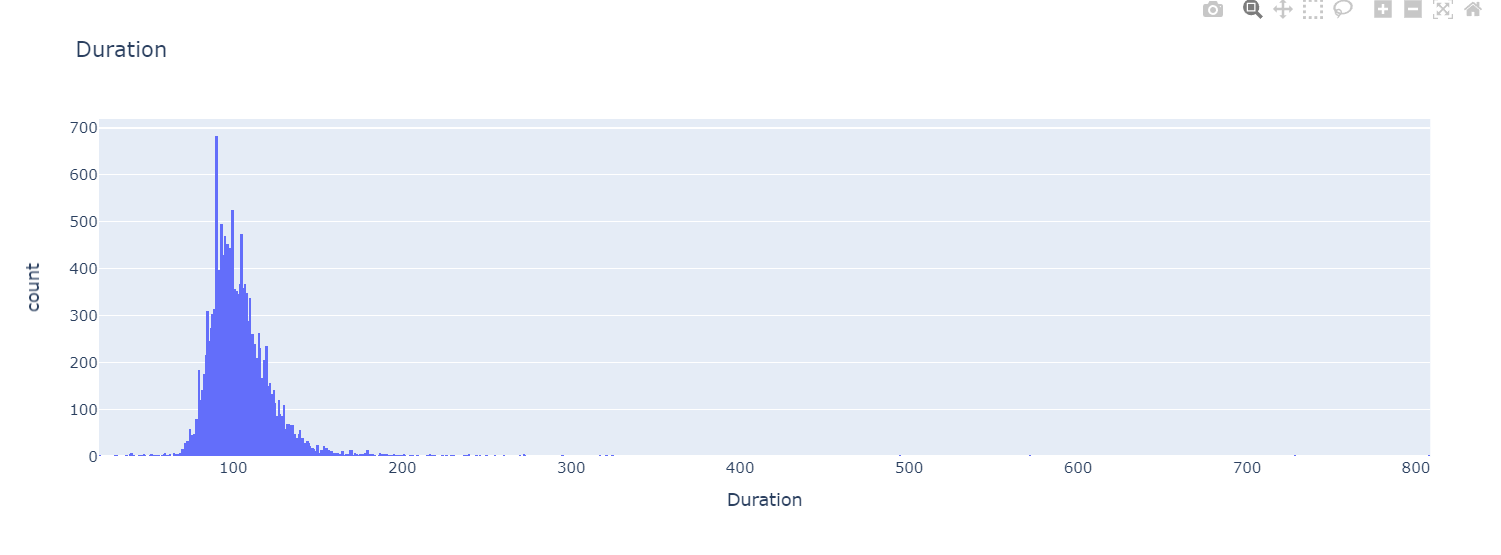
df = df.dropna(subset=['Duration'])

# Убедимся, что теперь столбец содержит только числовые значения в минутах

print("Тип данных столбца 'Duration':", df['Duration'].dtype)



Заново построим гистограмму для столбца Duration:



Получаем отличный результат без аномалий

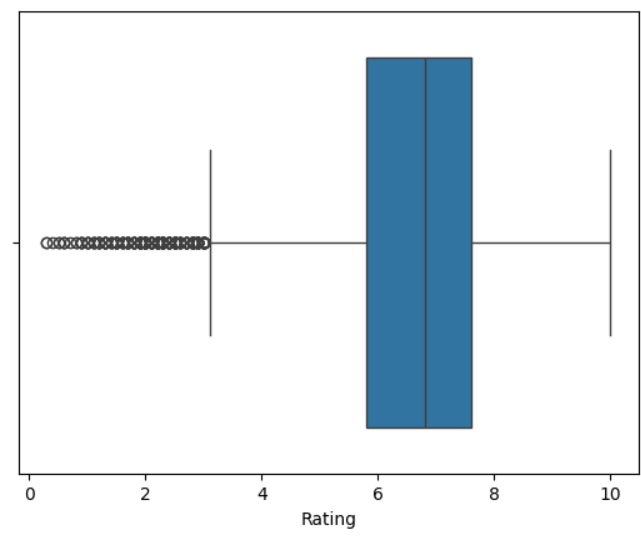
Сделаем BoxPlot визуализации числовых значений для просмотра выбросов:

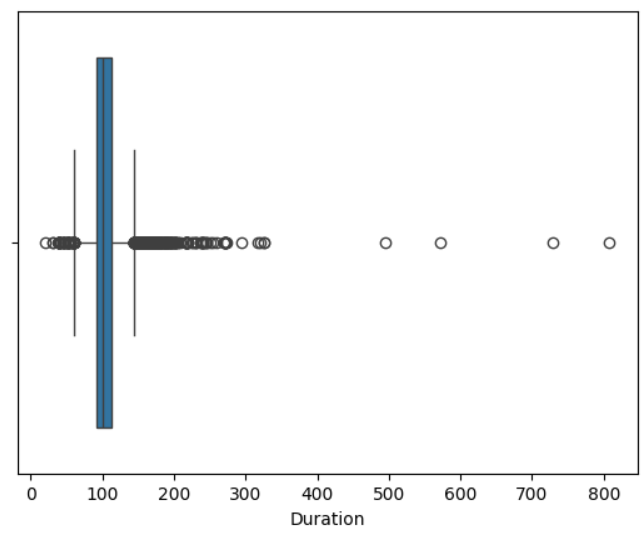
num\_cols = df.select\_dtypes(include=['float64', 'int64']).columns

for column in num\_cols:

    sns.boxplot(data=df, x=column)

    plt.show()





Посмотрим, как количество голосов на Metacritic связано с рейтингом фильма. Визуализация может показать, существует ли связь между популярностью фильма (в виде количества голосов) и его критической оценкой.

plt.figure(figsize=(15, 15))

sns.scatterplot(data=df, x='Rating', y='No of Persons Voted', alpha=0.7)

plt.xlabel('Рейтинг фильма')

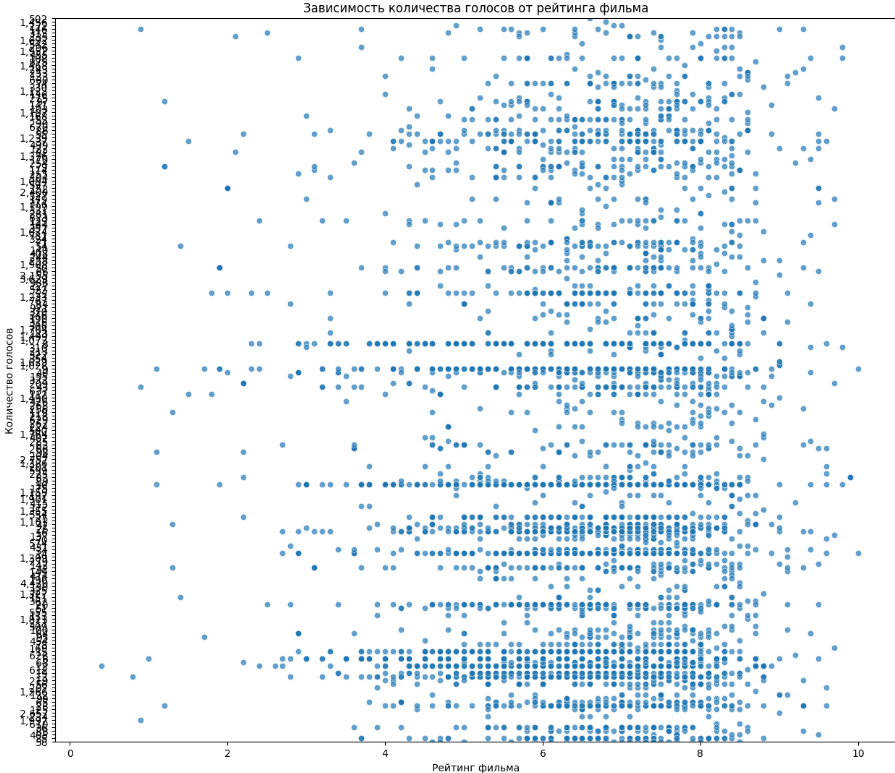
plt.ylabel('Количество голосов')

plt.title('Зависимость количества голосов от рейтинга фильма')

# Установка пределов оси Y

plt.ylim(100, 300)

plt.show()

 График показывает, что с увеличением рейтинга фильма количество голосов также увеличивается. Это может указывать на то, что фильмы с более высокими оценками, вероятно, вызывают больший интерес у зрителей, что приводит к большему количеству голосов.

Посмотрим, какие жанры фильмов имеют более длинное или короткое среднее время. Это может помочь понять, склонны ли определенные жанры к длинным или коротким фильмам. По итогу оказалось, что в среднем все жанры имеют одну и ту же продолжительность.

genre\_counts = df.explode('Genres')['Genres'].value\_counts()

# Отбираем жанры, у которых более 10 фильмов

popular\_genres = genre\_counts[genre\_counts > 18].index.tolist()

# Фильтруем df\_copy на основе популярных жанров

filtered\_df = df.explode('Genres')[df.explode('Genres')['Genres'].isin(popular\_genres)]

plt.figure(figsize=(12, 8))

sns.boxplot(data=filtered\_df, x='Genres', y='Duration')

plt.xlabel('Жанр')

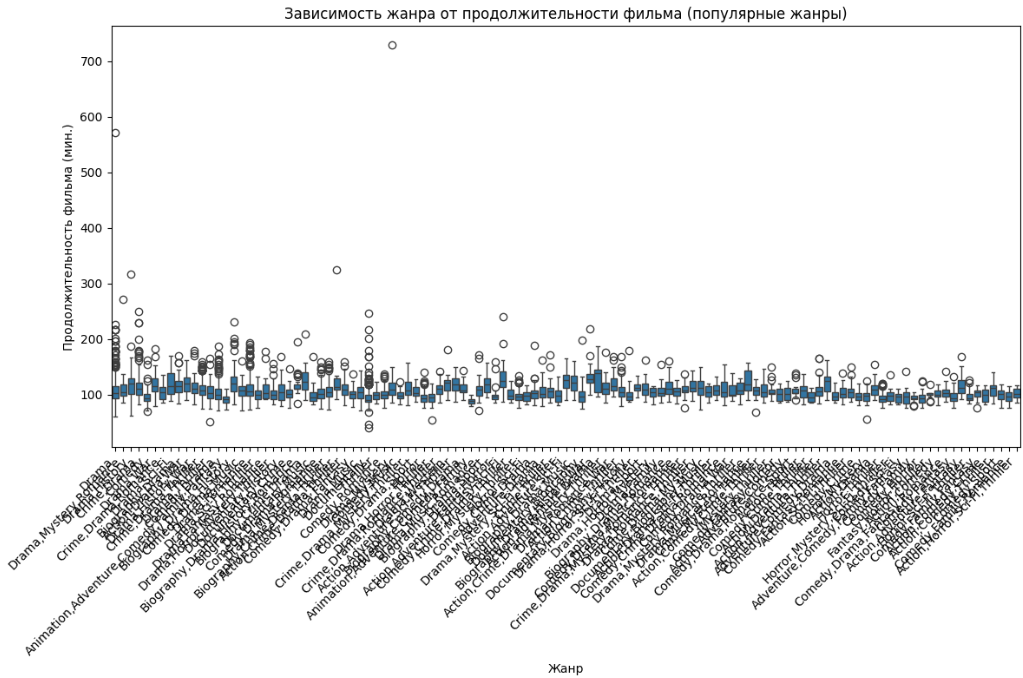
plt.ylabel('Продолжительность фильма (мин.)')

plt.title('Зависимость жанра от продолжительности фильма (популярные жанры)')

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()



Далее проверим, как средняя продолжительность фильмов изменялась с годами:

import re

# Функция для преобразования строки с продолжительностью в минуты

def convert\_to\_minutes(duration\_str):

    match = re.match(r'(?:(\d+)\s\*h)?\s\*(\d+)\s\*m', duration\_str)

    if match:

        hours = int(match.group(1)) if match.group(1) else 0

        minutes = int(match.group(2))

        return hours \* 60 + minutes

    return None

# Применим преобразование к столбцу 'Duration'

df['Duration\_Minutes'] = df['Duration'].apply(convert\_to\_minutes)

# Сгруппируем данные по году и посчитаем среднюю продолжительность фильмов

df\_grouped = df.groupby('Release Date')['Duration\_Minutes'].mean().reset\_index()

# Построим lineplot

plt.figure(figsize=(12, 6))

sns.lineplot(data=df\_grouped, x='Release Date', y='Duration\_Minutes', errorbar=None)

plt.xlabel('Год выпуска')

plt.ylabel('Средняя продолжительность (мин.)')

plt.title('Зависимость средней продолжительности фильмов от года выпуска')

plt.xticks(rotation=45, ha='right')  # Поворот меток на 45 градусов

plt.gca().set\_xticks(df\_grouped['Release Date'][::60])  # Каждую 2-ю метку

plt.tight\_layout()

plt.show()

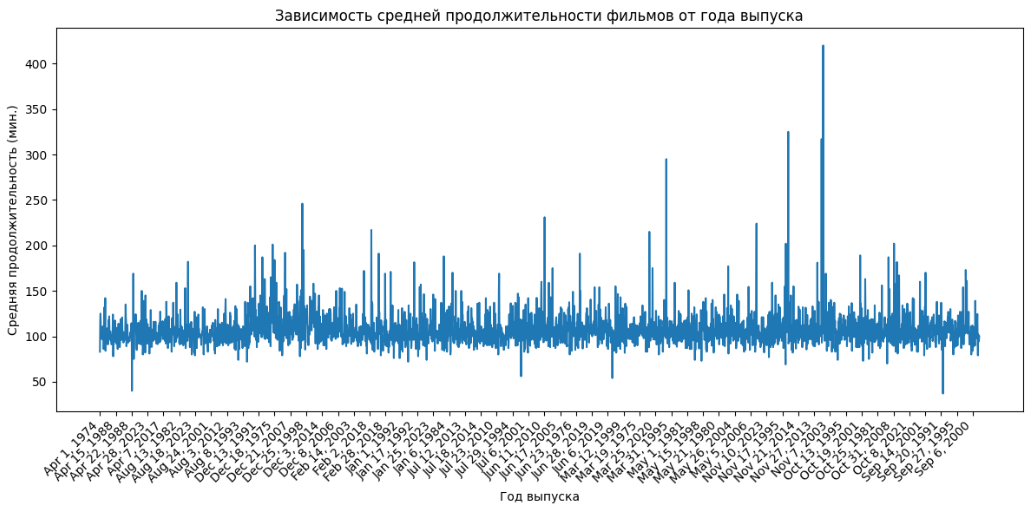


График показывает, что в некоторые годы средняя продолжительность значительно увеличивалась. Это может быть связано с выходом длинных фильмов или определенных жанров, которые требуют больше времени для раскрытия сюжета.

Проведем кодировку категориальных переменных (не будем использовать значения ‘Description’, так как можно использовать краткое описание фильма в виде его жанра):

df\_copy = df.loc[:, df.columns != 'Description'].copy()

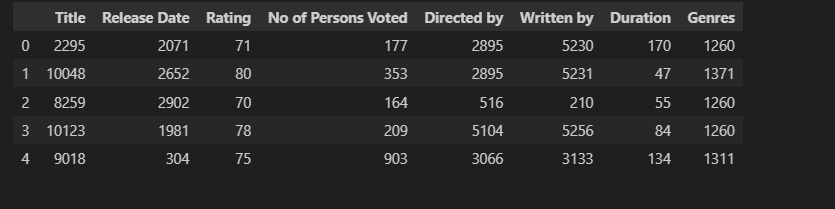
label\_encoder = LabelEncoder()

categorical\_columns = ['Title', 'Release Date', 'Rating', 'No of Persons Voted', 'Directed by', 'Written by', 'Duration', 'Genres']

for col in categorical\_columns:

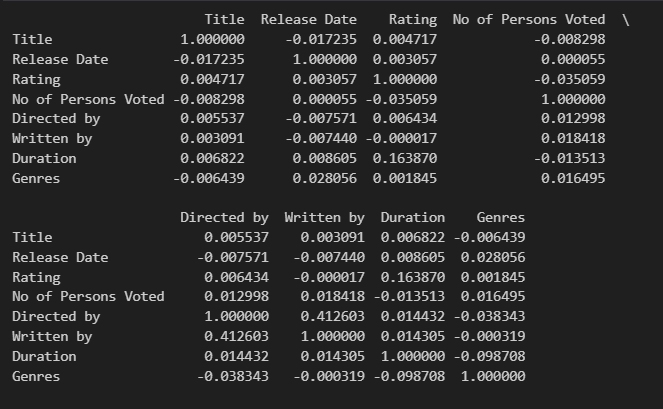
    df\_copy[col] = label\_encoder.fit\_transform(df[col])

df\_copy.head()



correlation\_matrix = df\_copy.corr()

print(correlation\_matrix)



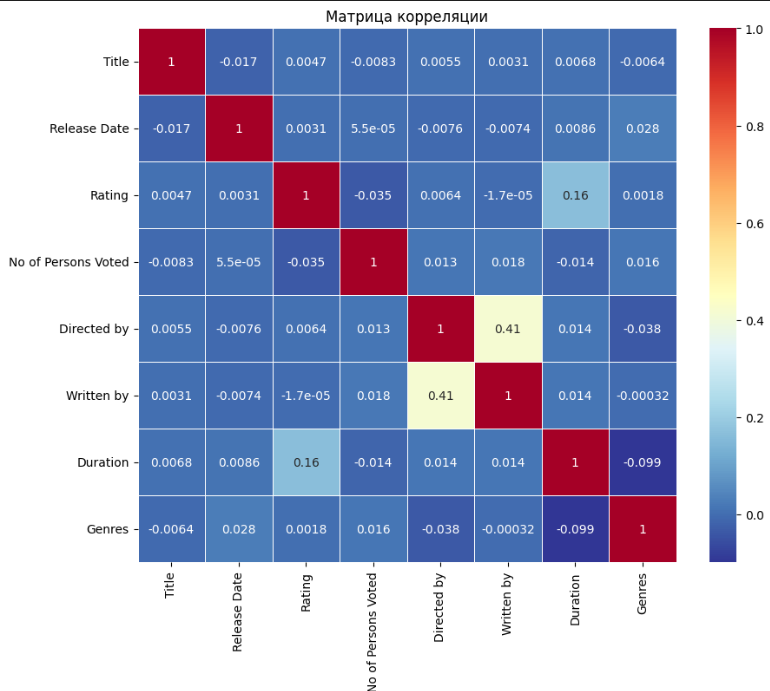
Выведем графическую визуализацию матрицы корреляции:

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, linewidths=0.5, cmap="RdYlBu\_r")

plt.title('Матрица корреляции')

plt.show()



По итогу целевой столбец Rating немного коррелирует с Duration. Это значит, что чуть большее влияние на рейтинг фильма будет оказывать его длительность.

Также была проанализирована обработка датасета в работе, представленной по ссылке: [IMDB-Reviews-Analysis Using Logistic Regression (kaggle.com)](https://www.kaggle.com/code/dhrv04/imdb-reviews-analysis-using-logistic-regression)

При обработке датасета были удалены пропущенные значения, а также выполнена векторизация текстовых данных отзывов с использованием методов преобразования текста в числовые признаки. В качестве основной модели использовалась **логистическая регрессия**, которая была обучена на разреженной матрице с помощью 10-кратной кросс-валидации. Средняя точность модели составила **0.8788**, что свидетельствует о ее высокой эффективности при классификации отзывов на основе рейтингов. Метрика ROC AUC также была использована для оценки качества модели, что подчеркивает надежность результатов в задаче предсказания отзывов.

**3. Построить и оценить качество бейзлайна**

В рамках данного пункта необходимо выбрать и обучить бейзлайн-модель,   
а также измерить ее качество.

Критерии оценки:

* Произведено измерение качества константного предсказания (например, наиболее частотный класс для классификации, среднее/медиана   
  для регрессии)
* Бейзлайновая модель обучена на тренировочной выборке, учтены особенности предобработки данных для модели, если они есть
* Произведено измерение качества на отложенной выборке   
  с использованием ранее выбранной метрики

В данной работе будет использована бейзлайн-модель, которая будет предсказывать результат отбора. Класс 1 представляет положительный результат, указывающий что фильм достаточно хорош (будем считать, что хороший фильм – это тот, у которого рейтинг больше 64), соответственно класс 0 будет представлять противоположный результат.

from sklearn.preprocessing import StandardScaler

# Создание целевой переменной

df\_copy['Good\_Movie'] = (df\_copy['Rating'] > 64).astype(int)

X = df\_copy.drop(['Rating', 'Good\_Movie'], axis=1)  # Измените имена, если нужно

y = df\_copy['Good\_Movie']

# Разделение на обучающую и тестовую выборки

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=200)

# Нормализация данных

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Создание классификатора логистической регрессии

logistic\_regression = LogisticRegression()

logistic\_regression.fit(X\_train\_scaled, y\_train)

y\_pred\_lr = logistic\_regression.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred\_lr)

print(f'Логистическая регрессия - Accuracy: {accuracy:.6f}')

# ROC-AUC

if len(set(y\_test)) > 1:

    roc\_auc\_lr = roc\_auc\_score(y\_test, logistic\_regression.predict\_proba(X\_test\_scaled)[:, 1])

    print(f'Логистическая регрессия - ROC-AUC: {roc\_auc\_lr:.6f}')

else:

    print('В тестовой выборке присутствует только один класс, ROC-AUC не может быть рассчитан.')

# Вывод матрицы несоответствий

cm\_lr = confusion\_matrix(y\_test, y\_pred\_lr)

plt.figure(figsize=(6, 4))

sns.heatmap(cm\_lr, annot=True, fmt="d", cmap="Blues", xticklabels=['Плохой фильм', 'Хороший фильм'], yticklabels=['Плохой фильм', 'Хороший фильм'])

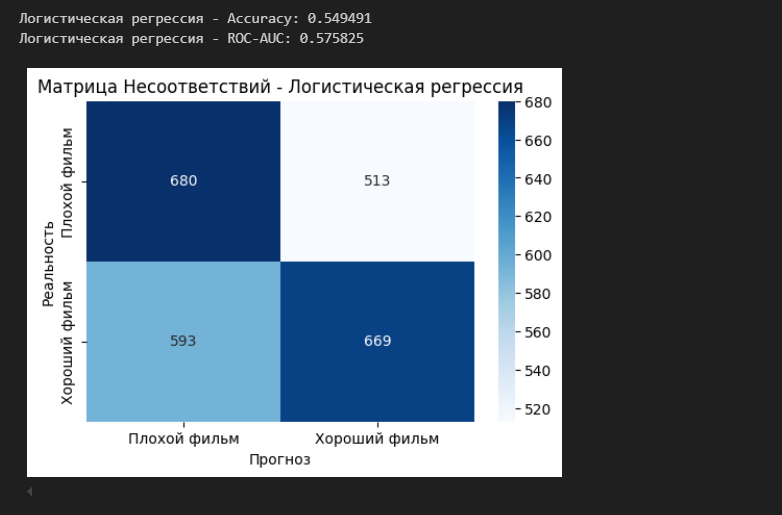
plt.xlabel('Прогноз')

plt.ylabel('Реальность')

plt.title('Матрица Несоответствий - Логистическая регрессия')

plt.show()

Вывод:



Также построим график, для просмотра доли ложно положительных примеров (false positive rate) в сравнении с долей истинно положительных примеров (англ. true positive rate):

# ROC Curve

y\_scores\_lr = logistic\_regression.predict\_proba(X\_train\_scaled)[:, 1]

fpr\_lr, tpr\_lr, thresholds\_lr = roc\_curve(y\_train, y\_scores\_lr)

auc\_lr = auc(fpr\_lr, tpr\_lr)

plt.figure()

plt.plot(fpr\_lr, tpr\_lr, color='darkorange', lw=2, label=f'ROC curve (area = {auc\_lr:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

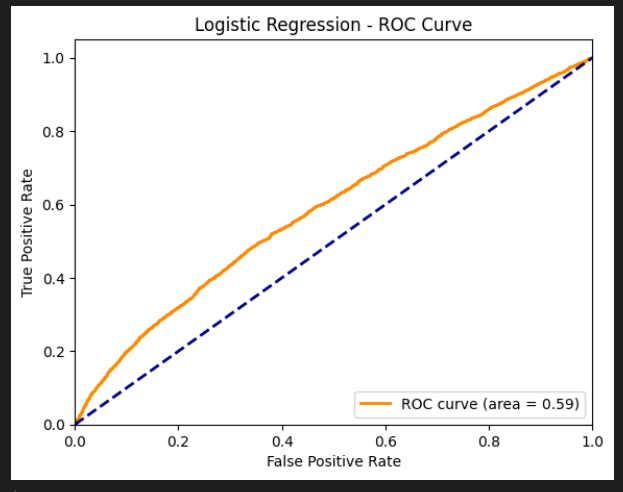
plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Logistic Regression - ROC Curve')

plt.legend(loc='lower right')

plt.show()



Подводя итоги, можно сделать вывод, что модель с неплохой вероятностью правильно ранжирует случайно выбранные объекты из разных классов.