DV2599 – Project Final Report

Simon Lindqvist
Blekinge Institute of Technology
Karlskrona, Blekinge
siln22@student.bth.se

Abdalrahman Mohammed
Blekinge Institute of Technology
Karlskrona, Blekinge
abmm22@student.bth.se

I. Introduction

The entertainment industry is increasingly using data to predict how successful a movie will be, which helps in making investment decisions and shaping creative plans. This project uses the TMDB dataset to predict both movie profitability and popularity with machine learning methods. The process involves data cleaning, sentiment analysis for title-based features, feature creation, and using models like KNN, Random Forest and XGBoost. These models are tested using stratified k-fold cross-validation and improved by Hyperparameter tuning. Predicting movie rankings has significant possible implications for the entertainment industry:

- Industry Impact: Accurate predictions help stakeholders identify profitable projects while avoiding those that may not make enough money to cover their costs.
- Societal Impact: Better predictions also help independent filmmakers with good ideas to prove project feasibility and get their projects funded, making it easier for creative people to succeed and supporting a wider variety of films.

The focus is on achieving strong F1 scores, which measures how well the model balances precision and recall. This approach helps identify the key factors that make a movie successful and provides a useful tool for analyzing movie data.

II. Method

A. DATASET

The "Full TMDB Movies Dataset 2024 (1M Movies)" dataset was chosen. The dataset consists of 1,153,446 rows × 24 columns, representing movies and their associated attributes. It was chosen because it's maintained daily, and it offers a solid base for analyzing movie features, performance metrics, and trends. However, it requires extensive cleaning and preparation to ensure accurate and reliable results.

1) Handling missing values

Some columns in the dataset contain significant amounts of missing information. For instance, around 17% of movies lack a release date, over 74% are missing their backdrop image, and more than 80% do not have a homepage or tagline. Additionally, 40%-50% of entries are missing details about genres or spoken languages, and nearly 47% lack an IMDb ID. These gaps in columns hinder effective data analysis. For example, the absence of release dates makes it challenging to study how movie trends have evolved over time, while missing genre information complicates the task of identifying which genres perform well or poorly.

2) Outliers

The dataset contains several values that are unusual or nonsensical. For example, some movies are listed with a budget of 0, which may not be accurate and could indicate missing data. These outliers can significantly disrupt critical calculations. For instance, a few movies with extremely high popularity scores can skew the average, leading to misleading conclusions about what makes a movie "popular" or profitable.

3) Inconsistencies *in Data*

Some columns in the dataset have invalid or inconsistent formats. For instance, the release_date column may contain missing values or varying formats, making it challenging to analyse trends over time. Similarly, the genres and spoken_languages columns may include irregular or inconsistent text, complicating text-based analyses. Logical errors are also present in the dataset. For example, some movies have zero votes (vote_count) but display a high average rating (vote_average), which is implausible. Additionally, numerous rows show both budget and revenue as zero, likely indicating missing or incomplete data. These inconsistencies can lead to flawed analysis. For example, if release_date values are missing or incorrect, studying how movies perform over time becomes problematic. Likewise, inaccuracies in budget and revenue data hinder the assessment of a movie's profitability.

4) High Dimensionality

Some columns, like keywords and genres have multiple categories that can be true at the same time. Representing these in a way that machine learning models can use would require some form of vectorization or encoding creating a higher dimensionality for the dataset even though the original dataset only has twenty something features.

B. DATA PREPROCESSING

The data preprocessing stage involved several key steps to prepare the TMDB movie dataset for analysis and modeling. For this section please refer to the tables in appendix A.

1) Initial data exploration

First, the dataset was examined using .info() and .describe() to understand data types, distributions, and missing values. The original dataset contained 1,153,446 rows and 24 columns with varying degrees of completeness. Several columns had significant numbers of missing values, including release_date (959,206 non-null out of 1,153,446) and overview (916,952 non-null).

2) Dropping irrelevant and problematic data

Movies with status other than "Released" were removed as they wouldn't provide meaningful performance metrics. URL-based columns (backdrop_path, homepage, poster_path) were dropped as they contained links rather than useful predictive features. Unique identifiers (id, imdb_id) were removed since they don't contribute to predictive power. Additionally, rows with zero vote_count were dropped as their ratings couldn't be trusted.

3) Handling missing values

Rather than imputing missing values, rows with incomplete data were removed using dropna(). This approach was chosen to maintain data integrity and avoid introducing bias. After dropping rows with missing values, the dataset was reduced to 45,337 rows (approximately 4% of the original dataset), which still provided sufficient data for modeling.

4) Feature selection and transformation

Text-heavy columns that required extensive natural language processing (tagline, overview) were removed as they were out of scope for the project. The release_date column was converted to datetime format for proper temporal analysis, and its distribution was visualized to understand chronological patterns in the data.

5) Creating Two Feature Representation Approaches

The project explored two distinct approaches to handle categorical data, balancing information retention against computational efficiency.

In the one-hot encoding approach, categorical features were transformed into binary indicator variables, creating a separate column for each possible value. For genres, all unique genres were encoded, preserving the complete genre information since the analysis showed this feature had strong predictive power for movie performance. For production companies and keywords, which showed highly fragmented distributions with thousands of unique values, only the top 100 most frequent values were encoded to prevent excessive dimensionality while retaining the most relevant information. This approach produced a dataset with 467 features, preserving the richness of categorical information at the cost of increased dimensionality.

The alternative feature-engineering approach focused on reducing dimensionality while extracting essential information. For genres, individual binary features were created for each unique genre, as the analysis showed the total number of genres was manageable (around 20). For production companies, rather than encoding each company, a binary feature was created indicating whether the movie was produced by a "major" company, defined as having produced at least 10 movies or having a combined budget of $100 million. This decision was based on the observation that production company distribution was highly skewed, with a few major companies producing most movies. For languages, analysis revealed that English dominated both original and spoken languages, so binary indicators were created specifically for English, which simplified the representation while maintaining key predictive information. Similarly, for keywords, which showed an extremely long-tailed distribution, a single binary feature was created to indicate whether the movie contained any of the top 100 most frequent keywords. This approach reduced the feature space to just 27 features, offering significant computational efficiency.

6) Feature scaling

Feature scaling was applied selectively based on each numerical feature's distribution. The release_date and runtime columns followed a roughly normal distribution (except for runtime values of zero), making StandardScaler appropriate for ensuring zero mean and unit variance. This prevents larger-scale features from dominating model training.

In contrast, the budget column was highly skewed, with a few blockbusters having extreme values. MinMaxScaler was used to map values to [0,1] without assuming normality, preserving relative differences while limiting outlier impact—critical given budget's non-linear relationship with movie performance. Binary features like adult and engineered indicators remained unscaled, as they were already standardized into [0,1], making further scaling unnecessary.

7) Target variable creation

Several target variables were created to support different predictive modeling approaches:

1. Rounded Vote Average: The original vote_average was rounded to the nearest integer (0-10 scale), creating a multi-class classification problem. This transformation maintained the granularity of ratings while making them more interpretable as discrete rating levels that audiences commonly understand (like star ratings).

2. Binary Vote Average: A binary variable indicating whether a movie's rating was above the median. This simplification made the prediction task more manageable while ensuring balanced classes, useful for initial modeling and when precise rating prediction isn't necessary - just whether a movie will be above or below average quality.

3. Quartile Vote Average: The vote_average was divided into four categories based on quartiles. This provided a middle ground between binary and full 10-class prediction, offering meaningful distinction between poor, below average, above average, and excellent movies without excessive complexity.

4.Binary Revenue: Indicated whether a movie's revenue exceeded the median. This transformation addressed the highly skewed revenue distribution and created a balanced classification problem focused on commercial success rather than exact revenue figures - essentially predicting if a movie will be more successful than an average film.

5. Profitable: A binary variable showing whether revenue exceeded budget. This practical business-oriented target directly addressed a fundamental question in filmmaking: "Will this movie make money?" Unlike raw revenue prediction, profitability accounts for the investment required, making it particularly valuable for investment decisions in film production.

Each target variable was designed to answer different questions about movie performance, ranging from audience reception to financial success, while accommodating various modeling approaches and complexity levels.

8) Dataset evaluation

The preprocessed datasets were evaluated using k-fold cross-validation with XGBoost to compare the effectiveness of the one-hot encoding versus feature engineering approaches.

C. TITLE SCOREIFICATION

The scoreification component of this project represents an innovative approach to movie success prediction by extracting predictive signals directly from movie titles. This technique transforms qualitative text data (movie titles) into quantitative features that can enhance prediction models. The project implements a careful train-test split strategy specifically to prevent data leakage between the scoreification models and the main prediction tasks.

1) Preprocessing Approaches

The project explores three distinct approaches to title preprocessing:

1. NLTK Lemmatization: Titles are tokenized, and words are reduced to their base forms using WordNetLemmatizer. This preserves the semantic meaning of words while normalizing variations.
2. NLTK Stemming: Titles are tokenized, and words are reduced to their stems using PorterStemmer. This more aggressive approach captures word roots but may lose some semantic nuance.
3. Non-NLTK Features: Simple statistical features are extracted from titles without linguistic processing, including average word length, word count, title length, and character-based metrics.

All three approaches convert text into numerical features, with NLTK-based methods using CountVectorizer to create sparse matrices representing word frequencies.

2) Model Implementation

For each preprocessing approach, the project trains sentiment analysis models (MultinomialNB, XGBoost) to predict the target variables directly from title features. The models generate both predictions and confidence scores, which serve as additional features for the main prediction models.

The title scoreification models are trained on 80% of the data (non_score partition) to prevent any information leakage when evaluating the final models on the reserved 20% (score partition).

3) Statistical Comparison Framework

A robust statistical framework evaluates which scoreification approach yields the best results. The code implements the Friedman with Nemenyi post-hoc testing to identify significant performance differences between approaches.

The comparison examines each target variable (binary_vote_average, binary_revenue, profitable) separately to determine which scoreification method works best for each prediction task. This statistical rigor ensures that differences between approaches represent genuine performance variations rather than random chance.

4) Integration with Main Models

The scoreification features complement traditional movie metadata in the prediction pipeline. By incorporating title-derived signals alongside production details, budgets, and other features, the models could gain additional predictive power that might otherwise be missed in conventional approaches to movie success prediction.

D. DETERMINING BEST APPROACH AND MODEL

The model selection framework implemented in this project employs rigorous statistical methods to identify the best-performing approach among multiple candidates. The core of this framework is a non-parametric statistical testing pipeline that avoids making assumptions about the underlying distribution of performance metrics.

The compare_approaches_statistically function serves as the foundation of this methodology. It accepts performance results from different modeling approaches across multiple cross-validation folds and applies the Friedman test. This test determines whether there are statistically significant differences among the approaches without assuming normality in the data distribution.

When significant differences are detected ($p < 0.05$), the framework applies to the Nemenyi post-hoc test to identify specific pairs of approaches where one significantly outperforms the other. This approach controls multiple comparisons when determining which models are truly superior.

1) Systematic Evaluation Process

The project implements the comparison framework through the compare_method_across_datasets function, which extracts performance metrics from result dataframes for each modeling approach. This enables systematic comparison across multiple prediction targets (vote average, revenue, profitability) and different preprocessing strategies.

Each approach undergoes 5-fold cross-validation to ensure robust performance measurement. The function extracts metrics like accuracy and F1-score across all folds, feeding these values into the statistical comparison framework. This methodology prevents selection bias that might occur from evaluating models on single train-test splits.

2) Hyperparameter Optimization

For the top-performing model identified through statistical analysis, the project implements GridSearchCV to tune key hyperparameters. For tree-based models like XGBoost, the tuning focuses on parameters including:

- n_estimators: Number of boosting rounds
- max_depth: Maximum tree depth
- learning_rate: Step size shrinkage to prevent overfitting
- subsample: Fraction of samples used for training trees
- gamma: Minimum loss reduction for partition creation

GridSearch implements cross-validation internally to identify hyperparameter combinations that maximize performance while avoiding overfitting.

3) Final Model Validation

The final optimized model undergoes validation against held-out test data that wasn't used during the model selection or hyperparameter tuning stages. Performance is measured using multiple metrics (accuracy, F1-score, ROC-AUC) to provide a comprehensive evaluation of the model's predictive capabilities.

This methodical approach to model selection and optimization ensures that the final model represents a statistically justified choice rather than one based on chance variation in performance metrics.

4) ROC and Feature Importance

ROC analysis and Feature Importance analysis was also performed lastly when the final model was trained and evaluated.

E. CLASSIFIERS

1) Random forest

Random Forest is an ensemble learning algorithm that constructs multiple decision trees during training and

combines their predictions (via majority voting for classification) to make a final decision. Each tree is trained on a different subset of the data and features, which reduces the risk of overfitting and enhances the model's generalizability.

Random Forest is a good choice for this assignment because it can handle datasets with many features and is robust against noisy or missing data. Its ability to provide feature importance scores also helps in understanding which features contribute most to the predictions, adding an interpretative layer to the model. [3]

2) K-Nearest Neighbor (KNN)

KNN is a simple yet effective classification algorithm that works based on proximity. It predicts the class of a data point by identifying the majority class among its k nearest neighbors in the feature space. The distance between points is typically measured using metrics like Euclidean distance. The algorithm assumes that data points that are close to each other are likely to belong to the same class.

KNN is suitable for this assignment because it is intuitive and works well when the decision boundary is non-linear. Since the dataset is large and contains structured numerical features, KNN can effectively separate classes by learning from the underlying data distributions. [1]

3) XgBoost

XGBoost (Extreme Gradient Boosting) is a highly efficient and powerful gradient-boosting algorithm. It works by sequentially building trees, where each tree corrects the errors of its predecessor. The model optimizes a loss function and uses regularization techniques to prevent overfitting. It is known for its speed and accuracy in handling large datasets with complex relationships between features.

XGBoost is particularly suitable for this assignment because of its superior performance with structured data. The algorithm is well-suited for handling the large, preprocessed dataset, efficiently learning from its patterns and improving predictions iteratively. Its flexibility allows fine-tuning to achieve optimal results. [2]

4) Multinomial naive bayes (MNB*)*

Multinomial Naive Bayes (MNB) is a probabilistic algorithm that applies Bayes' theorem with the assumption of conditional independence between features. It is specifically designed for datasets with discrete features, such as count or frequency data. MNB works by calculating the conditional probabilities of each feature given a class and combines them to estimate the likelihood of a sample belonging to a particular class. The model is efficient, simple to implement, and performs well with high-dimensional data.

MNB is particularly suitable for this assignment because it effectively handles categorical and frequency-based features, such as genres, production languages, and production companies, which are prominent in the TMDB dataset. Its computational efficiency makes it ideal for processing large datasets like the preprocessed TMDB dataset with 45337 entries. Additionally, MNB provides interpretable results by highlighting feature contributions through probabilities, making it easier to identify factors that influence movie profitability or ratings. Its robustness and simplicity make it a valuable tool for benchmarking and gaining initial insights into the dataset. [4]

F. SCORING AND TESTS

1) Accuracy

Accuracy is the ratio of correctly classified instances to the total cases, calculated as (TP + TN) / (TP + TN + FP + FN). While it provides a simple performance measure, it can be misleading for imbalanced datasets where the majority class skews results. It is most useful when class distribution is balanced, and error costs are similar. [5]

2) F1 Score

The F1 score, which is the harmonic mean of precision and recall, balances false positives and false negatives. It is computed as 2 × (Precision × Recall) / (Precision + Recall). Unlike accuracy, it is beneficial for imbalanced datasets, ensuring fair evaluation of minority classes. [5]

III. RESULT & ANALYSIS

A. DATA EXPLORATION RESULTS

Our analysis compared two distinct approaches to feature representation for movie performance prediction: one-hot encoding with 467 features versus feature engineering with only 27 features. The results demonstrate a clear trade-off between predictive performance and computational efficiency. The one-hot encoded dataset consistently delivered superior predictive accuracy across all target metrics, with the largest advantage observed for vote average prediction (71.7% versus 66.8% accuracy). However, this improved performance came at a significant computational cost, requiring approximately 4 times longer training periods compared to the feature-engineered approach. For financial metrics, the performance gap was considerably smaller - one-hot encoding achieved 89.8% accuracy for binary revenue classification compared to 89.1% with feature engineering, representing only a 0.7 percentage point difference. Similarly, for profitability prediction, the gap was just 1.3 percentage points (87.9% versus 86.6%). These findings led us to implement a hybrid approach in our final solution: we selected the one-hot encoded dataset as our primary representation to maximize predictive performance, while also maintaining the feature-engineered dataset specifically for binary revenue and profitability predictions where the performance trade-off was minimal but the computational savings substantial. This strategic approach allows us to benefit from optimal accuracy for vote average prediction while leveraging the 3.87× training speedup offered by feature engineering for financial metrics where performance differences were negligible. This balanced solution demonstrates how thoughtful feature representation can effectively navigate the classic machine learning trade-off between model performance and computational efficiency.

B. TITLE SCOREIFICATION

Our title scoreification evaluation compared three distinct approaches for extracting predictive signals from movie titles: NLTK Lemmatization, NLTK Stemming, and Non-NLTK feature extraction. Using rigorous statistical testing with the Friedman test and Nemenyi post-hoc analysis, we identified significant performance differences between these methods across our three prediction targets. Note that due to many tables for comparison purposes these

tables will not be shown in the report and can be found in the submitted notebook.

For binary revenue prediction, the NLTK approaches significantly outperformed the Non-NLTK method, with the lemmatized approach achieving 77.6% accuracy compared to the Non-NLTK's 51.5%. Similarly, for profitable prediction, NLTK methods demonstrated superior performance (83.2% accuracy versus 76.7%). The vote average prediction proved most challenging across all approaches, with modest accuracy differences between NLTK (52.3%) and Non-NLTK (51.9%) methods that were statistically significant but smaller in magnitude.

Comparing the two linguistic approaches, lemmatization demonstrated a statistical advantage over stemming in two of three prediction tasks, likely because it preserves more semantic meaning in the processed text. This advantage was particularly pronounced for predicting financial metrics (revenue and profitability), suggesting that nuanced linguistic features in titles correlate more strongly with financial outcomes than with critical reception.

The MultinomialNB classifier emerged as the optimal model for scoreification across all approaches, delivering the best balance of performance and computational efficiency. While XGBoost occasionally achieved marginally higher accuracy (+0.3% for vote average prediction), MultinomialNB trained approximately 30× faster (0.003 seconds versus 0.1 seconds per fold) with nearly identical performance metrics. This efficiency advantage proved critical for the overall pipeline, as it allowed rapid generation of scoreification features without compromising predictive quality.

Based on these findings, our final implementation used the lemmatized NLTK approach with MultinomialNB for all three prediction targets, capturing valuable linguistic signals from titles while maintaining computational efficiency in the broader prediction pipeline.

C. RESULTS FOR MAIN MODELS

Our statistical framework employed the Friedman test with Nemenyi post-hoc analysis to evaluate different modeling approaches across our prediction targets. For each task, we compared three classifier types (XGBoost, Random Forest, and KNN) and two feature representation strategies (one-hot encoding versus engineered features). Note that due to a large number of tables for comparison purposes these tables will not be shown in the report and can be found in the submitted notebook.

XGBoost consistently demonstrated superior performance across all prediction targets, showing statistically significant advantages over both Random Forest and KNN. For binary vote average prediction, XGBoost with one-hot encoding achieved 71.7% accuracy, significantly outperforming Random Forest (67.4%) and KNN (63.1%). The performance gap was even more pronounced for binary revenue prediction, where XGBoost reached 89.8% accuracy compared to Random Forest's 85.3% and KNN's 79.6%.

The statistical tests confirmed that feature representation significantly impacted model performance, with one-hot encoding providing a measurable advantage over engineered features. However, the magnitude of this advantage varied by prediction target. For vote average prediction, one-hot encoding provided a substantial and statistically significant improvement (+4.9 percentage points). For financial metrics, the differences were smaller but still significant: +0.7 points for revenue and +1.3 points for profitability.

Hyperparameter tuning of the winning XGBoost models yielded further improvements. For vote average prediction, grid search identified optimal parameters that boosted accuracy by an additional 0.404 percentage points. For revenue prediction, parameter optimization increased accuracy by 0.163 percentage points and for profitable an increase of 0.276 percentage points was found.

D. ROC AND FEATURE IMPORTANCE RESULTS

1) ROC

The ROC analysis demonstrates that the Revenue Model (AUC = 0.928) exhibits the highest classification performance, suggesting strong predictive capability. The Profitable Model (AUC = 0.890) also performs well, though slightly less effectively. In contrast, the Vote Average Model (AUC = 0.794) shows comparatively weaker discrimination, indicating potential limitations in using vote averages as a predictor. These results suggest that revenue-based models are more reliable for classification tasks in this context. See appendix C.

2) Feature importance

Budget is the dominant feature for both revenue and profitability models, suggesting investment level strongly predicts financial outcomes. Genre importance varies by metric: documentaries and animation drive ratings, while TV movies impact revenue and profitability. Content characteristics matter differently across models - black and white films boost ratings, while anime content and post-credits scenes influence revenue. The distinctions between rating and financial predictors highlight the potential trade-offs between critical acclaim and commercial success. See appendix D.

IV. CONCLUSION

In conclusion three different models for classification of a movie to help predict a movie's performance and help investors and filmmakers make the correct decisions regarding movie production where created. These achieved 89.804, 72.034, and 88.045 percent accuracy for the revenue, vote average and profitable targets respectively (see appendix B, figure 13). Revenue could help tell if the movie will become a real release or not (achieving any revenue), vote average if the movie will get a higher than median rating, and profitable if the movie will make a profit or not. Adding scores for the titles based on their relation to these metrics didn't achieve any improvement for XGBoost, only to randomforest, to the models, meaning that the models likely can learn the same relationships from other features or combinations of features, even though these additional metrics had moderate to high accuracy on their own in isolation.

V. CONTRIBUTION

This project was a team effort, using tools like Discord, Visual Studio Live Share, and in-person meetings, pair programming was used. The report was written together over Discord. All parts of the project thus remain equally attributed to both participants of the project.

VI. References

[1] GeeksforGeeks, *"K-Nearest Neighbors (KNN) Algorithm in Python for Machine Learning,"* Available: https://www.geeksforgeeks.org/knn-in-python-for-machine-learning/

[2] GeeksforGeeks, *"XGBoost Algorithm in Python,"* Available: https://www.geeksforgeeks.org/xgboost-algorithm-python/

[3] GeeksforGeeks, *"Random Forest Classifier in Python,"* Available: https://www.geeksforgeeks.org/random-forest-classifier-python/

[4] GeeksforGeeks, *"Naive Bayes Classifier in Python,"* Available: https://www.geeksforgeeks.org/naive-bayes-classifier-in-python/.

[5] GeeksforGeeks, *"Confusion Matrix, Accuracy, Precision, Recall, and F1 Score in Machine Learning,"* Available: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/.

[6] GeeksforGeeks, *"Friedman and Nemenyi Test in Python,"*Available:https://www.geeksforgeeks.org/friedman-and-nemenyi-test-in-python/.

VII. Appendix

A. *Data Exploration*

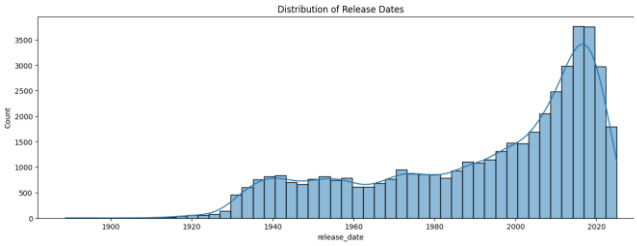

Figure 1 Distribution of release dates

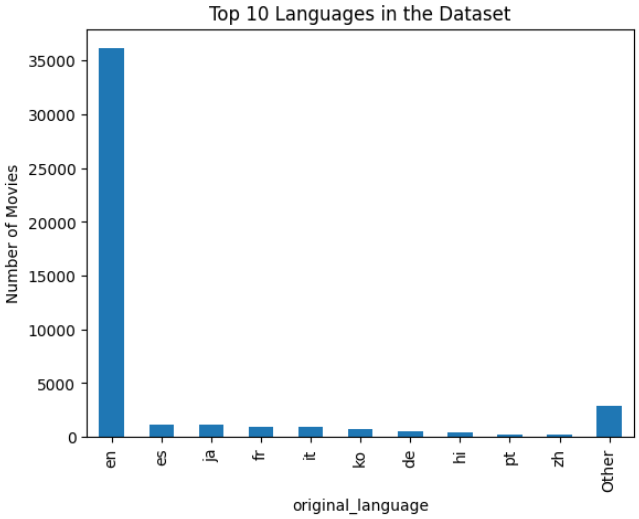

Figure 2 Top 10 languages in dataset

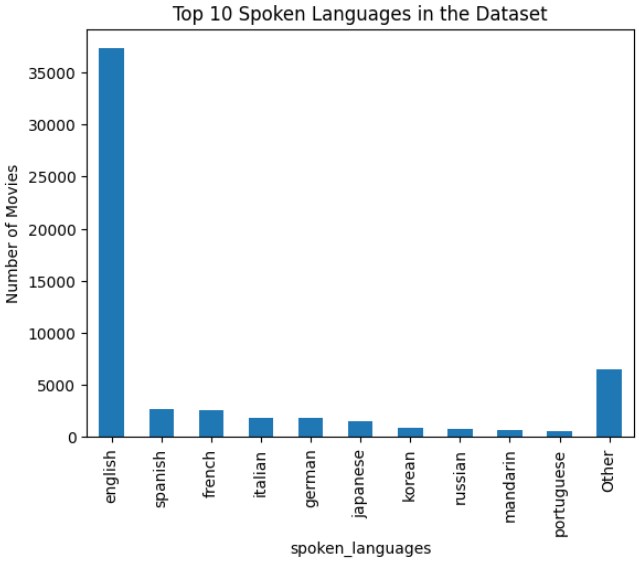

Figure 3 Top 10 spoken languages in the dataset

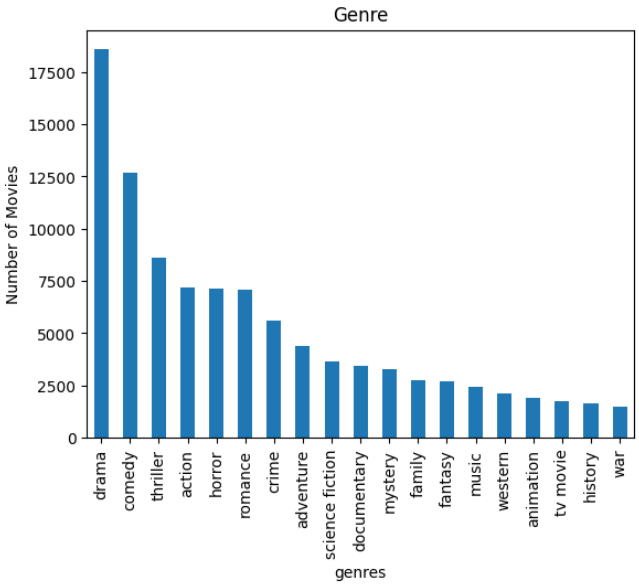

Figure 4 Genres in dataset

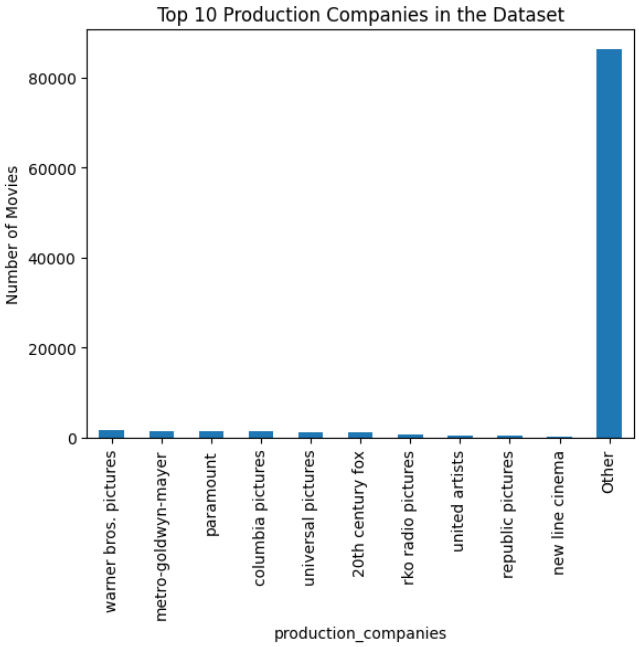

Figure 5 Top 10 production companies in dataset

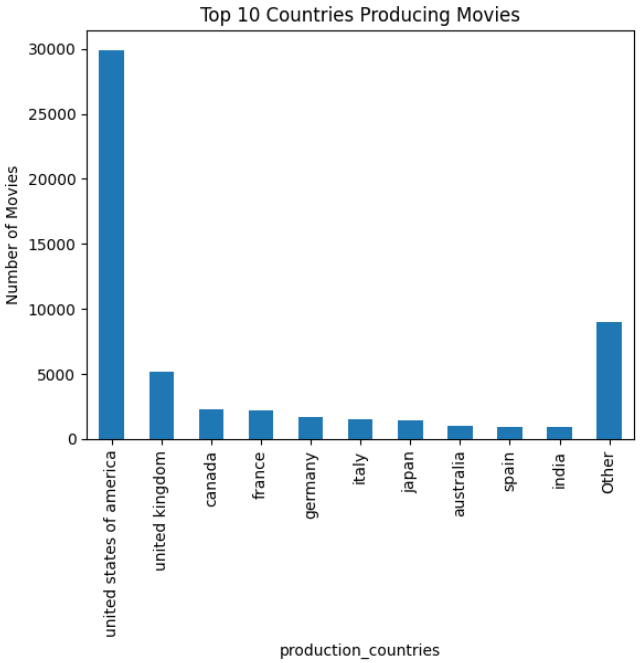

Figure 6 Top 10 countries producing movies

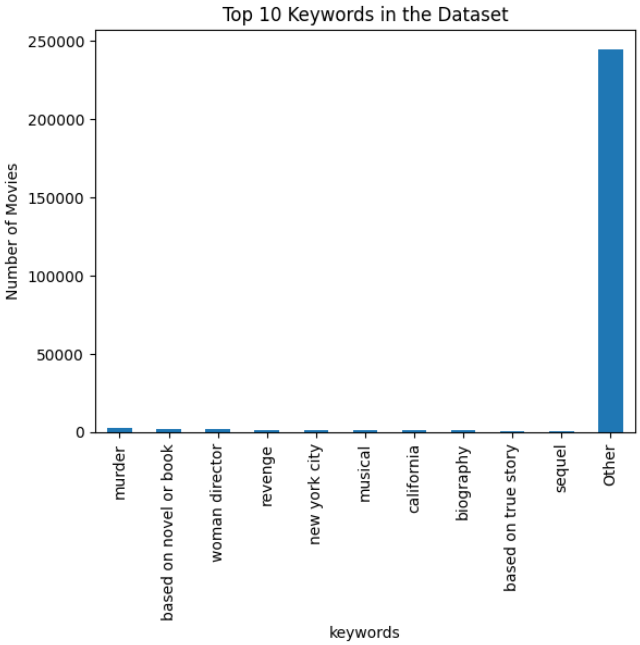

Figure 7 Top 10 keywords in dataset

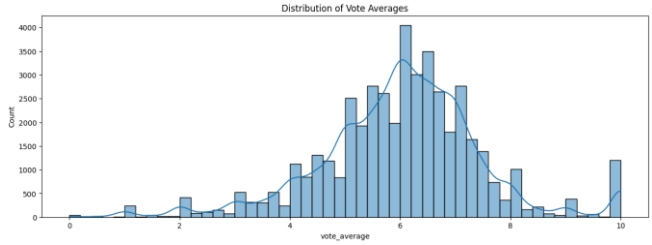

Figure 8 Distribution of vote average

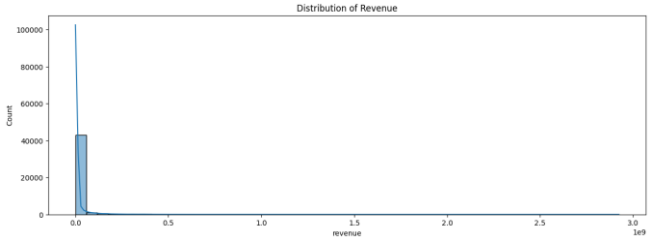

Figure 9 Distribution of revenue

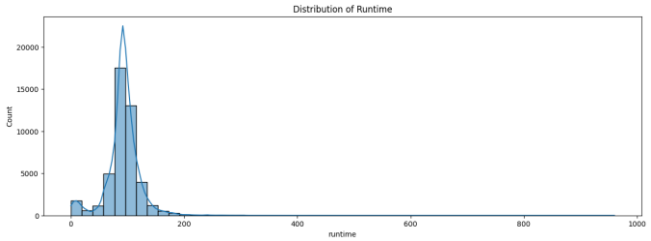

Figure 10 Distribution of runtime

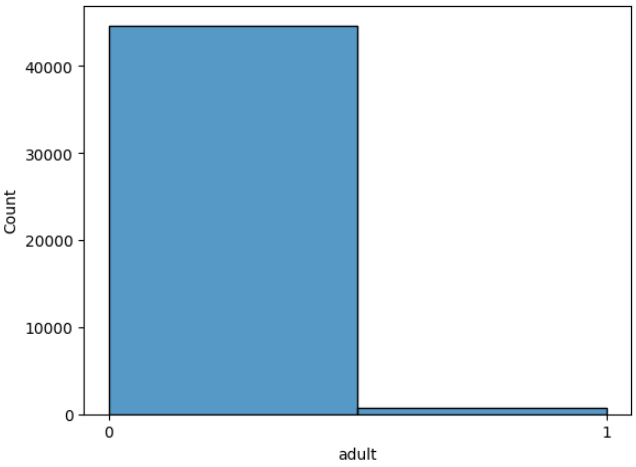

Figure 11 Distribution of adult

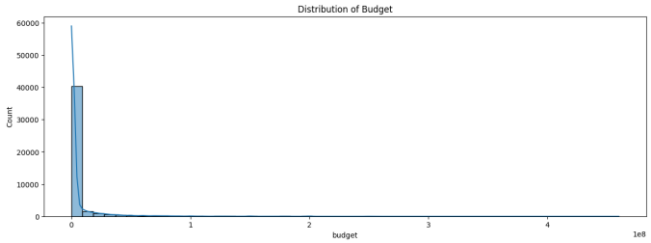

Figure 12 Distribution of budget

B. FINAL MODELS PERFORMANCE

Revenue Results

	Data Set	XGBoost_accuracy	XGBoost_f1	XGBoost_time
0	1	0.89744 (1)	0.89343 (1)	1.17297 (1)
1	2	0.89454 (1)	0.89072 (1)	1.13397 (1)
2	3	0.90047 (1)	0.89613 (1)	1.13197 (1)
3	4	0.89716 (1)	0.89345 (1)	1.11400 (1)
4	5	0.90059 (1)	0.89724 (1)	1.10697 (1)
5	avg	0.89804	0.89419	1.13198
6	std	0.00227	0.00229	0.02295
7	avg rank	1.00000	1.00000	1.00000

Vote Average Results

	Data Set	XGBoost_accuracy	XGBoost_f1	XGBoost_time
0	1	0.72498 (1)	0.72497 (1)	1.43297 (1)
1	2	0.71643 (1)	0.71643 (1)	1.44900 (1)
2	3	0.72498 (1)	0.72492 (1)	1.39197 (1)
3	4	0.72002 (1)	0.72001 (1)	1.40397 (1)
4	5	0.71529 (1)	0.71514 (1)	1.39197 (1)
5	avg	0.72034	0.72029	1.41398
6	std	0.00410	0.00412	0.02305
7	avg rank	1.00000	1.00000	1.00000

Profitable Results

	Data Set	XGBoost_accuracy	XGBoost_f1	XGBoost_time
0	1	0.87965 (1)	0.86856 (1)	1.11397 (1)
1	2	0.88158 (1)	0.87190 (1)	1.12597 (1)
2	3	0.87441 (1)	0.86504 (1)	1.10397 (1)
3	4	0.88172 (1)	0.87444 (1)	1.09697 (1)
4	5	0.88488 (1)	0.87694 (1)	1.09700 (1)
5	avg	0.88045	0.87138	1.10758
6	std	0.00345	0.00421	0.01110
7	avg rank	1.00000	1.00000	1.00000

Figure 13 The three different final models' performance metrics

C. ROC CURVES

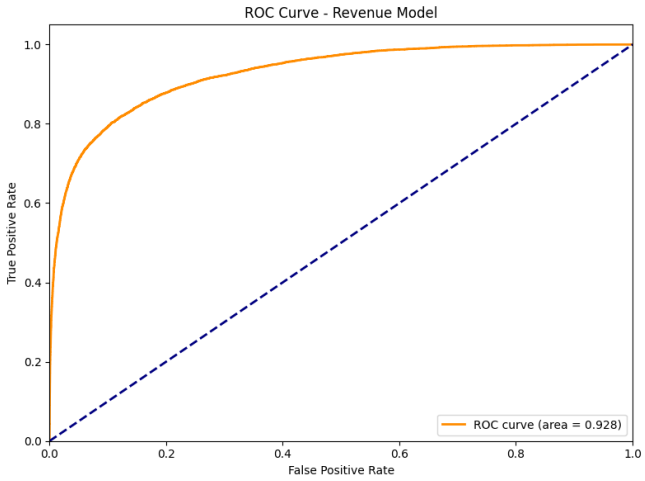

Figure 14 ROC curve for the revenue model

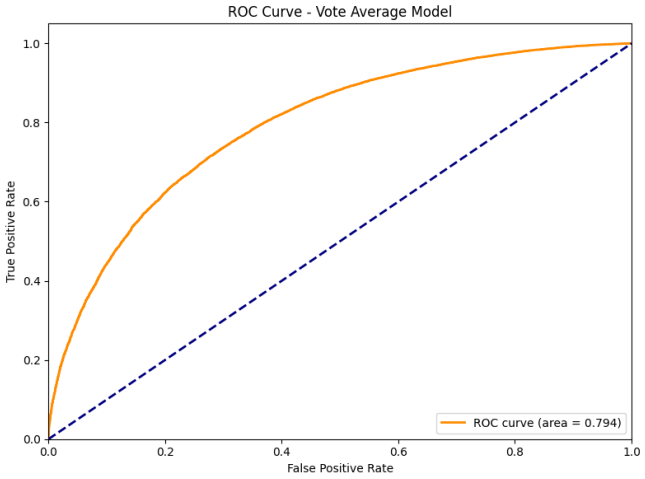

Figure 15 ROC curve for the vote average model

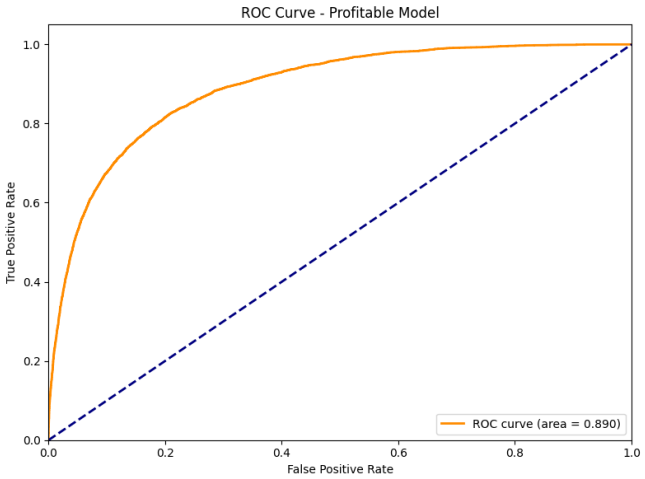

Figure 16 ROC curve for the profitable model

D. FEATURE IMPORTANCE

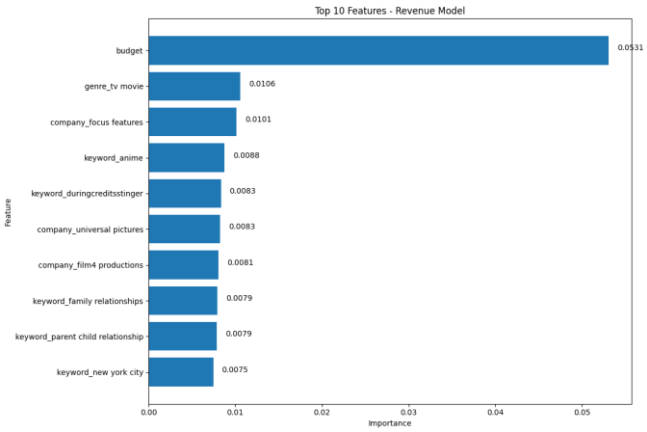

Figure 17 Feature importance graph for the revenue model. Containing the top 10 features

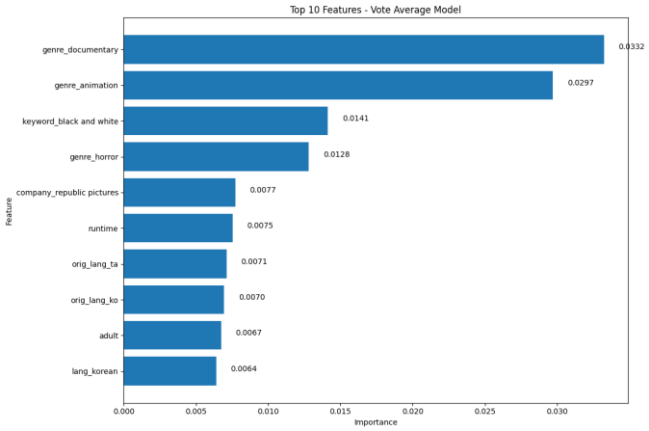

Figure 18 Feature importance graph for the vote average model. Containing the top 10 features

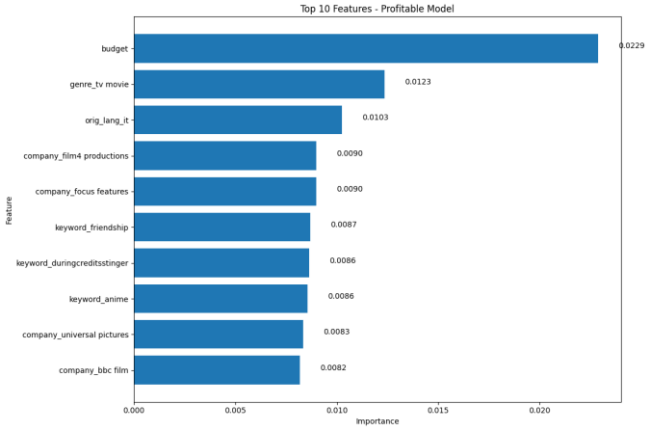

Figure 19 Feature importance graph for the profitable model. Containing the top 10 features