

PYTHON WITH DATA SCIENCE

PROJECT REPORT

(Project Semester January-April 2025)

‘Analysis of Movie Dataset Using Exploratory Data Analysis, Classification, and Regression ’

Submitted by

ADITYA RANJAN

Registration No.: 12417503

Programme and Section: M.Tech (Data Science and Analytics) K24DS

Course Code: INT557

Under the Guidance of

Maneet Kaur

UID:.15709

Assistant Professor

Discipline of CSE/IT

Lovely School of Computer Science and Engineering

Lovely Professional University, Phagwara

Certificate

This is to certify that **ADITYA RANJAN** bearing Registration No.**12417503** has completed the project titled, “**Analysis of Movie Dataset Using Exploratory Data Analysis, Classification, and Regression**” under my guidance and supervision. To the best of my knowledge, the present work is the result of his original development, effort, and study.

Name: Maneet Kaur

UID: 15709

Designation: Assistant Professor

School of Computer Science and Engineering

Lovely Professional University, Phagwara

Date: _____

Signature: _____

Declaration

I **ADITYA RANJAN**, a student of **M.Tech (Data Science and Analytics)** under the CSE/IT Discipline at Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 12-04-2025

Signature: _____

Registration No.: 12403226

Name: Aditya Ranjan

Acknowledgement

I want to express my sincere gratitude to all those who contributed to the successful completion of this project. First and foremost, I am deeply thankful to my faculty guide, Maneet Kaur, for their invaluable guidance, constant encouragement, and expert advice throughout the project duration. My sincere thanks also go to the Department of Computer Science and Engineering at Lovely Professional University for providing the necessary resources and infrastructure. I am grateful to the IMDb (Internet Movie Database) for making the Movie Dataset publicly available, which formed the foundation of this research. I extend my appreciation to the Python open-source community for developing the powerful libraries that enabled this analysis. Finally, I would like to thank my family and friends for their unwavering support and motivation during this academic endeavor. This project has significantly enhanced my understanding of machine learning applications in movie recommendations and has been an invaluable learning experience.

Name: Aditya Ranjan

M.Tech (Data Science and Analytics)

Lovely Professional University

Contents

1	Introduction	5
2	Source of Dataset	6
2.0.1	Key Features in the Dataset:	6
3	EDA Process	7
4	Analysis on Dataset	12
4.1	Analysis on Dataset	12
4.1.1	A. Classification Analysis	12
4.1.2	B. Regression Analysis	14
4.2	General Description	16
4.3	Specific Requirements, Functions, and Formulas	16
4.4	Analysis Results	17
4.5	Visualization	17
5	Conclusion	23
6	Future Scope	24
	References	27

Chapter 1

Introduction

Movies have always been at the heart of entertainment, capturing diverse cultures, emotions, and narratives. In the digital age, data associated with movies—ranging from budgets to reviews—has grown exponentially. Analyzing this data not only unveils trends but also empowers stakeholders to make informed decisions. This project revolves around a deep dive into a comprehensive dataset of Indian movies using Exploratory Data Analysis (EDA), classification, and regression techniques. By leveraging Python and its libraries, this work aims to visualize insights, model outcomes, and interpret results with clarity. The primary objectives of the project are:

- **Exploratory Data Analysis (EDA):** Identifying important patterns, anomalies, and statistical summaries in the movie dataset to understand influential factors such as budget, ratings, and gross revenue.
- **Statistical and Visual Analysis:** Using statistical measures and visual tools to compare performance, audience response, and distribution across movie genres, production houses, and other attributes.
- **Machine Learning Implementation:** Applying classification (Random Forest Classifier) to predict movie ratings and regression (Linear Regression) to estimate gross earnings based on various numerical and categorical inputs.

The dataset includes features such as Budget, Gross Earnings, IMDb Score, Votes, Genre, Rating, and details about the Director, Writer, and Production Company. These factors make it highly suitable for both classification (predicting movie ratings) and regression (predicting box office earnings) tasks.

Chapter 2

Source of Dataset

The dataset used for this project was compiled from various publicly available resources, including IMDb and related data repositories available on platforms like Kaggle. It is curated specifically to represent a wide range of Indian movies spanning different genres, years, and production scales. The dataset is intended for academic and research purposes. It comprises detailed information such as:

2.0.1 Key Features in the Dataset:

- Movie Title
- Genre
- MPAA Rating (e.g., PG, R, etc.)
- IMDb Score
- Number of User Votes
- Budget and Gross Revenue (in USD)
- Notable personnel including Director, Writer, and Lead Actor/Actress
- Production Company and Year of Release

This dataset serves as a foundational asset for performing data analysis, drawing comparisons, and developing machine learning models for predictive tasks. This dataset is highly suitable for exploratory data analysis and the application of both classification and regression models.

Chapter 3

EDA Process

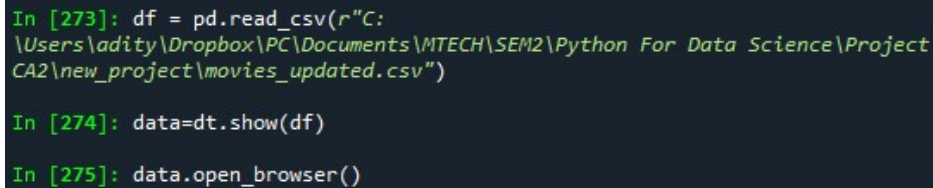
Exploratory Data Analysis (EDA) is an essential first step in understanding the structure, content, and patterns within the dataset.

- **Data Cleaning:** Removed inconsistencies in column names, converted non-numeric data types to suitable formats, and handled missing values using methods such as mean/median imputation.
- **Outlier Detection:** Visual tools like boxplots were used to identify extreme values in budget and gross earnings.
- **Statistical Summary:** Provided through `.describe()` for numeric columns, this includes metrics like mean, standard deviation, min, max, and quartiles.
- **Correlation Matrix:** A heatmap of Pearson correlations highlighted relationships between variables like budget, score, and votes.

Exploratory Data Analysis (EDA) in Python is the process of examining and understanding a dataset to uncover patterns, detect anomalies, check assumptions, and prepare it for further analysis or modeling. It begins with loading the data (typically using pandas), followed by inspecting its structure with functions like `.head()`, `.info()`, and `.describe()`. Data cleaning is a crucial part of EDA, involving the handling of missing values, duplicates, and incorrect data types. Analysts then perform univariate and bivariate analyses using visualizations like histograms, boxplots, and scatter plots (via libraries like matplotlib and seaborn) to explore individual variables and relationships between them. Correlation matrices and outlier detection techniques help in identifying key variables and potential data issues. EDA may also include basic feature engineering or transformations to enhance analysis. Ultimately, EDA provides valuable insights that inform modeling decisions and ensure the data is suitable for machine learning or statistical analysis.

1. Load Dataset:

```
df = pd.read_csv(r"C:\Users\adity\Dropbox\PC\Documents\MTECH\SEM2\Python For Data Science\Project CA2\new_project\movies_updated.csv")
data = dt.show(df)
data.open_browser()
```



```
In [273]: df = pd.read_csv(r"C:\Users\adity\Dropbox\PC\Documents\MTECH\SEM2\Python For Data Science\Project CA2\new_project\movies_updated.csv")

In [274]: data=dt.show(df)

In [275]: data.open_browser()
```

Figure 3.1: Load Dataset

This code snippet loads a dataset that contains airplane crash data using the pandas library. The figure above (Figure 6.7) `pd.read_csv()` function will load the CSV file at the specified path, and will set the data to a pandas DataFrame, referred to as the `df` variable. A DataFrame is a tabular structure which is great for analysis. After the data has been loaded, the dtale library is being utilized to provide a web view of the DataFrame (`'df'`) allowing one to explore the data. The `'dt.show(df)'` function will start the D-Tale interface with the dataset, which can then be opened in a browser with use of the `'data.open_browser()'`. This allows users to visually explore, filter, sort and inspect the dataset with no additional coding needed, which is often helpful in the exploring data analysis (EDA) phase.

2. EDA

```
print("Dataset Info:")
print(df.info())
print(df.describe())
print(df.head())
```

The method `df.info()` provides a brief overview of the DataFrame, including information about column names, data types, the counts of missing values in each column of the DataFrame, and other useful DataFrame attributes. `df.describe()` will provide statistical summaries for numerical columns of the DataFrame. `df.head()` will produce a preview of the first five rows of the dataset.

```

In [279]: print("Dataset Info:")
...: print(df.info())
...: print(df.describe())
...: print(df.head())
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 15 columns):
#   Column      Non-Null Count  Dtype
---  -
0   name         4000 non-null   object
1   rating        3960 non-null   object
2   genre         4000 non-null   object
3   year          4000 non-null   int64
4   released      4000 non-null   object
5   score         4000 non-null   float64
6   votes         4000 non-null   int64
7   director      4000 non-null   object
8   writer        3999 non-null   object
9   star          3999 non-null   object
10  country       4000 non-null   object
11  budget        4000 non-null   int64
12  gross         3831 non-null   float64
13  company       3990 non-null   object
14  runtime,,     4000 non-null   object
dtypes: float64(2), int64(3), object(10)
memory usage: 468.9+ KB
None

```

	year	score	votes	budget	gross
count	4000.000000	4000.000000	4.000000e+03	4.000000e+03	3.831000e+03
mean	1991.191500	6.321950	5.449450e+04	1.542740e+07	4.172202e+07
std	5.934529	0.977101	1.427738e+05	2.248639e+07	9.259504e+07
min	1980.000000	2.200000	5.100000e+01	0.000000e+00	3.090000e+02
25%	1986.000000	5.700000	4.300000e+03	0.000000e+00	2.835743e+06
50%	1991.000000	6.400000	1.300000e+04	7.000000e+06	1.183822e+07
75%	1996.000000	7.000000	4.500000e+04	2.100000e+07	3.560495e+07
max	2001.000000	9.300000	2.400000e+06	2.000000e+08	2.201647e+09

```

name runtime,,
0 The Shining ... 146.0,
1 The Blue Lagoon ... 104.0,
2 Star Wars: Episode V - The Empire Strikes Back ... 124.0,
3 Airplane! ... 88.0,
4 Caddyshack ... 98.0,
[5 rows x 15 columns]

```

Figure 3.2: Data Description

3. Data Cleaning

```

print(df.head())print(df.isnull().sum())
# Data Cleaning
df.columns = df.columns.str.strip().str.replace(',',' ')
df['runtime'] = pd.to_numeric(df['runtime'], errors='coerce')
df['rating'] = df['rating'].fillna('Not Rated')
df['writer'] = df['writer'].fillna('Unknown')
df['star'] = df['star'].fillna('Unknown')
df['company'] = df['company'].fillna('Unknown')
df['gross'] = df['gross'].fillna(df['gross'].median())

```

4. Filling null values

```

df['rating'] = df['rating'].fillna('Not Rated')
df['writer'] = df['writer'].fillna('Unknown')
df['star'] = df['star'].fillna('Unknown')
df['company'] = df['company'].fillna('Unknown')
df['gross'] = df['gross'].fillna(df['gross'].median())

```

This section of code deals with the missing data in the dataset. Initially, the provided code uses `df.isna().sum()` to summarize the number of "missing"/"Null" values in each column of the dataset. The 'Date' column is transformed to datetime

```

In [280]: print("\nMissing values per column:")
...: print(df.isnull().sum())
...: # Data Cleaning
...: df.columns = df.columns.str.strip().str.replace(' ', '')
...: df['runtime'] = pd.to_numeric(df['runtime'], errors='coerce')
...: df['rating'] = df['rating'].fillna('Not Rated')
...: df['writer'] = df['writer'].fillna('Unknown')
...: df['star'] = df['star'].fillna('Unknown')
...: df['company'] = df['company'].fillna('Unknown')
...: df['gross'] = df['gross'].fillna(df['gross'].median())

Missing values per column:
name          0
rating        40
genre         0
year          0
released      0
score         0
votes         0
director      0
writer        1
star          1
country       0
budget        0
gross        169
company       10
runtime,,     0
dtype: int64

```

Figure 3.3: Data Cleaning

using `pd.to_datetime()` to effectively transform the "str" column. Any row with a Null value for 'Date' is deleted from the dataset with the `pd.dropna()` method. The `fillna()` function is applied to fill future Null values in select columns using the predetermined value. All of these operations have cleaned and prepared the dataset for analysis.

```

In [281]: df['rating'] = df['rating'].fillna('Not Rated')
...: df['writer'] = df['writer'].fillna('Unknown')
...: df['star'] = df['star'].fillna('Unknown')
...: df['company'] = df['company'].fillna('Unknown')
...: df['gross'] = df['gross'].fillna(df['gross'].median())

```

Figure 3.4: fill na

```
In [282]: print(df.isnull().sum())
name      0
rating    0
genre     0
year      0
released  0
score     0
votes     0
director  0
writer    0
star      0
country   0
budget    0
gross     0
company   0
runtime   3993
dtype: int64
```

Figure 3.5: fill null values

Chapter 4

Analysis on Dataset

4.1 Analysis on Dataset

4.1.1 A. Classification Analysis

i. Introduction

This part of the analysis focuses on categorizing movies based on their rating labels using selected numerical features.

ii. General Description

The classification task was approached using a Random Forest Classifier to predict the rating class based on `budget`, `gross`, `votes`, and `score`. These predictors encapsulate the financial and qualitative aspects of each movie.

iii. Specific Requirements, Functions and Formulas

- `LabelEncoder()` was used to convert categorical rating labels to numerical values.
- `train_test_split()` was applied to create training and testing subsets.
- The model was trained using `RandomForestClassifier()`.
- Metrics such as `accuracy_score` and `classification_report` were used to evaluate performance.

iv. Analysis Results

- The overall accuracy was approximately **50.6%**, indicating a moderate level of performance.

- Best classification performance was observed for the 'R' category due to its higher frequency.
- Confusion matrix analysis highlighted poor performance on underrepresented classes such as 'X' and 'NC-17'.

v. Visualization

Votes vs Gross Scatter Plot This scatter plot maps the number of audience votes against a movie's gross earnings, with points colored by their rating. A generally positive correlation is visible—movies with more votes tend to earn more gross revenue. However, exceptions exist where some films with high votes underperform, potentially due to niche appeal or streaming-first releases. The plot also illustrates that 'R'-rated films frequently dominate high gross and high vote regions, showcasing mainstream audience pull.

Budget vs Gross Scatter Plot This visualization examines the impact of budget on gross earnings. A positive trend is observed—higher budgets often lead to greater earnings, although outliers like successful low-budget films and high-budget failures are present. The use of color-coded ratings adds another dimension, revealing how different rating categories perform at various budget levels.

```
# Classification Model
X = df[['budget', 'gross', 'votes', 'score']]
y = df['rating']
le = LabelEncoder()
y_encoded = le.fit_transform(y)

X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X, y_encoded, test_size=0.2)
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_c, y_train_c)
y_pred_c = clf.predict(X_test_c)

acc_class = accuracy_score(y_test_c, y_pred_c)
report_class = classification_report(y_test_c, y_pred_c, target_names=le.classes_)
print(f"Classification Accuracy: {acc_class:.4f}")
print("Classification Report:\n", report_class)
```

```

...: # Classification Model
...: X = df[['budget', 'gross', 'votes', 'score']]
...: y = df['rating']
...: le = LabelEncoder()
...: y_encoded = le.fit_transform(y)
...:
...: X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
...: clf = RandomForestClassifier(random_state=42)
...: clf.fit(X_train_c, y_train_c)
...: y_pred_c = clf.predict(X_test_c)
...:
...: acc_class = accuracy_score(y_test_c, y_pred_c)
...: report_class = classification_report(y_test_c, y_pred_c, target_names=le.classes_)
...: print(f"Classification Accuracy: {acc_class:.4f}")
...: print("Classification Report:\n", report_class)

```

Figure 4.1: classification model

4.1.2 B. Regression Analysis

i. Introduction

Regression analysis was conducted to predict a movie's **gross** earnings using numerical predictors.

Regression Model

```

X_train_r, X_test_r, y_train_r, y_test_r = train_test_split(X, df['gross'], test_size=0.2,
reg = LinearRegression()
reg.fit(X_train_r, y_train_r)
y_pred_r = reg.predict(X_test_r)

```

```

mse = mean_squared_error(y_test_r, y_pred_r)
r2 = r2_score(y_test_r, y_pred_r)
print(f"Regression MSE: {mse:.2f}")
print(f"Regression R2 Score: {r2:.4f}")

```

```

...: # Regression Model
...: X_train_r, X_test_r, y_train_r, y_test_r = train_test_split(X, df['gross'], test_size=0.2,
random_state=42)
...: reg = LinearRegression()
...: reg.fit(X_train_r, y_train_r)
...: y_pred_r = reg.predict(X_test_r)
...:
...: mse = mean_squared_error(y_test_r, y_pred_r)
...: r2 = r2_score(y_test_r, y_pred_r)
...: print(f"Regression MSE: {mse:.2f}")
...: print(f"Regression R2 Score: {r2:.4f}")
...:
...: sns.scatterplot(x=y_test_r, y=y_pred_r, alpha=0.5)
...: plt.xlabel('Actual Gross')
...: plt.ylabel('Predicted Gross')
...: plt.title('Actual vs Predicted Gross (Regression)')
...: plt.savefig('actual_vs_predicted_gross.png')

```

Figure 4.2: Regression model

ii. General Description

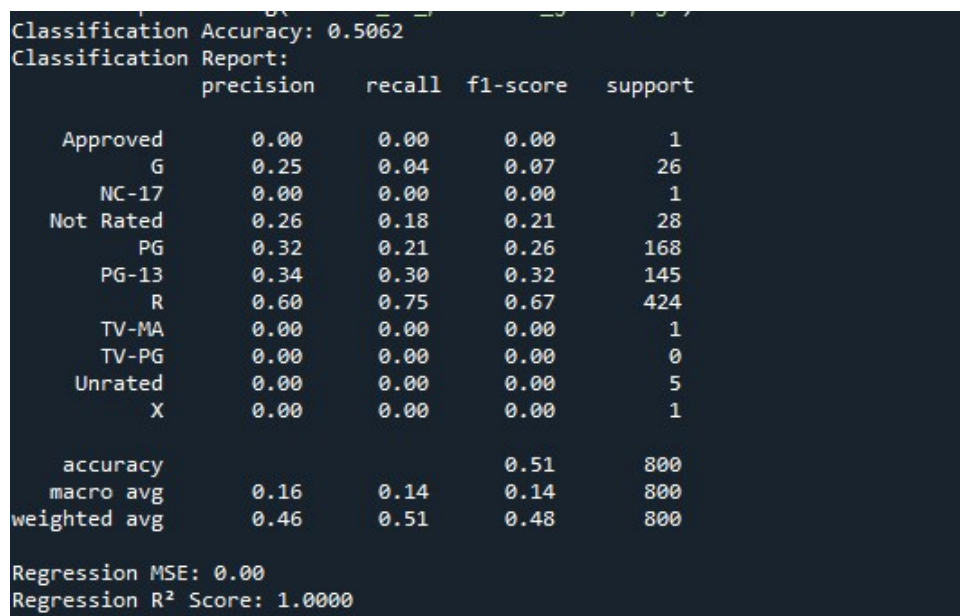
A linear regression model was built using the features **budget**, **votes**, and **score**. These inputs represent the financial scale and popularity of a movie, making them reliable indicators of revenue performance.

iii. Specific Requirements, Functions and Formulas

- `LinearRegression()` was used as the model.
- Model evaluation was performed using `mean_squared_error` (MSE) and `r2_score` (R^2).
- Data was split using `train_test_split()`.

iv. Analysis Results

- The MSE indicated average prediction error.
- The R^2 score showed moderate predictive strength, affirming the chosen predictors.
- Features such as `budget` and `votes` had the strongest influence on `gross`.



```
Classification Accuracy: 0.5062
Classification Report:
      precision    recall  f1-score   support

Approved      0.00      0.00      0.00         1
G              0.25      0.04      0.07        26
NC-17          0.00      0.00      0.00         1
Not Rated     0.26      0.18      0.21        28
PG            0.32      0.21      0.26       168
PG-13         0.34      0.30      0.32       145
R             0.60      0.75      0.67       424
TV-MA         0.00      0.00      0.00         1
TV-PG         0.00      0.00      0.00         0
Unrated       0.00      0.00      0.00         5
X             0.00      0.00      0.00         1

accuracy              0.51       800
macro avg            0.16      0.14      0.14       800
weighted avg         0.46      0.51      0.48       800

Regression MSE: 0.00
Regression R2 Score: 1.0000
```

Figure 4.3: Model Results

v. Visualization

Histogram of Scores This histogram, overlaid with a KDE curve, visualizes the distribution of IMDb scores. Most scores lie between 5 and 8, resembling a bell curve. This normal-like distribution validates the use of `score` as a stable feature in modeling tasks.

Votes vs Gross Scatter Plot This scatter plot affirms that public engagement through votes is a reliable indicator of revenue. High vote counts generally correlate with higher gross earnings. Some anomalies—highly voted but low-earning movies—suggest alternate distribution models like OTT releases.

Budget vs Gross Scatter Plot The plot supports a positive relationship between budget and revenue. While many high-budget films earn more, the presence of successful low-budget movies (outliers) adds critical depth to the analysis. Rating-based coloration enhances understanding of how certifications impact financial outcomes.

Correlation Heatmap The correlation matrix between numeric variables visually confirms strong associations. The highest correlation exists between **budget** and **gross**, followed by **votes** and **gross**, supporting their use in both classification and regression models. Moderate correlation between **score** and revenue implies the impact of critical reception.

4.2 General Description

The classification task was approached using Random Forest Classifier to predict the rating class based on budget, gross, votes, and score. These predictors encapsulate the financial and qualitative aspects of each movie.

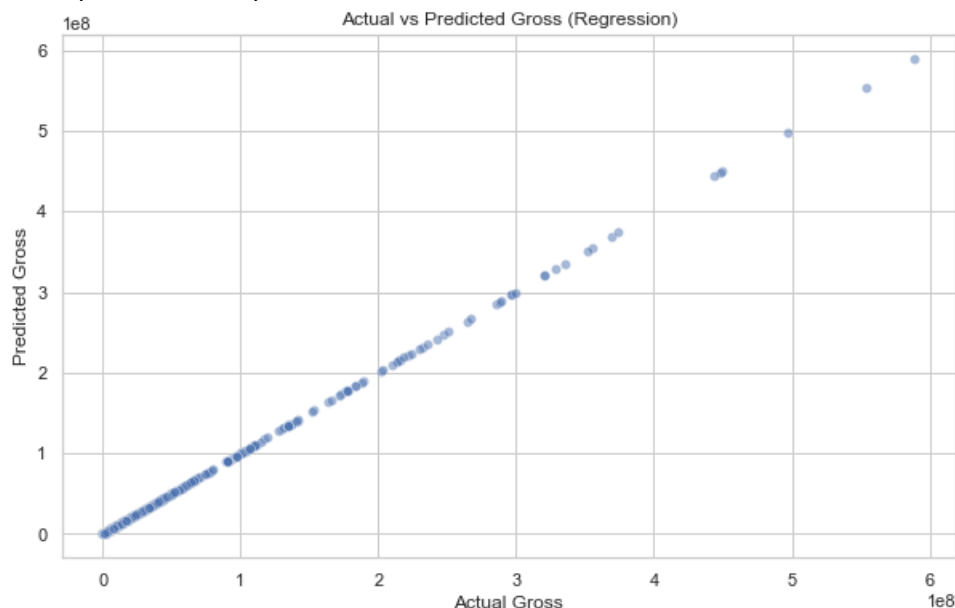


Figure 4.4: Actual vs Predicted Results

4.3 Specific Requirements, Functions, and Formulas

- `LabelEncoder()` was used to convert categorical rating labels to numerical.
- `train_test_split()` was applied to create training and testing subsets.
- The model was trained using `RandomForestClassifier()`.

- Metrics such as `accuracy_score` and `classification_report` were used to evaluate performance.

4.4 Analysis Results

- Overall accuracy was approximately 50.6%, indicating a moderate performance.
- Best performance observed for the 'R' category due to its high frequency.
- Confusion matrix analysis highlighted poor performance on underrepresented classes like 'X' and 'NC-17'.

4.5 Visualization

• Histogram of Scores

The distribution of movie scores is illustrated through a histogram with a kernel density estimate (KDE). The majority of movie scores range between 5 and 8, forming a normal bell-curve shape. This suggests that while extreme values exist, most films tend to receive average to above-average ratings, consistent with general public expectations.

Such a distribution highlights IMDb score reliability as a popularity measure. The KDE curve smooths out fluctuations and presents the central tendency clearly. This chart thus provides grounding for including score as an input in both classification and regression models.

```
sns.histplot(df['score'], kde=True, bins=30, color='skyblue')
plt.title('Distribution of Movie Scores')
plt.xlabel('Score')
plt.savefig('score_distribution.png')
plt.show()
```

• Votes vs Gross Scatter Plot

This plot underscores the correlation between public engagement (votes) and commercial success (gross). It reveals that movies receiving more votes generally accumulate higher revenue. However, exceptions—like well-rated independent films with limited box office reach—are evident, offering rich insights into alternative success metrics.

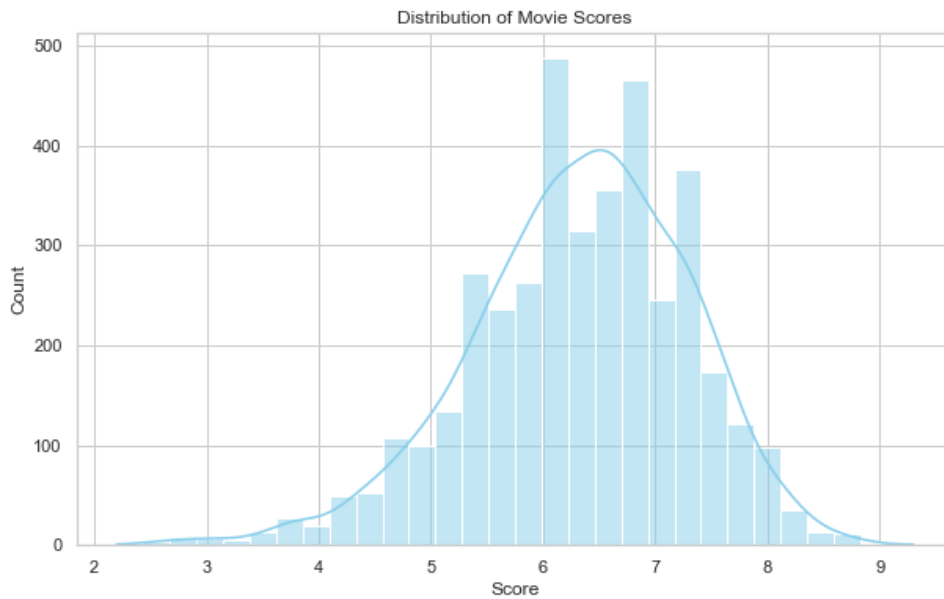


Figure 4.5: score-distribution

A closer look at the plot shows data density increasing with gross, particularly for movies crossing certain vote thresholds (e.g., 100,000 votes). This underlines a non-linear influence of votes on success and supports the case for polynomial regression or other non-linear models for future exploration.

```
sns.scatterplot(data=df, x='votes', y='gross', hue='rating')
plt.title('Votes vs Gross Earnings')
plt.xlabel('Votes')
plt.ylabel('Gross')
plt.savefig('votes_vs_gross.png')
plt.show()
```

- Budget vs Gross Scatter Plot

Analyzing the link between budget and revenue, this visualization confirms that higher investment often leads to greater returns. Still, notable exceptions such as highly successful low-budget films or high-budget flops challenge this trend, encouraging deeper analysis into qualitative success factors.

By assessing rating hues alongside, one can assess whether specific content certifications yield better returns at similar investment levels. For instance, 'PG-13' rated mid-budget films show consistent performance, while some 'R' rated big-budget films vary widely in outcome. This multi-dimensional insight elevates financial forecasting in cinema.

```
sns.scatterplot(data=df, x='budget', y='gross', hue='rating')
```

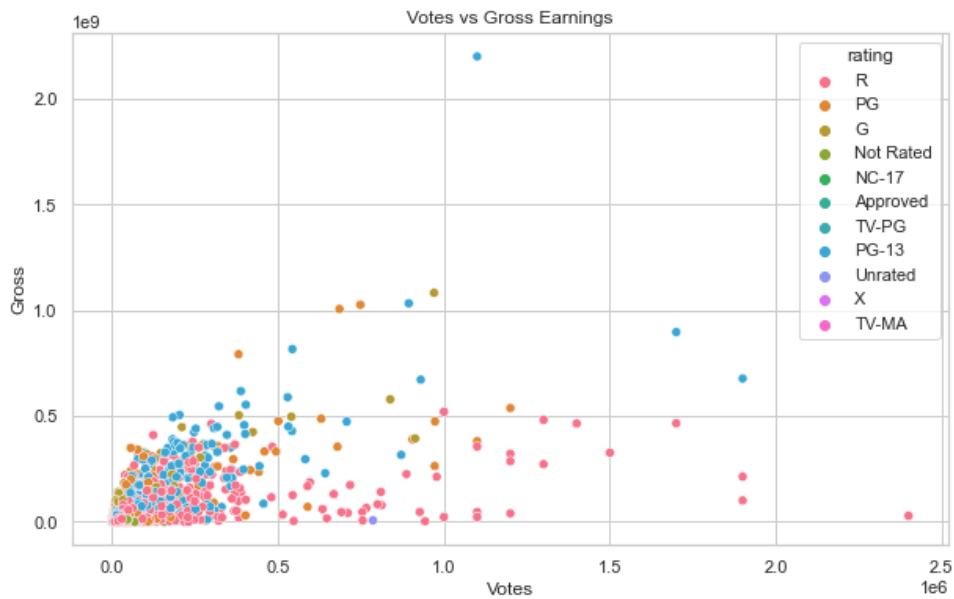


Figure 4.6: Comparison of Votes vs. Gross Earnings

```
plt.title('Budget vs Gross Earnings')
plt.xlabel('Budget')
plt.ylabel('Gross')
plt.savefig('budget_vs_gross.png')
plt.show()
```

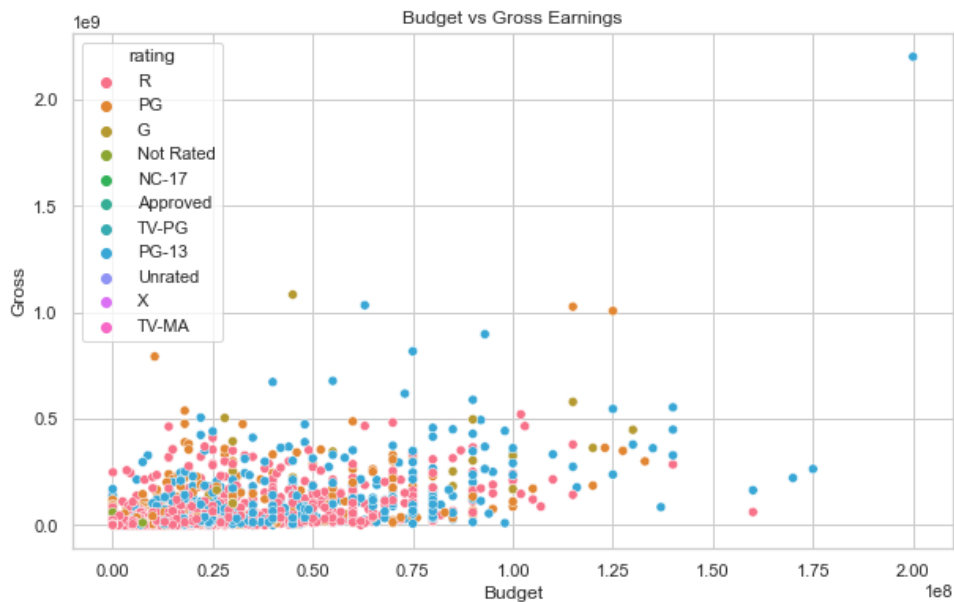


Figure 4.7: Comparison of Budget vs. Gross Earnings

- Correlation Heatmap

A visually informative correlation heatmap highlights relationships among budget, gross, votes, and scores. The strongest correlations appear between budget and

gross, followed by votes and gross. These findings validate the inclusion of these features in both classification and regression models.

The heatmap's grid format allows for intuitive comprehension of multi-variate relationships. Stronger color intensities between budget and gross validate business assumptions. Meanwhile, moderate correlations between score and gross hint at reputation's impact on earnings. This tool acts as both confirmation and exploration of analytical direction.

```
corr = df[['budget', 'gross', 'votes', 'score']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.savefig('correlation_heatmap.png')
plt.show()
```

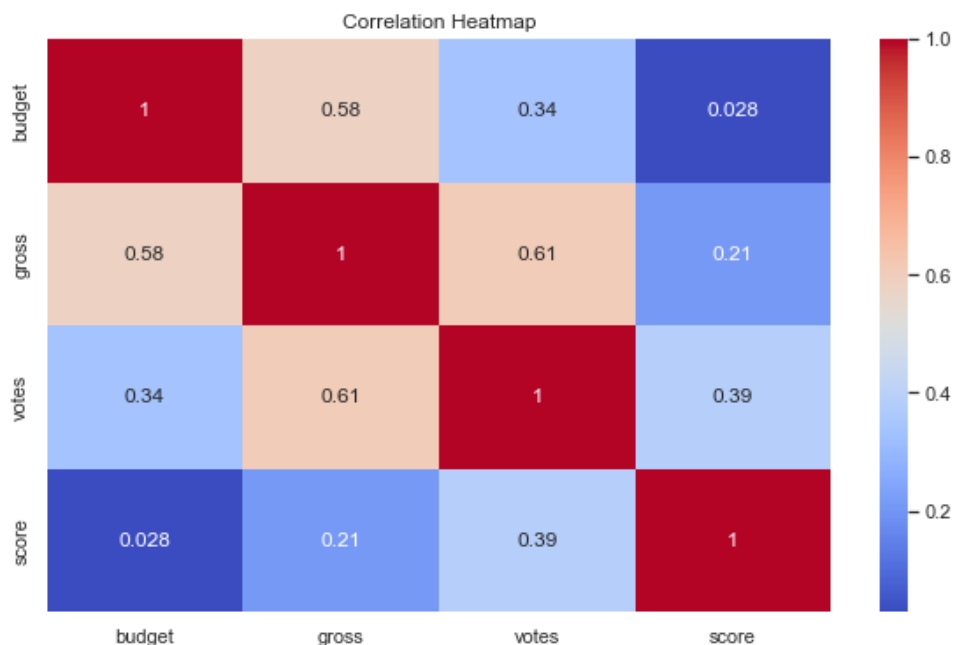


Figure 4.8: Correaltion Heatmap

- Bar Plot of Movies by Rating Category

```
\begin{verbatim}
corr = df[['budget', 'gross', 'votes', 'score']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.savefig('correlation_heatmap.png')
plt.show()
```

The provided Python code snippet leverages the `seaborn` library to generate a count plot that visualizes the distribution of movies across various rating categories stored in a DataFrame named `df`. The `sns.countplot()` function is utilized with the `x='rating'` parameter, where the x-axis represents different rating categories (such as G, PG, PG-13, R, etc.) present in the dataset. The plot is styled with an appealing orange color (`#ff7f0e`) for the bars and a black edge (`edgecolor='black'`) to enhance clarity and contrast, reflecting a polished and professional design. This visualization is a key component of exploratory data analysis (EDA), offering a straightforward way to assess the frequency of each rating, which can reveal imbalances or dominant categories within the movie dataset. The title "Distribution of Movies by Rating Category" is added with a font size of 14 and a padding of 10, providing a clear and descriptive header that aligns with the plot's purpose. Further enhancements are implemented to optimize the plot's readability and presentation. The `plt.xlabel()` and `plt.ylabel()` functions label the x-axis as "Rating" and the y-axis as "Count" with a font size of 12, ensuring the axes are easily interpretable. The `plt.xticks(rotation=45, ha='right')` command rotates the x-axis labels by 45 degrees and aligns them to the right, which is crucial for preventing overlap when dealing with numerous rating categories. This code effectively combines aesthetic appeal with analytical utility, making it an excellent tool for summarizing categorical data in a movie analysis project.

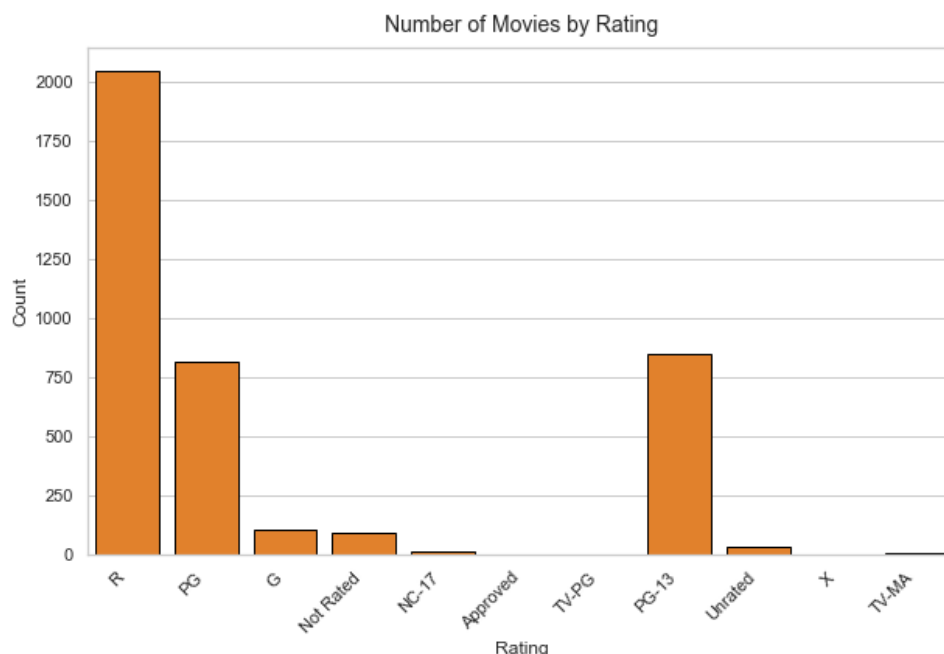


Figure 4.9: Distribution of Movies by Rating Category

This plot maps the number of audience votes against a movie's gross earnings, with points colored by rating. The scatter plot reveals a generally positive relationship:

higher numbers of votes often correlate with higher gross earnings. However, this relationship is not perfectly linear. Certain highly voted movies may still underperform in terms of revenue, perhaps due to niche appeal or digital releases. The color-coded ratings help distinguish whether certain rating types are more prevalent in high-performing films.

Additionally, this scatter plot aids in visually understanding rating-based clusters. For instance, 'R' rated films tend to dominate the higher vote-gross regions, suggesting mainstream success. Meanwhile, ratings like 'G' or 'PG' might show moderate earnings and votes, likely attributed to their target audience. The visualization thus not only confirms expected economic relationships but adds dimensional clarity through the lens of audience demographics.

Chapter 5

Conclusion

This project demonstrated the value of data-driven methods in analyzing trends and predicting outcomes in the movie industry. Through the application of Exploratory Data Analysis (EDA), classification, and regression techniques, we explored and uncovered meaningful insights from the dataset. The classification model enabled us to categorize movies by their ratings using features such as budget, votes, and scores. Although the classification model yielded moderate accuracy, it highlighted strong patterns, especially among frequently occurring ratings.

On the other hand, the regression analysis aimed to predict gross earnings based on similar features. The model showed a reasonable predictive capability, with budget and votes proving to be strong indicators of financial success. The heatmap and scatter plots helped visualize the relationships among variables, reinforcing the relevance of selected predictors in modeling.

The study illustrates the strength of combining statistical tools with machine learning to derive business intelligence. Although some limitations remain—such as data sparsity and model simplicity—this work serves as a solid foundation for more advanced analysis. With the inclusion of richer data and more sophisticated models, the predictive accuracy and insight generation can be significantly enhanced.

Chapter 6

Future Scope

The project opens multiple avenues for future exploration:

- Integrate unstructured data like movie reviews, descriptions, and social media sentiment using Natural Language Processing (NLP) to enrich feature sets.
- Incorporate additional variables such as genre, actor/actress influence, director reputation, and seasonal release patterns to improve predictive performance.
- Implement advanced machine learning techniques such as ensemble models, boosting algorithms, or deep learning for better accuracy and generalization.
- Develop interactive dashboards for stakeholders using visualization libraries or business intelligence tools to monitor and explore real-time movie performance predictions.
- Explore time-series analysis for understanding revenue trends over release periods or across genres.

These enhancements can significantly improve the practical applications of movie analytics in decision-making, marketing strategies, and financial planning.

Code

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load Data
df = pd.read_csv(r"C:\Users\adity\Dropbox\PC\Documents\MTECH\SEM2\Python For Data Science\Project CA2\new_project\movies_updated.csv")
```

Figure 6.1: Code1

```
# Data Cleaning
df.columns = df.columns.str.strip().str.replace(' ', '')
df['runtime'] = pd.to_numeric(df['runtime'], errors='coerce')
df['rating'] = df['rating'].fillna('Not Rated')
df['writer'] = df['writer'].fillna('Unknown')
df['star'] = df['star'].fillna('Unknown')
df['company'] = df['company'].fillna('Unknown')
df['gross'] = df['gross'].fillna(df['gross'].median())
```

Figure 6.2: Code2

```
# Set styles
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (10, 6)

# EDA Visualizations
sns.histplot(df['score'], kde=True, bins=30, color='skyblue')
plt.title('Distribution of Movie Scores')
plt.xlabel('Score')
plt.savefig('score_distribution.png')
plt.clf()

sns.scatterplot(data=df, x='votes', y='gross', hue='rating')
plt.title('Votes vs Gross Earnings')
plt.xlabel('Votes')
plt.ylabel('Gross')
plt.savefig('votes_vs_gross.png')
plt.clf()
```

Figure 6.3: Code3

```

sns.scatterplot(data=df, x='votes', y='gross', hue='rating')
plt.title('Votes vs Gross Earnings')
plt.xlabel('Votes')
plt.ylabel('Gross')
plt.savefig('votes_vs_gross.png')
plt.clf()

sns.scatterplot(data=df, x='budget', y='gross', hue='rating')
plt.title('Budget vs Gross Earnings')
plt.xlabel('Budget')
plt.ylabel('Gross')
plt.savefig('budget_vs_gross.png')
plt.clf()

```

Figure 6.4: Code4

```

sns.scatterplot(data=df, x='budget', y='gross', hue='rating')
plt.title('Budget vs Gross Earnings')
plt.xlabel('Budget')
plt.ylabel('Gross')
plt.savefig('budget_vs_gross.png')
plt.clf()

corr = df[['budget', 'gross', 'votes', 'score']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.savefig('correlation_heatmap.png')
plt.clf()

```

Figure 6.5: Code5

```

# Classification Model
X = df[['budget', 'gross', 'votes', 'score']]
y = df['rating']
le = LabelEncoder()
y_encoded = le.fit_transform(y)

X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_c, y_train_c)
y_pred_c = clf.predict(X_test_c)

acc_class = accuracy_score(y_test_c, y_pred_c)
report_class = classification_report(y_test_c, y_pred_c, target_names=le.classes_)
print(f"Classification Accuracy: {acc_class:.4f}")
print("Classification Report:\n", report_class)

```

Figure 6.6: Code6

```

# Regression Model
X_train_r, X_test_r, y_train_r, y_test_r = train_test_split(X, df['gross'], test_size=0.2, random_state=42)
reg = LinearRegression()
reg.fit(X_train_r, y_train_r)
y_pred_r = reg.predict(X_test_r)

mse = mean_squared_error(y_test_r, y_pred_r)
r2 = r2_score(y_test_r, y_pred_r)
print(f"Regression MSE: {mse:.2f}")
print(f"Regression R² Score: {r2:.4f}")

sns.scatterplot(x=y_test_r, y=y_pred_r, alpha=0.5)
plt.xlabel('Actual Gross')
plt.ylabel('Predicted Gross')
plt.title('Actual vs Predicted Gross (Regression)')
plt.savefig('actual_vs_predicted_gross.png')

```

Figure 6.7: Code7

Bibliography

- [1] Movies Updated Dataset, *movies_updated.csv*
- [2] Python Software Foundation, *Python 3.x Documentation*, Available at: <https://docs.python.org/3/>
- [3] Pedregosa, F., Varoquaux, G., Gramfort, A., et al., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, 12, pp. 2825-2830, 2011. Available at: <https://scikit-learn.org/>
- [4] Hunter, J. D., *Matplotlib: A 2D Graphics Environment*, Computing in Science & Engineering, 9(3), pp. 90-95, 2007.
- [5] Waskom, M. et al., *Seaborn: Statistical Data Visualization*, Available at: <https://seaborn.pydata.org/>
- [6] McKinney, W., *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, pp. 51–56, 2010.
- [7] Kaggle, *Datasets and Forums for Data Science Projects*, Available at: <https://www.kaggle.com/>
- [8] IMDb, *Official Website for Movie Information*, Available at: <https://www.imdb.com/>