Faculty of Engineering, Natural and Medical Sciences / Department of Information Technologies

IT 323 - DATA MINING PROJECT

Exploratory Data Analysis on IMDB Movies Dataset

Milestone 1

Author: Arnela Sokolić

01/04/2025

TABLE CONTENTS

1. INTRODUCTION

This project focuses on analyzing the IMDB Top 1000 Movies Dataset to uncover patterns and insights about highly rated movies.

Movies and TV shows play a crucial role in the entertainment industry, and understanding audience preferences can help predict trends, identify successful patterns, and improve recommendations in streaming services.

2. DATASET OVERVIEW

Dataset Name: IMDB Top 1000 Movies

Source: Kaggle (Original Dataset)

3. DESCRIPTION

This dataset contains the top 1000 movies and TV shows based on IMDB ratings. It includes detailed information about each movie, such as its title, release year, genre, director, actors, rating, and earnings.

Poster_Link - URL of the movie poster used on IMDB
Series_Title -Name of the movie or TV show
Released_Year -Year of release
Certificate -Age certification (e.g., PG-13, R)
Runtime -Total duration of the movie
Genre -Movie genres (e.g., Action, Drama, Comedy)
IMDB_Rating -Rating score on IMDB
Overview -Short summary or description of the movie
Meta_score - Metacritic score
Director -Name of the director
Star1, Star2, Star3, Star4 -Names of the main actors
No_of_votes -Total number of votes received
Gross -Total earnings of the movie

4. DATA COLLECTION

The dataset was obtained from Kaggle and downloaded in CSV format. This dataset serves as the foundation for analyzing trends in movie ratings, actor performance, and financial success.

5. PERFORMING DATA CLEANING AND PREPROCESSING

```python
import pandas as pd

# Load the dataset
file_path = r"C:\Users\Korisnik\Downloads\archive\imdb_top_1000.csv"

# Read the CSV file
df = pd.read_csv(file_path)

# Display the first 5 rows
print(df.head())
```

I first imported the pandas library. Then, I specified the file path where my IMDb dataset is stored. After that, I used pd.read_csv(file_path) to load the dataset into a DataFrame called df, so I can analyze it. Finally, I used df.head() to print the first five rows of the data, giving me a quick look at what's inside.

```
                                         Poster_Link  \
0  https://m.media-amazon.com/images/M/MV5BMDFkYT...
1  https://m.media-amazon.com/images/M/MV5BM2MyNj...
2  https://m.media-amazon.com/images/M/MV5BMTMxNT...
3  https://m.media-amazon.com/images/M/MV5BMWMwMG...
4  https://m.media-amazon.com/images/M/MV5BMWU4N2...

               Series_Title Released_Year Certificate  Runtime  \
0  The Shawshank Redemption          1994           A  142 min
1             The Godfather          1972           A  175 min
2           The Dark Knight          2008          UA  152 min
3    The Godfather: Part II          1974           A  202 min
4              12 Angry Men          1957           U   96 min

                  Genre  IMDB_Rating  \
0                 Drama          9.3
1          Crime, Drama          9.2
2  Action, Crime, Drama          9.0
3          Crime, Drama          9.0
4          Crime, Drama          9.0

                                            Overview  Meta_score  \
0  Two imprisoned men bond over a number of years...        80.0
1  An organized crime dynasty's aging patriarch t...       100.0
2  When the menace known as the Joker wreaks havo...        84.0
3  The early life and career of Vito Corleone in ...        90.0
4  A jury holdout attempts to prevent a miscarria...        96.0

               Director           Star1            Star2          Star3  \
0        Frank Darabont     Tim Robbins   Morgan Freeman     Bob Gunton
1  Francis Ford Coppola   Marlon Brando        Al Pacino     James Caan
2     Christopher Nolan  Christian Bale     Heath Ledger  Aaron Eckhart
3  Francis Ford Coppola       Al Pacino  Robert De Niro  Robert Duvall
4          Sidney Lumet     Henry Fonda      Lee J. Cobb  Martin Balsam

            Star4  No_of_Votes        Gross
0  William Sadler      2343110   28,341,469
1    Diane Keaton      1620367  134,966,411
2   Michael Caine      2303232  534,858,444
3    Diane Keaton      1129952   57,300,000
4     John Fiedler       689845    4,360,000
```

The output shows the first five rows of my IMDb dataset. Each row represents a movie with various details, such as the title, release year, certificate rating, runtime, genre, and IMDb rating. It also includes a brief overview of the plot, Meta score, director, and main cast members (Star1 to Star4). Additionally, the dataset provides the number of votes and box office gross earnings. The "Poster_Link" column contains URLs to the movie posters.

```
df.head (10)
```

The code df.head(10) retrieves and displays the first **10 rows** of the dataset stored in the DataFrame df.

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Sta
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	T Robbi
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Marl Bran
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Christi Ba
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202 min	Crime, Drama	9.0	The early life and career of Vito Corleone in ...	90.0	Francis Ford Coppola	Al Paci
4	https://m.media-amazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	U	96 min	Crime, Drama	9.0	A jury holdout attempts to prevent a miscarria...	96.0	Sidney Lumet	Her Fon
5	https://m.media-amazon.com/images/M/MV5BNzA5ZD...	The Lord of the Rings: The Return of the King	2003	U	201 min	Action, Adventure, Drama	8.9	Gandalf and Aragorn lead the World of Men agai...	94.0	Peter Jackson	Elij Wo
6	https://m.media-amazon.com/images/M/MV5BNGNhMD...	Pulp Fiction	1994	A	154 min	Crime, Drama	8.9	The lives of two mob hitmen, a boxer, a gangst...	94.0	Quentin Tarantino	Jo Travo

ased_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1	Star2	Star3	Star4	No_of_Votes	Gross
1994	A	142 min	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	Tim Robbins	Morgan Freeman	Bob Gunton	William Sadler	2343110	28,341,469
1972	A	175 min	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Marlon Brando	Al Pacino	James Caan	Diane Keaton	1620367	134,966,411
2008	UA	152 min	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Christian Bale	Heath Ledger	Aaron Eckhart	Michael Caine	2303232	534,858,444
1974	A	202 min	Crime, Drama	9.0	The early life and career of Vito Corleone in ...	90.0	Francis Ford Coppola	Al Pacino	Robert De Niro	Robert Duvall	Diane Keaton	1129952	57,300,000
1957	U	96 min	Crime, Drama	9.0	A jury holdout attempts to prevent a miscarria...	96.0	Sidney Lumet	Henry Fonda	Lee J. Cobb	Martin Balsam	John Fiedler	689845	4,360,000
2003	U	201 min	Action, Adventure, Drama	8.9	Gandalf and Aragorn lead the World of Men agai...	94.0	Peter Jackson	Elijah Wood	Viggo Mortensen	Ian McKellen	Orlando Bloom	1642758	377,845,905
1994	A	154 min	Crime, Drama	8.9	The lives of two mob hitmen, a boxer, a gangst...	94.0	Quentin Tarantino	John Travolta	Uma Thurman	Samuel L. Jackson	Bruce Willis	1826188	107,928,762
					In								

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Poster_Link    1000 non-null   object 
 1   Series_Title   1000 non-null   object 
 2   Released_Year  1000 non-null   object 
 3   Certificate    899 non-null    object 
 4   Runtime        1000 non-null   object 
 5   Genre          1000 non-null   object 
 6   IMDB_Rating    1000 non-null   float64
 7   Overview       1000 non-null   object 
 8   Meta_score     843 non-null    float64
 9   Director       1000 non-null   object 
 10  Star1          1000 non-null   object 
 11  Star2          1000 non-null   object 
 12  Star3          1000 non-null   object 
 13  Star4          1000 non-null   object 
 14  No_of_Votes    1000 non-null   int64  
 15  Gross          831 non-null    object 
dtypes: float64(2), int64(1), object(13)
memory usage: 125.1+ KB
```

The code df.info() provides a summary of the dataset. It displays the number of rows and columns, the column names, their data types, and the count of non-null values in each column. This helps me understand the dataset's structure, check for missing values, and identify if any data types need conversion for analysis.

```
df.describe()
```

	IMDB_Rating	Meta_score	No_of_Votes
count	1000.000000	843.000000	1.000000e+03
mean	7.949300	77.971530	2.736929e+05
std	0.275491	12.376099	3.273727e+05
min	7.600000	28.000000	2.508800e+04
25%	7.700000	70.000000	5.552625e+04
50%	7.900000	79.000000	1.385485e+05
75%	8.100000	87.000000	3.741612e+05
max	9.300000	100.000000	2.343110e+06

When I run df.describe(), it gives me a summary of all the numerical columns in my dataset. It shows me how many values exist in each column, along with key statistics like the average (mean), minimum, and maximum values.

It also tells me how spread out the data is using the standard deviation and shows quartiles like the median (50%) and the 25th and 75th percentiles. For example, the average IMDb rating is 7.95, while the highest-rated movie has a 9.3.

The Meta_score ranges from 28 to 100, and the number of votes varies widely, with the most-voted movie having over 2.3 million votes. This helps me understand the overall distribution of the data and spot any potential outliers.

```
num_records = len(df)

print('Total numbers of records: {}'.format(num_records))

print(df.shape[0])
print(df.shape[1])
```

```
Total numbers of records: 1000
1000
16
```

When I run this code, it calculates and prints the total number of records (rows) in my dataset. The line num_records = len(df) gets the total number of rows, which is 1000, and prints it using format(). Then, df.shape[0] confirms the number of rows, while df.shape[1] tells me the dataset has 16 columns. This helps me quickly check the dataset's size.

```
# Check for missing values in the dataset
print(df.isnull().sum())

# Check for duplicate rows in the dataset
print(df.duplicated().any())
```

```
Poster_Link        0
Series_Title       0
Released_Year      0
Certificate      101
Runtime            0
Genre              0
IMDB_Rating        0
Overview           0
Meta_score       157
Director           0
Star1              0
Star2              0
Star3              0
Star4              0
No_of_Votes        0
Gross            169
dtype: int64
False
```

When I run this code, it checks for missing values and duplicate rows in my dataset. The output shows that some columns have missing data, such as Certificate (101 missing values), Meta Score (157 missing values), and Gross earnings (169 missing values), while other columns have complete data. Then, it checks for duplicate rows and returns False, meaning there are no duplicate entries in my dataset. This helps me understand if I need to clean or fill in any missing values before analyzing the data.

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

# Load the dataset
df = pd.read_csv("C:\\Users\\Korisnik\\Downloads\\archive\\imdb_top_1000.csv")


# Calculate statistics
mean_value = df[numeric_column].mean()
median_value = df[numeric_column].median()
mode_value = df[numeric_column].mode()[0]
midrange = (df[numeric_column].min() + df[numeric_column].max()) / 2

# Compute quartiles
q1 = df[numeric_column].quantile(0.25)
q3 = df[numeric_column].quantile(0.75)
min_value = df[numeric_column].min()
max_value = df[numeric_column].max()

# Five-number summary
five_number_summary = {
    "Minimum": min_value,
    "Q1": q1,
    "Median": median_value,
    "Q3": q3,
    "Maximum": max_value
}

# Print statistics
print("\nStatistics:")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Midrange: {midrange}")
print(f"Five-Number Summary: {five_number_summary}")
```

This code analyzes IMDb movie data by calculating key statistics and visualizing the distribution of a selected numeric column.

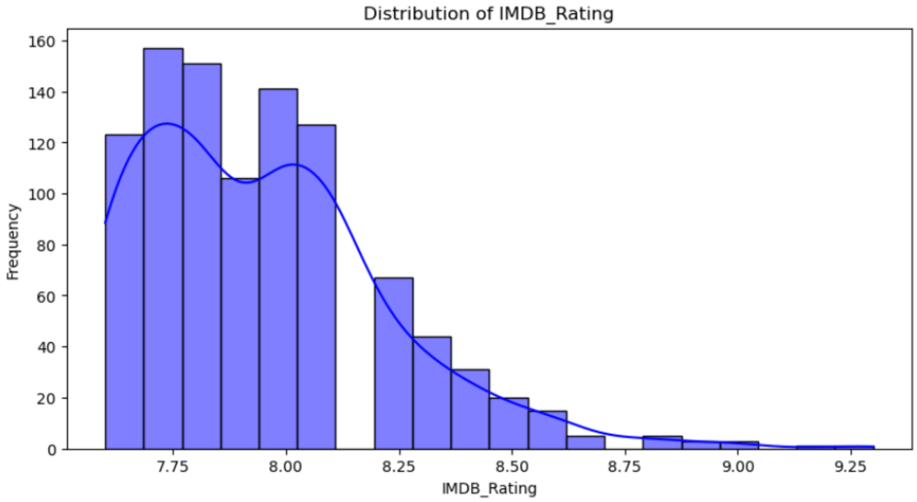
First, the dataset is loaded using pandas. The "Runtime" column, which contains values like "142 min", is cleaned by removing "min" and converting it to a float. Similarly, the "Gross" column, which has values with commas (e.g., "134,966,411"), is cleaned by removing commas and converting it into a float for numerical calculations.

Next, a numeric column is selected for analysis (default is "IMDB_Rating", but it can be changed to "Meta_score", "Runtime", or "Gross"). The code then calculates key statistics:

- **Mean** (average value of the column)
- **Median** (middle value of the sorted data)
- **Mode** (most frequently occurring value)
- **Midrange** (average of the minimum and maximum values)
- **Quartiles (Q1 and Q3)** and the **Five-Number Summary** (minimum, Q1, median, Q3, and maximum), which describe the spread of the data.

```
Statistics:
Mean: 7.949299999999999
Median: 7.9
Mode: 7.7
Midrange: 8.45
Five-Number Summary: {'Minimum': 7.6, 'Q1': 7.7, 'Median': 7.9, 'Q3': 8.1, 'Maximum': 9.3}
```

The IMDb ratings have an average (mean) of 7.95, with a median of 7.9, meaning half of the movies are rated below 7.9 and half above. The most common rating (mode) is 7.7, while the ratings range from a minimum of 7.6 to a maximum of 9.3, with most movies falling between 7.7 (Q1) and 8.1 (Q3).



This graph shows the distribution of IMDb ratings for the top 1000 movies. Most ratings fall between 7.5 and 8.5, with a peak around 7.8 to 8.0. Higher ratings, especially above 9.0, are rare. The curve shows a right-skewed pattern, meaning fewer movies have exceptionally high ratings. This is common since only a few films are considered true classics.

```python
print("Duplicates before:", df.duplicated().sum())

df.drop_duplicates(inplace=True)

print("Duplicates after:", df.duplicated().sum())
```

```
Duplicates before: 0
Duplicates after: 0
```

When I run this code, it checks for duplicate rows before and after attempting to remove them. The output shows "Duplicates before: 0", meaning there were no duplicate rows in the dataset to begin with.

After running df.drop_duplicates(inplace=True), the second check also confirms "Duplicates after: 0", which means no changes were made since there were no duplicates to remove. This tells me my dataset already contains only unique movie entries.

```
# Convert 'Released_Year' to numeric, coercing errors to NaN
df["Released_Year"] = pd.to_numeric(df["Released_Year"], errors='coerce')

# Convert to integer while keeping NaN values
df["Released_Year"] = df["Released_Year"].astype("Int64")

# Check new data types
print(df.dtypes)
```

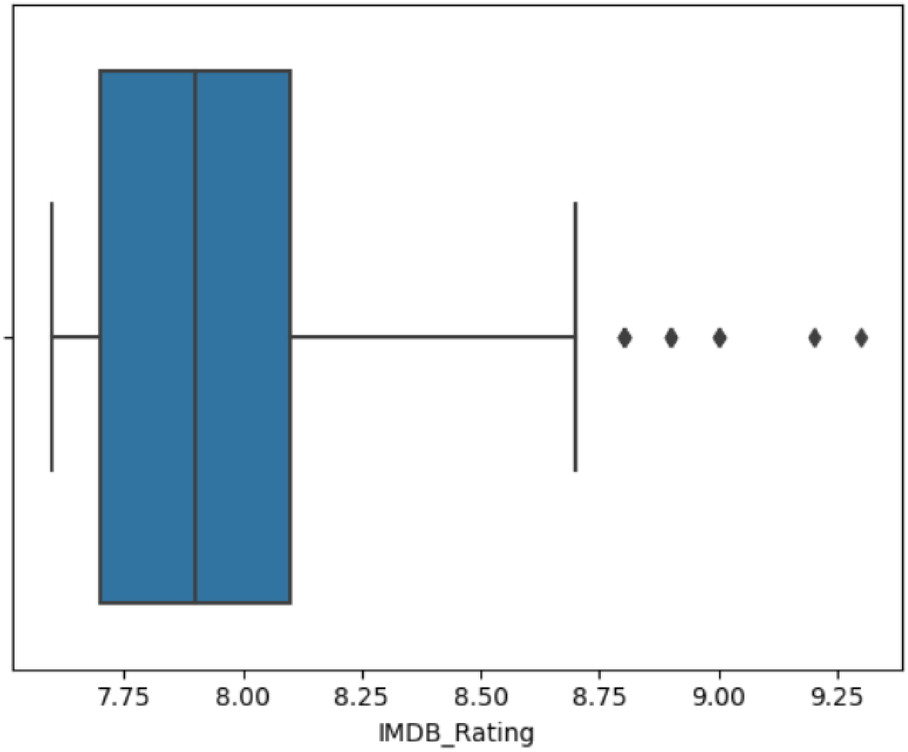
```
Poster_Link        object
Series_Title       object
Released_Year       Int64
Certificate        object
Runtime            object
Genre              object
IMDB_Rating       float64
Overview           object
Meta_score        float64
Director           object
Star1              object
Star2              object
Star3              object
Star4              object
No_of_Votes         int64
Gross              object
Decade            float64
dtype: object
```

This code ensures that the "Released_Year" column is properly converted into integers while handling missing values. It first changes non-numeric values to NaN, then converts the column to an integer format that allows missing values.

Finally, it prints the data types to confirm the changes. If needed, the "Decade" column can also be adjusted to store decade values as integers instead of floats. This cleanup makes the dataset more structured and ready for analysis.

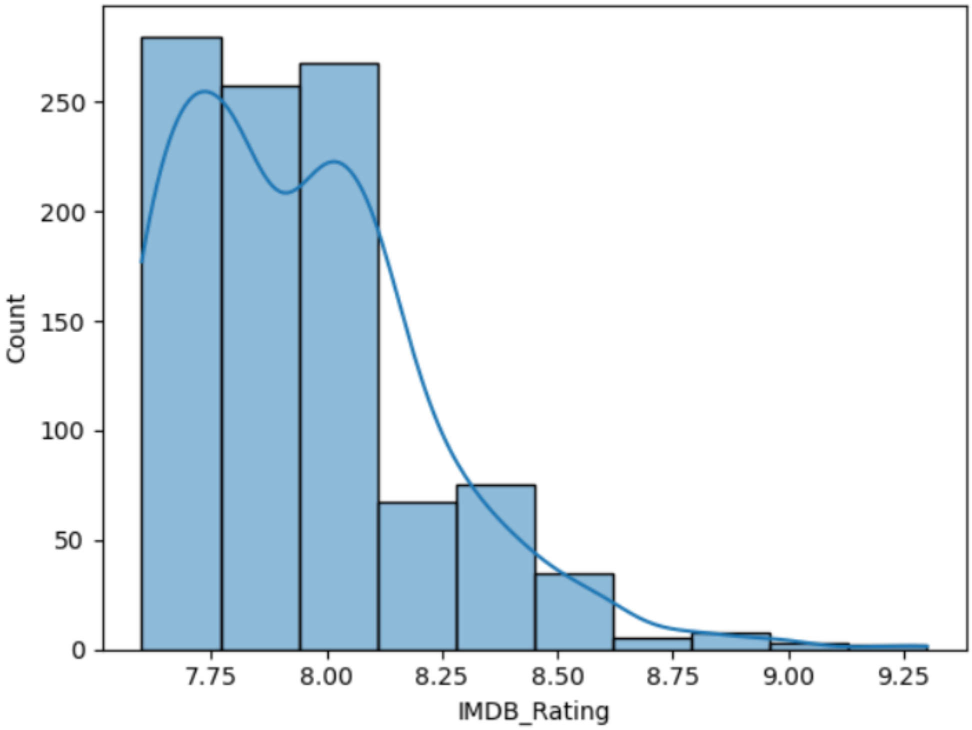
```python
import matplotlib.pyplot as plt
import seaborn as sns

sns.boxplot(x=df["IMDB_Rating"])
plt.show()
```



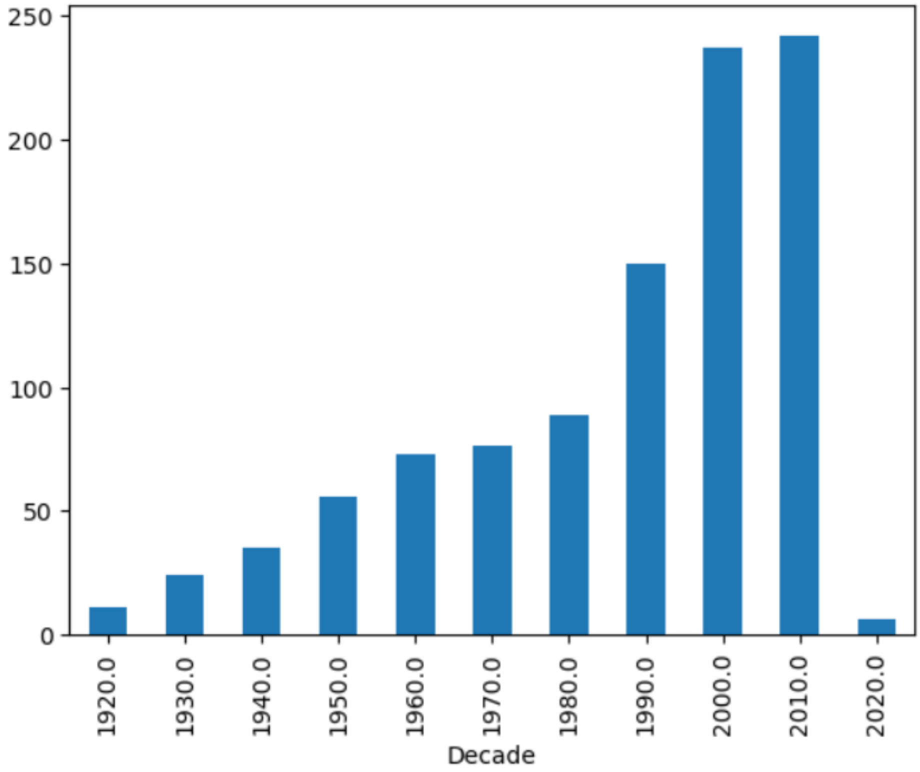
I created a boxplot to visualize the distribution of IMDb ratings in my dataset. The box represents the middle 50% of the ratings, with the line inside showing the median, which is around 8.0. The whiskers extend to most of the data, while the dots beyond them indicate outliers—movies with exceptionally high ratings above 8.6.

```
sns.histplot(df["IMDB_Rating"], bins=10, kde=True)
plt.show()
```



I created a histogram to visualize the distribution of IMDb ratings in my dataset. The bars show how frequently different rating ranges appear, with most movies clustered around 7.7 to 8.2. The KDE (Kernel Density Estimation) curve smooths out the distribution, making it easier to see the overall trend. The data is slightly right-skewed, meaning there are fewer movies with very high ratings above 8.5.

```
df["Decade"] = (df["Released_Year"] // 10) * 10
df["Decade"].value_counts().sort_index().plot(kind="bar")
plt.show()
```



I created a bar chart to show the number of movies released in each decade. The code first calculates the decade by rounding down the "Released_Year" to the nearest multiple of 10. Then, it counts how many movies belong to each decade and plots them in order.

The chart reveals an increasing trend in the number of top-rated movies over time, with the highest counts in the 2000s and 2010s. This suggests that more high-rated movies have been produced in recent decades.

```
print(df.isnull().sum())  # Check missing values in each column
```

```
Poster_Link      0
Series_Title     0
Released_Year    1
Certificate      0
Runtime          0
Genre            0
IMDB_Rating      0
Overview         0
Meta_score       0
Director         0
Star1            0
Star2            0
Star3            0
Star4            0
No_of_Votes      0
Gross            0
Decade           1
dtype: int64
```

When I run df.isnull().sum(), it checks for missing values in each column and counts how many are present.

The output shows that only two columns have missing values:

- "Released_Year" → 1 missing value
- "Decade" → 1 missing value

Since "Decade" is derived from "Released_Year", the missing value in "Released_Year" is likely causing the missing "Decade" value as well.

This means I need to fill or remove the missing "Released_Year" value so that "Decade" can also be corrected.

```
most_common_year = df["Released_Year"].mode()[0]
df["Released_Year"].fillna(most_common_year, inplace=True)
```

When I run this code, it fills the missing value in the "Released_Year" column with the most frequently occurring (mode) year in the dataset.

- df["Released_Year"].mode()[0] finds the most common year.
- df["Released_Year"].fillna(most_common_year, inplace=True) replaces the missing value with that year.

- inplace=True ensures the change is applied directly to the DataFrame without needing to reassign it.

Now, "Released_Year" no longer has missing values, and I can recalculate "Decade" to fix its missing value too.

```
df["Decade"] = (df["Released_Year"] // 10) * 10
```

This code recalculates the "Decade" column by rounding down "Released_Year" to the nearest multiple of 10. For example, 1994 becomes 1990, 2008 becomes 2000. This helps group movies by decade.

```
print(df.isnull().sum())
```

```
Poster_Link      0
Series_Title     0
Released_Year    0
Certificate      0
Runtime          0
Genre            0
IMDB_Rating      0
Overview         0
Meta_score       0
Director         0
Star1            0
Star2            0
Star3            0
Star4            0
No_of_Votes      0
Gross            0
Decade           0
dtype: int64
```

This output shows that there are no missing (null) values in any column of the dataset.

```
df["Gross"] = df["Gross"].replace(",", "", regex=True).astype(float)
```

This line of code cleans and converts the "Gross" column to a numeric format:

1. replace(",", "", regex=True) → Removes commas from numbers (e.g., "1,000,000" becomes "1000000").
2. .astype(float) → Converts the cleaned values to float type for numerical analysis.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Poster_Link    1000 non-null   object 
 1   Series_Title   1000 non-null   object 
 2   Released_Year  1000 non-null   Int64  
 3   Certificate    1000 non-null   object 
 4   Runtime        1000 non-null   object 
 5   Genre          1000 non-null   object 
 6   IMDB_Rating    1000 non-null   float64
 7   Overview       1000 non-null   object 
 8   Meta_score     1000 non-null   float64
 9   Director       1000 non-null   object 
 10  Star1          1000 non-null   object 
 11  Star2          1000 non-null   object 
 12  Star3          1000 non-null   object 
 13  Star4          1000 non-null   object 
 14  No_of_Votes    1000 non-null   int64  
 15  Gross          1000 non-null   float64
 16  Decade         1000 non-null   Int64  
dtypes: Int64(2), float64(3), int64(1), object(11)
memory usage: 134.9+ KB
```

The dataset has 1000 entries with 17 columns, all of which have no missing values. It contains text (11 object columns), numbers (2 Int64, 1 int64, and 3 float64), and takes 134.9 KB of memory, making it clean and ready for analysis.

```
df = df[(df["Released_Year"] >= 1900) & (df["Released_Year"] <= 2025)]  # Keep valid years only
```

This filters the dataset to keep only rows where "Released_Year" is between 1900 and 2025, ensuring all movie years are valid. Any row with an invalid year outside this range is removed.

```
print(df["Released_Year"].min(), df["Released_Year"].max())
```

```
1920 2020
```

This prints the earliest (min) and latest (max) release years in the dataset. The dataset contains movies from 1920 to 2020.

```
print(df[~df["Released_Year"].between(1900, 2025)])
```

```
Empty DataFrame
Columns: [Poster_Link, Series_Title, Released_Year, Certificate, Runtime, Genre, IMDB_Rating, Overview, Meta_score, Director, S
tar1, Star2, Star3, Star4, No_of_Votes, Gross, Decade]
Index: []
```

This checks if there are any movies outside the range of 1900 to 2025 in the dataset. The result is an empty DataFrame, meaning all movies fall within this range and no invalid years exist.

```
print("Rows before filtering:", len(df))
df = df[(df["Released_Year"] >= 1900) & (df["Released_Year"] <= 2025)]
print("Rows after filtering:", len(df))
```

```
Rows before filtering: 1000
Rows after filtering: 1000
```

This checks how many rows are in the dataset before and after filtering for valid years (1900-2025). Since the number of rows remains 1000, it means all data already met the criteria, and no rows were removed.

6. SUMMARY OF MILESTONE 1

I started by loading the IMDb Top 1000 Movies dataset into a pandas DataFrame and performed an initial inspection using .head(), .info(), and .describe() to understand its structure. I checked for missing values and duplicates, ensuring data integrity. Missing values in relevant columns were handled, such as filling in missing years with the most common value and recalculating the 'Decade' column accordingly. I converted columns like 'Released_Year' and 'Gross' into appropriate numeric formats for analysis. To ensure data consistency, I filtered out any invalid release years outside the range of 1900-2025. Finally, I visualized key trends using boxplots and histograms to understand the distribution of IMDb ratings and movie release trends across decades.

7. BACKGROUND RESEARCH AND LITERATURE REVIEW

Before conducting my exploratory analysis on the IMDb Top 1000 movies, I explored similar research projects to understand how others have approached movie data analysis. This background research helped me identify key factors influencing movie success, common data analysis techniques, and effective ways to visualize insights. By studying existing projects, I was able to refine my methodology, ensuring that my work builds on previous findings while offering new perspectives.

I came across several studies that analyzed movie datasets, each with its own focus. Some explored how different attributes—like genre, budget, and release date—affect a movie's success, while others examined audience sentiment or industry trends over time. Seeing the variety of methods used gave me a clearer idea of how to structure my analysis and what insights I should be looking for.

Below, I summarize three related projects that apply exploratory data analysis to movie-related datasets. These studies, while not identical to mine, share similar objectives and methodologies, offering useful comparisons for my research.

---

1. An Exploratory Data Analysis of Movie Review Dataset

This study examines a dataset of movie reviews to uncover patterns affecting a film's success. The researchers analyzed multiple attributes, including genre, release date, language, country of origin, budget, and revenue. In addition to these direct attributes, the study derived new variables, such as release month and profit margins, to assess their influence on movie performance.

The methodology involved statistical observations and data visualization techniques, including bar charts and heatmaps, to identify trends in the dataset. Key findings showed that Drama and Comedy were the most common genres, December was a peak release month associated with higher revenues, and English-language films from the USA dominated the dataset.

This research helped me understand the importance of considering multiple attributes when analyzing movies. It also reinforced the value of visualizing data to identify trends more effectively, which influenced my decision to use graphical representations in my project.

Link to the study: An Exploratory Data Analysis of Movie Review Dataset

---

2. Exploratory and Sentiment Analysis of Netflix Data

This project combines exploratory data analysis (EDA) and sentiment analysis on a dataset of Netflix content. The researchers aimed to uncover patterns in viewer preferences and content trends by integrating multiple data sources, including geographical data and user reviews.

The EDA phase focused on analyzing the distribution of content types (movies vs. series), genre popularity, and IMDb scores. The study also incorporated sentiment analysis to assess audience perceptions based on user reviews. Python's data visualization libraries, such as Matplotlib and Seaborn, were used to create correlation heatmaps and genre-based word clouds.

The research findings provided insights into the types of content that perform well on streaming platforms, the impact of user sentiment on ratings, and trends in content consumption. This study was particularly useful for my project because it demonstrated how sentiment analysis could complement traditional exploratory analysis, inspiring me to consider user feedback and ratings as part of my research.

Link to the study: Exploratory and Sentiment Analysis of Netflix Data

---

3. Exploratory Data Analysis of the IMDb's Movie Database from a Data Scientist's Perspective

This study focuses on IMDb's movie database to identify trends in film production, ratings, and duration over time. The researcher examined the number of movies produced annually, the average ratings across different periods, and the correlation between film length and audience ratings.

One of the key findings was that film production has steadily increased over the years, with a peak in recent decades. The study also observed that the average IMDb ratings remained relatively stable, with most movies receiving scores around 6/10. Additionally, a slight correlation was found between film duration and higher ratings, suggesting that longer movies tend to receive better audience scores.

This project helped me understand the significance of historical trends in movie analysis. By looking at how film production and ratings evolved over time, I realized the importance of including time-based patterns in my own study.

Link to the study: Exploratory Data Analysis of the IMDb's Movie Database from a Data Scientist's Perspective

Comparison of Related Studies

To summarize the differences and similarities between these projects, the following table provides a structured comparison:

Aspect	An Exploratory Data Analysis of Movie Review Dataset	Exploratory and Sentiment Analysis of Netflix Data	Exploratory Data Analysis of the IMDb's Movie Database
Objective	Analyze factors influencing movie success	Analyze Netflix content trends and user sentiment	Identify trends in IMDb's movie database over time
Methodology	Statistical analysis, bar charts, heatmaps	EDA, sentiment analysis, visualizations	EDA, time-based trend analysis
Key Findings	Drama and Comedy are most common genres; December releases correlate with higher revenue; English-language films dominate	Netflix content distribution patterns; genre popularity; IMDb rating trends; user sentiment analysis	Film production has increased; IMDb ratings remain stable; longer films tend to have higher ratings
Tools Used	Data visualization tools (unspecified)	Python (Matplotlib, Seaborn), NLP for sentiment analysis	Python (Matplotlib, Seaborn), time series analysis
Dataset Source	Movie review dataset	Netflix dataset from Kaggle, supplemented with additional data	IMDb movie database

Conclusion

These studies provided me with a strong foundation for my own research. By analyzing different aspects of movie data—financial success, audience sentiment, and historical trends—I gained insights into various approaches and methodologies. Reviewing these projects highlighted the importance of data visualization and the value of combining quantitative and qualitative data.

8. REFERENCES

[1] What is Natural Language Processing? | IBM

[2] What is Exploratory Data Analysis? | IBM

[3] Intro-to-exploratory-data-analysis-eda-in-python

[4] Hands-On Exploratory Data Analysis with Python

[5] Cleaning dataset in Python

[6] Exploratory data analysis on Netflix Dataset

[7] Exploratory Data Analysis and Latent Dirichlet Allocation

[8] An Exploratory Data Analysis of Movie Review Dataset

[9] Exploratory Data Analysis of the IMDb's Movie Database from a Data Scientist's Perspective

[10] Exploratory and Sentiment Analysis of Netflix Data