Министерство цифрового развития

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт

по лабораторной работе № 1 «Первичный анализ и предобработка данных»

Выполнил:

студент группы ИП-312

Прозоренко Константин Владиславович

Работу проверил: старший преподаватель

Дементьева Кристина Игоревна

Новосибирск 2025 г.

Введение (задание)

В данной работе проводится первичный анализ и предобработка данных (EDA) на примере датасета IMDb Top 1000 Movies Dataset с платформы Kaggle.
Этот набор данных содержит информацию о популярных фильмах, включая их название, год выпуска, жанр, рейтинг, метаоценку, режиссёра, актёрский состав и коммерческие показатели.

Основная цель работы — освоить на практике методы первичного анализа и визуализации данных с использованием библиотек pandas, matplotlib и seaborn, а также выполнить базовую предобработку:

- выявление и обработку пропусков,
- анализ распределений и выбросов,
- исследование взаимосвязей между признаками,
- подготовку данных к дальнейшему использованию в моделях машинного обучения.

Структура датасета:

Poster_Link	Ссылка на постер фильма
Series_Title	Название фильма
Released_Year	Год выхода
Certificate	Возрастной рейтинг
Runtime	Продолжительность фильма (в минутах)
Genre	Жанр
IMDB_Rating	Рейтинг IMDb
Overview	Краткое описание сюжета
Meta_score	Оценка критиков
Director	Режиссёр
Star1–Star4	Основные актёры
No_of_votes	Количество голосов пользователей
Gross	Суммарный доход фильма

Основная часть

```
import pandas as pd
import numpy as np

from scipy import stats

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

imdb = pd.read_csv('imdb.csv')

imdb.head(10)

print(f"Количество строк: {imdb.shape[0]}")

print(f"Количество столбцов: {imdb.shape[1]}")

print(f"Общее количество элементов: {imdb.size}")

print("Статистическая сводка\n")

imdb.describe(include="all")

 print("Анализ пропусков\n")
total_rows = len(imdb)

missing_data = imdb.isnull().sum()

missing_percentage = (missing_data / total_rows) * 100



missing_df = pd.DataFrame({

    'Столбец': missing_data.index,

    'Количество_пропусков': missing_data.values,

    'Процент_пропусков': missing_percentage.values

})

missing_df

 # 3. Анализ числовых признаков
# Приведём названия столбцов к удобному виду

imdb['Gross'] = (imdb['Gross'].astype(str).str.replace('[\$,]', '',
regex=True).replace('nan', None).astype(float))
```

```
imdb['Runtime'] = (imdb['Runtime'].astype(str).str.replace(' min', '',
regex=False).replace('nan', None).astype(float))


imdb.columns = [c.strip().lower().replace(' ', '_') for c in imdb.columns]

print('Columns:', imdb.columns.tolist())

for col in ['no_of_votes','meta_score','imdb_rating','released_year']:

    if col in imdb.columns:

        imdb[col] = pd.to_numeric(imdb[col], errors='coerce')


numeric_cols = imdb.select_dtypes(include=[np.number]).columns.tolist()

print('Numeric columns:', numeric_cols)


# Гистограммы и boxplot'ы

for col in numeric_cols:

    plt.figure(figsize=(8,3))

    plt.hist(imdb[col].dropna(), bins=50)

    plt.title(f'Histogram: {col}')

    plt.xlabel(col)

    plt.ylabel('count')

    plt.tight_layout()

    plt.show()


    plt.figure(figsize=(6,2))

    sns.boxplot(x=imdb[col])

    plt.title(f'Boxplot: {col}')

    plt.tight_layout()

    plt.show()


# Статистики

stats = []

for col in numeric_cols:
```

```
    s = imdb[col].dropna()

    if len(s)==0: continue

    stats.append((col, s.mean(), s.median(), s.std(), s.skew(), s.min(),
s.max()))

stats_df = pd.DataFrame(stats,
columns=['col','mean','median','std','skew','min','max']).set_index('col')

display(stats_df)

 # 4. Анализ категориальных признаков
cat_cols = imdb.select_dtypes(include=['object']).columns.tolist()

print('Categorical columns:', cat_cols)



# Уникальные значения

for c in cat_cols:

    print(f'- {c}: {imdb[c].nunique(dropna=True)} unique')



# Построим countplot для топ-20 категорий в каждом столбце

for c in cat_cols:

    vc = imdb[c].fillna('NaN').value_counts().head(20)

    plt.figure(figsize=(10, max(3, len(vc)*0.25)))

    sns.barplot(x=vc.values, y=vc.index)

    plt.title(f'Топ 20 категорий для {c}')

    plt.tight_layout()

    plt.show()

 # 5. Анализ взаимосвязей
numeric_cols = imdb.select_dtypes(include=[np.number]).columns.tolist()



if len(numeric_cols) >= 2:

    corr = imdb[numeric_cols].corr()

    plt.figure(figsize=(8,6))

    sns.heatmap(corr, annot=True, fmt='.2f', cmap='coolwarm', vmin=-1,
vmax=1)

    plt.title('Correlation matrix (numeric features)')
```

```
    plt.tight_layout()
    plt.show()
    display(corr)

# Scatterplots для ключевых пар
pairs = []
if 'imdb_rating' in numeric_cols and 'no_of_votes' in numeric_cols:
    pairs.append(('imdb_rating','no_of_votes'))
if 'imdb_rating' in numeric_cols and 'gross' in numeric_cols:
    pairs.append(('imdb_rating','gross'))
if 'meta_score' in numeric_cols and 'imdb_rating' in numeric_cols:
    pairs.append(('meta_score','imdb_rating'))

for x,y in pairs:
    plt.figure(figsize=(6,4))
    sns.scatterplot(data=imdb, x=x, y=y, alpha=0.6)
    plt.title(f'{y} vs {x}')
    plt.tight_layout()
    plt.show()

# Boxplot числового по категориальному
if 'certificate' in imdb.columns and 'imdb_rating' in imdb.columns:
    plt.figure(figsize=(10,4))
    sns.boxplot(data=imdb, x='certificate', y='imdb_rating')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
 # 6. Базовая предобработка
# Приведение названий столбцов к нижнему регистру
df_processed = imdb.copy()
df_processed.columns = [col.lower() for col in df_processed.columns]
```

```
print("\nНовые названия столбцов:")
print(df_processed.columns.tolist())


# Обработка пропусков
print("\nОбработка пропусков...")


df_processed['runtime'] = (
    df_processed['runtime']
    .astype(str)                          #
    .str.extract(r'(\d+)')[0]
    .astype(float)
)


df_processed['released_year'] =
pd.to_numeric(df_processed['released_year'], errors='coerce')
df_processed['imdb_rating'] = pd.to_numeric(df_processed['imdb_rating'],
errors='coerce')
df_processed['meta_score'] = pd.to_numeric(df_processed['meta_score'],
errors='coerce')
df_processed['no_of_votes'] = pd.to_numeric(df_processed['no_of_votes'],
errors='coerce')


df_processed['gross'] = (
    df_processed['gross']
    .astype(str)
    .str.replace(',', '', regex=False)
    .str.replace('$', '', regex=False)
)
df_processed['gross'] = pd.to_numeric(df_processed['gross'],
errors='coerce')


# Для числовых признаков заполняем медианой
```

```
numerical_to_fill = ['meta_score', 'gross']

for col in numerical_to_fill:

    if col in df_processed.columns:

        df_processed[col] =
df_processed[col].fillna(df_processed[col].median())


# Для категориальных - модой

categorical_to_fill = ['certificate']

for col in categorical_to_fill:

    if col in df_processed.columns:

        mode_value = df_processed[col].mode()

        df_processed[col] = df_processed[col].fillna(mode_value[0] if
len(mode_value) > 0 else 'Unknown')


print("Пропуски после обработки:")

print(df_processed.isnull().sum())

 # 7. Обработка выбросов
print("\nОбработка выбросов для признака 'gross'...")


# Boxplot до обработки

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

sns.boxplot(data=df_processed, y='gross')

plt.title('Boxplot Gross до обработки')


# Метод IQR для обработки выбросов

Q1 = df_processed['gross'].quantile(0.25)

Q3 = df_processed['gross'].quantile(0.75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR

upper_bound = Q3 + 1.5 * IQR
```

```
print(f"Границы выбросов: [{lower_bound:.2f}, {upper_bound:.2f}]")



# Обрезка выбросов

df_processed['gross_processed'] =
df_processed['gross'].clip(lower=lower_bound, upper=upper_bound)



# Boxplot после обработки

plt.subplot(1, 2, 2)

sns.boxplot(data=df_processed, y='gross_processed')

plt.title('Boxplot Gross после обработки')

plt.tight_layout()

plt.show()



print(f"Количество выбросов до обработки: {((df_processed['gross'] <
lower_bound) | (df_processed['gross'] > upper_bound)).sum()}")

print(f"Количество выбросов после обработки:
{((df_processed['gross_processed'] < lower_bound) |
(df_processed['gross_processed'] > upper_bound)).sum()}")
```

Заключение

В ходе выполнения лабораторной работы был проведён исследовательский анализ данных (EDA) на основе набора данных IMDb, содержащего информацию о фильмах — включая название, год выпуска, рейтинг IMDb, метаоценку, сборы, жанры, длительность и другие характеристики.

Основная цель работы заключалась в том, чтобы подготовить данные к дальнейшему использованию в моделях машинного обучения или аналитике — провести очистку, преобразование и выявить потенциальные проблемы в исходных данных.

Основные выявленные проблемы данных:

Некоторые числовые признаки (например, runtime, gross, meta_score) были представлены как строки или содержали нечисловые символы (min, $, ,). Это делало невозможным их использование в расчетах без предварительного преобразования.
В данных присутствовали пропуски в таких столбцах, как meta_score, gross и certificate. Их наличие могло негативно повлиять на построение моделей, поэтому требовалась корректная обработка.
Некоторые значения года выпуска (released_year) содержали ошибки преобразования или отсутствовали, а длительность фильмов (runtime) была указана в текстовом виде.
В процессе анализа были замечены аномально большие значения сборов или рейтингов, что может потребовать дополнительного анализа перед обучением моделей.

Применённые методы предобработки:

Приведение всех названий столбцов к **нижнему регистру** для удобства работы и унификации. Преобразование признаков runtime, gross, meta_score, released_year и imdb_rating в **числовой формат** с помощью регулярных выражений и функции pd.to_numeric().

Очистка денежных значений — удаление символов $ и , перед преобразованием к числовому типу.

Обработка пропусков:

Для числовых признаков применено заполнение **медианой**, чтобы сохранить распределение и уменьшить влияние выбросов.

Для категориальных признаков (например, certificate) пропуски заполнены **модой** — наиболее часто встречающимся значением.

Выводы о проделанной работе:

В результате анализа и предобработки данные были приведены к чистому и унифицированному виду. Теперь набор данных готов к дальнейшему применению в аналитических и машинных задачах — например, для: построения моделей предсказания рейтинга фильма по характеристикам; анализа зависимости между доходами (gross) и рейтингами (imdb_rating); кластеризации фильмов по жанрам и популярности.

Проведённая предобработка устранила ключевые проблемы (несоответствие типов, пропуски и шум), что позволит существенно повысить **качество и устойчивость будущих моделей**. Кроме того, сформированный пайплайн очистки может быть использован повторно при обновлении или расширении набора данных.

Ссылка на выбранный датасет Kaggle

https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows

Ссылка на Google Colab

https://colab.research.google.com/drive/1ywzyf1vokB3uj0GKEXGtrWJz7BnYybn3?usp=sharing