HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

PROJECT REPORT

MOVIE EVALUATION

Group 32

Hoang Trung Khai	20225502
Trinh Duy Phong	20220065
Vu Viet Long	20225508
Luu Hoang Phan	20225516
Nguyen Phan Thang	20225529
Pham Minh Tien	20225555

Supervisor: Assoc. Prof. Than Quang Khoat

Course: Machine Learning – IT3190E

HANOI, 05/2024

TABLE OF CONTENT

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

Predicting the revenue of a movie is a critical aspect of the film industry, influencing decisions from production to marketing. Accurate revenue predictions can guide filmmakers, producers, and investors in resource allocation, risk assessment, and strategic planning. This report delves into the methodologies and challenges associated with forecasting movie revenues, leveraging data analytics and machine learning techniques to enhance prediction accuracy.

The motivation behind this project stems from the high stakes involved in movie production and distribution. The film industry is characterized by significant investments, with substantial financial risks and rewards. Traditional methods of revenue prediction often rely on intuition and historical data, which may not adequately capture the complexities of modern cinema. By utilizing advanced data-driven approaches, this project aims to provide a more reliable and nuanced revenue prediction model, thereby aiding stakeholders in making informed decisions and maximizing their return on investment.

Predicting movie revenue is fraught with challenges due to the myriad factors that can influence a film's financial performance. These factors include genre, cast, director, release timing, marketing efforts, competition, and critical reception, among others. Additionally, external variables such as economic conditions and social trends can also impact box office earnings. The main problems to address in this project are data variability, as movies are influenced by a wide range of factors, making it difficult to isolate variables that consistently predict revenue; non-linear relationships, since the relationship between predictors and revenue is often non-linear and complex; data availability and quality, as accurate and comprehensive data is essential but often difficult to obtain; and dynamic market conditions, because the film industry is continuously evolving, and models need to adapt to changes in audience preferences and market dynamics.

The primary aim of this project is to develop a robust and accurate model for predicting movie revenue using advanced data analytics and machine learning techniques. Specific objectives include identifying key predictors to determine the most influential factors that affect movie revenue, gathering and preprocessing relevant data from various sources to ensure its accuracy and completeness,

employing machine learning algorithms to create predictive models and testing various approaches to identify the most effective one, rigorously evaluating the model's performance using historical data to adjust for accuracy and reliability, and providing a user-friendly tool or framework that industry stakeholders can use to forecast movie revenues. By achieving these aims, this project seeks to bridge the gap between traditional revenue prediction methods and modern data-driven approaches, offering a valuable resource for the film industry. Through improved prediction accuracy, stakeholders can better navigate the complexities of movie production and distribution, ultimately contributing to the financial success and sustainability of the industry.

2. METHODOLOGY

2.1. Work Flow

The first phase is data acquisition. This is the initial step where data is collected from various sources such as databases, APIs, web scraping, or direct input. During our project, the data source chosen is TMDB – The Movie Database. The goal is to gather relevant information for the analysis or project ar hand.

Once the data is collected, it often needs to be cleaned to ensure accuracy and consistency. This involves identifying and correcting errors, handling missing or incomplete data, removing duplicates, and standardizing formats.

After cleaning, the data may need to be transformed or manipulated to make it suitable for analysis. This can include converting data types or standardize the data.

In the next phase, Features Extraction, relevant features or attributes are extracted from the dataset to be used for analysis or modeling. It involves selecting the most informative variables that will help achieve the project's objectives. And then, with the prepared dataset and extracted features, various analytical techniques are applied to uncover insights, patterns, or relationships within the data. Where we applyied machine learning algorithms on our dataset.

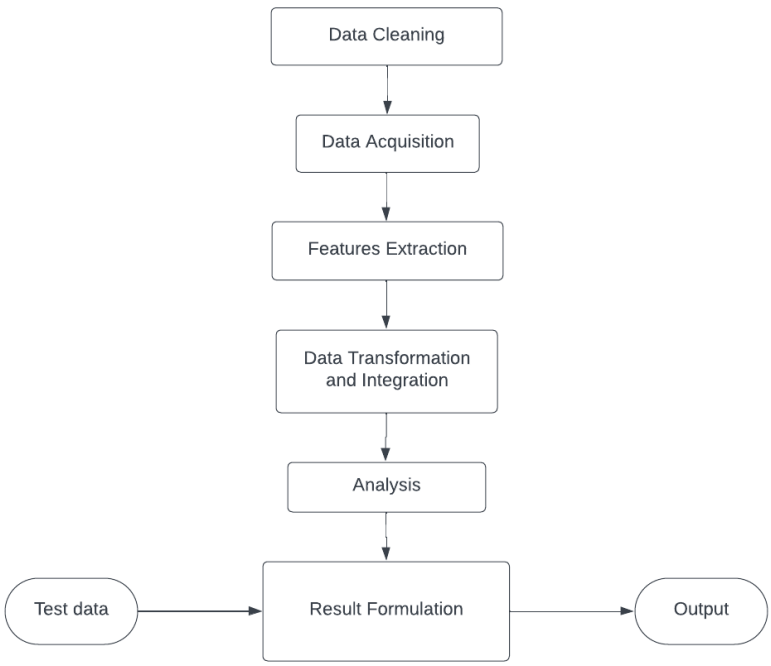


Figure 2.1. Work Flow

2.2. Algorithms

2.2.1. Grid Search

Grid search is a fundamental technique in machine learning for hyperparameter optimization, a critical aspect of model refinement. Hyperparameters are parameters that are set before the learning process begins, affecting the model's behavior and performance. Finding the optimal values for these hyperparameters is essential for maximizing a model's effectiveness.

Grid search systematically explores a predefined set of hyperparameter values for a given machine learning algorithm. It constructs a "grid" of all possible combinations of hyperparameters and evaluates each combination using a specified performance metric, such as accuracy, precision, or mean squared error. The combination that yields the best performance on a validation set is selected as the optimal configuration.

2.2.2. Random Search

Random search is a popular technique for hyperparameter optimization in machine learning, offering an alternative to exhaustive methods like grid search. Hyperparameter optimization involves finding the best combination of hyperparameters for a given machine learning algorithm, crucial for maximizing model performance.

Unlike grid search, which systematically explores a predefined grid of hyperparameter values, random search samples hyperparameter values randomly from specified distributions. This random sampling approach has several advantages and can often yield comparable or even better results with fewer evaluations.

The combination of Grid Search and Random Search effectively addresses the limitations inherent in each technique. Random Search's stochastic nature and Grid Search's systematic exploration of parameter ranges complement each other's strengths. In our approach, for each Machine Learning algorithm, we begin by employing Grid Search to exhaustively explore all feasible hyperparameter combinations. Subsequently, we leverage Random Search to refine the search space and attain optimal results.

2.2.3. SVR

Overview

Ridge regression, also known as Tikhonov regularization, is a technique used to analyze multiple regression data that suffer from multicollinearity. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value. Ridge regression adds a degree of bias to the regression estimates, which reduces the standard errors.

Multicollinearity

Multicollinearity occurs when predictor variables in a regression model are highly correlated. This can cause issues with the stability and interpretation of the regression coefficients.

Regularization

Regularization is a technique used to introduce additional information to prevent overfitting. Ridge regression is a form of regularization.

Ridge Regression Formula

The ridge regression estimator is given by:

$$\beta ridge = (XTX+\lambda I)\text{-}1XTy$$

Here, X is the matrix of predictor variables, y is the vector of response variables, I is the identity matrix, and λ (lambda) is the regularization parameter.

Regularization Parameter (λ):

λ controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage, and thus the coefficients become more biased but have lower variance.

Objective Function

The objective function in ridge regression is:

$$min(||y\text{-}X\beta||2 + \lambda||\beta||2$$

The first term represents the residual sum of squares, and the second term is the penalty (or shrinkage term).

Hyperparameter tuning

Hyperparameter tuning for a ridge regression model involves selecting the optimal value for the regularization parameter λ (also known as alpha in many libraries) to minimize the prediction error. This process typically uses techniques

like cross-validation to evaluate model performance across different values of λ. This helps improve the performance and generalizability.

2.2.4. Ridge Regression

Overview

Support Vector Regression (SVR) is a type of support vector machine (SVM) that is specifically used for regression tasks. Similar to linear SVMs, SVR identifies a hyperplane with the maximum margin between data points. SVR aims to find a function that approximates the relationship between input variables and a continuous target variable while minimizing prediction error. SVR can use both linear and non-linear kernels to model complex relationships.

Hyperparameter tuning

Hyperparameter tuning in Support Vector Regression (SVR) is crucial for achieving optimal performance. The main hyperparameters that need to be tuned in SVR are:

- C (Regularization parameter): This parameter controls the trade-off between achieving a low error on the training data and minimizing the model complexity. A small C value makes the decision surface smooth, while a large C aims to classify all training examples correctly.
- Epsilon (ε): This parameter defines the margin of tolerance where no penalty is given to errors. It specifies the width of the epsilon-tube within which no penalty is associated in the training loss function. Larger values of ϵ result in fewer support vectors.
- Kernel Parameters: Depending on the kernel chosen (e.g., linear, polynomial, radial basis function), each kernel has its own set of parameters that may need tuning. Common kernel parameters include:
    - Kernel type: The choice of kernel function (e.g., linear, polynomial, RBF).

- Gamma (γ): This parameter defines the influence of a single training example. Low values mean ‘far’ and high values mean ‘close’. It is a parameter for non-linear hyperplanes.
- Degree: For polynomial kernels, this parameter represents the degree of the polynomial function.

SVR Algorithm: The SVR (Support Vector Regression) algorithm aims to find a function f(x) that has at most ϵ deviation from the actual targets for all training data points, while also ensuring that the function is as flat as possible.

Input:

- A set of n examples forming the Training Set.

$$TSn=\{(x1,y1),(x2,y2),\ldots,(xn,yn)\}$$

- A kernel function $K(xi,xj)$
- Regularization parameter C.
- Epsilon parameter ϵ.

Initialize: Compute the kernel matrix K using the chosen kernel function.

Prediction:

For a new data point x:

$$f\ (x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i) K(x_i, x) + b$$

Where b is determined from the support vectors.

2.2.5. Decision Tree Regression

Overview

Decision tree regression is a machine learning method that builds regression models in the form of a tree structure. Unlike classification trees that predict categorical outcomes, decision tree regression predicts continuous outcomes by partitioning the feature space into regions and assigning a constant value to each region.

Building the Tree

The process begins with the selection of a feature that best splits the data into two subsets, aiming to minimize the variance of the target variable within each subset. This splitting continues recursively until a stopping criterion is met, such

as reaching a maximum tree depth or having a minimum number of samples in each leaf node.

Predicting Continuous Values

To predict the outcome for a new data point, the algorithm traverses the tree from the root node to a leaf node, assigning the constant value associated with that leaf node as the predicted value for the input data.

Hyperparameters in Decision Tree Regression

- 'max_depth': Controls the maximum depth of the tree. Limiting the depth can prevent overfitting. In this model, we selected these values: [5, 10, 15, 20, 25].
- 'min_samples_split': Minimum number of samples required to split an internal node. Higher values can lead to a more generalized model. In this we selected these values: [2, 5, 10].
- 'min_samples_leaf': Minimum number of samples required to be at a leaf node. This parameter prevents the model from learning overly specific patterns. In this we selected these values: [1, 2, 5].

2.2.6. Random Forest Regression

Overview

Random Forest Regression is an ensemble learning method that builds upon the principles of decision tree regression. It is designed to mitigate the overfitting tendencies of individual decision trees while maintaining their predictive power.

Ensemble of Decision Trees

In Random Forest Regression, multiple decision trees are trained on different subsets of the training data and using different subsets of features. This ensemble approach helps to reduce variance and increase the generalization performance of the model.

Bootstrap Aggregating (Bagging)

Random Forest employs a technique called Bootstrap Aggregating, or bagging, to create diverse subsets of the training data. Each decision tree is trained on a bootstrapped sample of the original data, where samples are drawn with

replacement. This results in each tree being exposed to slightly different variations of the data.

Combining Predictions

Once all decision trees are trained, predictions from each tree are combined through averaging (for regression tasks) to produce the final ensemble prediction. This averaging process helps to smooth out the predictions and reduce the impact of outliers or noise in the data.

Hyperparameters in Random Forest Regression

The performance of a Random Forest Regression model is also highly dependent on its hyperparameters. The following parameters were tuned in this project:

- 'n_estimators': Number of trees in the forest. More trees can improve performance but increase computational cost. In this we selected these values: [100, 200 , 300].
- 'max_depth': Maximum depth of each tree. Limiting depth helps to prevent overfitting. In this we selected these values: [5,10,15, 20].
- 'min_samples_split': Minimum number of samples required to split an internal node. This helps to control the depth of the tree. In this we selected these values: [2, 5].
- 'min_samples_leaf': Minimum number of samples required to be at a leaf node. This parameter ensures that each leaf has a sufficient number of samples. In this we selected these values: [1, 2].

2.2.7. Gradient Boosted Decision Trees

Overview

Gradient Boosting Decision Trees (GBDT) is a machine learning method belonging to the ensemble learning class, specifically boosting, which is a method in machine learning aimed at combining multiple weak models into a stronger one. During boosting, weak models are built sequentially, with each new model attempting to optimize the residual of the previous model.

Base Learner: Decision Trees

Decision trees are often chosen as the base learner or weak model in GBDT. These decision trees are relatively simple models that make predictions based on a series of binary decisions. Each decision tree is a weak learner because it tends to have high variance and low bias, meaning it might not generalize well to unseen data on its own.

Training Process in Gradient Boosting

In gradient boosting, at each step, a new weak model is trained to predict the "error" of the current strong model (which is called the pseudo response). We will detail "error" later. For now, assume "error" is the difference between the prediction and a regressive label. The weak model (that is, the "error") is then added to the strong model with a negative sign to reduce the error of the strong model.

Gradient boosting is iterative. Each iteration invokes the following formula:

$$Fi+1 = Fi – fj$$

where:

- F_i is the strong model at step i
- f_j is the weak model at step i

This operation repeats until a stopping criterion is met, such as a maximum number of iterations or if the (strong) model begins to overfit as measured on a separate validation dataset.

Hyperparameters in GBDT

There are 7 hyperparameters used to optimize our GBDT algorithm:

- 'learning_rate': This parameter controls the contribution of each tree in the ensemble. A smaller learning rate makes the model training more gradual, preventing overfitting but requiring more iterations.
- 'n_estimators': This represents the number of trees in the ensemble.
- 'max_depth': It determines the maximum depth of each decision tree in the ensemble.
- 'min_samples_split': It sets the minimum number of samples required to split an internal node during tree building.
- 'min_samples_leaf': This parameter sets the minimum number of samples required to be at a leaf node.

- 'max_features': It determines the number of features to consider when looking for the best split at each node. There are 3 options: ‘sqrt’, ‘log2’ and ‘None’.
- 'subsample': This parameter specifies the fraction of samples to be used for fitting the individual base learners.

2.2.8. XGBoost

Overview

XGBoost, or Extreme Gradient Boosting, is a powerful and versatile machine learning algorithm that has gained widespread popularity due to its exceptional performance across various tasks. The methodology of XGBoost builds upon the principles of gradient boosting, introducing several innovative techniques and optimizations to enhance its predictive capabilities. Both XGBoost and GBDT optimize the model by reducing the gradient of the loss function.

Hyperparameters in GBDT

Beside the same hyperparameters used in GBDT: ‘learning_rate’, ‘n_estimators’, ‘max_depth’, ’subsample’, XGBoost includes several unique hyperparameters that contribute to its effectiveness:

- 'colsample_bytree': This parameter allows for random subsampling of features at each tree-building step, contributing to improved model generalization.
- 'gamma': Represents the minimum loss reduction required to make a further partition on a leaf node of the tree.
- 'reg_alpha': Lasso regularization – which shrinks the magnitude of weights and induces sparsity in the feature space, reducing model complexity.
- 'reg_lambda': Ridge regularization - Similar to 'reg_alpha', but penalizes the square of the weights, encouraging smaller weight values and further reducing model complexity.

3. DATA DESCRIPTION

3.1. Data Acquisition

The dataset is sourced from TMDB (The Movie Database), a popular and user-editable database for movies and TV shows. This site was chosen because it offers all the essential features required for our project's goal of predicting box office revenue. Each movie in the database has a unique ID, which allows us to easily access comprehensive information collected about that movie. This makes TMDB an ideal source for gathering the detailed data necessary for accurate revenue prediction.

Table 3.1. Dataset Summary

Feature	Type	Count	Mean	Median	Min	Max	Std. Dev
Duration	Integer	6234	107.86	104	3	310	21.98
User score	Integer	6234	63.41	64	0	100	10.50
Number of Votes	Integer	6234	1561.44	516	0	35686	2898.90
Restriction	Integer	6234	3.61	4	0	5	1.53
Year	Integer	6234	1995.62	2000	1915	2023	17.03
Month	Integer	6234	6.98	7	1	12	3.42
Budget	Integer	6234	23623210	12000000	1	379000000	32986370
Revenue	Integer	6234	63672320	17969840	1	1362000000	126067600
Actor Star Power	Float	6234	576491800	450849500	1	5554789000	638543300
Director Star Power	Float	6234	437609200	102486900	1	9105032000	894363400

3.2. Data Cleaning

Initially, the dataset contained 91,463 movies. However, since the focus of the project is on revenue prediction, we decided to exclude all movies without information on budget and revenue. This reduced the dataset to 6,241 movies.

3.3. Features Extraction

There are two types of features considered in this project: pre-release features and post-release features. Pre-release features are available before a movie is released and are used to predict its success. These include the budget, rating, cast (actors/actresses), director, and release date. Post-release features, which become available after the movie's release, can improve prediction accuracy, normally use to . These include user ratings and the number of votes. Additionally, post-release features help determine whether a film meets its expected performance.

3.3.1. Release Date

The year and month on revenue and budget is an important aspect to consider in the analysis. The year of release can influence factors such as market trends, audience preferences, and overall economic conditions, all of which can impact a movie's revenue and budget. Additionally, the month of release may coincide with specific seasons, holidays, or blockbuster periods, which can also influence revenue and budget allocation.

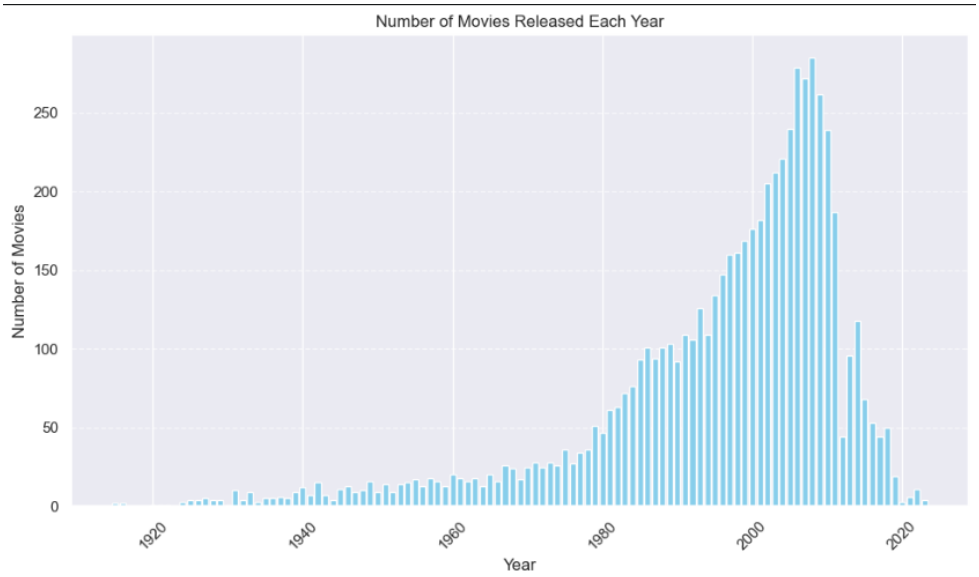


Figure 3.1. Number of Movies released each year

3.3.2. Star Power

This paper delves into the significance of star power in shaping the success of movies, focusing on actors/actresses and directors. Recognizable figures in the industry often bring a considerable following to their projects. For instance, renowned actors like Tom Hanks and esteemed directors like Christopher Nolan command significant attention due to their track record of successful movies.

In this context, a star is defined as an actor/actress whose previous movies have achieved notable box office success, indicating their established appeal among audiences. By aggregating the total earnings from all movies in an individual's career, this paper quantifies their star power. Essentially, the more successful projects associated with an actor/actress/director, the higher their perceived popularity and potential for future success.

3.3.3. Budget

Budget is a key factor in predicting a movie's success before its release. Movies with bigger budgets, covering both production and marketing, often generate more excitement and anticipation. Therefore, higher-budget movies often have a greater potential for earning higher revenues. Moreover, higher budgets often allow for the casting of more valuable and popular actors, which can further enhance the movie's appeal and its potential for success at the box office.

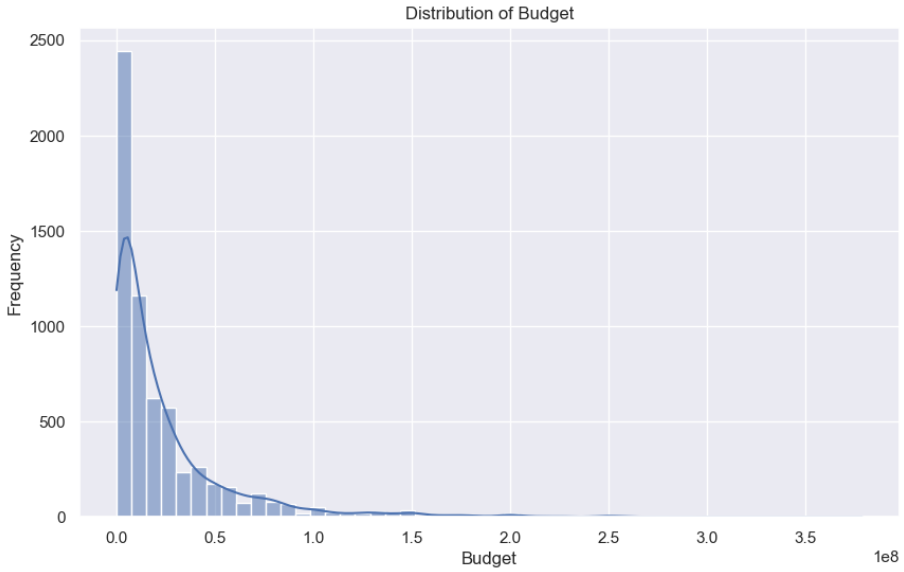


Figure 3.2. Distribution of Budget

As shown on the figure, it is right-skewed, which means there are a few movies with very high budgets compared to the majority of movies. The distribution shows that most movies have a budget of less than $100 million, while a few movies have budgets over $200 million. Undertaking a transformation before model training is indispensable.

3.3.4. Restrictions

This paper also examines the implications of content restrictions on the success of a movie. Content restrictions, such as age ratings and censorship guidelines, can influence the audience demographics and the movie's accessibility in various markets. A movie subject to less stringent content restrictions may have a broader appeal and a potentially larger audience base, which could positively impact its commercial performance. Therefore, understanding the impact of

content restrictions is essential in predicting a movie's box office performance accurately.

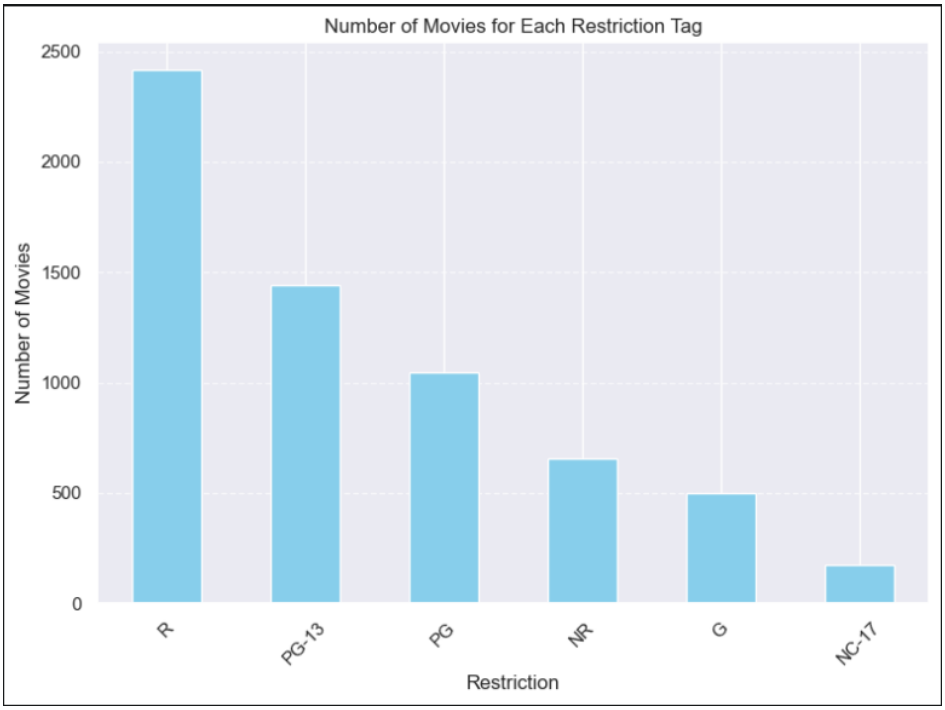


Figure 3.3. Number of Movies each Restriction Tag

3.3.5. Number of votes & User Score

User scores play a crucial role in assessing audience reception towards a movie. As scrapped, user votes typically range from integers 1 to 10, representing the rating given by individual viewers. To gauge the overall reception accurately, calculating the mean (average) of these user votes provides a comprehensive understanding of the movie's popularity among audiences. This mean score serves as a valuable metric for evaluating the movie's overall performance and audience satisfaction. The number of votes a movie receives reflects the level of audience engagement and interest in that particular film. A higher number of votes typically indicates a greater level of audience interaction and involvement with the movie. This metric is important because it provides insight into the movie's popularity and how widely it has been viewed and discussed among audiences. Additionally, a higher number of votes can contribute to a the assessment of the movie's overall reception and success.

3.3.6. Genres

Genres categorize movies into groups that reflect the primary focus or tone of the film. Understanding the genre of a movie is essential for both filmmakers and audiences alike, as it helps set expectations regarding the content, themes, and narrative style. Common genres include action, comedy, drama, horror, romance, science fiction, and thriller, among others. Each genre appeals to specific audiences with particular preferences, and the choice of genre can significantly influence a movie's reception, box office performance, and critical acclaim. Analyzing the genre distribution within a dataset allows for insights into audience preferences, trends in filmmaking, and can inform predictions about a movie's success based on its genre.

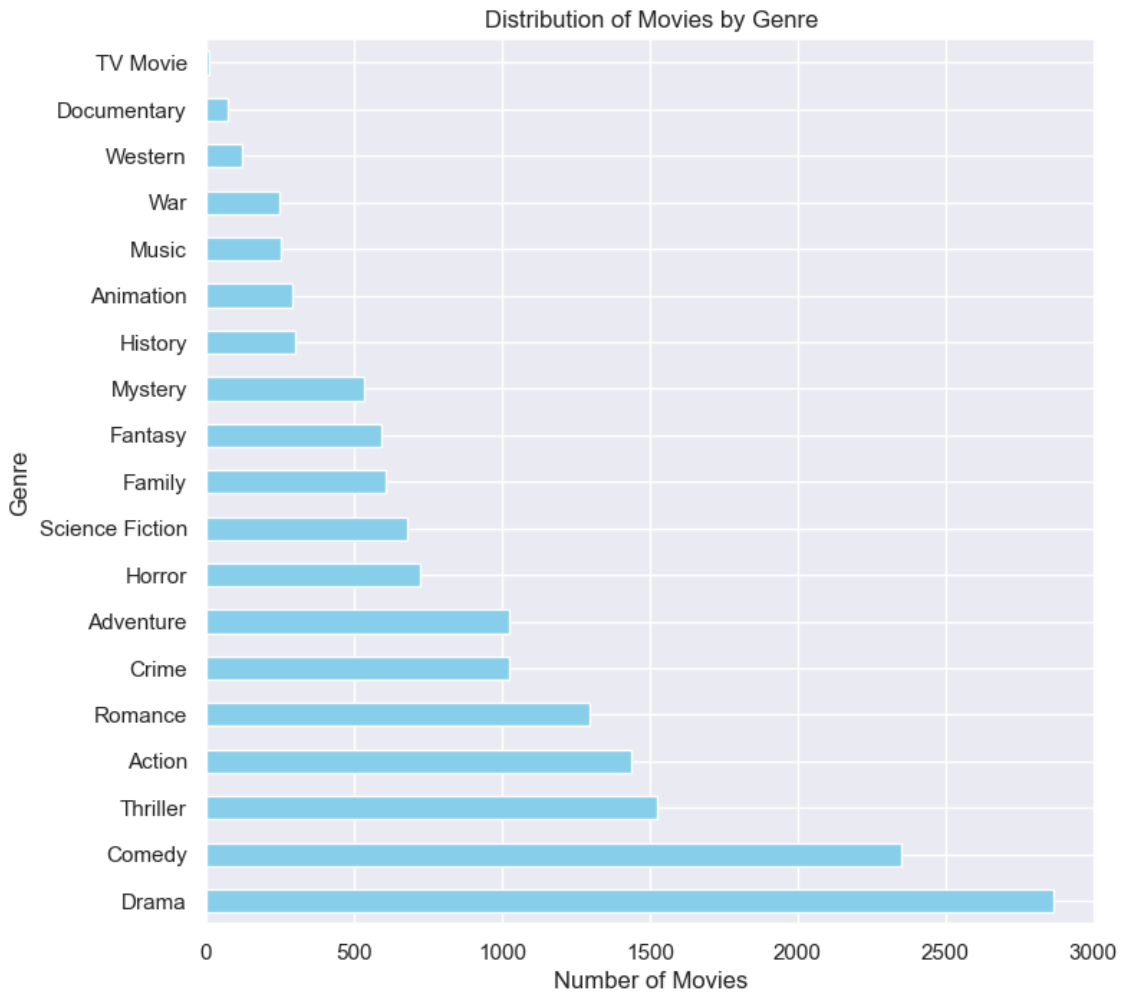


Figure 3.4. Distribution of Movies by Genre

3.3.7. Duration

The duration of a movie refers to its length in terms of running time, typically measured in minutes. Movie durations can vary widely, ranging from short films of just a few minutes to epic productions lasting several hours. Understanding the duration of a movie is important for both filmmakers and audiences, as it can impact factors such as pacing, storytelling depth, and audience engagement. Shorter films may be more suitable for certain genres or storytelling formats, such as animation or experimental cinema, while longer durations are often associated with epic narratives or in-depth character development. Additionally, movie theaters and streaming platforms may have specific constraints or preferences regarding the duration of films they distribute or showcase.

The duration of a movie can also influence its ticket price. Generally, longer movies may have higher ticket prices due to the perceived value of a more extended entertainment experience.

3.4. Data Integration and Transformation

In this project, the primary aim is around revenue of movies. Therefore, the integration process focuses on understanding how different features affect movie revenue. By integrating various features such as budget, cast, director, release date, genre, content restrictions, and user ratings, we aim to uncover the relationships between these factors and movie revenue.

3.4.1. Restriction

Movies originating from different nations often adhere to distinct standards and practices regarding content restriction. As a result, the level of restriction imposed on movies may differ significantly between nations. So, we must have a general Standard restriction for movies in our database. Specifically, we focus on standardizing the restriction ratings to the MPAA (Motion Picture Association of America) rating system and encoding them into numerical labels.

Firstly, we integrate the various restriction ratings present in the dataset into a standardized MPAA rating system. This involves mapping the existing ratings to their corresponding MPAA equivalents, ensuring consistency and clarity in the

dataset. Following integration, we employ label encoding to represent the MPAA ratings numerically. Each distinct MPAA rating is assigned a unique numerical label. We employ the LabelEncoding library of Scikit-learn's preprocessing module for this purpose. Table 2.1 includes the transformation of restrictions to numeric values.

Table 3.2. Restriction

MPAA Restriction	Label Encode
G	**0**
NC-17	**1**
NR	**2**
PG	**3**
PG-13	**4**
R	**5**

3.4.2. Release Date

This feature is seperated into 2 different, year and month. Since the release month represents a cyclical pattern, exploring its relationship with movie income enables us to identify any seasonal trends or variations in audience preferences. Otherwise, year represents some overarching patterns such as changes in audience preferences or shifts in industry dynamics.

3.4.3. Actors/Actresses & Directors Star Power

We calculate the star power of actors/actresses and directors separately, but both using a similar approach:

1. Calculate the total revenue of movies for each actor/actress or director: For each actor/actress, sum the revenue of all movies they have participated in. Similarly, we also calculate total director’s movie revenue.
2. Calculate the mean revenue of all actors/actresses or directors involved in the film. This provides the average revenue contribution of all actors/actresses or directors participating in the movie.

This approach allows us to quantify and compare the star power or influence of different individuals within the film industry, which may affect to the revenue of the movie they involved.

3.4.4. Revenue

As shown in table 2.1, revenue data spans a wide range of values, from very small to very large numbers (from 1 to over 1 billion). Taking the logarithm of the revenue values can help normalize the data, making it easier to compare and analyze.

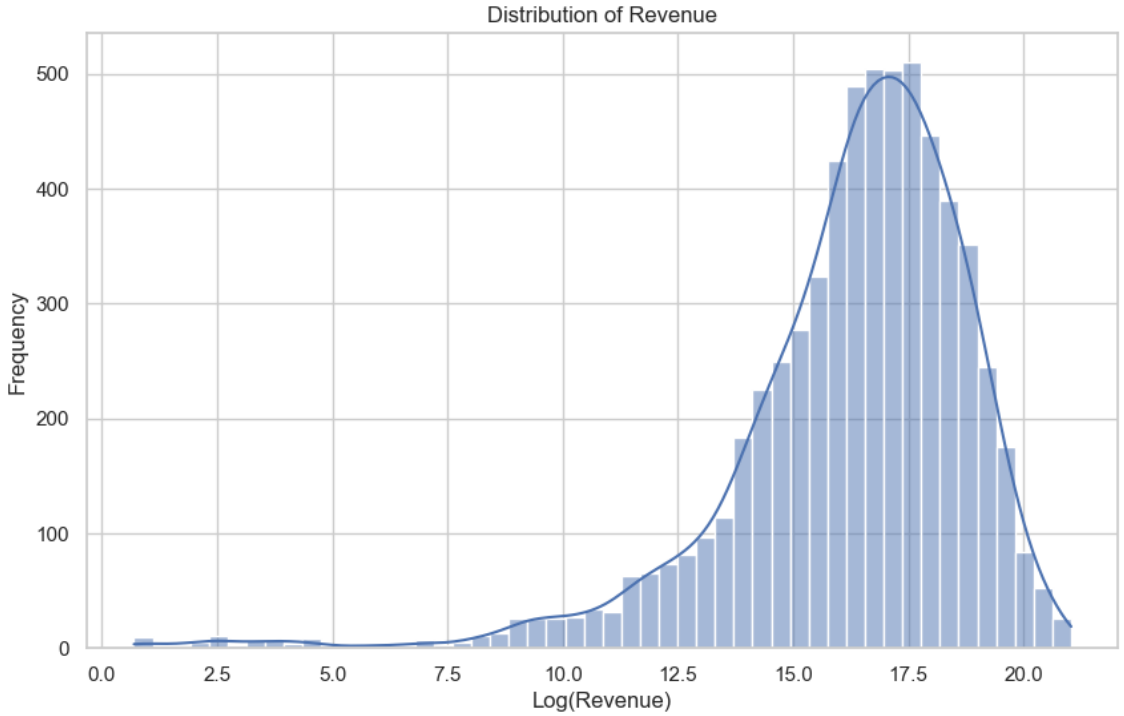


Figure 3.5. Distribution of Log(Revenue)

Moreover, logarithmic transformation can reduce the impact of extreme values or outliers, which may skew the analysis or distort interpretations. By compressing the scale of the data, logarithmic transformation can mitigate the influence of these outliers.

4. RESULTS AND EVALUATION

4.1. Regression Algorithms

4.1.1. SVR

In this model training progress, we used GridSearch to find best kernal, gamma and C in combination with 5-fold cross validation. This methodology was chosen to mitigate overfitting and furnish a more dependable estimate of the model's generalizability across diverse datasets. Additionally, feature scaling was employed utilizing StandardScaler, a standard practice aimed at enhancing algorithm convergence and model accuracy.

The culmination of these efforts yielded a set of best parameters: {'C': 10, 'epsilon': 0.2, 'gamma': 0.01, 'kernel': 'rbf'}. Subsequently, the model underwent training with the overarching objective of maximizing the R-squared metric, attaining a commendable value of 0.81 for the Support Vector Regression (SVR) algorithm. Furthermore, ancillary performance metrics such as Mean Absolute Error outperformed those of the Ridge Regression model, registering at 27 million, and the Root Mean Squared Error stood at 55 million.

Notably, in comparison to the Ridge Regression model, the SVR algorithm exhibited a slight enhancement in predicting revenue against actual revenue, evidencing a closer alignment with the underlying trend. However, it is acknowledged that there may exist alternative algorithms with the potential to further refine predictive accuracy, suggesting avenues for continued exploration and optimization in model development.

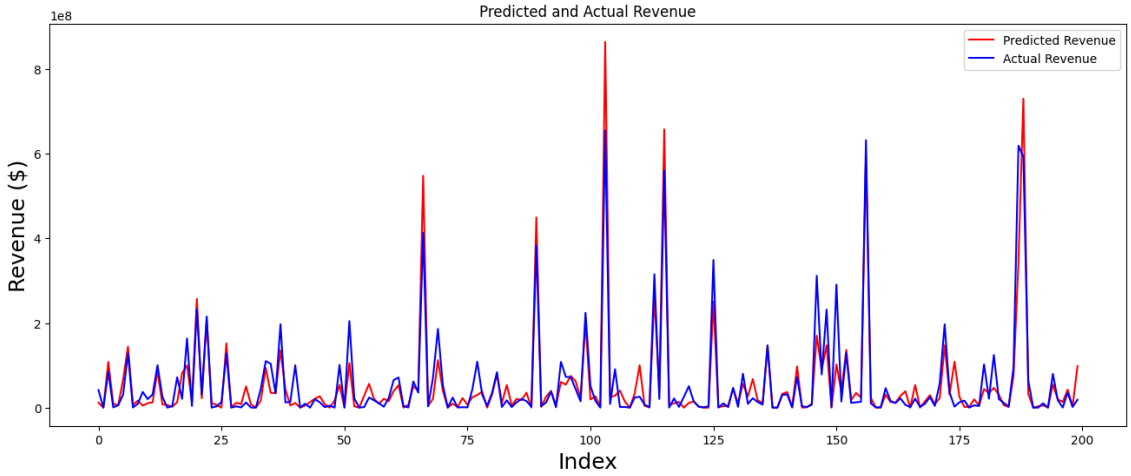


Figure 4.1. SVR - Predicted vs. Actual Revenue

4.1.2. Ridge Regression

For this algorithm, we used GridSearch to find best lambda (alpha in sklearn library). We trained the model as to make R-squared as high as possible and achieved 0.76 for Ridge Regression model. Other metrics such as Mean Absolute Error was 36 million and Root Mean Squared Error was 64 million. In such instances, there is a noticeable alignment between the predicted revenue and the actual revenue, indicating a strong predictive capability. However, as the revenue diminishes, the model's accuracy appears to decline, with a discernible disparity between the predicted and actual revenue figures. This discrepancy suggests a limitation in the model's ability to accurately forecast revenue in scenarios characterized by lower revenue streams. Thus, while the model demonstrates proficiency in certain contexts, its predictive accuracy may be variable depending on the magnitude of the revenue being forecasted.

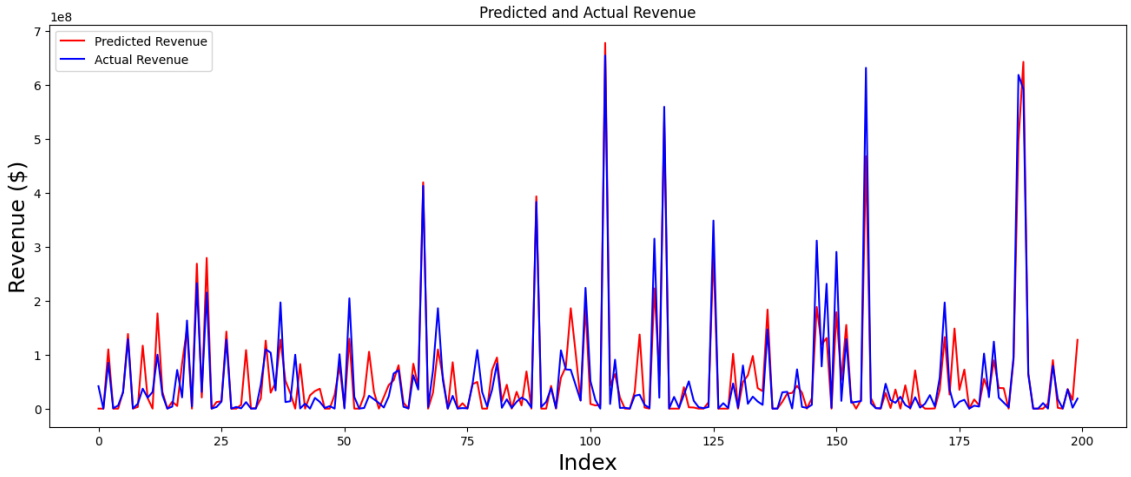


Figure 4.2. Ridge Regression - Predicted vs. Actual Revenue

4.1.3. Decision Tree Regression

In evaluating the performance of our predictive model, we conducted a 5-fold cross-validation. The results are detailed in the table below:

Fold	R^2	MAE	RMSE
1	0.77	0.47	1.25
2	0.83	0.53	1.18
3	0.75	0.51	1.28
4	0.81	0.49	1.15
5	0.83	0.48	1.1
Mean	**0.80**	**0.50**	**1.19**

a. Coefficient of Determination R2

- The R2 values across the folds range from 0.75 to 0.83.
- The mean R2 is 0.80, indicating that, on average, the model explains 80% of the variance in the target variable.

b. Mean Absolute Error (MAE):

- The MAE values range from 0.47 to 0.53 across the folds.
- The mean MAE is 0.50, reflecting that, on average, the absolute difference between the predicted values and the actual values is 0.50 units.

c. Root Mean Squared Error (RMSE):

- The RMSE values range from 1.10 to 1.28 across the folds.
- The mean RMSE is 1.19, indicating that the standard deviation of the residuals is 1.19 units.

Interpretation:

- R2 : The model shows a high level of predictive power, consistently explaining around 80% of the variability in the data across different folds.
- MAE : The model’s predictions are quite accurate with an average absolute error of 0.50 units.
- RMSE: The error magnitude is relatively low, with an average error of 1.19 units, demonstrating that the model's predictions are close to the actual values.

Overall, the performance metrics indicate that the model performs well with good consistency across different subsets of the data, suggesting its robustness and reliability in making predictions.

4.1.4. Random Forest Regression

The performance of our predictive model was assessed using a 5-fold cross-validation approach. The results are detailed in the table below:

Fold	R^2	MAE	RMSE
1	0.81	0.45	1.13
2	0.88	0.47	1.0
3	0.8	0.41	1.14
4	0.85	0.41	1.01
5	0.87	0.42	0.96
Mean	**0.84**	**0.43**	**1.05**

a. Coefficient of Determination R2 :

- The R^2 values across the folds range from 0.80 to 0.88.
- The mean R2 is 0.84, indicating that, on average, the model explains 84% of the variance in the target variable.

b. Mean Absolute Error (MAE):

- The MAE values range from 0.41 to 0.47 across the folds.
- The mean MAE is 0.43, reflecting that, on average, the absolute difference between the predicted values and the actual values is 0.43 units.

c. Root Mean Squared Error (RMSE):

- The RMSE values range from 0.96 to 1.14 across the folds.
- The mean RMSE is 1.05, indicating that the standard deviation of the residuals is 1.05 units.

Interpretation:

- R2 : The model demonstrates a high level of predictive power, consistently explaining around 84% of the variability in the data across different folds.

- MAE: The model's predictions are highly accurate with an average absolute error of 0.43 units.
- RMSE: The error magnitude is low, with an average error of 1.05 units, suggesting that the model's predictions are very close to the actual values.

Overall, these performance metrics indicate that the model performs exceptionally well with consistent and reliable results across different subsets of the data. This suggests the model's robustness and effectiveness in making accurate predictions.

4.1.5. Gradient Boost Decision Tree

We have implemented 5 fold cross validation in each of our experiments. In 5 fold cross validation, all the elements in our dataset are divided into 5 groups. Each group consecutively be chosen as the testing group while remainder are used for training. In this way all data are tested and mean is calculated from the accuracy of each fold.

In this algorithm, we train and compare two types of machine learning models based on different input feature sets: one using all available features and the other using a subset of pre-released features (excluding user score and number of votes).

1. Model Training with All Features:
   - Feature Set: Includes all features.
   - Hyperparameter Optimization: Employ Random Search for hyperparameter tuning.
2. Model Training with Pre-Released Features:
   - Feature Set: Excludes user score and number of votes.
   - Hyperparameter Optimization: Use Grid Search for initial hyperparameter tuning, followed by fine-tuning.

Table 4.1. GBDT (Pre-released Features)

Fold	R^2	MAE	RMSE
1	0.73	12854476	62846127
2	0.74	15468811	64600342
3	0.75	13611195	61646168
4	0.75	13997054	68125112
5	0.74	14085942	62587247
Mean	**0.74**	**14003495**	**63960999**

Table 4.2. GBDT (All Features)

Fold	R2	MAE	RMSE
1	0.81	7431829	56446224
2	0.93	7099429	44010212
3	0.90	5924753	47427212
4	0.92	5821911	53242866
5	0.92	5703825	45652173
Mean	**0.90**	**6396349**	**49355737**

Table 4.1 show the results of each fold in 5-fold cross validation with pre-released data, while in table 4.2, representing the data of GBDT model trained on all features.

The model for pre-released performance is quite good. The R^2 values are all above 0.7, indicating that the model is able to explain a significant portion of the variance in the data. The mean values across all folds provide a good overall assessment of the model's performance.

Beside, the all features' model performs well across all folds, with average of $R^2 = 0.9$. The MAE and RMSE values indicate that the model's predictions are relatively accurate.

We can clearly see that the model trained with all features outperforms the model trained with pre-released features both in 3 scoring of R^2_score, MAE and

RMSE. This suggests that user score and number of votes are significant features that contribute to better model performance.

4.1.6. XGBoost Regression

This algorithm adopts the same approach as Gradient Boosting Decision Tree (GBDT) algorithms. Additionally, we implemented 5-fold cross-validation in all our experiments. In 5-fold cross-validation, the dataset is divided into five groups. Each group is sequentially chosen as the testing set, while the remaining groups are used for training. This method ensures that all data points are tested, and the mean accuracy across all folds is calculated.

1. Model Training with All Features:
   - Feature Set: Incorporating all available features.
   - Hyperparameter Optimization: Employing Random Search for optimizing hyperparameters.
2. Model Training with Pre-Released Features:
   - Feature Set: Excluding user score and number of votes.
   - Hyperparameter Optimization: Initiating with Grid Search for initial hyperparameter exploration, succeeded by fine-tuning.

Table 4.3. XGBoost (Pre-released Features)

Fold	R^2	MAE	RMSE
1	0.73	12854476	62846127
2	0.74	15468811	64600342
3	0.75	13611195	61646168
4	0.75	13997054	68125112
5	0.74	14085942	62587247
Mean	**0.74**	**14003495**	**63960999**

Table 4.4. XGBoost (All Features)

Fold	R2	Real_MAE	Real_RMSE
1	0.81	6863173	52955007
2	0.96	5216215	34450903
3	0.95	4410665	36427157
4	0.96	4069888	38614170
5	0.96	4378360	34023101
Mean	**0.93**	**4987660**	**39294067**

The 4.2 table examines the results of XGBoost model with pre-released features. According to the average of … in R^2 score, we can a significant portion of the variance in the data. The average MAE and RMSE values is relatively low, in compare to that of the GBDT model.

The XGBoost model for all features exhibits decent performance, with an average R^2 of 0.93, indicating a strong fit. The mean of MAE is 4987660, suggesting the model's predictions are fairly accurate and mean RMSE is 39294067, also indicating good accuracy.

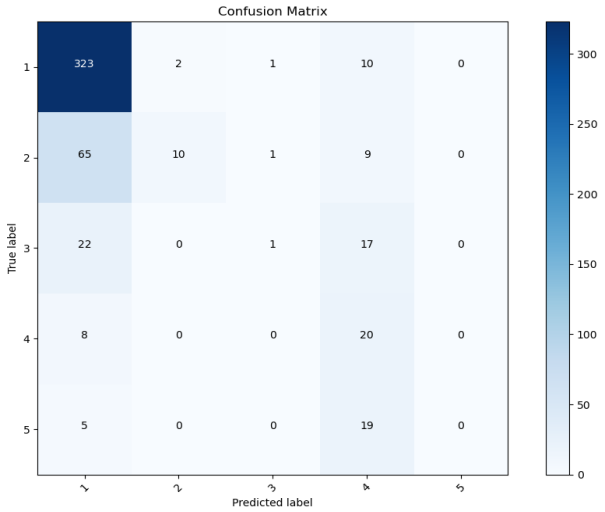
4.2. Classification Algorithms

4.2.1. Decision Tree

To classification task, we use DecisionTreeClassifier function. In our project, our algorithm is the combination of 3 hyperparameters: max_features, max depth, max_leaf_nodes.

- max_features: This hyperparameter represents the number of features to consider when looking for the best split with 3 conditions ‘sqrt’, ‘log2’, None.
- max_depth: This parameter determines the maximum depth of the tree. Limiting the depth of the tree helps prevent overfitting. If max_depth is None, the nodes are expanded until all leaves are pure or until all leaves contain fewer than the minimum number of samples required to split.
- max_leaf_nodes: This hyperparameter sets the maximum number of leaf nodes in the tree. Constraining the number of leaf nodes can help reduce

the model's complexity and enhance its ability to generalize to new data. If max_leaf_nodes is None, there is no limit on the number of leaf nodes.



	precision	recall	f1-score	support
0	0.76	0.96	0.85	336
1	0.83	0.12	0.21	85
2	0.33	0.03	0.05	40
3	0.27	0.71	0.39	28
4	0.00	0.00	0.00	24
accuracy			0.69	513
macro avg	0.44	0.36	0.30	513
weighted avg	0.68	0.69	0.62	513

Figure 4.3. Decision Tree, combination (None, 6, 9)

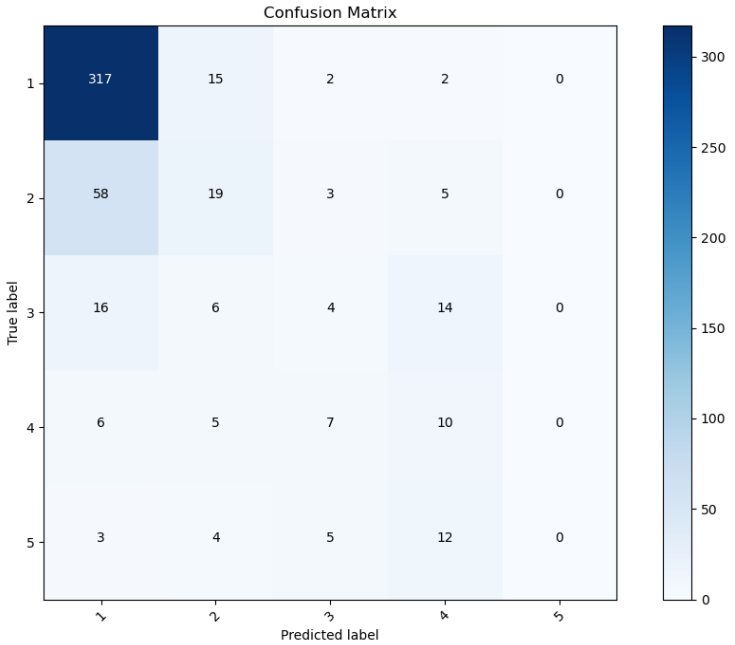
4.2.2. Random Forest

Random Forest algorithm, an algorithm that is highly appreciated for its scalability, ease of use, and especially its high applicability to many levels of problems in the field of machine learning. The idea of Random Forest is to creat several decision trees, combine and vote to make predictions.

To classification task, we use RandomForestClassifier function. In our project, our algorithm is the combination of 3 hyperparameters: max_features, max depth, max_leaf_nodes, n_estimators

- "n_estimators": This parameter defines the number of decision trees in the random forest. It randomly selects an integer value between 25 and 150.

- Max_depth: This parameter determines the maximum depth of the tree. Limiting the depth of the tree helps prevent overfitting. If max_depth is None, the nodes are expanded until all leaves are pure or until all leaves contain fewer than the minimum number of samples required to split.
- Max_features: This hyperparameter represents the number of features to consider when looking for the best split with 3 conditions "sqrt", ‘log2’,None.if max_features = sqrt, it will use the square root of the number of features., if ‘log2’, it will use the logarithm (base 2) of the number of features, else it will consider all the features available for each split.
- Max_leaf_nodes: This hyperparameter sets the maximum number of leaf nodes in the tree. Constraining the number of leaf nodes can help reduce the model's complexity and enhance its ability to generalize to new data. If max_leaf_nodes is None, there is no limit on the number of leaf nodes.



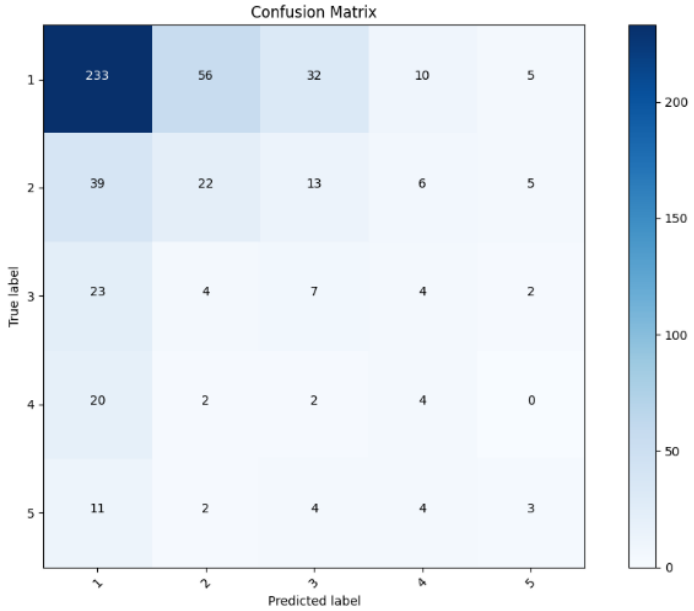
	precision	recall	f1-score	support
0	0.79	0.94	0.86	336
1	0.39	0.22	0.28	85
2	0.19	0.10	0.13	40
3	0.23	0.36	0.28	28
4	0.00	0.00	0.00	24
accuracy			0.68	513
macro avg	0.32	0.32	0.31	513
weighted avg	0.61	0.68	0.64	513

Figure 4.4. Decision Tree, combination (150, 6, None, 9)

4.2.3. SVC kernal RBF

SVM algorithm, support vector machine can solve problems related to operations in high-dimensional space, which is a state prone to overfitting of random forest because its regularization parameter helps control the complexity of the random forest. model. This is a powerful algorithm and is especially effective in situations where the data is linearly separable or can be transformed into a higher dimensional space where the data is separable. The idea of SVM is to find the optimal hyperplane that best separates different classes in the feature space while maximizing the margin, which is the distance between the hyperplane and the closest data points of each class.

- "C": This parameter represents the regularization parameter which controls the trade-off between high bias and high variance. C is randomly select in defined set.
- "gamma": This parameter defines the kernel coefficient for 'rbf' and 'poly' kernels which influences the shape of the decision boundary. The values 'scale' and 'auto' are predefined options: 'scale': Scale gamma value according to the inverse of the number of features. 'auto': Chooses 1 / n_features as the gamma value.
- "kernel": This parameter specifies the kernel type in the algorithm. Common kernel functions include: Linear: Uses a linear kernel function $K(x,y) = x^T y$, RBF (Radial Basis Function): Uses a Gaussian kernel function $K(x,y) = exp(-\gamma||xy||2)$. Poly (Polynomial): Uses a polynomial kernel function $K(x,y) = (x^2y+r)^d$.



	precision	recall	f1-score	support
0	0.71	0.69	0.70	336
1	0.26	0.26	0.26	85
2	0.12	0.17	0.14	40
3	0.14	0.14	0.14	28
4	0.20	0.12	0.15	24
accuracy			0.52	513
macro avg	0.29	0.28	0.28	513
weighted avg	0.54	0.52	0.53	513

Figure 4.5. SVC kernel RBF, combination (100, 0.3, 'rbf')

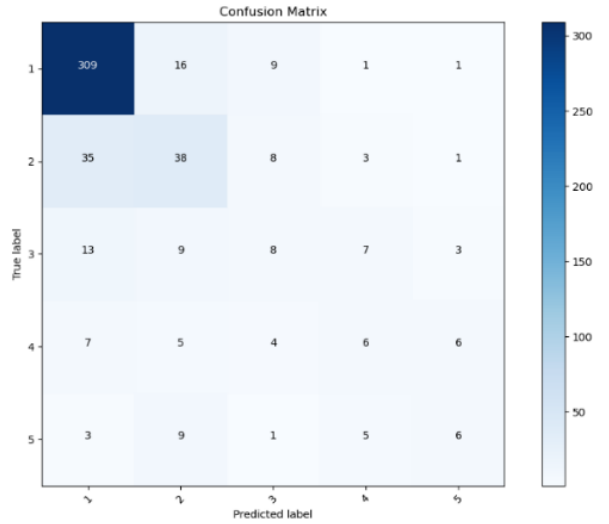
4.2.4. XGBoost Classification

XGBoost (Extreme Gradient Boosting) is another highly acclaimed algorithm in the field of machine learning, known for its exceptional performance and versatility in a wide range of problems, including classification tasks.

The key idea behind XGBoost is to create an ensemble of weak predictive models, typically decision trees, and then iteratively improve the overall model by adding new trees that focus on correcting the mistakes made by the previous trees. This process, known as gradient boosting, allows XGBoost to construct a powerful and accurate predictive model.

To classification task, we use XBGClassifier function. In our project, our algorithm is the combination of 5 hyperparameters: learning_rate, max depth,min_child_weight, n_estimators , sub_sample:

- Learning rate: This hyperparameter controls the step size at each iteration while moving toward a minimum of the loss function.
- Max_depth: This hyperparameter defines the maximum depth of a tree, i.e., the maximum number of nodes between the root node and the farthest leaf node.
- Min_child_weight: This parameter specifies the minimum sum of instance weight (hessian) needed in a child. It is used to control overfitting.
- n_estimators: This parameter sets the number of trees (or boosting rounds) to be used in the model.
- subsample: This hyperparameter specifies the fraction of the training data to be used for growing each tree. It introduces randomness into the model building process.



	precision	recall	f1-score	support
0	0.84	0.92	0.88	336
1	0.49	0.45	0.47	85
2	0.27	0.20	0.23	40
3	0.27	0.21	0.24	28
4	0.35	0.25	0.29	24
accuracy			0.72	513
macro avg	0.45	0.41	0.42	513
weighted avg	0.69	0.72	0.70	513

Figure 4.6. XGBClassification, combination (0.1, 3, 3, 200, 0.6)

4.2.5. Neural Network

A Multi-Layer Perception Neural Network (MLP) has been used for prediction. This MLP model is developed using Keras, a very popular Python API for neural networks. The Keras sequential model has been utilized to construct the model. In the proposed model, there are three hidden layers: the first layer has 58 neurons, the second layer has 35 neurons along with a Dropout layer with a dropout rate of approximately 0.43, and the final hidden layer has 32 neurons with a Dropout layer having a dropout rate of around 0.09. The input layer consists of 28 nodes, and the final output layer contains five nodes for five outputs. Softmax and ReLU activation functions are used for the final output layer and the hidden layers, respectively. The Adam optimizer has been used to train the model, selected for its efficiency and effectiveness in managing sparse gradients on noisy problems. To determine the optimal architecture, including the number of layers, the number of neurons in each layer, and the dropout rates, we employed the Optuna framework.

In detail:

- ***Hidden Layer:*** A hidden layer in a neural network is any layer between the input layer and the output layer. The purpose of hidden layers is to capture and learn complex patterns and representations in the data.
- ***Dropout Layer:*** A Dropout layer is a regularization technique used to prevent overfitting in neural networks. During training, dropout randomly sets a fraction of input units to zero at each update cycle. This prevents neurons from becoming too dependent on specific patterns, thus forcing the network to learn more robust features.
- ***Activation Functions:*** Softmax and ReLU
  - ReLU (Rectified Linear Unit): ReLU is an activation function defined as $f(x)$ = max(0,x). It is simple yet effective, allowing the network to capture non-linear relationships.
  - Softmax: The Softmax activation function is commonly used in the output layer of a classification network. It converts raw output scores (logits) into probabilities, with each value between 0 and 1, and the sum of all output probabilities equals to 1. Softmax is defined as $\sigma(z_i) = \frac{e^{z_i}}{\sum_j e^{z_i}}$.
- ***Adam Optimizer:***

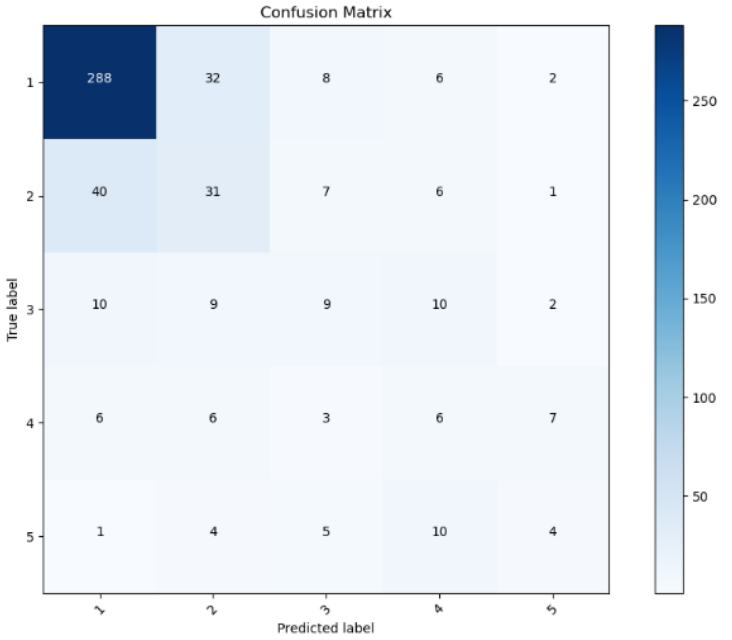
  The Adam (Adaptive Moment Estimation) optimizer is an extension of stochastic gradient descent that computes adaptive learning rates for each parameter.
- ***Optuna Framework***

  Optuna is an open-source hyperparameter optimization framework designed to automate the search for the best hyperparameters in machine learning models. It uses an efficient sampling-based optimization approach to explore the hyperparameter space. Optuna leverages techniques like Bayesian optimization, Tree-structured Parzen Estimator (TPE), and other methods to balance exploration and exploitation during the search process. In our project, I use Optuna to

find number of layer, number or neurons in each layers and Dropout rate for dropout layer.

Here is a figure for combination of:

{‘number layer’: 3;
layer_size_0: 58,
layer_size_1: 35;
dropout_rate_1: 0,43;
layer_size_2: 32;
dropout_rate_2: 0.09
}



	precision	recall	f1-score	support
0	0.83	0.86	0.85	336
1	0.38	0.36	0.37	85
2	0.28	0.23	0.25	40
3	0.16	0.21	0.18	28
4	0.25	0.17	0.20	24
accuracy			0.66	513
macro avg	0.38	0.37	0.37	513
weighted avg	0.65	0.66	0.65	513

Figure 4.7. Neural Network

5. EVALUATION METRICS

5.1. Classification

5.1.1. Accuracy

Classification accuracy serves as a pivotal metric in evaluating the performance of our models. It quantifies the model's effectiveness in correctly assigning labels to instances. Accuracy is computed as the ratio of correct predictions to the total predictions made:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

While it stands as a fundamental measure for assessing model efficacy, accuracy metrics do not work well with imbalanced datasets and do not provide a clear view of each class's performance. For instance, in a dataset where one class significantly outnumbers others, a model predicting the majority class most of the time will have high accuracy but poor predictive power for the minority class.

5.1.2. Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a classification model by comparing the predicted labels to the actual labels in the dataset. The matrix is organized into rows and columns, where each row represents the actual class, and each column represents the predicted class.

The confusion matrix provides a more detailed analysis than accuracy alone by showing how the model is performing on each class. It includes four key metrics:

- True Positives (TP): The number of instances correctly predicted as the positive class.
- True Negatives (TN): The number of instances correctly predicted as the negative class.
- False Positives (FP): The number of instances incorrectly predicted as the positive class.
- False Negatives (FN): The number of instances incorrectly predicted as the negative class.

The structure of the confusion matrix can be illustrated as follows for a binary classification problem:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

From the confusion matrix, several important metrics can be derived, such as Precision, Recall, and F1 Score, which are critical for a comprehensive evaluation of model performance, especially in cases of imbalanced datasets.

5.1.3. Precision, Recall, and F1 Score

- Precision: Precision is the ratio of true positive predictions to the total predicted positives. It indicates how many of the predicted positive instances are actually positive.

$$\text{Precision} = \frac{TP}{TP+FP}$$

- Recall (Sensitivity): Recall is the ratio of true positive predictions to the total actual positives. It measures the model's ability to correctly identify all positive instances.

$$\text{Recall} = \frac{TP}{TP+FN}$$

- F1 Score: The F1 Score is the harmonic mean of Precision and Recall, providing a single metric that balances both concerns. It is particularly useful when the classes are imbalanced.

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

5.2. Regression

5.2.1. Root Mean Squared Error (RMSE)

RMSE is a measure of the differences between the predicted values and the actual values. It is the square root of the average of the squared differences between the predicted and actual values. RMSE gives a relatively high weight to large errors, meaning it is more sensitive to outliers.

The formula for RMSE is:

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

where:

- y_i is the actual value,
- $\hat{y}_i$ is the predicted value,
- n is the number of observations.

5.2.2. Mean Absolute Error (MAE)

MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It is the average over the test sample of the absolute differences between the predicted and actual values.

The formula for MAE is:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where:

- y_i is the actual value,
- $\hat{y}_i$ is the predicted value,
- n is the number of observations.

5.2.3. R_2 score

R2 score, also known as the coefficient of determination, is a statistical measure that represents the proportion of the variance for the dependent variable that's explained by the independent variables in a regression model. R_2 provides an indication of goodness of fit and ranges from 0 to 1.

Formula of R2 score:

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

where: y_i is the actual value, $\hat{y}_i$ is the predicted value,

$\bar{y}_i$ is the predicted value, n is the number of observations.

6. Features Importance

Here we have a figure of importance scoring by F_score, draw by built-in function of xgboost.

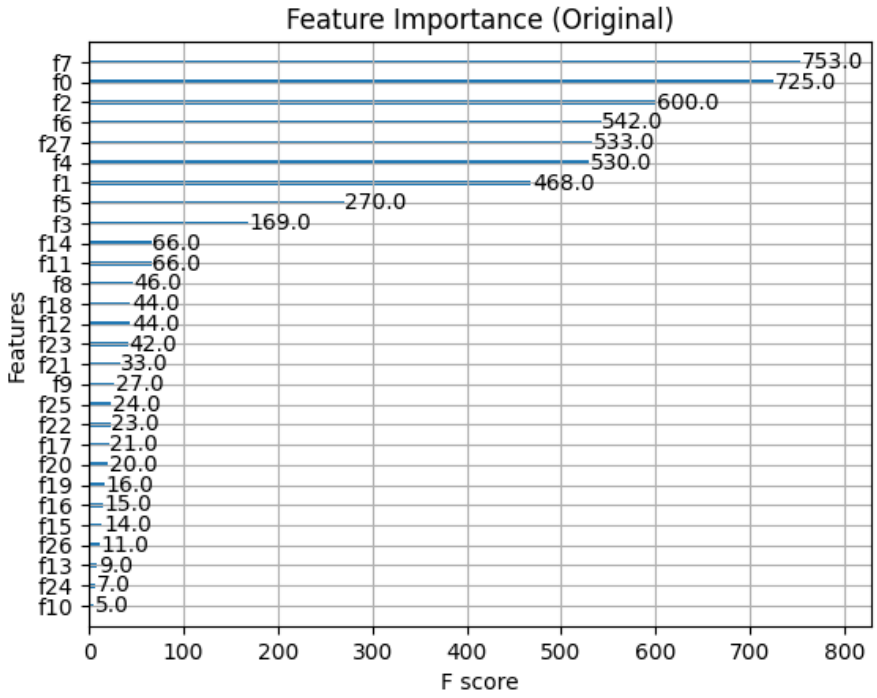


Figure 6.1. Feature Importance (XGBoost's plot_importance)

The F-score here represents the frequency with which a feature is used to make key decisions with decision trees in the ensemble. The more frequently a feature is used in the splits, the higher its importance. We'll eliminate features with insignificant importance and then provide a suitable title for the filtered feature importance plot, then change to the inportance.

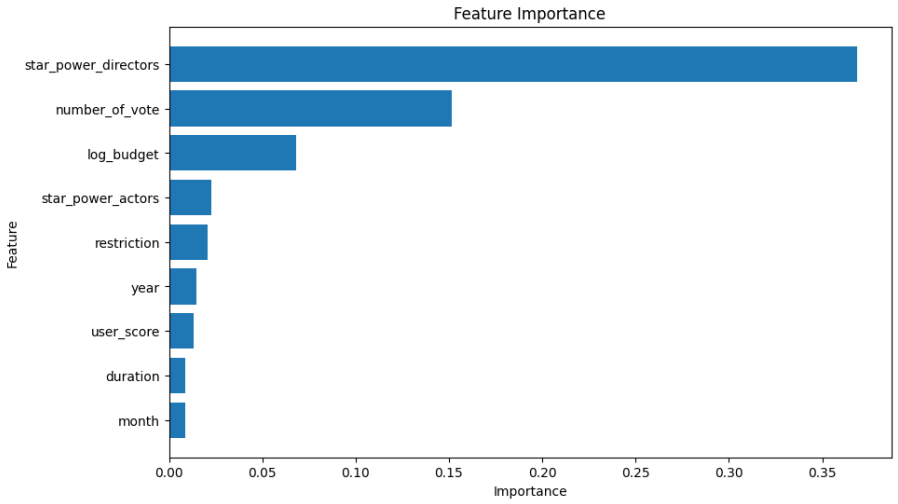


Figure 6.2. Feature Importance (Converted)

The provided bar chart displays the feature importance scores for a model predicting movie revenue. Feature importance is a metric that indicates how much each feature contributes to the model's predictive power. The features are listed on the y-axis, while the x-axis represents the importance, which quantifies the importance of each feature.

The most important feature is "star_power_directors" with the figur of over 35%, suggesting it significantly influences the prediction model. The following are "number_of_votes", “log_budget” and "star power actors", with 15%, 6% and 2% consecutively. These features highlight the substantial impact that the popularity and influence of directors and actors have on a movie's revenue and also the money invested in the movie.

In summary, the chart reveals that audience engagement metrics (like the number of votes and star power) and financial inputs (like budget) are the most influential factors in predicting movie revenue, while genre-related features play a comparatively minor role.

7. REFERENCES

Gradient Boosted Decision Trees . (2022, 09 28). Retrieved from Google Developer: https://developers.google.com/machine-learning/decision-forests/intro-to-gbdt

Introduction to XGBoost Algorithm in Machine Learning. (2024, 5 27). Retrieved from Analytic Vidhya: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

Nahid Quader, M. O. (2018). *A Machine Learning Approach to Predict Movie Box-Office Success.* Dhaka, Bangladesh: Research Gate.

Rijul Dhir, A. R. (2018). *Movie Success Prediction using Machine Learning Algorithms and their Comparison.* Jalandhar, India: IEEE.

MEMBER CONTRIBUTION

- Hoang Trung Khai
  - Collecting data
  - Preprocessing
  - 2 models in Regression: GBDT, XGB Regression
  - Writing Report
- Trinh Duy Phong
  - 5 models in Classification: Decision Tree, Random Forest, SVC kernel RBF, XGBoost Classification, Neural Network.
  - UI
- Vu Viet Long
  - 2 models in Regression: SVR, Ridge Regression
- Luu Hoang Phan
  - 2 models in Regression: Decision Tree, Random Forest
- Pham Minh Tien
  - Preprocessing
- Nguyen Phan Thang
  - Slide designing
  - Feature Importance