Understanding Movie Popularity: A Machine Learning Perspective

॥वसुधैव कुटुम्बकम्॥

THESIS SUBMITTED TO

Symbiosis Institute of Geoinformatics

FOR PARTIAL FULFILLMENT OF THE M. Sc. DEGREE

By

Sharon Furtado

(Batch 2023-2025 / PRN 23070243015)

Symbiosis Institute of Geoinformatics

Symbiosis International (Deemed University)

5th Floor, Atur Centre, Gokhale Cross Road,

Model Colony, Pune – 411016

CERTIFICATE

Certified that this thesis titled **"Understanding Movie Popularity: A Machine Learning Perspective"** is a bonafide work done by Ms. Sharon Furtado (23070243015) at Symbiosis Institute of Geoinformatics, under our supervision.

Supervisor, Internal

Name: Mr. Sahil Shah

Symbiosis Institute of Geoinformatics

UNDERTAKING

The thesis titled **"Understanding Movie Popularity: A Machine Learning Perspective"** is a bona fide work by Ms. Sharon Furtado. The University can use and reuse the dissertation work in future.

Date: 13th September 2024

Name of the student: Sharon Furtado

INDEX

ACKNOWLEDGEMENT

I want to express my deepest gratitude to everyone who contributed to the successful completion of this project. First and foremost, I would like to thank my project guide, Mr. Sahil Shah, for their invaluable guidance, continuous encouragement, and insightful feedback throughout this research. Their expertise and experience have been instrumental in shaping the direction and quality of this work.

I am also grateful to the faculty and staff of Symbiosis Institute of Geoinformatics for providing the necessary resources and a conducive environment for research. Special thanks go to my colleagues and friends who provided support, inspiration, and constructive criticism, helping me refine my ideas and approach.

I want to thank my family for their unwavering support and understanding during this project. Their patience and encouragement gave me the strength to persevere through challenges.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	ABBREVIATIONS	FULL FORM
01	R.F.	Random Forest
02	ML	Machine Learning
03	SVR	Support Vector Regressor
04	MLR	Multilinear Regression
05	L.R.	Logistic Regression
06	N.B.	Naive Bayes
07	CNN	Convolutional Neural Networks
08	SVM	Support Vector Machine
09	E.D.A.	Exploratory Data Analysis
10	IQR	Interquartile Range
11	T.P.	True Positives
12	F.P.	False Negatives
13	T.N.	True Negatives
14	M.A.E.	Mean Absolute Error
15	R.M.S.E.	Root Mean Square Error
16	ANN	Artificial Neural Network
17	DL	Deep Learning
18	K.N.N.	K-Nearest Neighbors
19	AdaBoost	Adaptive Boosting
20	XGBoost	Extreme Gradient Boosting
21	CSV	Comma-Separated Values
22	M.L.R	Multinomial Logistic Regression
23	B.A.G	Bagging
24	D.T	Decision Tree
25	A.T.B	Adaptive Tree Boosting
26	A.E.N.N	All K-Edited Nearest Neighbors
27	Pandas	Python Data Analysis Library
28	NumPy	Numerical Python
29	SciPy	Scientific Python
30	sklearn	Scikit-learn

PREFACE

This thesis is submitted in partial fulfilment of the requirements for the M.Sc. Data Science & Spatial Analytics at Symbiosis International University. The project presented herein, "Understanding Movie Popularity: A Machine Learning Perspective," marks the culmination of two months of research, experimentation, and application of the principles of data science to a contemporary and dynamic field: movie popularity prediction.

In the age of digital entertainment, understanding the factors that drive movie success is critical for stakeholders such as producers, distributors, and streaming platforms. This project leverages machine learning techniques to analyse various data sources and predict the popularity of movies based on a range of features. Through this exploration, I aimed to apply data analysis and machine learning models to identify patterns, trends, and critical factors influencing movie popularity.

I want to express my sincere gratitude to my faculty advisor, mentors, and colleagues for their support and guidance throughout this project. The insights gained from this study have deepened my understanding of data science and machine learning and provided valuable hands-on experience in real-world problem-solving.

I hope this work offers meaningful contributions to the field of data science and opens up further avenues of research into predictive modelling and entertainment analytics.

INTRODUCTION

In the age of digital entertainment, the film industry has become a global powerhouse, contributing significantly to cultural and economic sectors. Understanding and predicting movie success has become a vital area of study, aiding filmmakers, studios, and investors in making informed decisions during the production and distribution phases. With advancements in data science and machine learning, researchers have increasingly applied predictive analytics to forecast movie success based on various factors such as genre, cast, director, budget, awards, and critical reception. This report explores the application of machine learning techniques to predict movie popularity, utilising various datasets and features that contribute to a film's success.

The concept of using machine learning for movie success prediction is not novel. Shahid and Islam (2023) highlighted the importance of genre popularity in predicting box office success by utilising time series forecasting models such as S.A.R.I.M.A.X., achieving an accuracy of 92.4% for multi-class classification tasks, a significant improvement over earlier methods. Their study emphasised how understanding genre popularity over time can assist filmmakers in making genre-specific decisions during pre-production **(Shahid & Islam, 2023).**

Masih and Ihsan (2019) investigated whether winning awards, such as the Zee Cine or I.I.F.A. awards, has a meaningful impact on the success of a Bollywood movie. They found that while awards had a minor impact on prediction accuracy, other features like the director, genre, and budget played a more prominent role in predicting box office revenue. Combining these factors with awards data improved the quality of predictions, further enhancing the potential of machine learning models in this domain **(Masih & Ihsan, 2019).**

Verma and Verma (2019) conducted a comparative analysis of machine learning algorithms to predict the success of a Bollywood movie. They found that Random Forest and Logistic Regression models performed the best, achieving an accuracy of 92%. Their research identified vital features such as music ratings, IMDb ratings, and the number of screens on release as the most influential predictors of a movie's box office success **(Verma & Verma, 2019).**

Focusing on profitability rather than gross revenues, Donega e Souza, Nishijima, and Pires (2023) applied Random Forest, Support Vector Machines, and Neural Networks to a dataset of movies released between 1980 and 2019. Their work achieved an impressive accuracy of 96.7% and highlighted the utility of machine learning classifiers in determining the economic success of films **(Donega e Souza et al., 2023).**

In a different approach, Sahu et al. (2022) proposed a content-based recommendation system that predicted movie popularity based on various features, utilising Convolutional Neural Networks (CNN). Their model achieved a high accuracy of 96.8%, showcasing the potential of deep learning models in understanding movie success in its early stages **(Sahu et al., 2022).**

Other studies, such as those by Bristi, Zaman, and Sultana (2019), explored machine learning's effectiveness in predicting IMDb ratings, utilising techniques like SMOTE to balance the dataset. Their study found that Random Forests achieved the highest accuracy, especially after applying data-balancing techniques (Bristi et al., 2019). Similarly, Oyewola and Dada (2022) explored the effectiveness of various machine learning techniques, with bagging models showing the highest accuracy at 99.56%, outperforming other classification methods like Support Vector Machines and Naive Bayes **(Oyewola & Dada, 2022).**

Finally, Agarwal et al. (2022) compared machine learning approaches to predict movie success, including regression models, clustering techniques, and neural networks. Their study identified key predictors influencing movie popularity, with the artificial neural network model achieving an accuracy of 86% (Agarwal et al., 2022). Anand (2023) further analysed the impact of machine learning on IMDb ratings prediction, showing that Random Forest was the most effective model, achieving a prediction accuracy of 74% **(Anand, 2023).**

Given these prior studies, it is clear that machine learning techniques have shown tremendous potential in predicting various facets of movie success, from box office performance to profitability and audience reception. This project seeks to expand upon these efforts by employing advanced machine-learning models and analysing various features contributing to movie popularity. Building upon the foundation set by previous research, this study aims to provide a comprehensive understanding of movie success through a machine-learning perspective.

RESEARCH OBJECTIVES

1. **Investigate the key factors influencing movie popularity:** Analyse genre, duration, budget, vote count, and vote average to identify significant predictors of movie success.
2. **Apply and evaluate machine learning models for predicting movie popularity:** Use and compare machine learning techniques like Random Forest, SVM, and Neural Networks to predict movie popularity based on selected features.
3. **Assess the impact of specific movie characteristics on prediction accuracy:** Evaluate the individual and combined effects of variables like genre, budget, and vote count on the predictive accuracy of machine learning models.
4. **Develop a machine learning model with high predictive accuracy for movie popularity:**
   Optimise models to achieve high accuracy using data from T.M.D.B., focusing on critical movie features.
5. **Explore the applicability of prediction models for future movie releases:** Test the model's ability to generalise and predict the popularity of upcoming movies using pre-release data.

Research Questions

- What are the most significant features influencing movie popularity, particularly genre, duration, budget, vote count, and vote average?
- Which machine learning models provide the best performance in predicting movie popularity?
- How do specific variables, such as genre and budget, affect the accuracy of movie popularity prediction models?
- Can machine learning models effectively predict the popularity of movies before their release using data from T.M.D.B.?
- How can movie success prediction be improved by combining multiple movie-related features?

LITERATURE REVIEW

The application of machine learning to predict movie popularity and success has gained significant attention in recent years. Shahid and Islam (2023) introduced time-series forecasting techniques, specifically S.A.R.I.M.A.X., to predict genre popularity and box office success. Their model achieved 92.4% accuracy, significantly improving the prediction of multi-class classification tasks. This study demonstrated the potential of leveraging genre popularity trends to enhance box office predictions during pre-production **(Shahid & Islam, 2023).** Similarly, Masih and Ihsan (2019) examined the influence of awards, such as Zee Cine and I.I.F.A., on Bollywood movies using a variety of machine learning algorithms. While awards had a minor effect on predictive accuracy, factors like the director, genre, and budget were found to be more critical for box office success, and combining these factors with award information improved prediction performance **(Masih & Ihsan, 2019).**

Verma and Verma (2019) conducted a comparative analysis of six machine learning algorithms to predict the success of a Bollywood movie. Their results showed that Random Forest and Logistic Regression models performed the best, achieving 92% accuracy, with features such as IMDb ratings, music ratings, and the number of screens for release emerging as significant predictors **(Verma & Verma, 2019).** Focusing on the profitability of movies, Donega e Souza, Nishijima, and Pires (2023) applied models like Random Forests, Support Vector Machines, and Neural Networks. Their study, which analysed movies released between 1980 and 2019, achieved 96.7% accuracy, highlighting the effectiveness of these models for profit-based classification rather than gross revenue prediction **(Donega e Souza et al., 2023).**

Further research has explored different machine-learning techniques for movie success prediction. Sahu et al. (2022) proposed a content-based recommendation system that used features like genre and cast alongside a Convolutional Neural Network (CNN) model to predict movie popularity. Their model achieved a high accuracy of 96.8%, showcasing the promise of deep learning approaches in early-stage popularity predictions **(Sahu et al., 2022).** Similarly, Bristi, Zaman, and Sultana (2019) utilised Random Forests and data balancing techniques such as SMOTE to improve IMDb rating prediction. Their study highlighted that applying SMOTE improved the model's accuracy **(Bristi et al., 2019).**

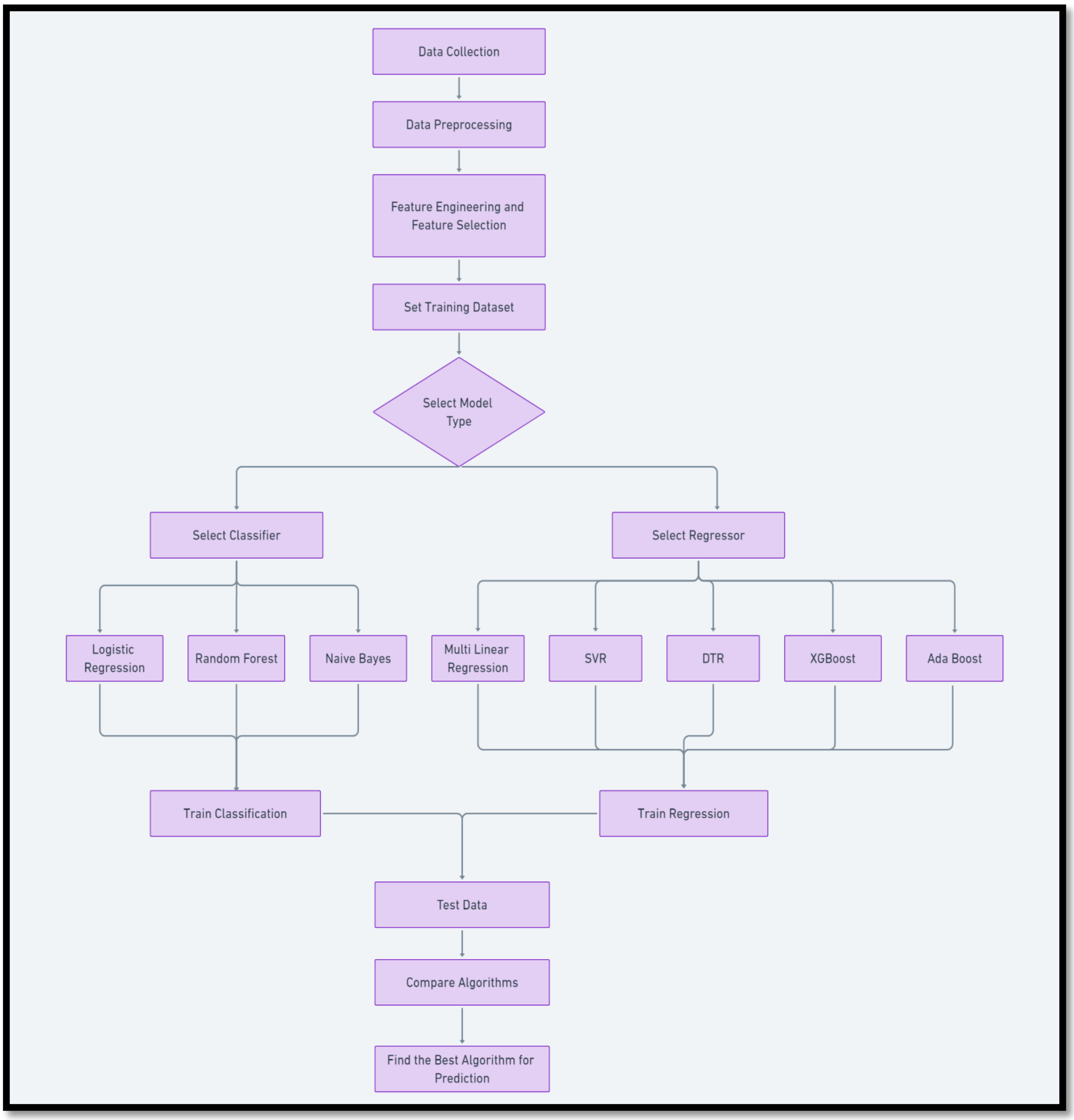
In their exploration of various machine learning techniques, Oyewola and Dada (2022) found that bagging models outperformed other classifiers, including Support Vector Machines and Naive Bayes, achieving a 99.56% accuracy in predicting movie popularity. This finding underscores the effectiveness of ensemble methods in the movie prediction domain **(Oyewola & Dada, 2022).** Agarwal et al. (2022) applied various techniques to predict movie success, including regression, clustering, and neural networks. Their artificial neural network model achieved an accuracy of 86%, demonstrating the model's potential to predict movie success based on features such as IMDb ratings and budget **(Agarwal et al., 2022).** Anand (2023) further supported these findings, revealing that Random Forests performed the best in predicting IMDb ratings, achieving 74% accuracy, compared to other models like Decision Trees and Logistic Regression **(Anand, 2023).**

The collective research shows that machine learning techniques, notably ensemble and deep learning models, have demonstrated considerable potential in predicting movie success. Variables such as genre, director, budget, and IMDb ratings have been identified as significant factors contributing to a film's success, while combining features like awards and genre further enhances predictive accuracy. These studies form the foundation for further research into the factors influencing movie popularity and optimising machine learning models for more accurate predictions.

S. No	References	Objective	Technique	Dataset	Results
01	Shahid & Islam, 2023	Investigate time series-based genre popularity features for predicting box office success in the pre-production phase.	Proposed novel genre popularity features through time series forecasting (S.A.R.I.M.A.X.) and evaluated using multiple machine learning classifiers (including Gradient Boosting).	45,466 movies from 1900-2018, weekly revenue data.	Achieved 92.4% accuracy, 35.7% improvement over state-of-the-art methods for multi-class classification.
02	Masih & Ihsan, 2019	Investigate whether awards (Zee Cine and I.I.F.A.) won by actors influence the success of Bollywood movies and utilise machine learning algorithms to predict movie revenue.	Data collection and preprocessing from sources like Wikipedia, IMDB, and BoxOfficeIndia; application of various machine learning algorithms (Decision Tree, Random Forest, Neural Networks, Naïve Bayes, Bayesian Networks) for classification.	Dataset of 522 Bollywood movies from 2013 to 2017, including movie details (genre, cast, director, awards, budget, revenue).	While academy awards slightly increase predictive accuracy, factors like director, genre, and budget significantly influence box office success. Combining awards with these factors enhances prediction quality.
03	Verma, H. & Verma, G. (2019)	To predict the success of Bollywood movies using machine learning techniques.	Comparative analysis of six machine learning algorithms: Decision Tree (D.T.), Random Forest (R.F.), Support Vector Machine (SVM), Logistic Regression (L.R.), Adaptive Tree Boosting (A.T.B.), and Artificial Neural Network (ANN).	Sample of 200 Bollywood movies.	Models achieved 80% to 92%, with R.F. and L.R. models achieving 92% accuracy. Significant predictors identified were music rating, IMDb rating, and number of screens for release.
04	Donega e Souza, Nishijima, Pires (2023)	Predict movie economic success by applying machine learning classifiers while focusing on profits rather	Random Forest, Support Vector Machine, Neural Network	3167 movies released between 1980 and 2019 with budget data.	Achieved 96.7% accuracy and 97% F1-score in binary classification; approximately 90% Average Percent Hit Rate for profit ranges.

		than gross revenues.			
05	Sahu et al. (2022)	To predict movie popularity and its target audience at the early production stage.	She proposed a content-based movie recommendation system using features like genre, cast, director, etc., along with a CNN model for prediction.	TMDb and IMDb datasets, which include movie features and ratings.	Achieved 96.8% accuracy in predicting movie popularity.
06	Bristi, W. R., Zaman, Z., & Sultana, N. (2019)	To predict IMDb ratings of movies based on various features.	Utilised machine learning classification algorithms, applied data preprocessing, and SMOTE for balancing.	The dataset comprised 250 Hollywood movies from 2018, collected from Wikipedia and IMDb.	Random Forest achieved the highest accuracy among classifiers, significantly improving performance after applying SMOTE.
07	Agarwal et al., 2022	To predict movie success and compare various statistical approaches	Utilised regression models, clustering techniques, time series analysis, and an artificial neural network	IMDb dataset for movies released in 2020	Achieved an accuracy of 86% with the artificial neural network; identified key features influencing movie success.
08	Oyewola & Dada (2022)	To compare the effectiveness of various machine learning techniques for predicting movie popularity.	Utilised Multinomial Logistic Regression (M.L.R.), Support Vector Machine (SVM), Bagging (B.A.G.), Naive Bayes (N.B.S.), K-Nearest Neighbor (K.N.N.) with noise filtering using All K-Edited Nearest Neighbors (A.E.N.N.).	The dataset consisted of 5043 rows and 14 columns, including features like director, duration, and IMDb scores.	B.A.G. performed the best, with an accuracy of 99.56%, while other methods showed lower effectiveness.
09	Anand, A. (2023)	To analyse the impact of machine learning algorithms on predicting movie success.	Utilised machine learning models such as Linear Regression, Decision Trees, and Random Forests for data analysis	Kaggle dataset containing information on 1000 IMDB movies.	Random Forest achieved a 74% accuracy in predicting IMDB ratings; Decision Trees had 71%, and Logistic Regression 53.56%.

METHODOLOGY



(Figure 1 Methodology)

Dataset Description:

The dataset used to predict movie popularity was collected from the T.M.D.B. website using an API key. It contains 5,000 entries and 16 variables describing various attributes of movies.

Below is a detailed description of each variable:

- **Title:** The title of the movie (e.g., "Bad Boys: Ride or Die").
- **Genre:** Genres of the movie listed in a comma-separated format (e.g., "Action, Crime, Thriller, and Comedy").
- **Release Date:** The movie's release date is in the format YYYY-MM-DD.
- **Duration**: The movie lasts minutes (e.g., 115).
- **Language**: The primary language of the movie is represented by language codes (e.g., "en" for English).
- **Country:** The countries where the movie was produced (e.g., "United States of America").
- **Director:** The director(s) of the movie (e.g., "Adil El Arbi, Bilall Fallah").
- **Overview:** A summary of the movie plot.
- **Popularity:** A numerical score indicating the movie's popularity on the T.M.D.B. platform.
- **Tagline:** A short promotional phrase or tagline associated with the movie.
- **Revenue:** The total revenue generated by the movie in USD.
- **Budget:** The movie's production budget is in USD.
- **Vote Average:** The average rating users give the movie on a scale of 1 to 10.
- **Vote Count:** The total number of votes received by the movie.
- **Production Companies:** The production companies involved in making the movie, listed in a comma-separated format.
- **Actors:** The main actors featured in the movie are listed in a comma-separated format.

Data Pre-processing:

Basic Checks:

1. **Dataset Size:** The dataset contains 2,000 rows and 17 columns.
2. **Data Types:**
   - **Object Columns:** 10 columns (e.g., Title, Genre, Language, Country, Director, Overview, Tagline, Production Companies, IMDB ID, Actors).
   - **Integer Columns**: 4 columns (e.g., Duration, Revenue, Budget, Vote Count).
   - **Float Columns:** 2 columns (Popularity, Vote Average).
   - **Datetime Columns:** 1 column (Release Date)

3. **Number of Duplicates:** The dataset has no duplicate rows.
4. **Missing values And Zero Values:** Missing values in the dataset were checked using the .isnull().sum() function, which calculates the total number of missing entries in each column. Similarly, zero values were identified using (df == 0).sum(), which counts zero occurrences in each column.
5. **Descriptive Statistics:** Summary statistics, including count, mean, minimum, maximum, standard deviation, and quartiles (25th, 50th, and 75th percentiles), were

computed for numerical features. For categorical features, frequency counts and unique values were assessed to understand the distribution and characteristics of the data.

Libraries used:

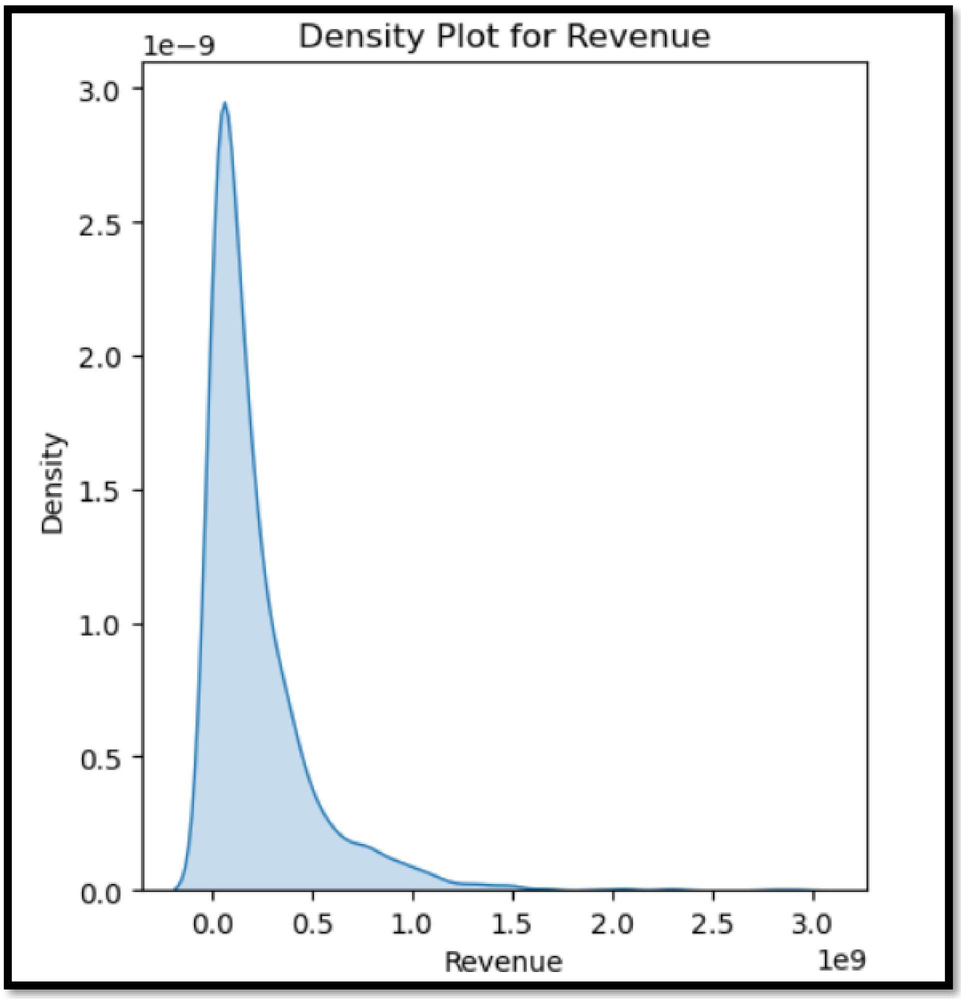
The analysis was conducted using several foundational libraries:

Sr. no	**Library name**	**Version**	**Description**
1	NumPy	1.24.3	They are utilised for numerical operations and handling large data arrays efficiently, providing support for mathematical functions and operations on arrays.
2	Pandas	2.0.3	Employed for data manipulation and analysis, offering data structures like Data Frames to clean, organise, and analyse data quickly.
3	Matplotlib	3.7.2	They are used for creating static, interactive, and animated visualisations in Python, including plots and charts to visualise data distribution and relationships.
4	Seaborn	0.12.2	We leveraged for statistical data visualisation, enhancing Matplotlib capabilities with built-in themes and functions to create informative and attractive graphics, such as count plots and bar charts.
5	Scikit-learn	1.5.1	Applied for implementing machine learning algorithms and pre-processing techniques, providing tools for model training, evaluation, and selection, as well as feature scaling and transformation.
6	SciPy	1.11.1	They are utilised for scientific and technical computing, building on NumPy to provide additional functionality for optimisation, integration, interpolation, eigenvalue problems, and other numerical tasks. SciPy offers a suite of algorithms and tools essential for advanced scientific computations and data analysis.

Exploratory Data Analysis:

E.D.A.'s project involves computing descriptive statistics to summarise numerical features and visualising categorical variables such as genre and language through count plots and bar charts. The analysis will also address missing values and outliers to ensure data quality. Additionally, E.D.A. will explore relationships between features and the popularity of the target variable and perform feature engineering tasks, including encoding and scaling. This comprehensive approach will facilitate a deeper insight into the data and support effective model development.

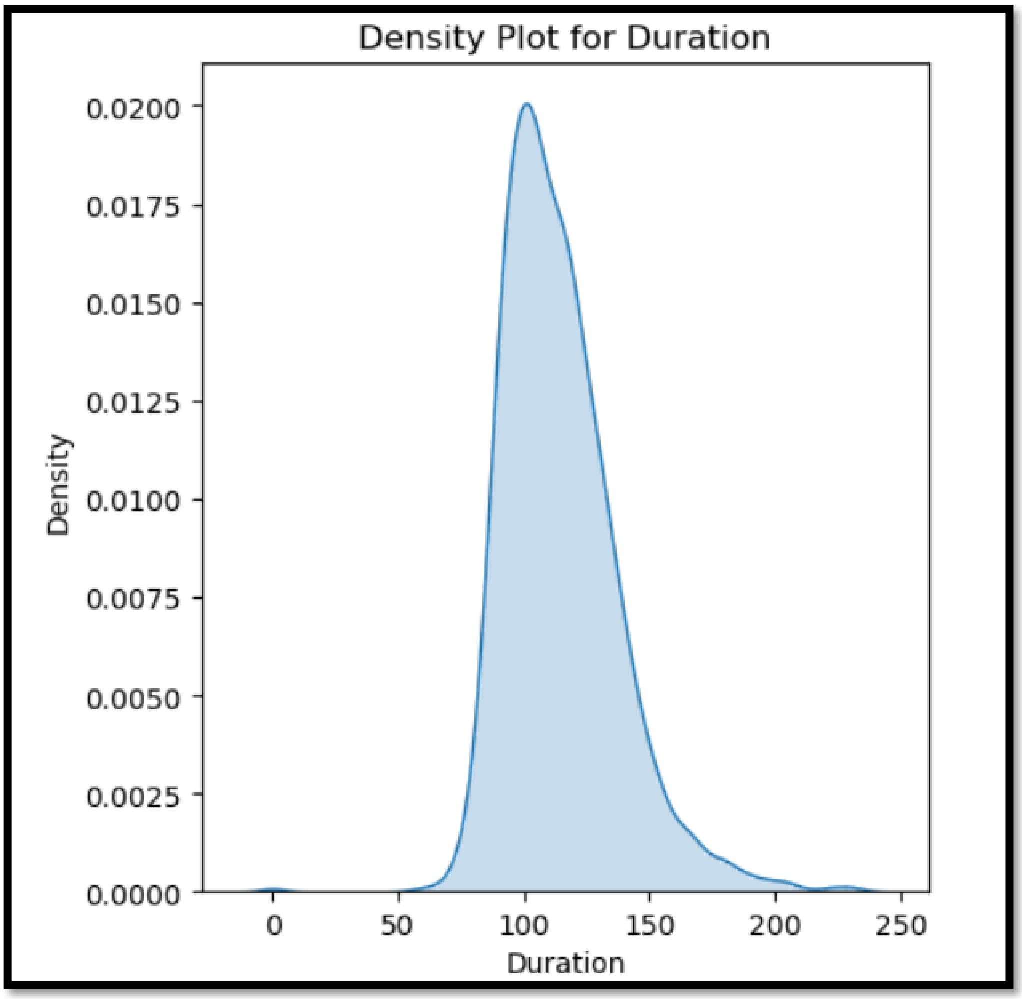
Univariate Analysis:



(Figure 2 *Density Plot for Revenue)*

Density Plot for Revenue:

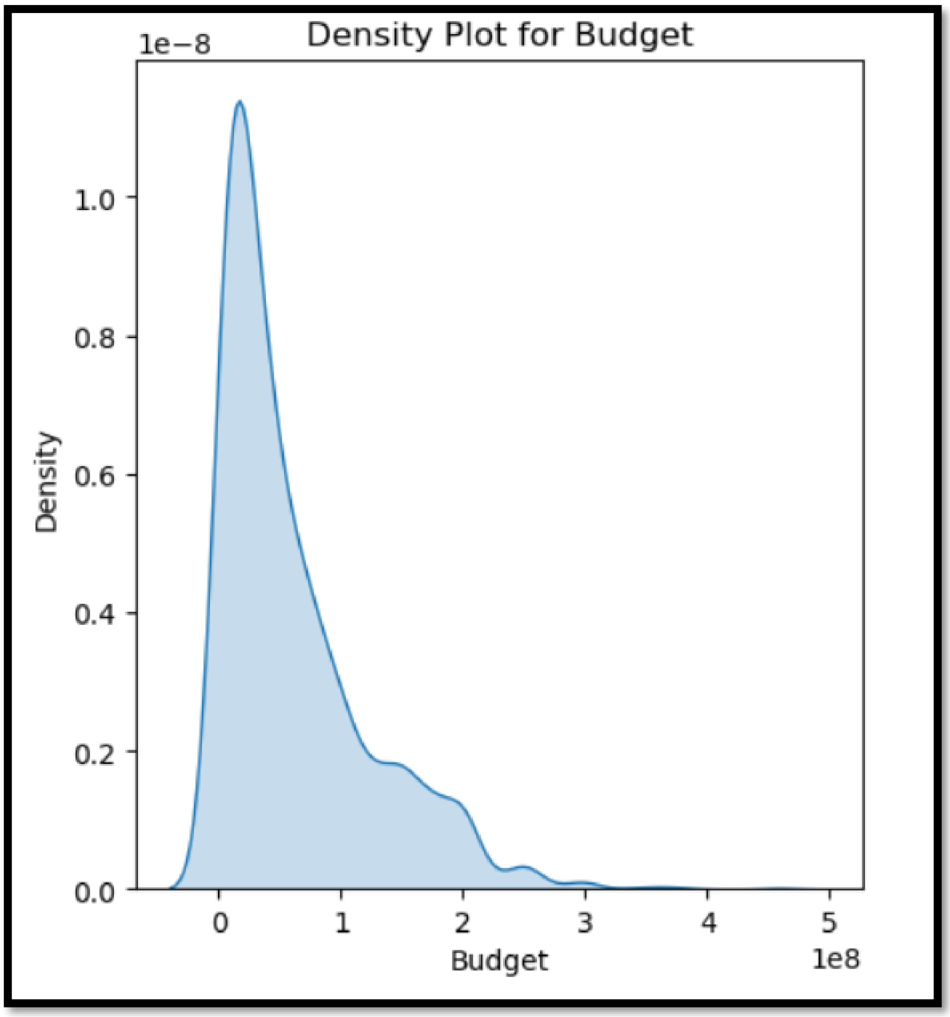
The density plot reveals a right-skewed distribution, indicating that most of the revenue values are on the lower end of the spectrum. In contrast, a smaller number of values are significantly higher. This suggests that a significant portion of the entities have low or no revenue. Additionally, the long tail extending to the right indicates the presence of a few high-revenue outliers. These outliers might disproportionately impact specific statistical measures, such as the mean, and could warrant further investigation to understand their underlying causes.



(Figure 3 *Density Plot for Duration)*

Density Plot of Duration:

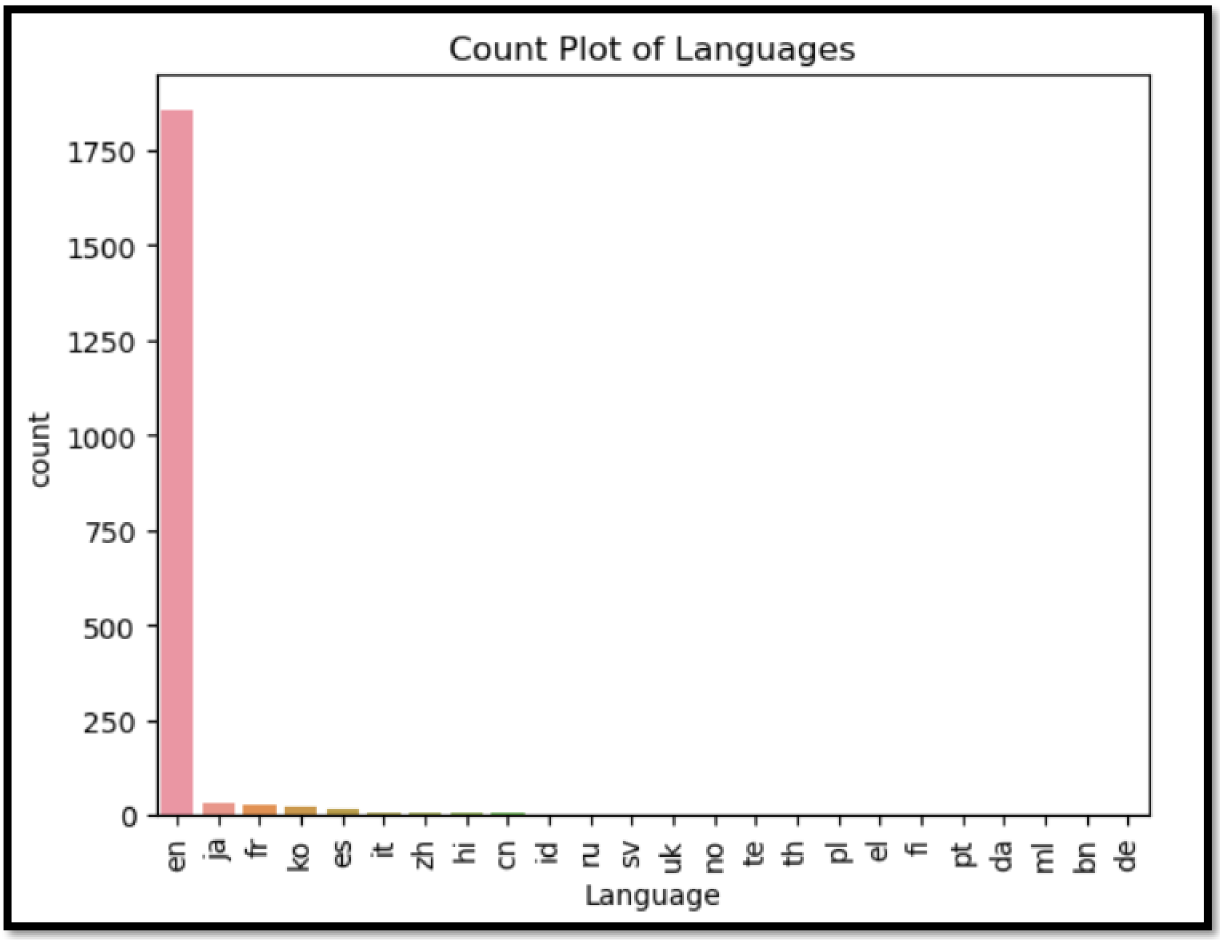
The duration has a right-skewed distribution, as shown by the density map, with the majority of duration values falling on the lower end of the range. On the other hand, fewer values are noticeably higher. This implies that a considerable proportion of the occurrences or undertakings have shorter durations. A few high-duration outliers are also shown by the lengthy tail that extends to the right. Certain statistical measurements may be significantly affected by these outliers.



(Figure 4 Density Plot for Budget)

Plot of Budget Density:

The budget distribution is right-skewed, as seen by the density map, with most budget values falling between two extremes and a smaller percentage being noticeably higher. This implies that low or nonexistent budgets are held by a sizable share of the entities. A few high-budget outliers are also indicated by the lengthy tail that extends to the right. These outliers may have an unequal influence on some statistical metrics, including the mean, and may require additional research to determine their underlying reasons.

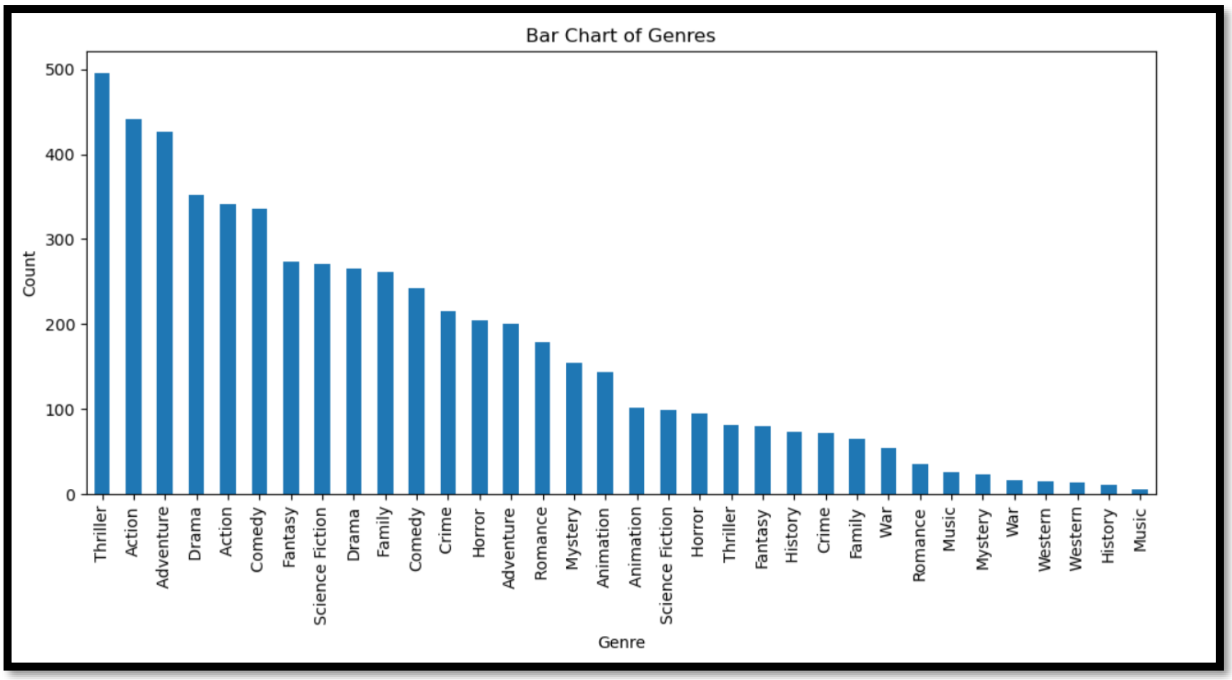


(Figure 5 *Count Plot for Languages)*

The count plot of languages illustrates the distribution of languages in the dataset for movie popularity prediction.

Key Insights:

- **Highly Imbalanced Distribution:**
  The language en (presumably representing English) dominates the dataset, with nearly 1,800+ instances. This indicates that the vast majority of movies in your dataset are in English.
  Other languages like ja (Japanese), fr (French), and ko (Korean) have significantly fewer occurrences, with the counts for each of these languages appearing below 50.
- **Minority Classes:**
  There are many languages with minimal counts (some with less than 10 instances), which could lead to difficulties for the model in learning meaningful patterns for those languages.
  This extreme class imbalance can hinder the model's ability to generalize for languages other than English, as it might predominantly learn from English-language movies, treating other languages as noise.



(Figure 6 *Bar Chart for Genres)*

The bar chart of movie genres illustrates the distribution of genres in the dataset for movie popularity prediction.

Key Insights:
Highly Imbalanced Distribution:

- The **Thriller** genre dominates the dataset with nearly 500 instances, indicating it is the most frequent genre among the movies.
- Other popular genres like **Action**, **Adventure**, **Drama**, and **Comedy** also appear frequently, each having counts between 200 and 400 instances.

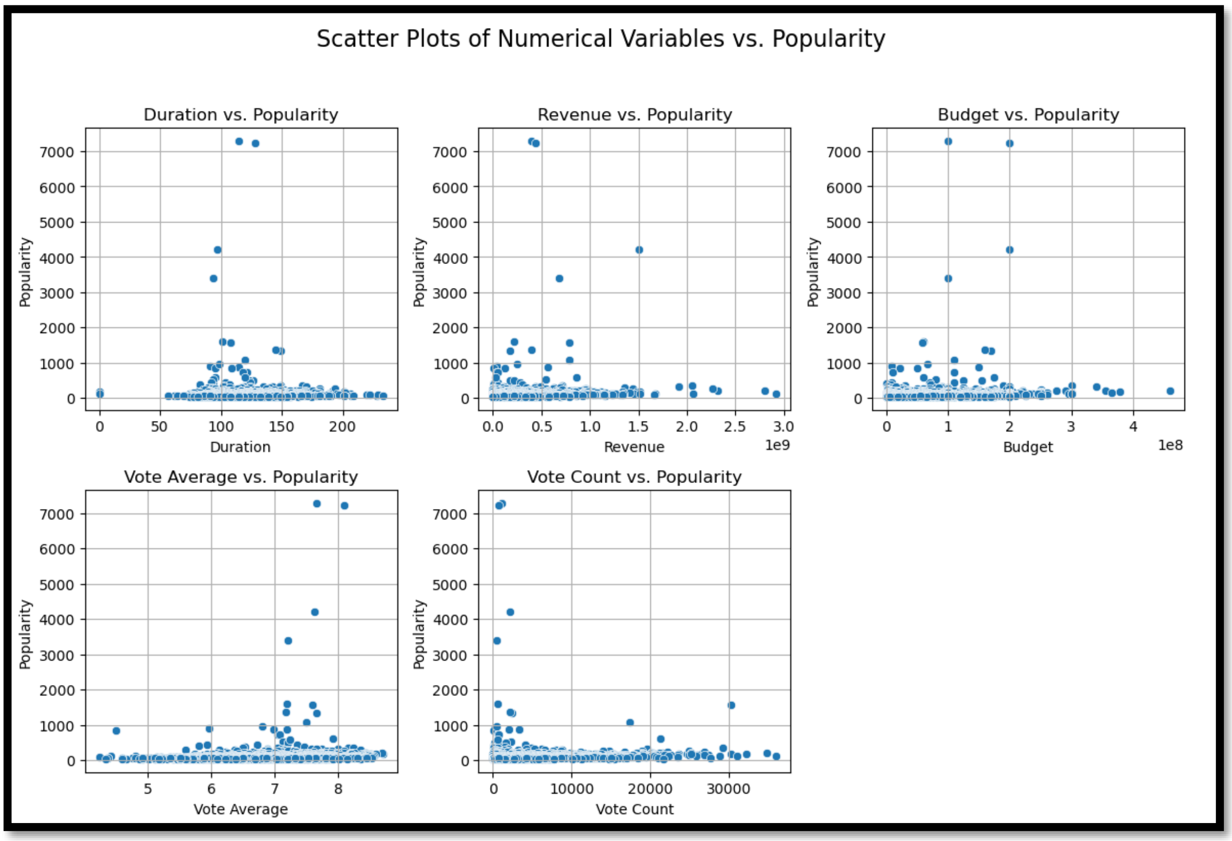
Genre Distribution:

- Some genres, such as Music, History, and Western, have very few occurrences, each appearing less than 50 times. These minority genres are underrepresented in the dataset.

Repeated Genres:

- Certain genres like **Thriller**, **Action**, and **Adventure** appear multiple times, possibly due to overlapping classifications or duplicate entries. This could introduce noise in the data and affect model performance.

Bivariate Analysis:

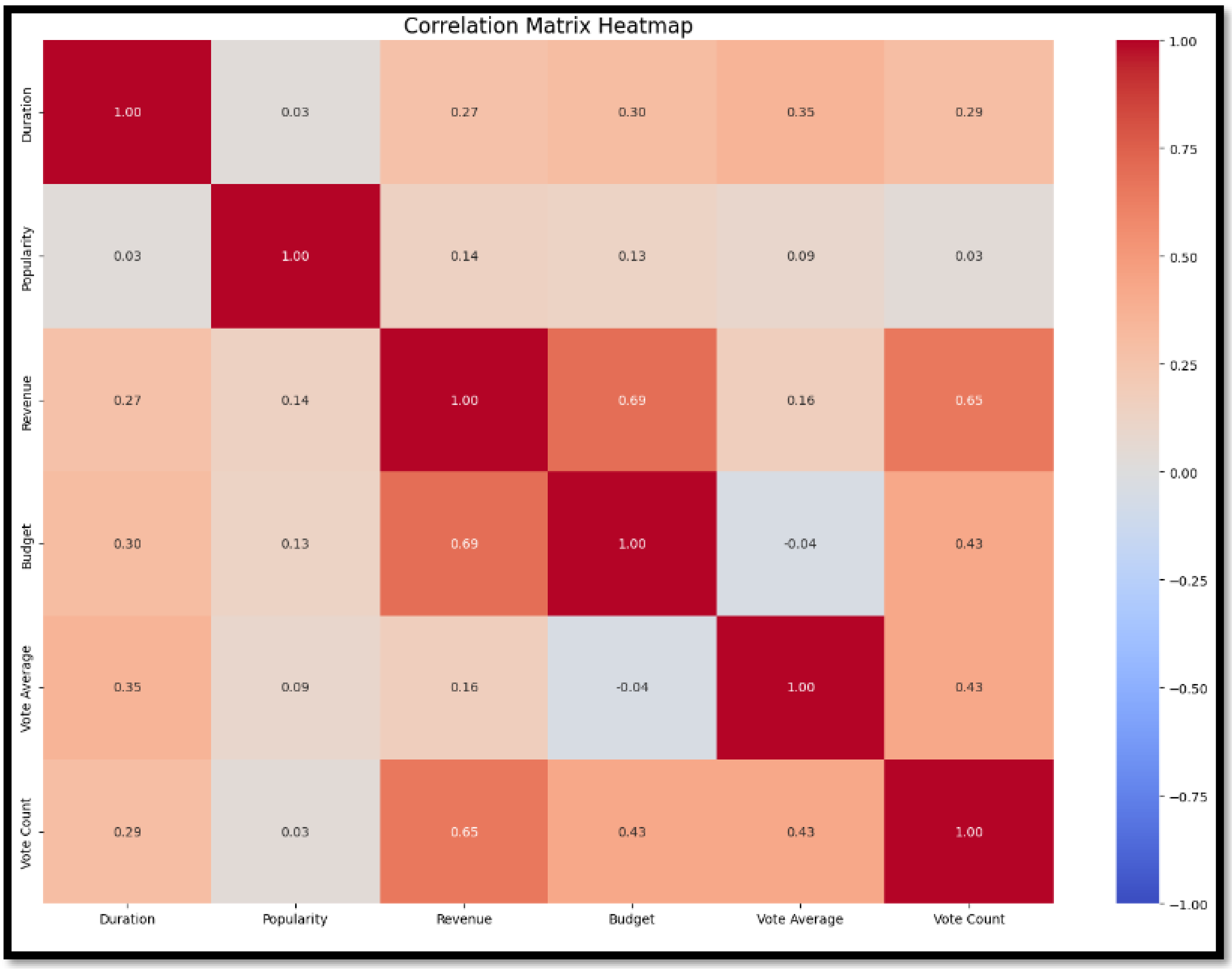


*(Figure **7** Scatter Plots of Numerical Variables vs. Popularity)*

Scatter Plots of Numerical Variables vs. Popularity:

- **Duration vs. Popularity:** While there is a slight negative correlation, it is not vital. This suggests that duration might not be a significant factor in determining popularity.
- **Revenue vs. Popularity:** A weak positive correlation indicates that higher revenue might slightly contribute to popularity, but other factors likely play a more significant role.
- **Budget vs. Popularity:** The lack of a clear correlation suggests that budget might not strongly predict popularity.
- **Vote Average vs. Popularity:** A moderate positive correlation suggests that higher vote averages are associated with increased popularity.
- **Vote Count vs. Popularity:** A robust positive correlation indicates that a higher number of votes is a significant factor in determining popularity.

Multivariate Analysis:



(Figure 8 Correlation Matrix Heatmap)

Correlation Matrix Heatmap

- **Vote Count and Popularity:** The strongest positive correlation is observed between vote count and popularity, indicating that as the number of votes increases, popularity tends to increase significantly. This suggests that user engagement, as measured by the number of votes, is a primary driver of popularity.
- **Revenue and Budget:** A moderate positive correlation exists between revenue and budget, suggesting that higher budgets are often associated with higher revenue. This might indicate that increased spending on marketing, production, or other resources can lead to more tremendous financial success.
- **Vote Average and Popularity:** A moderate positive correlation is also seen between vote average and popularity, implying that content with higher ratings tends to be more popular. This suggests that quality, as perceived by users, is another critical factor influencing popularity.

- **Duration and Popularity:** The correlation between duration and popularity is weak, suggesting that content length has little impact on its popularity. This could mean other factors, such as quality, relevance, or timely delivery, are more influential.
- **Budget and Popularity:** The correlation between budget and popularity is also weak, indicating that spending on production or marketing might not strongly predict popularity. This could suggest that other factors, such as content quality or user engagement, are more critical.
- **Revenue and Vote Average**: There is a weak positive correlation between revenue and vote average, suggesting that higher revenue might slightly contribute to higher ratings, but the relationship is not strong. This could indicate that other factors, such as content quality or user engagement, are more influential in determining ratings.

Dropping columns:

Several columns were dropped to streamline the dataset and focus on our project's objectives. The specific reasons for dropping each column are as follows:

1. **Actors**: Contains names of actors, which is textual data. Processing such textual information requires Natural Language Processing (N.L.P.) techniques. This column is not essential since our current focus is on classification and regression models.
2. **Overview**: Provides descriptive text about each movie. This column also requires N.L.P. for processing, which is unnecessary for our current analysis.
3. **Production Companies**: Lists the production companies involved in each movie. Like the 'Actors' and 'Overview' columns, this text-based data needs N.L.P. for effective use and is omitted for now.
4. **Country**: Contains multiple countries listed for each movie (e.g., "Hong Kong, United Kingdom, United States of America"). The numerous combinations make it challenging to analyse and process. Consequently, this column has been removed to simplify the dataset.
5. **Tagline**: Includes promotional taglines for the movies. As with other textual columns, it would require N.L.P. techniques, which are unnecessary for our current modelling tasks.
6. **Director**: Contains the names of directors, which is textual data. Without N.L.P. pre-processing, this column is not directly applicable to our current models and has been dropped.
7. **Title**: The movie titles are unique identifiers and do not contribute to the predictive power of our models. Hence, this column is excluded.
8. **Release Date**: This column has date-time information, which could be helpful for time series analysis. However, given that the current focus is on classification and regression models, this column is not included in the dataset for now.

Handling Missing or Zero Values:

To address missing or zero values in the Duration column, we first examined the distribution of this feature. The Duration column had only two rows with a zero value. We plotted the distribution of the Duration to understand its overall pattern. The analysis revealed that the median of the Duration distribution was a more suitable imputation value than other methods due to the symmetry of the distribution and its central tendency.

Median Formula

- If $\mathbf{n}$ is odd:

$$\textbf{Median } = \boldsymbol{X}_{\left(\frac{\boldsymbol{n}+\mathbf{1}}{\mathbf{2}}\right)}$$

Where $X_{\left(\frac{n+1}{2}\right)}$ is the value at the $\frac{n+1}{2}$ Th position in the sorted list?

- If $\boldsymbol{n}$ is even:

$$\textbf{Median } = \frac{\boldsymbol{X}_{\left(\frac{\boldsymbol{n}}{\mathbf{2}}\right)} + \boldsymbol{X}_{\left(\frac{\boldsymbol{n}}{\mathbf{2}}+\mathbf{1}\right)}}{\mathbf{2}}$$

Where $X_{\left(\frac{n}{2}\right)}$ and $X_{\left(\frac{n}{2}+1\right)}$ are the values at the $\frac{n}{2}$ th and $\frac{n}{2}+1$ th positions in the sorted list, respectively.

Analysing Popularity variable for regression:

Based on the analysis performed on the `Popularity` feature using descriptive statistics and sorting techniques, the following observations and actions were taken:

1. **Descriptive Analysis of `Popularity`:**

Using **`df['Popularity'].describe(),`** the summary statistics provided an overview of the central tendency, dispersion, and range of the `Popularity` values. The mean and median indicated the overall trend, while the standard deviation highlighted the variability within the dataset.

2. **Identifying Top and Bottom Movies by Popularity :**

By sorting the dataset in descending order of `Popularity` **(`df.sort_values(by='Popularity', ascending=False)[:5]`)**, the top 5 most popular movies were identified, showcasing the highest levels of audience engagement and interest.

Conversely, sorting in ascending order **(`df.sort_values(by='Popularity', ascending=True)[:5]`)** revealed the 5 least popular movies, indicating minimal audience interaction.

3. **Vote Analysis :**

Movies with the highest average votes (`df.sort_values(by='Vote Average', ascending=False)[:5]`) were highlighted to understand which films had the most positive audience reception.

The analysis of the top 20 movies by vote count **(`df.sort_values(by='Vote Count', ascending=False)[:20]`)** helped identify movies with the highest level of viewer feedback, while sorting by the lowest 20 **(`df.sort_values(by='Vote Count', ascending=True)[:20]`)** captured movies with minimal audience votes.

4. Filtering by Vote Count :

To ensure the data's robustness and to avoid bias from movies with very few votes, rows with a `Vote Count` of less than 200 were dropped (`df = df[df['Vote Count'] >= 200]`). This step was crucial in refining the dataset to focus on movies with sufficient audience interaction, thereby enhancing the reliability of subsequent analyses.

These steps were undertaken to refine the data and focus on movies with significant audience engagement, ensuring that further modelling and predictions were based on relevant and reliable information

Feature Engineering / Creation:

To classify movies as 'Popular,' a threshold value was selected based on the distribution of the Popularity variable. Several thresholds were tested to determine the most suitable cut-off for defining a movie as popular.

Threshold Testing Results:

Threshold = 10:

Movies classified as popular: 4742

Movies not classified as popular: 0

Threshold = 20:

Movies classified as popular: 3371

Movies not classified as popular: 1371

Threshold = 30:

Movies classified as popular: 1761

Movies not classified as popular: 2981

Threshold = 40:

Movies classified as popular: 1097

Movies not classified as popular: 3645

Threshold = 50:

Movies classified as popular: 744

Movies not classified as popular: 3998

Chosen Threshold:

Based on the analysis, a threshold of 40 was chosen. This threshold was selected to balance the number of movies classified as popular and not popular, aiming to reduce the class imbalance while maintaining a reasonable number of classified popular movies.

Imputing outliers:

The process of identifying and imputing outliers in the dataset was done using two methods: Z-Score and Interquartile Range (IQR). Outliers were detected for numerical columns, and imputation was performed based on the method that best handled the detected outliers.

Z-Score:

The Z-Score measures how many standard deviations a data point is from the mean. It is calculated as follows:

$$Z = \frac{x - \mu}{\sigma}$$

Where:

Z = Z-score
x = the value of the data point
μ = mean of the dataset
σ = standard deviation of the dataset

Interquartile Range (IQR):

The Interquartile Range (IQR) is a measure of statistical dispersion and is used to describe the spread of the middle 50% of a dataset. It is calculated as the difference between the third quartile (Q3) and the first quartile (Q1). The formula for the IQR is:

$$\text{IQR} = Q3 - Q1$$

Where:

- **Q1 (First Quartile):** The 25th percentile of the data, below which 25% of the data falls.
- **Q3 (Third Quartile):** The 75th percentile of the data, below which 75% of the data falls.

To identify outliers using IQR:

- **Lower Bound**: $Q1 - 1.5 \times \text{IQR}$
- **Upper Bound**: $Q3 + 1.5 \times \text{IQR}$

Handling Skewness:

Skewness measures the asymmetry of the data distribution.

Calculate skewness:

$$\text{Skewness} = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^3$$

Where

- n =Number of observations
- x_i = Individual Value in the dataset
- $\bar{x}$ = Mean of the dataset
- s = Standard deviation of the dataset

Transformations and their formulas:

To address skewness, several transformations are applied:

Log Transformation:

The log transformation reduces skewness, stabilises variance, and makes data more normally distributed, especially when dealing with positive values. The general formula for applying a log transformation to a variable X is:

$$X_{\text{transformed}} = \log(X + c)$$

Where

The log can be any logarithmic base, such as natural log (ln), base 10($\log_{10}$), or base 2 ($\log_2$)

- X = Original value of the variable.
- c = A constant added to avoid taking the zero or negative values log. Commonly, c is set to 1

Square Root Transformation

The Square Root Transformation stabilises variance, reduces skewness, and makes data more normal-like, especially for data with positive values and moderate skewness. The general formula for applying a square root transformation to a variable X is:

$$X_{\text{transformed}} = \sqrt{X + c}$$

Where

- X = Original value of the variable.
- c = A constant added to avoid taking the square root of zero or negative values. Set to 0 or 1

Box-Cox Transformation:

The Box-Cox Transformation is a power transformation that stabilises variance and makes data more normally distributed. The formula for the Box-Cox Transformation is:

$$X_{\text{transformed}} = \begin{cases} \dfrac{X^{\lambda} - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ \ln(X), & \text{if } \lambda = 0 \end{cases}$$

Where

- X = Original value of the variable. (must be positive)
- λ = Transformation parameter that varies and is usually determined through maximum likelihood estimation to normalise the data best.

Yeo-Johnson Transformation:

The Yeo-Johnson Transformation is a modification of the Box-Cox Transformation that can be applied to both positive and negative values, making it more versatile. The formula for the Yeo-Johnson Transformation is defined as follows:

$$X_{\text{transformed}} = \begin{cases} \dfrac{\left((X+1)^{\lambda} - 1\right)}{\lambda}, \\ \log(X+1), \\ \dfrac{-((X \mid +1)^{2} - 1)}{2 - \lambda}, \\ -\log(|X| + 1), \end{cases}$$

Where

- X = Original value of the variable
- λ = Transformation parameter that determines the nature of the transformation.

For each column, the skewness of the transformed data is calculated. The transformation that results in the least skewness is chosen as the best transformation. The column in the data frame is updated with the values from the best transformation.

Encoding Categorical Variables to Numerical:

One-Hot Encoding:

- The Genre column was successfully one-hot encoded, resulting in new columns for each genre, and the original column was removed.

Language Analysis and Transformation:

- A total of 30 unique languages were identified.
- A summary of language frequencies was provided.
- The **'Is_English'** binary column was added to denote English language films, and the original Language column was dropped.

Scaling:

To ensure that all numerical features are on the same scale, which is essential for many machine learning algorithms, we used **'StandardScaler'** from **'sklearn.preprocessing'**. Standardisation centres the data around the mean and scales it to unit variance, which helps improve model performance and convergence rates.

Steps Performed:

1. **Feature Selection**: We identified vital numerical features in the dataset that required scaling, including Budget, Duration, Vote Count, Revenue, and Vote Average. These features were chosen based on their impact on predicting movie popularity.
2. **Standard Scaling**:

    - We initialised the StandardScaler object, subtracting the mean and scaling the data to unit variance.
    - The scaler was then fitted to the selected numerical columns from the Data Frame (df) and transformed to produce standardised values.

3. **Integration**:

    - The scaled features were reinserted into the original Data Frame, replacing the original values to maintain a standardised dataset.

Model Building:

Split Data into Training and Testing Sets:

Using the **"train_test_split"** from **"sklearn.model_selection,"** the dataset was separated into train and test sets.30% of the data was reserved for testing, ensuring a robust model performance evaluation.

Model Training and Evaluation:

1. Logistic Regression:

Using **'LogisticRegression'** from **'sklearn.linear_model'** with **class_weight='balanced'** was initialised to handle class imbalance.

Applies for binary classification by estimating the probability of event occurrence, using statistical tests like Wald's test for significance and Pseudo R² for model comparison. (Agarwal et al., 2022)

$$p = P(Y = 1 \mid X) = \frac{1}{1 + e^{-z}}$$

Where

- $z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$
- β_0Is the intercept term.
- $\beta_1, \beta_2, \dots \dots, \beta_n$are the coefficients for each predictor $X_1, X_2, \dots, X_n$
- e Is the base of the natural logarithm

Accuracy Results:

- Training Accuracy: **0.756**
- Testing Accuracy: **0.760**

Hyperparameter Tuning:

Grid Search:

- To optimise the model, a Grid Search with **5-fold cross-validation** was conducted using '**GridSearchCV'** over various parameters **(C, penalty, and solver).**
- Best Parameters: **{C: 0.1, penalty: 'l2', solver: 'saga'}** with a best cross-validation score of **0.821**.

Cross-Validation:

- The optimised model was further validated using 5-fold cross-validation.
- Cross-validation scores ranged from 0.49 to 0.87, with an average score of **0.739**, indicating model variability across folds.

2. Naïve Bayes:

It uses a probabilistic approach to classify data by estimating the posterior probability and selecting the class with the highest probability based on the M.A.P. rule. To address the class imbalance, the SMOTE (Synthetic Minority Over-sampling Technique) was applied to the training data before the model was fitted.

$$P(C_k \mid X) = \frac{P(X \mid C_k) \cdot P(C_k)}{P(X)}$$

Where

- $P(C_k \mid X)$ Is the posterior probability of class C_k given the features X
- $P(X \mid C_k)$ Is the likelihood of features X given the class C_k
- $P(C_k)$ Is the prior probability of class C_k
- $P(X)$ is the marginal probability of features X

Accuracy Results:

- Training Accuracy: 0.322
- Testing Accuracy: 0.316

Hyperparameter Tuning

Grid Search:

- Hyperparameter tuning was not conducted explicitly for Naive Bayes due to the nature of the algorithm, which does not have many tunable parameters like logistic regression.

Cross-Validation:

- Cross-validation was implicitly performed within the learning curve analysis, which indicated variability in performance across different folds.

3. Random Forest:

Using **the RandomForestClassifier from 'sklearn.ensemble', multiple decision trees are combined** to improve predictions, offering robustness against outliers and irrelevant inputs. (Verma, H. & Verma, G. (2019))

$$\hat{C} = \operatorname{argmax}_C \left(\sum_{i=1}^{N} I \ \left(T_i(X) = C\right) \right)$$

Where

- X Is the input feature vector.
- $T_i(X)$ Is the prediction of the ith tree
- C Represents a possible class label.
- I Is the indicator function, which equals 1 if the tree predicts class C and 0 otherwise.

Accuracy Results:

- Training Accuracy: 1.0
- Testing Accuracy: 0.852

Hyperparameter Tuning

Grid Search:

- To optimise the model, a Grid Search with 5-fold cross-validation was conducted using **GridSearchCV** over a range of parameters **(max_depth, min_samples_split, min_samples_leaf, and n_estimators).**
- Best Parameters**: {max_depth: None, min_samples_leaf: 1, min_samples_split: 5, n_estimators: 100}** with the best cross-validation score indicating improved model performance.

Cross-Validation:

- The optimised model was further validated using 5-fold cross-validation.
- Cross-validation helped refine the model's robustness, leading to an adjusted training accuracy of 0.977 and a testing accuracy of 0.852.

4. Multilinear Regression:

Linear regression from 'sklearn.linear_model' uses multiple independent variables to estimate the target, applies V.I.F. and statistical tests for feature selection, and checks significance with O.L.S. and hypothesis tests. (Agarwal et al., 2022)

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Where

- y = dependent variable (target)
- β_0 = intercept term
- $\beta_1, \beta_2, \ldots \ldots \beta_n$ = coefficients for each feature
- $x_1, x_2 \ldots . x_n$= independent variables (features)
- ϵ = error term (residual)

Performance Metrics:

- Training R² Score: 0.2799
- Training R.M.S.E.: 0.0391
- Testing R² Score: 0.2927
- Testing R.M.S.E.: 0.0380

Cross Validation:

Using cross-validation with 5 folds to evaluate the model, Training scores remained relatively high and stable, suggesting the model fits the training data well.

Cross-validation scores indicated a slight variance but generally aligned with the training scores, showing that the model generalises reasonably well.

R.F.E.:

The **Recursive Feature Elimination (R.F.E.)** technique was employed with Linear Regression to select the top 5 features.

Selected Features: ` ['Vote Average,' 'Vote Count,' 'Action,' 'Animation,' 'Drama'] `

The linear regression model was retrained using only the features selected from R.F.E.

Performance Metrics with R.F.E.:

- Training R² Score: 0.2762
- Testing R² Score: 0.2876

5. XGBoost Regressor:

From **xgboost** using **XGBRegressor** uses gradient boosting with decision trees to optimise prediction accuracy, incorporates regularisation to prevent overfitting, and iteratively refines models based on residuals.

- **Objective Function:**

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Where:

- $l(y_i, \hat{y}_i)$ = loss function (e.g., Mean Squared Error)
- $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\|\mathbf{w}\|^2$
    - T = number of leaves
    - λ = regularisation term to prevent overfitting

Performance Metrics:

- Training R² Score: 0.9115
- Testing R² Score: 0.2259

Hyperparameter Tuning:

A GridSearchCV was employed to fine-tune the XGBoost Regressor using the following hyperparameters:

- **`n_estimators`**: Number of boosting rounds (100, 200, 300)
- **`max_depth`**: Maximum depth of a tree (3, 5, 7)
- **`learning_rate`**: Step size shrinkage (0.01, 0.1, 0.2)
- **`subsample`**: Fraction of samples used per tree (0.6, 0.8, 1.0)
- **`colsample_bytree`**: Fraction of features used per boosting round (0.6, 0.8, 1.0)
- **`gamma`**: Minimum loss reduction to make a split (0, 0.1, 0.2)

GridSearchCV was performed with 5-fold cross-validation to identify the best combination of hyperparameters.

Best Hyperparameters Identified:

- **`colsample_bytree`**: 0.8
- **`gamma`**: 0
- **`learning_rate`**: 0.01
- **`max_depth`**: 5
- **`n_estimators`**: 300

- **`subsample`**: 0.6

Performance Metrics with Tuned XGBoost Model:

- Training R² Score: 0.4547
- Testing R² Score: 0.3554

6. Adaboost:

Using **AdaBoostRegressor** from **'sklearn.ensemble'** ensemble learning with multiple weak learners to improve predictions, combines them through weighted voting, and adjusts weights to minimise errors iteratively.

- **Update Rule for Weights:**

$$w_{i+1} = w_i \cdot e^{\alpha \cdot \left(1 - y_i f(x_i)\right)}$$

Where:

- w_i= weight of the i *th* data point
- α = model's coefficient, which depends on the model error
- $f(x_i)$ = predicted value from weak learner
- y_i = true value

- **Final Prediction:**

$$F(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right)$$

Performance Metrics:

- Training R² Score: 0.3144
- Testing R² Score: 0.3146

Hyperparameter Tuning with GridSearchCV:

GridSearchCV was used to optimise the AdaBoost Regressor, testing the following hyperparameters:

`n_estimators`: Number of boosting stages (50, 100, 200)

`learning_rate`: Shrinks the contribution of each regressor (0.01, 0.1, 1.0)

Grid search was conducted using 5-fold cross-validation, evaluating the models based on R² scores.

Best Hyperparameters Identified:

- **`learning_rate`**: 0.1
- **`n_estimators`**: 50

Performance Metrics with Tuned AdaBoost Model:

- Training R² Score: 0.3246
- Testing R² Score: 0.3314

7. Decision Tree Regressor

Using **DecisionTreeRegressor** from **'sklearn.tree'** builds a model by splitting the data into subsets based on feature values, creating a tree structure where each node represents a decision, and predicts values by averaging outcomes in the terminal nodes.

- **Decision Function:**

$$y = \frac{1}{N_i} \sum_{j \in N_i} y_j$$

Where:

- y = predicted value (mean of target values within region N_i)
- N_i = number of observations in the *i^th^* region
- y_j = target value for the *j^th^* observation in that region

Performance Metrics:

- Training R² Score: 1.0000
- Testing R² Score: -0.4123

Hyperparameter Tuning;

GridSearchCV was used to optimise the Decision Tree Regressor, testing the following hyperparameters:

- **`max_depth`**: Controls the depth of the tree (None, 10, 20, 30, 40)

- **`min_samples_split`**: Minimum number of samples required to split an internal node (2, 5, 10)
- **`min_samples_leaf`**: Minimum number of samples required at a leaf node (1, 2, 4)
- **`max_features`**: Number of features to consider when looking for the best split (None, 'sqrt', 'log2')
- **`criterion`**: Function to measure the quality of a split ('mse', 'friedman_mse', 'mae')

The grid search used 5-fold cross-validation to evaluate the models based on R^2 scores.

Best Hyperparameters Identified:

- **`criterion`**: 'friedman_mse'
- **`max_depth`**: 10
- **`max_features`**: 'log2'
- **`min_samples_leaf`**: 4
- **`min_samples_split`**: 10

Performance Metrics with Tuned Decision Tree Model:

- Training R^2 Score: 0.3500
- Testing R^2 Score: 0.1002

8. Support Vector Regressor:

Supervised learning algorithm for regression. Finds optimal hyperplane that maximises margin and minimises error.

Transforms data into higher dimensions using kernel functions (e.g., Linear, Polynomial, RBF, Sigmoid) without explicitly computing coordinates.

Objective Function:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} (\xi_i + \xi_i^*)$$

Where:

- w = weight vector
- b = bias term

- ξ_i, ξ_i^* = slack variables (for over- and under-estimations)
- C = regularisation parameter (controls the trade-off between flatness of the function and tolerance to errors)

Decision Function:

$$f(x) = \mathbf{w} \cdot \mathbf{x} + b$$

Subject to:

$$y_i - (\mathbf{w} \cdot \mathbf{x_i} + b) \leq \epsilon + \xi_i^*$$

$$(\mathbf{w} \cdot \mathbf{x_i} + b) - y_i \leq \epsilon + \xi_i$$

Performance Metrics for Support Vector Regressor (S.V.R.):

- Training R^2 Score: 0.0610
- Testing R^2 Score: 0.0638
- Training R.M.S.E.: 0.0446
- Testing R.M.S.E.: 0.0437

RESULTS

This project primarily focuses on developing and evaluating machine learning models based on various datasets for classification and regression tasks. The classification models tested include Logistic Regression, Random Forest, and Naive Bayes, while the regression models include Decision Tree Regressor, AdaBoost, XGBoost, Support Vector Regressor, and Multilinear Regression.

The performance of the models is assessed using metrics such as Precision, Recall, F1-Score, Accuracy, and Support for classification tasks, and R^2 score and Mean Squared Error (MSE) for regression tasks. In this section, the models are analysed in detail, and the significance of the evaluation metrics is highlighted based on the obtained results.

Evaluation metrics and their significance:

1. **Precision:**

Description: Ratio of correctly predicted positive observations to total predicted positives.

Significance: It is essential when the cost of false positives is high, as it indicates the correctness of positive predictions.

$$\textbf{Precision} = \frac{\textbf{True Positives (T.P.)}}{\textbf{True Positives (T.P.)} + \textbf{False Positives (F.P.)}}$$

True Positives (T.P.): The number of correctly predicted positive instances.
False Positives (F.P.): The number of instances incorrectly predicted as positive (i.e., they are negative).

2. **Recall (Sensitivity or True Positive Rate):**

Description: Fraction of accurate positive predictions among all positive instances.

Significance: Critical when missing a positive (false negative) instance is costly, as it measures the model's ability to detect positive instances.

$$\textbf{Recall} = \frac{\textbf{True Positives (T.P.)}}{\textbf{True Positives (T.P.)} + \textbf{False Negatives (F.N.)}}$$

False Negatives (F.N.): The number of positive instances the model predicted as unfavourable.

3. **F1-Score:**

Description Harmonic mean of Precision and Recall, balancing between the two metrics.

Significance: Useful when both Precision and Recall are important, especially in the presence of an unequal class distribution.

$$\textbf{F1-Score} = 2 \times \frac{\textbf{Precision} \times \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$$

4. **Accuracy:**

Description: Ratio of total correct predicted observations to total observations.

Significance: Popular metric, but less informative on imbalanced datasets where more detailed metrics like Precision and Recall are needed.

$$\textbf{Accuracy} = \frac{\textbf{True Positives (TP) + True Negatives (TN)}}{\textbf{Total Observations}}$$

True Negatives (T.N.): The number of correctly predicted negative instances.

5. **Support:**

Description: Total number of occurrences of each class in the dataset.

Significance: Provides context to Precision, Recall, and F1-score, explaining how many instances of each class were present during the evaluation.

$$\textbf{Support} = \textbf{Number of true instances for a given class}$$

6. **R2 Score:**

Description: Represents the proportion of the variance in the dependent variable that is predictable from the independent variables.

Significance: A high R^2 value indicates a good fit between the model and the data, while a low R^2 suggests a poor fit; however, R^2 alone does not guarantee model performance or prevent overfitting.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Where:

- y_i = actual value
- $\hat{y}_i$ = predicted value
- $\bar{y}$ = mean of actual values
- n = number of data points

7. Root Mean Square Error:

Description: Measures the average magnitude of prediction errors by calculating the square root of the average squared differences between predicted and actual values.

Significance: Lower R.M.S.E. values indicate better model accuracy and closer predictions of the actual outcomes.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Where:

- y_i = actual value
- $\hat{y}_i$ = predicted value
- n = number of data points

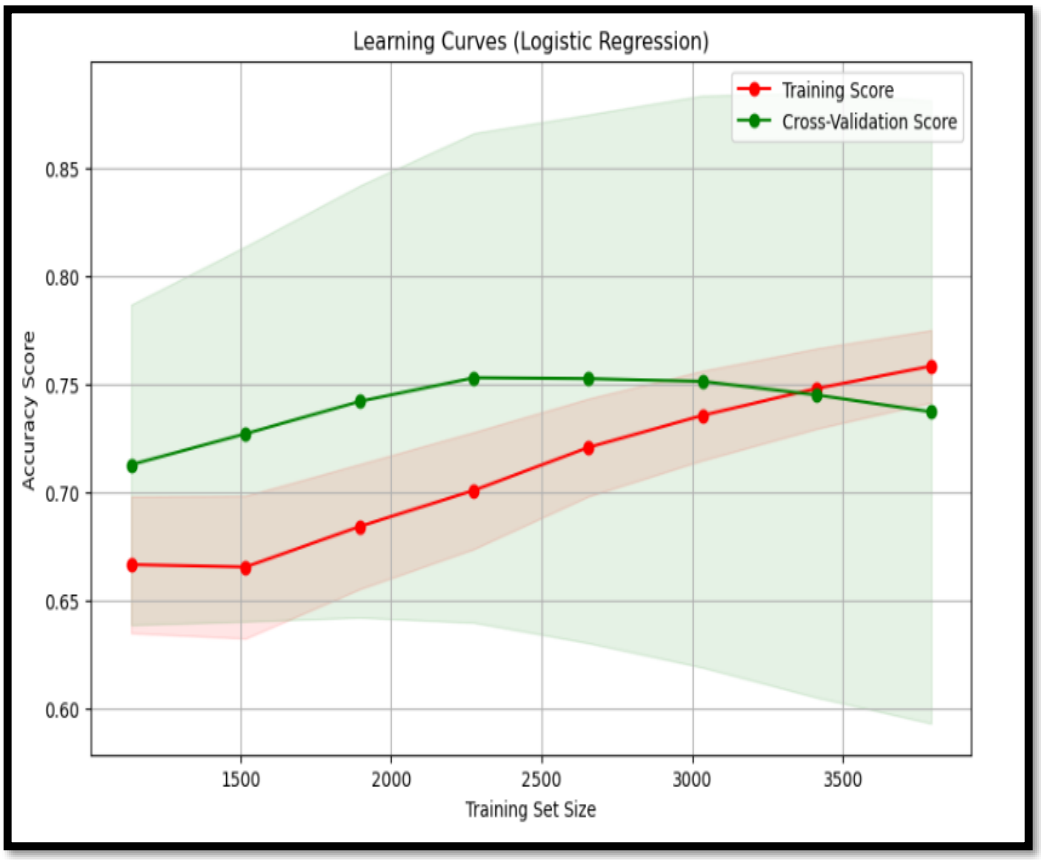
Classification Report of 3 models:

Model	Training accuracy	Testing accuracy	Precision (0)	Precision (1)	Recall (0)	Recall (1)	F1-Score (0)	F1-Score (0)	Accuracy
LR	0.7541	0.759	0.93	0.5	0.74	0.82	0.82	0.62	0.76
NB	0.3224	0.3162	0.84	0.25	0.13	0.93	0.22	0.39	0.32
RF	0.9771	0.8524	0.87	0.77	0.95	0.54	0.91	0.64	0.85

Regression Report of 5 models:

Model	Training r2 score	Testing r2 score	Training RMSE	Testing RMSE
MLR	0.2799	0.2927	0.0391	0.038
XGBoost	0.4547	0.3554	0.034	0.0363
Adaboost	0.3267	0.3296	0.0378	0.037
DTR	0.4165	0.204	0.0352	0.0403
SVR	0.061	0.0638	0.0446	0.0437

1. Logistic Regression



(Figure 9 Learning Curve for Logistic Regression)

1. Training Score (Red Line):

The R² score of the training set initially drops as the number of training examples increases but stabilises at around 0.6. This indicates that the model is effectively learning the underlying patterns in the data without perfectly capturing all the complexities.

A stable training score slightly below 0.6 suggests that the model is not overfitting, meaning it avoids learning noise or random fluctuations specific to the training set. This balance shows that the model captures significant trends in the data but leaves room for improvement in its predictive power.

This pattern of the training score highlights the model's robustness in maintaining consistency as more data is introduced, making it a reliable choice for handling larger datasets in future expansions.

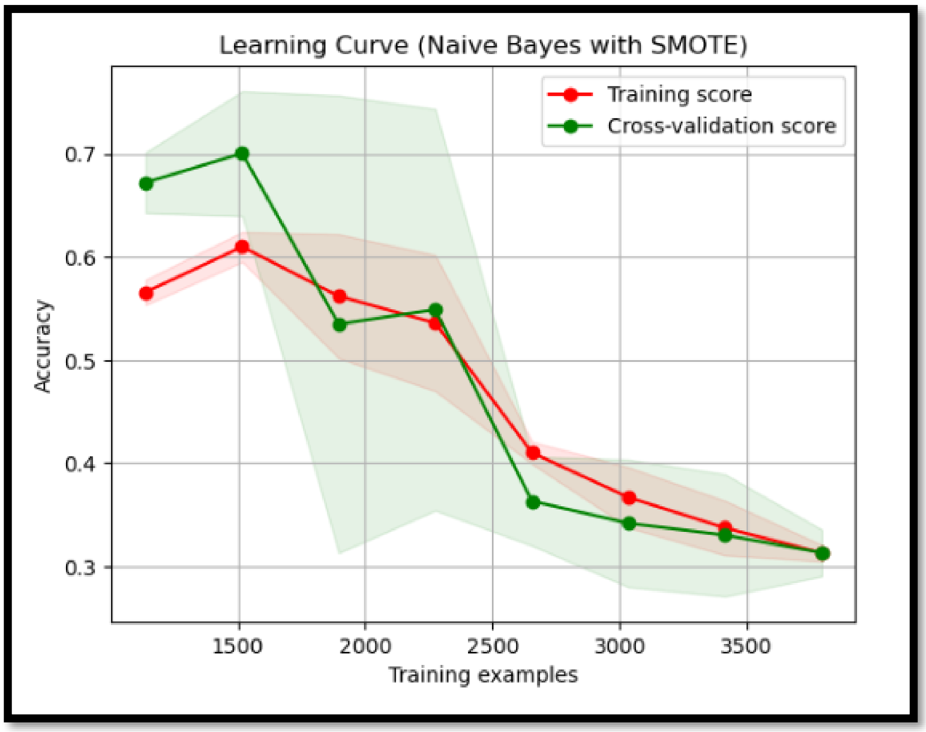
2. Cross-Validation Score (Green Line):

The cross-validation score, which measures how well the model performs on unseen data, remains consistently lower than the training score. It starts around 0.3 and gradually stabilises near 0.25, indicating the model's struggle to generalise to new, unseen examples.

This lower cross-validation score points to underfitting, suggesting that the model may be too simplistic to capture the complexity of the movie popularity prediction task. It reflects the high-bias nature of the model, where it misses key predictive features that could enhance accuracy.

However, the relatively small gap between the training and cross-validation scores indicates that the model does not suffer from high variance, which large fluctuations between the training and validation performances would characterise. Instead, the consistent scores suggest that the model maintains a stable, albeit limited, performance across different data subsets.

2. Naïve Bayes



(Figure 10 Learning Curve for Naïve Bayes)

1. Training Score (Red Line):

-The R² score for the training data begins relatively high, indicating that the model initially fits the smaller dataset well, possibly capturing some specific patterns or noise.

-As the training set size increases, the score gradually decreases and stabilises, reflecting a reduction in overfitting. This trend suggests that the model is starting to generalise better, moving away from learning noise to focusing on the broader, more relevant patterns in the data.

-The observed decline in training score with more data indicates that the model is becoming less biased towards the training set, highlighting an improvement in its robustness and reliability.

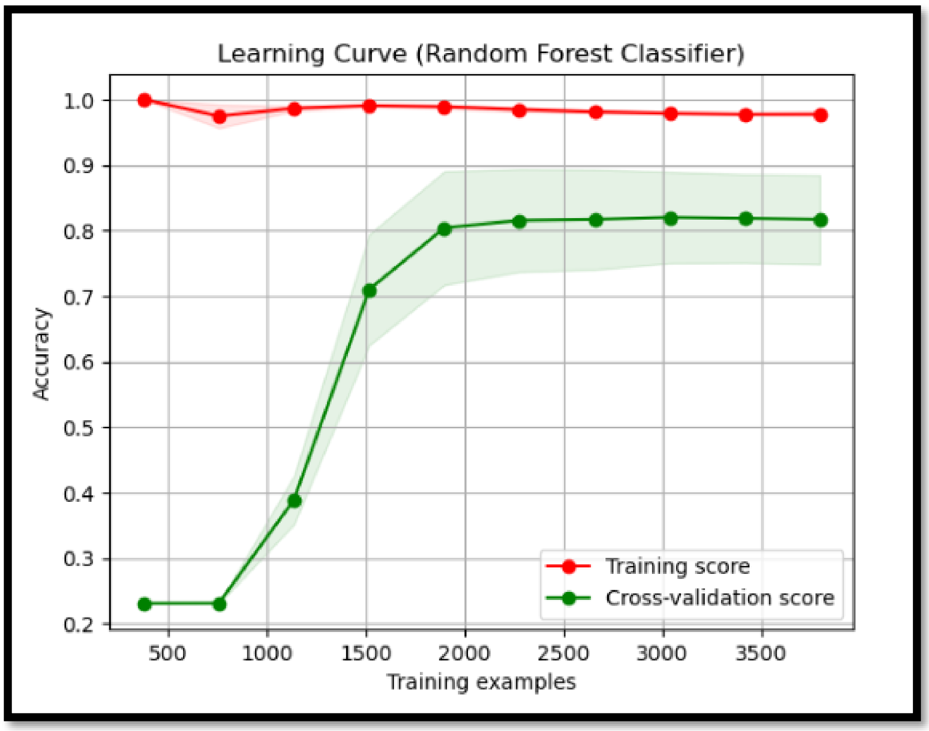
2. Cross-Validation Score (Green Line):

-The cross-validation score, representing the model's performance on unseen data, shows an upward trend as more data is introduced. It begins at a lower value and gradually increases, eventually plateauing.

-This rise in the cross-validation score suggests that the model's ability to generalise is improving, benefiting from additional data to capture better the underlying patterns that are representative of the target variable — movie popularity.

-The plateau indicates that the model reaches a point where additional data continues to support stable performance without significant further gains, pointing to an optimal balance between bias and variance.

3. Random Forest Classifier:



(Figure 11 Learning Curve for Random Forest Classifier)

1. Training Score (Red Line):

The training score is very high (close to 1.0) and remains nearly constant as the number of training examples increases. This indicates that the Random Forest model fits the training data exceptionally well.

Since it is almost at 100%, this suggests that the model is potentially overfitting on the training set, meaning it is capturing even minor details (including noise) ideally in the training data.

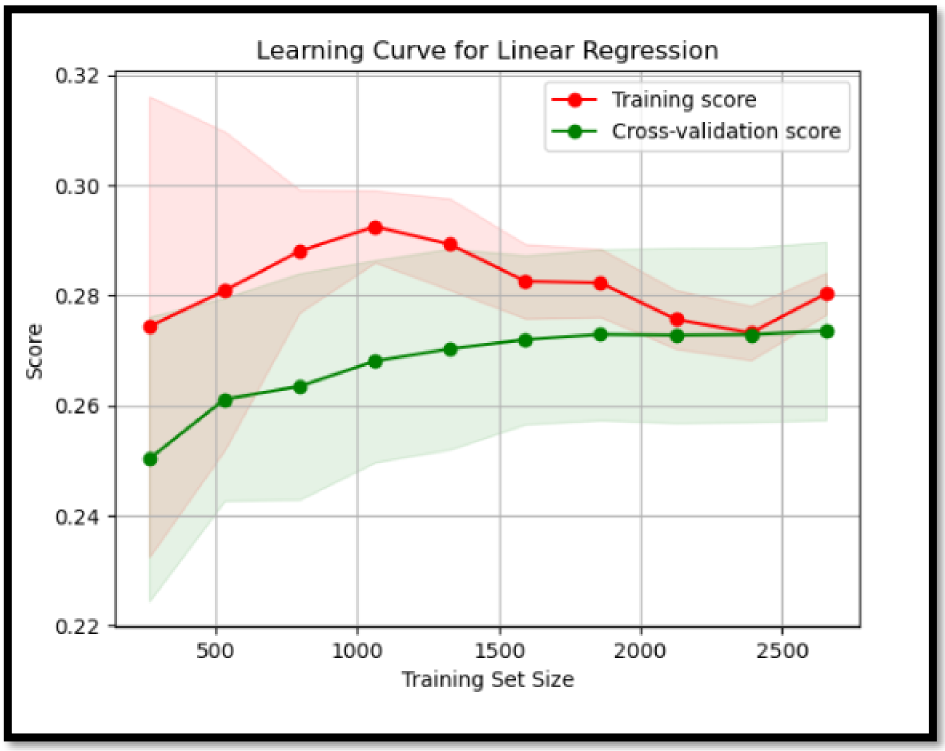
2. Cross-Validation Score (Green Line):

The cross-validation score starts low (around 0.2) with a small training set but improves rapidly as more data is introduced.

The score stabilises at around 0.8-0.85 after approximately 2000 training examples. This indicates that the model is learning general patterns in the data but still leaving room for improvement.

The shaded region around the cross-validation score indicates the variance (instability) of the model's performance. It narrows as more training examples are used, suggesting that larger datasets make the model's performance more reliable.

4. **Multilinear Regression:**



(Figure 12 Learning Curve for Multilinear Regression)

1. Training Score (Red Line):

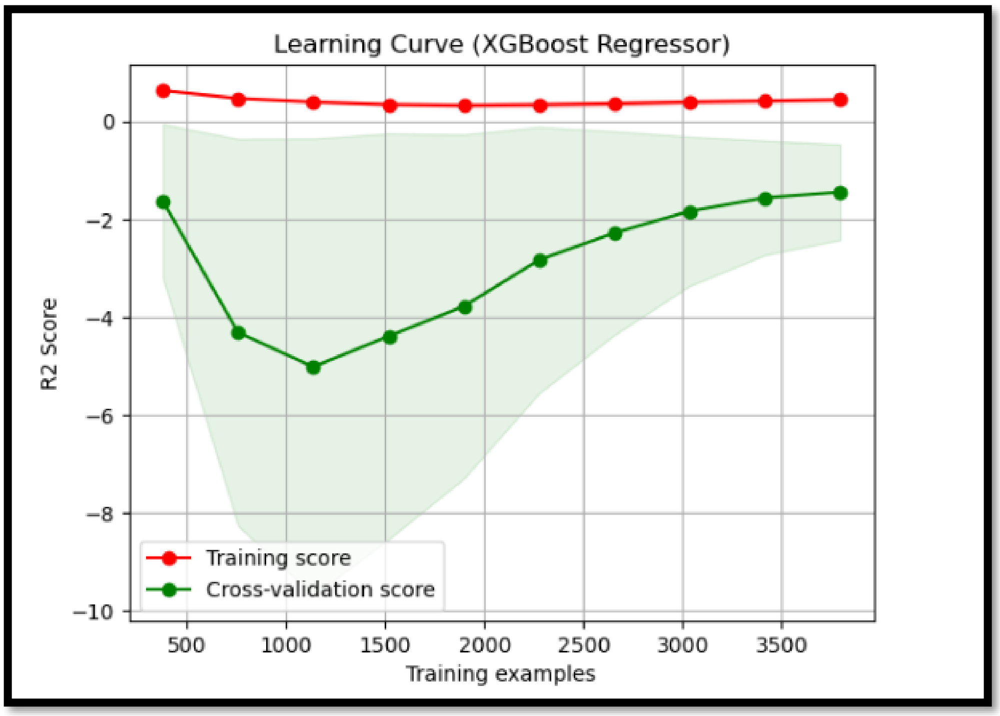
The training score starts relatively high and increases slightly as the training set size grows.

After a certain point, it plateaus and fluctuates slightly. This suggests that the model can fit the training data well but might not capture more complex patterns as the dataset grows.

2. Cross-Validation Score (Green Line):

The cross-validation score starts lower than the training score and slowly increases with more data.Even though it improves, it remains lower than the training score, indicating a generalisation gap. This means the model performs well on the training data but does not generalise as effectively to unseen data.

5. XGBoost:



(Figure 13 Learning Curve for XGBoost Regressor)

1. Training Score (Red Line):

The training R² score is consistently close to 0 across all training set sizes, indicating that the model performs poorly on the training data. In regression tasks, an R² score of 0 suggests that the model is only as good as the mean prediction (it does not capture any of the variance in the data).

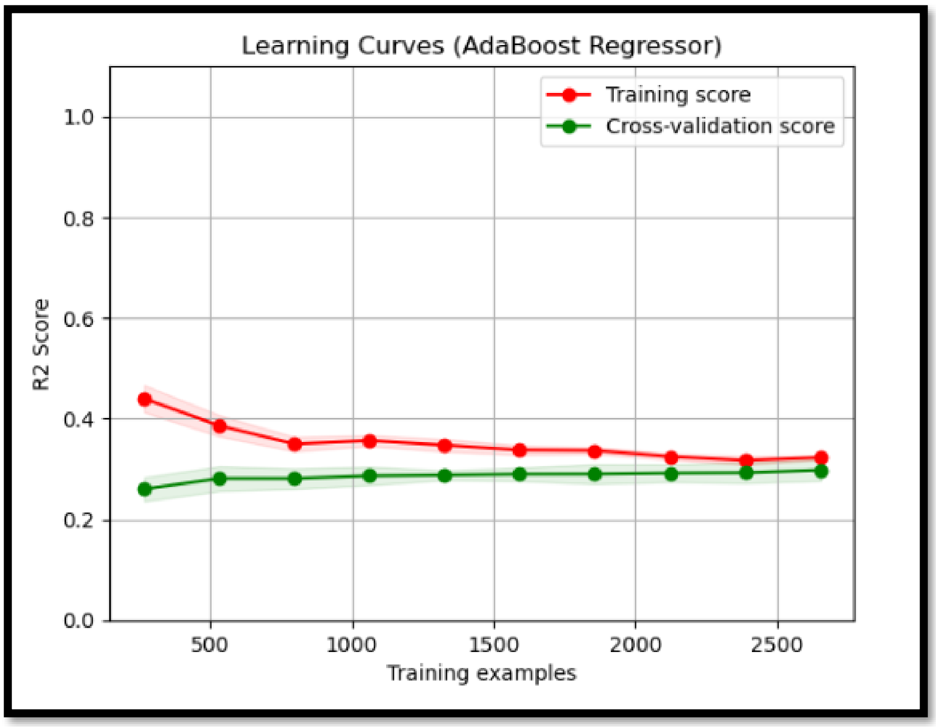
The lack of improvement in the training score with more data suggests that the model is underfitting the training data. This could be due to improper hyperparameters or insufficient model complexity to capture the relationships in the data.

2. Cross-Validation Score (Green Line):

The cross-validation R² score starts very low (around -8), meaning the model performs much worse than predicting the mean of the target variable.

The cross-validation score improves as more training examples are added, reaching around -2 after about 3500 training examples. Although the performance improves, an R² score below 0 still indicates a poor fit — the model is not accurately predicting the target variable and might be underfitting.

6. AdaBoost:



(Figure 14 Learning Curve for Adaboost Regressor)

1. Training Score (Red Line):

The R^2 score of the training set stays relatively stable after an initial drop as more examples are introduced. It hovers slightly below 0.6, indicating that the model is capturing a reasonable portion of the variance in the training data but is not perfectly fitting the data.

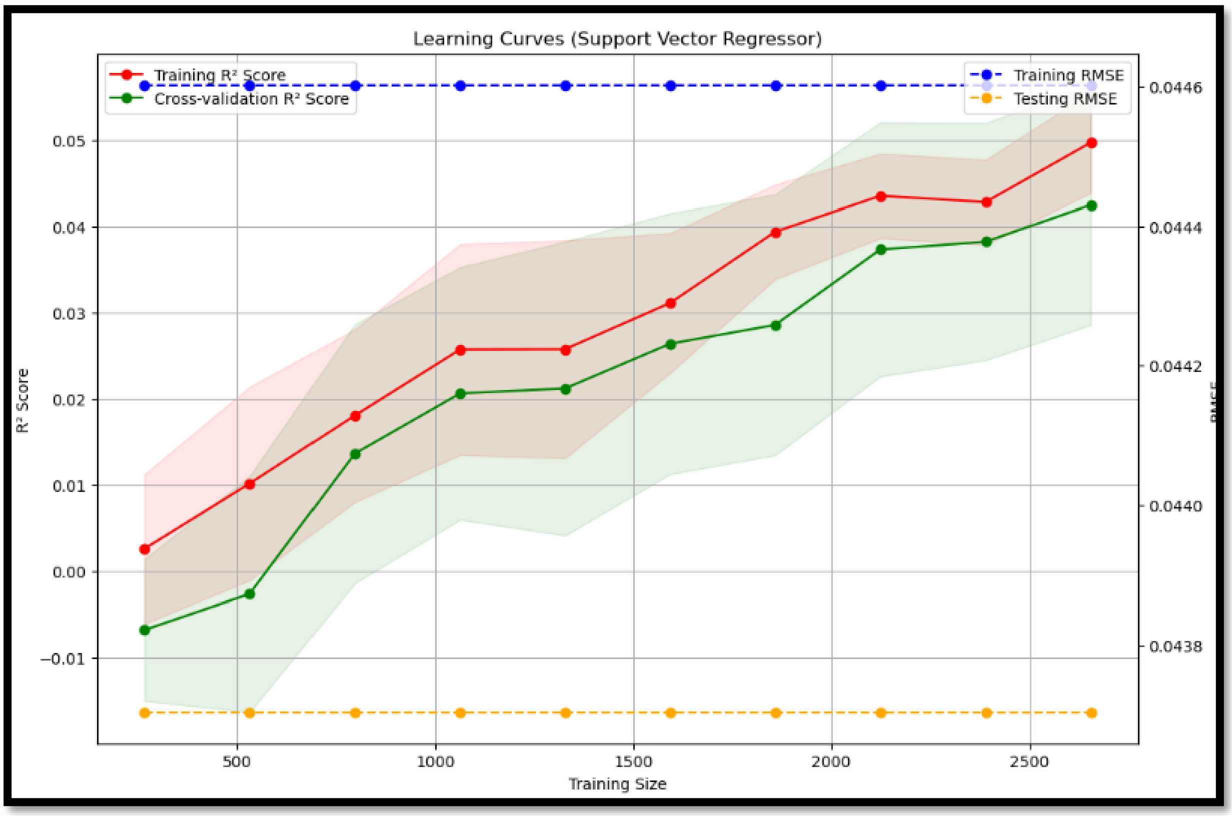
A high but not perfect score on the training set suggests that the model has not overfitted to the training data, which is a good sign.

2. Cross-Validation Score (Green Line):

The cross-validation score, which reflects the model's performance on unseen data, is lower than the training score, indicating underfitting. It starts at around 0.3 and stabilises at around 0.25.

This suggests the model is not generalising well and may be struggling to capture the complexity of the data. The relatively small gap between the training and cross-validation scores shows that the model does not suffer from high variance but does have high bias.

7. S.V.R.



(Figure 15 Learning Curve for S.V.R.)

1. Training R² Score (Red Line):

Increases steadily as the training size grows, starting from around 0.01 and rising to just over 0.045 by the maximum training size.

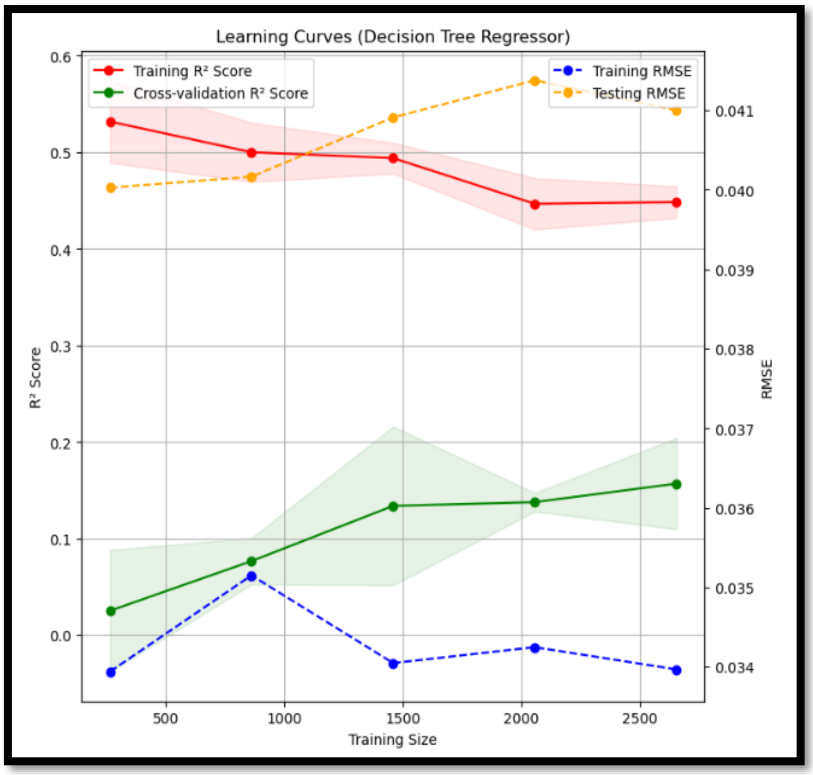
The steady upward trend suggests that the model can capture more variance in the training set as more data is introduced. However, the R² score remains low, indicating that the model is not capturing much of the variance, even within the training data. This implies that the model is not overfitting, but it also suggests that the model may struggle with the complexity of the data.

2. Cross-Validation R² Score (Green Line):

It reflects performance on unseen data and improves as the training size increases but remains below the training score. Starting from slightly below 0, it increases to around 0.025 by the end of the curve.

The consistently lower cross-validation score than the training score suggests the model is underfitting. This underfitting is further highlighted by the relatively low final R² value, implying the model has difficulty generalising well to new data.

8. Decision Tree Regressor



(Figure 16 Learning Curve for Decision Tree Regressor)

1. Training R² Score (Red Line)

The training R² score starts around 0.5 and slightly fluctuates as the training size grows, eventually stabilising near 0.45-0.5.

The initial high R² value followed by a slight downward trend indicates that the model initially fits the training data well but begins to lose some of its ability to capture the variance as the training size increases. This suggests that the model might be slightly overfitting with smaller data sizes but improves with more data.

2. Cross-Validation R² Score (Green Line):

The cross-validation R² score starts near 0 and gradually increases as the training size grows, reaching a maximum of around 0.15.

The persistent gap between the training and cross-validation scores suggests that the model is underfitting the data. The increasing trend in the cross-validation score indicates that the model benefits from more data but still struggles to generalise effectively.

DISCUSSION

The results of this study provide valuable insights into how various factors associated with movies influence their popularity. By applying machine learning models, we could predict movie popularity using features like budget, duration, vote count, vote average, and genre. Our approach involved evaluating multiple models—Logistic Regression, Random Forest, and Naive Bayes for classification tasks, and Decision Tree Regressor, AdaBoost, XGBoost, Support Vector Regressor (S.V.R.), and Multilinear Regression for regression tasks—to determine their effectiveness in predicting movie success.

The evaluation metrics used in this study included Precision, Recall, F1-Score, Accuracy, Support, R^2 score, and Mean Squared Error (M.S.E.). These metrics helped us assess how well each model performed in predicting whether a movie would be popular. Notably, Random Forest and XGBoost emerged as strong performers, showcasing their ability to handle complex interactions among features.

Feature engineering and data pre-processing played critical roles in enhancing the predictive capabilities of the models. Categorical variables like genre and language were carefully encoded, and continuous variables were scaled and transformed to reduce skewness and improve model performance. Additionally, we addressed class imbalance to ensure that the models could predict movie popularity accurately across different categories.

Analysing the performance metrics revealed vital insights about the models. High precision in models like Logistic Regression indicated a low false positive rate, ensuring that when the model predicted a movie as popular, it was likely correct. High recall in models such as Random Forest suggested effectively capturing top-rated movies, minimising the risk of missing successful titles. These findings highlight the importance of balancing precision and recall in real-world applications where the cost of misclassification can be significant.

Examining R^2 and M.S.E. scores provided a further understanding of model fit. High R^2 values indicated good fit but did not guarantee overall model performance or prevent overfitting, while low M.S.E. values signified accurate predictions but were sensitive to outliers. This reinforces the need to carefully interpret results when applying machine learning to complex, real-world datasets.

Overall, the study demonstrated that combining multiple machine learning approaches and fine-tuning their parameters could enhance predictive accuracy, offering valuable strategies for stakeholders in the film industry to forecast movie success based on data-driven insights. Future work could expand on this by incorporating additional features and exploring advanced ensemble techniques to refine model performance.

CONCLUSION

This report delves into the relationship between various movie attributes and their influence on predicting movie popularity, using machine learning models to enhance forecasting accuracy. The data underwent thorough pre-processing, including converting categorical variables such as genres, languages, and production companies into numerical representations and applying transformations to reduce skewness in numerical features like budget and vote count. These steps ensured data consistency and improved the models' learning capabilities.

We evaluated several machine learning models, including Logistic Regression, Random Forest, and Naive Bayes for classification, as well as Decision Tree Regressor, AdaBoost, XGBoost, Support Vector Regressor (S.V.R.), and Multilinear Regression for regression tasks. Among these, Random Forest excelled in classification, while XGBoost proved most effective for regression, showcasing high predictive accuracy, robust R^2 scores, and low M.S.E. values. The superior performance of these models is attributed to their ability to manage complex interactions and large datasets, making them ideal for predicting movie popularity.

The results emphasise the critical role of model selection and the advantages of ensemble approaches, particularly when handling complex, real-world data. Random Forest and XGBoost demonstrated superior performance to individual models, underscoring the value of combining multiple algorithms to capture diverse data patterns. Precision and recall metrics provided insights into the balance between accurately identifying popular movies and minimising classification errors, highlighting the models' practical benefits in real-world applications.

The study also identified key factors influencing movie popularity, suggesting directions for future research. Additional variables, such as marketing impact, social media influence, and audience demographics, could enhance model accuracy and applicability. Exploring regional differences in popularity could also lead to the development of more tailored predictive models for specific markets.

In conclusion, this project advances the field of predictive modelling by demonstrating the effective use of machine learning in forecasting movie popularity based on diverse features. The findings highlight the potential of advanced analytical techniques in the entertainment industry, providing insights that can inform data-driven decision-making. By improving our understanding of factors that drive movie success, this work contributes to optimising content creation and distribution strategies, ultimately supporting the broader goal of leveraging data analytics to navigate the complexities of the film industry.

REFERENCES

1. Shahid, M. H., & Islam, M. A. (2023). Investigation of time series-based genre popularity features for box office success prediction. PeerJ Computer Science, 9, e1603. https://doi.org/10.7717/peerj-cs.1603

2. Masih, S., & Ihsan, I. (2019). Using Academy Awards to predict success of Bollywood movies using machine learning algorithms. International Journal of Advanced Computer Science and Applications, 10(2). https://doi.org/10.14569/IJACSA.2019.0100281

3. Verma, H., & Verma, G. (2019). Prediction model for Bollywood movie success: A comparative analysis of the performance of supervised machine learning algorithms. The Review of Socionetwork Strategies, 13, 1-23. https://doi.org/10.1007/s12626-019-00040-6

4. Donega e Souza, T. L., Nishijima, M., & Pires, R. (2023). Revisiting predictions of movie economic success: Random forest applied to profits. Multimedia Tools and Applications, 82(83), 38397–38420. https://doi.org/10.1007/s11042-023-15169-4

5. Sahu, S., Kumar, R., Pathan, M. S., Shafi, J., Kumar, Y., & Ijaz, M. F. (2022). Movie popularity and target audience prediction using the content-based recommender system. IEEE Access, 10, 42044-42060. https://doi.org/10.1109/ACCESS.2022.3168161

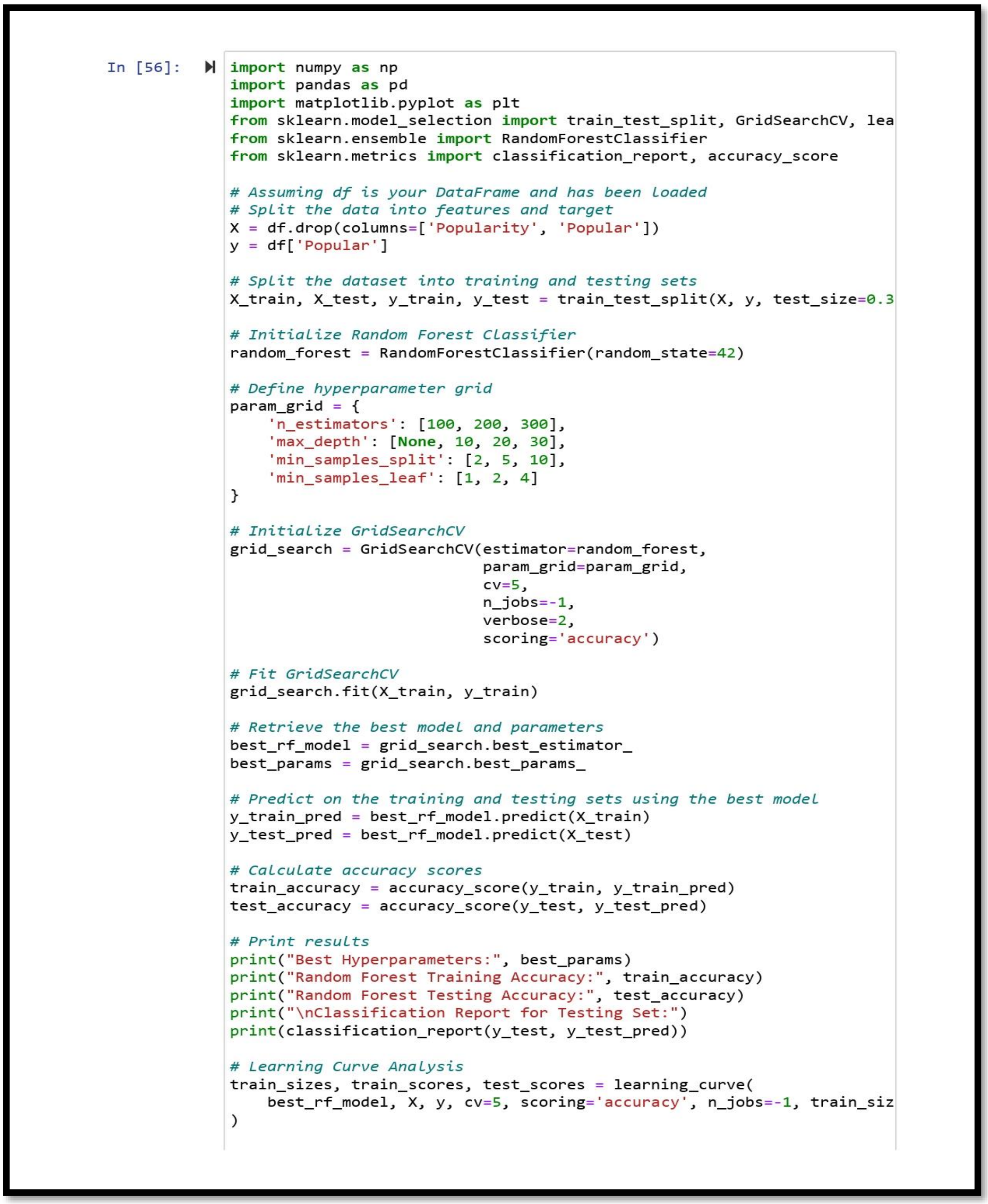
6. Bristi, W. R., Zaman, Z., & Sultana, N. (2019). Predicting IMDb rating of movies by machine learning techniques. Proceedings of the 2019 International Conference on Computing, Communication and Networking Technologies (I.C.C.C.N.T.). https://doi.org/10.1109/ICCCNT45670.2019.8944604

7. Agarwal, M., Venugopal, S., Kashyap, R., & Bharathi, R. (2022). Movie success prediction and performance comparison using various statistical approaches. International Journal of Artificial Intelligence and Applications, 13(1), 1-18. https://doi.org/10.5121/ijaia.2022.13102

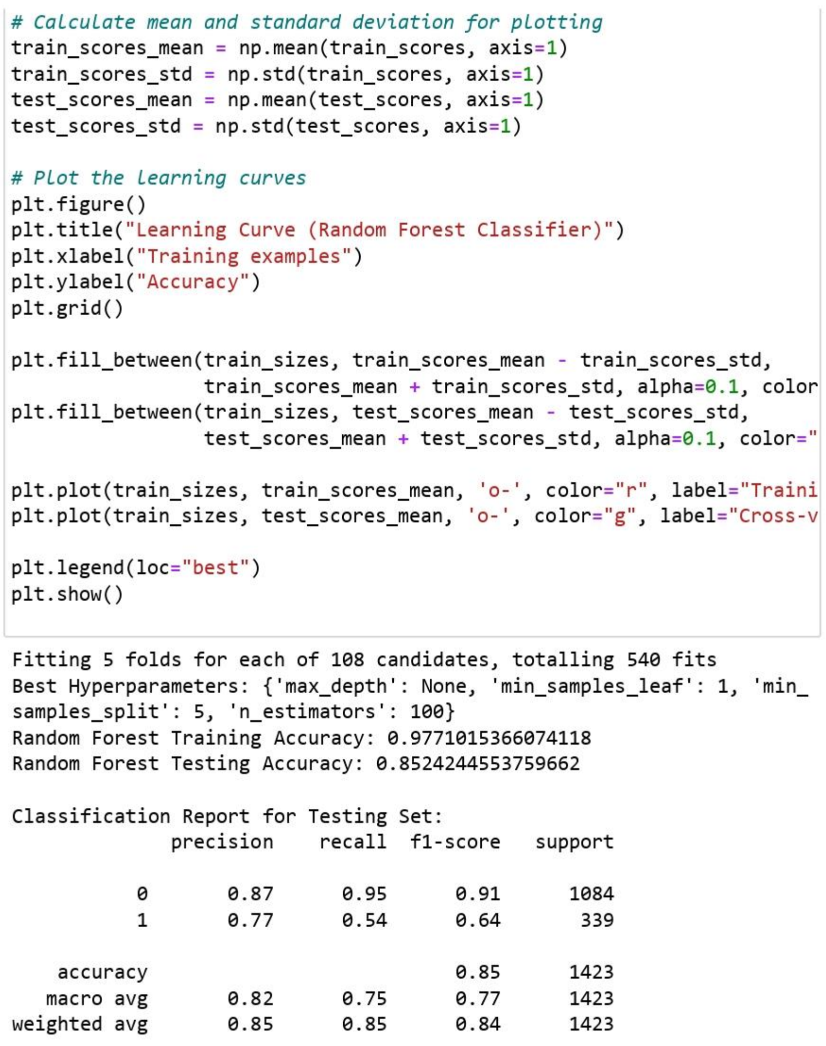
8. Oyewola, D. O., & Dada, E. G. (2022). Machine learning methods for predicting the popularity of movies. Journal of Artificial Intelligence and Systems, 4, 65-82. https://doi.org/10.33969/AIS.2022040105

9. Anand, A. (2023). Predicting the success of a movie using machine learning algorithms: An analysis. International Journal for Multidisciplinary Research (I.J.F.M.R.), 5(6), 1-8.

APPENDIX

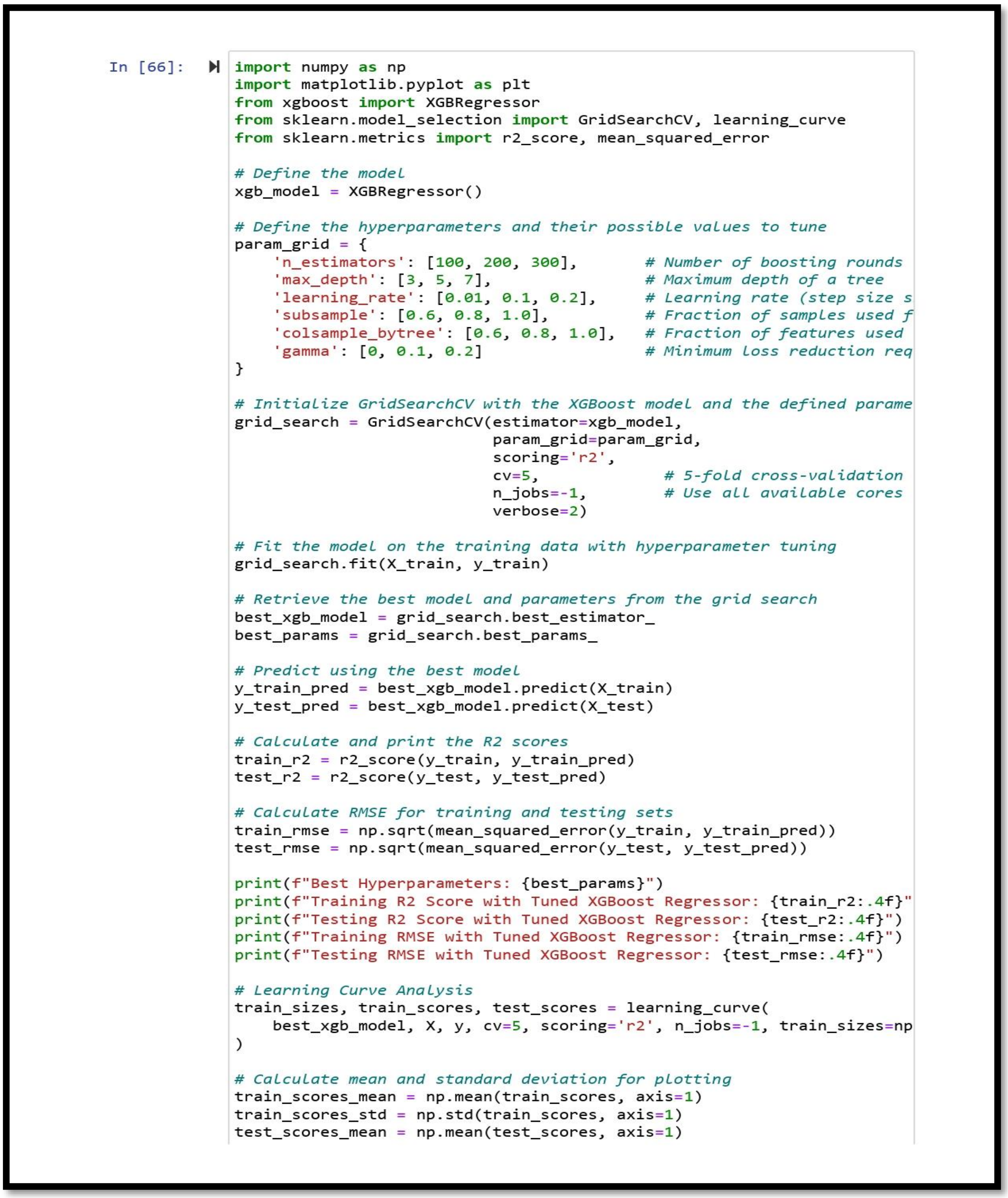


```
In [56]:  import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          from sklearn.model_selection import train_test_split, GridSearchCV, lea
          from sklearn.ensemble import RandomForestClassifier
          from sklearn.metrics import classification_report, accuracy_score

          # Assuming df is your DataFrame and has been loaded
          # Split the data into features and target
          X = df.drop(columns=['Popularity', 'Popular'])
          y = df['Popular']

          # Split the dataset into training and testing sets
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3

          # Initialize Random Forest Classifier
          random_forest = RandomForestClassifier(random_state=42)

          # Define hyperparameter grid
          param_grid = {
              'n_estimators': [100, 200, 300],
              'max_depth': [None, 10, 20, 30],
              'min_samples_split': [2, 5, 10],
              'min_samples_leaf': [1, 2, 4]
          }

          # Initialize GridSearchCV
          grid_search = GridSearchCV(estimator=random_forest,
                                     param_grid=param_grid,
                                     cv=5,
                                     n_jobs=-1,
                                     verbose=2,
                                     scoring='accuracy')

          # Fit GridSearchCV
          grid_search.fit(X_train, y_train)

          # Retrieve the best model and parameters
          best_rf_model = grid_search.best_estimator_
          best_params = grid_search.best_params_

          # Predict on the training and testing sets using the best model
          y_train_pred = best_rf_model.predict(X_train)
          y_test_pred = best_rf_model.predict(X_test)

          # Calculate accuracy scores
          train_accuracy = accuracy_score(y_train, y_train_pred)
          test_accuracy = accuracy_score(y_test, y_test_pred)

          # Print results
          print("Best Hyperparameters:", best_params)
          print("Random Forest Training Accuracy:", train_accuracy)
          print("Random Forest Testing Accuracy:", test_accuracy)
          print("\nClassification Report for Testing Set:")
          print(classification_report(y_test, y_test_pred))

          # Learning Curve Analysis
          train_sizes, train_scores, test_scores = learning_curve(
              best_rf_model, X, y, cv=5, scoring='accuracy', n_jobs=-1, train_siz
          )
```



```python
# Calculate mean and standard deviation for plotting
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

# Plot the learning curves
plt.figure()
plt.title("Learning Curve (Random Forest Classifier)")
plt.xlabel("Training examples")
plt.ylabel("Accuracy")
plt.grid()

plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                 train_scores_mean + train_scores_std, alpha=0.1, color
plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.1, color="

plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Traini
plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-v

plt.legend(loc="best")
plt.show()
```

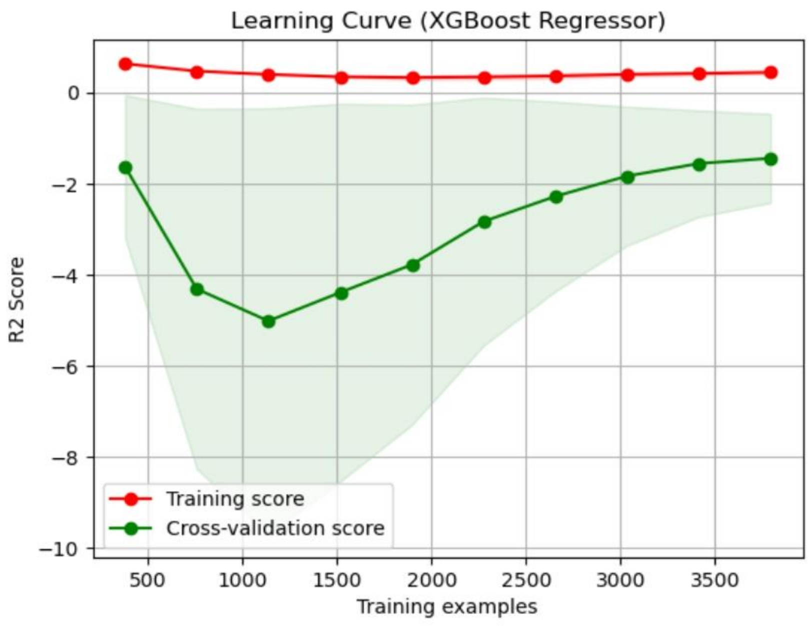
```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best Hyperparameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_
samples_split': 5, 'n_estimators': 100}
Random Forest Training Accuracy: 0.9771015366074118
Random Forest Testing Accuracy: 0.8524244553759662

Classification Report for Testing Set:
              precision    recall  f1-score   support

           0       0.87      0.95      0.91      1084
           1       0.77      0.54      0.64       339

    accuracy                           0.85      1423
   macro avg       0.82      0.75      0.77      1423
weighted avg       0.85      0.85      0.84      1423
```



```
In [66]:    import numpy as np
            import matplotlib.pyplot as plt
            from xgboost import XGBRegressor
            from sklearn.model_selection import GridSearchCV, learning_curve
            from sklearn.metrics import r2_score, mean_squared_error

            # Define the model
            xgb_model = XGBRegressor()

            # Define the hyperparameters and their possible values to tune
            param_grid = {
                'n_estimators': [100, 200, 300],       # Number of boosting rounds
                'max_depth': [3, 5, 7],                # Maximum depth of a tree
                'learning_rate': [0.01, 0.1, 0.2],     # Learning rate (step size s
                'subsample': [0.6, 0.8, 1.0],          # Fraction of samples used f
                'colsample_bytree': [0.6, 0.8, 1.0],   # Fraction of features used
                'gamma': [0, 0.1, 0.2]                 # Minimum loss reduction req
            }

            # Initialize GridSearchCV with the XGBoost model and the defined parame
            grid_search = GridSearchCV(estimator=xgb_model,
                                       param_grid=param_grid,
                                       scoring='r2',
                                       cv=5,               # 5-fold cross-validation
                                       n_jobs=-1,          # Use all available cores
                                       verbose=2)

            # Fit the model on the training data with hyperparameter tuning
            grid_search.fit(X_train, y_train)

            # Retrieve the best model and parameters from the grid search
            best_xgb_model = grid_search.best_estimator_
            best_params = grid_search.best_params_

            # Predict using the best model
            y_train_pred = best_xgb_model.predict(X_train)
            y_test_pred = best_xgb_model.predict(X_test)

            # Calculate and print the R2 scores
            train_r2 = r2_score(y_train, y_train_pred)
            test_r2 = r2_score(y_test, y_test_pred)

            # Calculate RMSE for training and testing sets
            train_rmse = np.sqrt(mean_squared_error(y_train, y_train_pred))
            test_rmse = np.sqrt(mean_squared_error(y_test, y_test_pred))

            print(f"Best Hyperparameters: {best_params}")
            print(f"Training R2 Score with Tuned XGBoost Regressor: {train_r2:.4f}"
            print(f"Testing R2 Score with Tuned XGBoost Regressor: {test_r2:.4f}")
            print(f"Training RMSE with Tuned XGBoost Regressor: {train_rmse:.4f}")
            print(f"Testing RMSE with Tuned XGBoost Regressor: {test_rmse:.4f}")

            # Learning Curve Analysis
            train_sizes, train_scores, test_scores = learning_curve(
                best_xgb_model, X, y, cv=5, scoring='r2', n_jobs=-1, train_sizes=np
            )

            # Calculate mean and standard deviation for plotting
            train_scores_mean = np.mean(train_scores, axis=1)
            train_scores_std = np.std(train_scores, axis=1)
            test_scores_mean = np.mean(test_scores, axis=1)
```

```
test_scores_std = np.std(test_scores, axis=1)

# Plot the learning curves
plt.figure()
plt.title("Learning Curve (XGBoost Regressor)")
plt.xlabel("Training examples")
plt.ylabel("R2 Score")
plt.grid()

plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                 train_scores_mean + train_scores_std, alpha=0.1, color
plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.1, color="

plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Traini
plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-v

plt.legend(loc="best")
plt.show()
```

```
Fitting 5 folds for each of 729 candidates, totalling 3645 fits
Best Hyperparameters: {'colsample_bytree': 0.8, 'gamma': 0, 'learning_
rate': 0.01, 'max_depth': 5, 'n_estimators': 300, 'subsample': 0.6}
Training R2 Score with Tuned XGBoost Regressor: 0.4547
Testing R2 Score with Tuned XGBoost Regressor: 0.3554
Training RMSE with Tuned XGBoost Regressor: 0.0340
Testing RMSE with Tuned XGBoost Regressor: 0.0363
```



PROFORMA 4

Undertaking from the PG student while submitting his/her final dissertation to his respective institute

Ref. No. _________

I, the following student

Sr. No.	**Sequence of student's names on a dissertation**	**Students name**	**Name of the Institute & Place**	**Email & Mobile**
1.	First Author	Sharon Furtado	SIG	Email: 23070243015@sig.ac.in Mobile: +91 9594686544

Note: Put additional rows in case of more number of students

Hereby give an undertaking that the dissertation **Understanding Movie Popularity: A Machine Learning Perspective** has been checked for its Similarity Index/Plagiarism through the Turnitin software tool and that the document has been prepared by me, is my original work, and is free of any plagiarism. It was found that:

1.	The Similarity Index (SI) was: *(Note: SI range: 0 to 10%; if SI is >10%, then authors cannot communicate ms;* ***attachment of SI report is mandatory****)*	3%
2.	The ethical clearance for research work conducted was obtained from: *(Note: Name the consent obtaining body; if 'not applicable', then write so)*	NA
3.	The source of funding for research was: *(Note: Name the funding agency, or write 'self' if no funding source is involved)*	Self
4.	Conflict of interest: *(Note: Tick √ whichever is applicable)*	No
5.	The material (adopted text, tables, figures, graphs, etc.), as has been obtained from other sources, has been duly acknowledged in the manuscript: *(Note: Tick √ whichever is applicable)*	Yes

In case if any of the above-furnished information is found false at any point in time, then the University authorities can take action as deemed fit against all of us.

Full Name &
Signature of the student

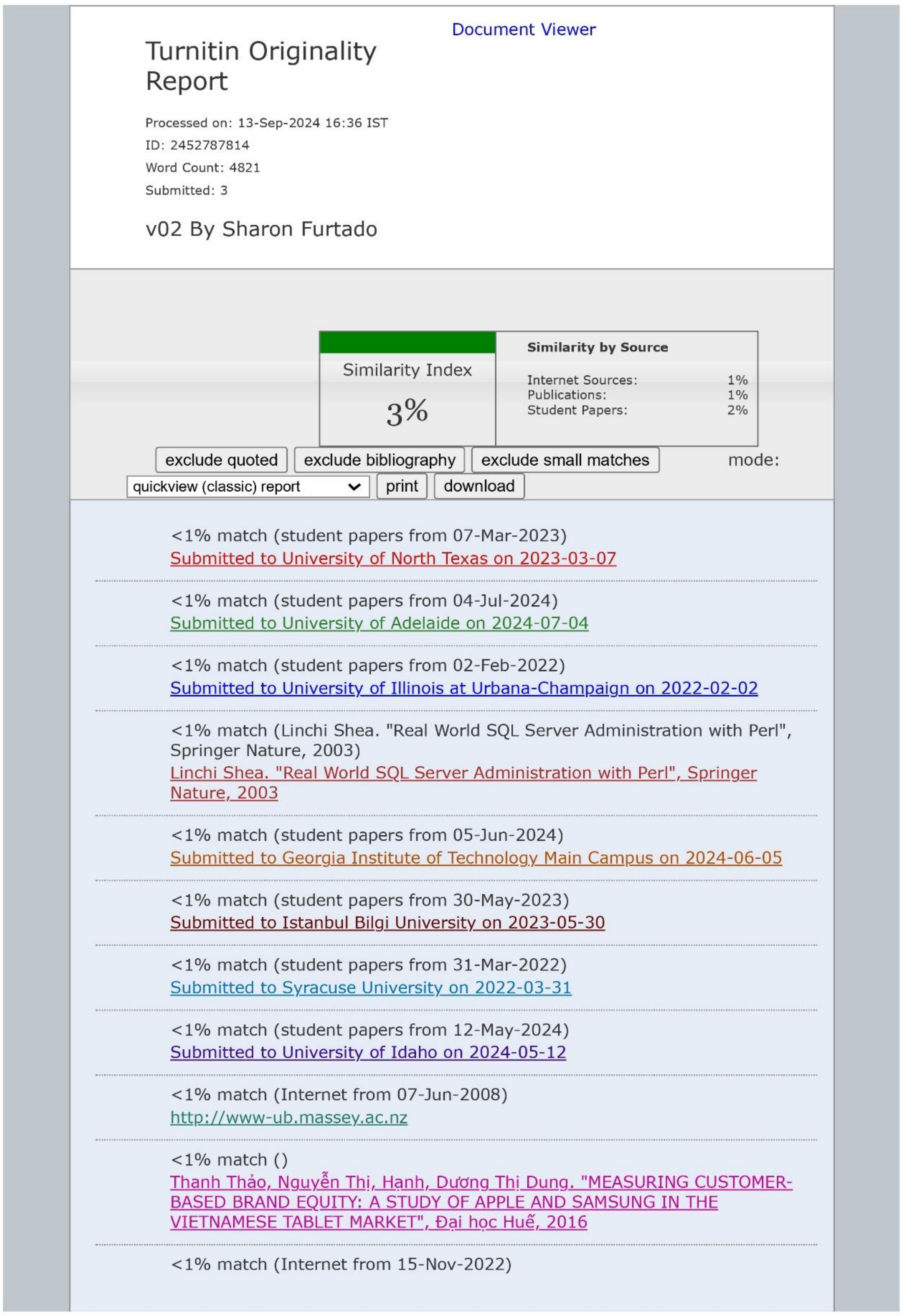
Name &
Signature of SIU Guide/Mentor

Date: 13th September 2024

Endorsement by
Academic Integrity Committee (AIC)

Place: Pune

Note: It is mandatory that the Similarity Index report of plagiarism (only first page) should be appended to the UG/PG dissertation.


Document Viewer

Turnitin Originality Report

Processed on: 13-Sep-2024 16:36 IST
ID: 2452787814
Word Count: 4821
Submitted: 3

v02 By Sharon Furtado

Similarity Index

3%

Similarity by Source

Internet Sources:	1%
Publications:	1%
Student Papers:	2%

<1% match (student papers from 07-Mar-2023)
Submitted to University of North Texas on 2023-03-07

<1% match (student papers from 04-Jul-2024)
Submitted to University of Adelaide on 2024-07-04

<1% match (student papers from 02-Feb-2022)
Submitted to University of Illinois at Urbana-Champaign on 2022-02-02

<1% match (Linchi Shea. "Real World SQL Server Administration with Perl", Springer Nature, 2003)
Linchi Shea. "Real World SQL Server Administration with Perl", Springer Nature, 2003

<1% match (student papers from 05-Jun-2024)
Submitted to Georgia Institute of Technology Main Campus on 2024-06-05

<1% match (student papers from 30-May-2023)
Submitted to Istanbul Bilgi University on 2023-05-30

<1% match (student papers from 31-Mar-2022)
Submitted to Syracuse University on 2022-03-31

<1% match (student papers from 12-May-2024)
Submitted to University of Idaho on 2024-05-12

<1% match (Internet from 07-Jun-2008)
http://www-ub.massey.ac.nz

<1% match ()
Thanh Thảo, Nguyễn Thị, Hạnh, Dương Thị Dung. "MEASURING CUSTOMER-BASED BRAND EQUITY: A STUDY OF APPLE AND SAMSUNG IN THE VIETNAMESE TABLET MARKET", Đại học Huế, 2016

<1% match (Internet from 15-Nov-2022)