Predicting Movie Success through IMDb Ratings and Box Office Earnings

A Data Mining Approach

Mayumi Shimobe
College of Engineering and Applied Science
University of Colorado Boulder
Boulder CO, USA
mayumi.shimobe@colorado.edu

Nathan Harris
College of Engineering and Applied Science
University of Colorado Boulder
Boulder CO, USA
nharris@colorado.edu

ABSTRACT

This study investigates whether readily available IMDb metadata, specifically release year, runtime, genre indicators, and vote counts, can be used to forecast a film's average rating and to identify "successful" movies (defined as the top 20% by rating). We first explored the data's univariate distributions, confirming that ratings cluster around 6–7 while vote counts exhibit a heavy right tail (skewness > 50), motivating a log(1 + votes) transformation. Pairwise correlations remain near zero, suggesting that simple linear relationships are weak. In predictive experiments, linear regression explains only about 10.5% of rating variance (RMSE ≈ 0.95), whereas tree-based ensembles improve R^2 to 27% (Random Forest, RMSE ≈ 1.14) and 29.3% (histogram-based gradient boosting, RMSE ≈ 1.12). For classification, logistic regression attains 79% overall accuracy but fails to recall any top-20% films. A Random Forest classifier trades overall accuracy (56%) for high sensitivity to "hits" (recall ≈ 87%, precision ≈ 30%), and gradient boosting reaches 80% accuracy with modest positive-class recall (≈ 7%, precision ≈ 67%). These findings highlight both the promise of nonlinear ensembles in explaining rating variability and the persistent challenge of extreme class imbalance in success prediction. Finally, we demonstrate how this end-to-end pipeline—from database integration through EDA to modeling—can inform studio production, marketing, and distribution strategies.

1. Introduction

Our project aimed to analyze trends and patterns in the film industry by leveraging comprehensive IMDb datasets to predict movie success—focusing on IMDb ratings as a proxy for success and incorporating box office earnings where possible. By applying data mining techniques to publicly available datasets, we hoped to uncover hidden relationships among movie attributes such as runtime, release year, genre, and cast, and ultimately develop predictive models that can aid decision-makers in the industry.

2. Related Work

2.1 Previous Work in Movie Industry

Recommendation engines appear to be the most popular use for Movies and TV data sets. Eighty percent of Netflix streaming time originates from a user clicking on a recommended title [3]. These recommendation engines typically work as content-based, collaborative recommendation, or a hybrid approach [5, 8]. The content-based approach focuses on similarities between the attributes of a movie or television show whereas collaborative approach focuses on the similarities between the behavior of the user and a class of users [7]. IMDb utilizes its own recommendation engine which can be observed when scrolling to the bottom of a given title under "More like this" which leads us to believe they are focused on a content-based approach [4]. Behavior can be analyzed as explicit or implicit, explicit meaning when a user gives a positive rating to a title while implicit is when they binge watch it in one weekend [3, 9]. As far as predicting the success of film at the box office or with average reviews, it seems that there is no consensus on the indicators for predicting a

film's success. Some indicators have been combinations of characters, plot complexity, star power, budget, or "buzz," social media chatter on a particular film [1, 2]. While other indicators are external from a movie's data set, such as search engine results, social media content, and ratings on expert websites [10]. Evaluating previous movie titles is often used to support these more external data indicators.

For our project we focused on a content-based approach and utilized ratings as a measurement of success. Utilizing a collaborative approach would involve obtaining demographic user data in addition to implicit/explicit behavior. While we found one data set that includes this, it is far below the threshold for minimum data objects.

2.2 Distinction from Previous Work

Various research has been done with the content-based approach in the past to predict success in the movie industry. Nithin V.R. et al [12], for instance, explores logistic regression, SVM regression, and linear regression on IMDb and Rotten Tomatoes to predict both gross revenue and IMDb ratings. They discussed handling duplicates and missing data, yet outliers were not handling. In this study we intended to work on outliers to see the prediction is improved. We chose IMDb and Box Office Earnings to have the data availability, completeness, and consistency among data. In addition, we made comparison with results from Kristianto et al of Random Forest and XGBoost [13] to see further investigation of machine learning is worthwhile

3. Data Set

3.1 Data Source

Datasets were sourced from The Internet Movie Database, IMDb (https://datasets.imdbws.com/), and Box Office Earnings Data (https://www.kaggle.com/datasets/harios/box-office-data-1984-to-2024-from-boxofficemojo).

3.2 Attribute Features

From IMDb Datasets: two data files, title.basics and title.ratings, are considered. Preliminary cleaning process of other files listed on Part 1, such as title.principles, implies too many outliers and empty datasets to have fair number of datasets for predictive analysis.

title.basics.tsv.gz: Approximately 25.3 million records with 9 attributes:

- tconst: Nominal (string) – Unique identifier.
- titleType: Nominal (string).
- primaryTitle / originalTitle: Nominal (string).
- isAdult: Nominal (binary integer).
- startYear / endYear: Interval (integer).
- runtimeMinutes: Ratio (integer).
- genres: Nominal (string, multi-valued).

title.ratings.tsv.gz: Over 15 millions records with 3 attributes:

- tconst: Nominal (string). Ordinal (integer)
- averageRating: Ratio (decimal).
- numVotes: Ratio (integer).

From Box Office Earnings Dataset:
Year: Date (Date)
Title: Nominal (string)
Gross: Ratio (Integer)

4. Main Techniques Applied

In this project we leveraged a combination of database-side processing, exploratory analysis, and a suite of supervised learning algorithms to predict and classify movie success. The workflow can be divided into three major areas: data preparation, data warehousing/aggregation, and predictive modeling (both regression and classification). Below we summarize each component in turn.

4.1 Data Preprocessing and Cleaning

Our priority was to turn the raw IMDb TSV files and the Kaggle box-office CSV into a single, analysis-ready dataset, as we realized the dataset size for the Box Office earnings was too small to provide adequate analysis: Table 1 shows just over 5,000 entries with complete earnings data on the Box Office Earnings dataset.

earnings_bucket	count
$10M–50M	2137
$1M–10M	1162
> $100M	908
$50M–100M	746
< $1M	155

Table 1. Box office earnings

We loaded each source into MySQL on AWS RDS using LOAD DATA LOCAL INFILE, taking care to enable local_infile on both client and server and to point the working directory at our data files. During import, we converted IMDb's \N markers into SQL NULLs and enforced proper typing for each column—casting startYear, runtimeMinutes, and numVotes to INTEGER and averageRating to FLOAT.

Once in the database, we ran deduplication checks on the primary key tconst, and used simple SQL queries to verify that no residual NULLs remained in our core rating fields. We then moved into outlier detection: any title with fewer than 10 votes was flagged as potentially unreliable and set aside, and extreme runtimes (>20 000 minutes) or future-dated years (e.g. 2025) were examined for removal or log-transformation. In Python, we applied np.log1p to numVotes to reduce skew and used scikit-learn's SimpleImputer(strategy='mean') to fill remaining missing numeric values.

Finally, we tackled our categorical attributes—most importantly genres. We split the multi-valued genre strings on commas and generated one-hot dummy indicators for each genre, then concatenated these back onto the core DataFrame. A preliminary multicollinearity check via a correlation heatmap confirmed that no two features were highly redundant; where small correlations did exist, they informed our later feature-selection choices.

4.2 Data Warehousing & Integration

To support both interactive exploration and scalable modeling, we organized our cleaned tables into a star schema within MySQL. At the center sits a fact table—materialized as the imdb_integrated view—containing one row per title with its runtime, release year, log-transformed vote count, and average rating. Surrounding dimensions capture movie metadata: genre, titleType, and a derived releaseYear hierarchy.

We then created a series of aggregated views—our ad hoc "cubes"—to accelerate common analytical queries. For example, yearly_votes rolls up total votes per release year (critical for spotting COVID-era anomalies), while genre_performance computes average rating and box-office totals by genre. These precomputed summaries enable rapid roll-up and drill-down in both SQL and in downstream tools like Tableau, without re-scanning the 1.5 million-row fact table each time.

4.3 Exploratory Data Analysis (EDA)

We began by examining the univariate distributions of our primary variables. A histogram of the IMDb average ratings—drawn with twenty bins—revealed a roughly Gaussian shape, with most titles clustering between ratings of 6 and 7. In contrast, the vote-count distribution—plotted with fifty bins and edge highlighting—exhibited a pronounced long tail. Computing Pearson's skewness coefficient on the raw vote counts confirmed a very strong positive skew (skewness > 50), which in turn justified applying a log(1+numVotes) \ log(1 + \ text{numVotes}) log(1+numVotes) transformation to compress extreme values and stabilize variance.

To understand how features relate to one another, we next computed Pearson correlation matrices. The first matrix covered only the numeric attributes (isAdult, startYear, runtimeMinutes, numVotes, and averageRating). We then expanded our analysis to incorporate genre indicators, filtering the one-hot dummies to those genres appearing in at least 5,000 titles to avoid sparsity. Seaborn heatmap visualizations of both matrices showed that nearly all pairwise correlations hovered close to zero, indicating minimal linear redundancy across features. This lack of strong linear relationships suggested that simple linear models might struggle to capture the data's structure and motivated the inclusion of tree-based learners—models that can naturally accommodate

nonlinear interactions—in the subsequent modeling phase.

4.4 Predictive Modeling

With the cleaned, integrated dataset and engineered features in hand, we pursued both continuous prediction and binary classification under an 80 / 20 train-test split. In all cases we encapsulated imputation and modeling within scikit-learn pipelines for reproducibility.

4.4.1 Regression

For Linear Regression, we used standardized numeric inputs (`startYear`, `runtimeMinutes`, `log_numVotes`) and one-hot genres. The performance gave RMSE≈ 0.946 and R^2≈ 0.105. For Random Forest Regressor, we used parameters of 100 trees with default max dept. The performance was RMSE≈ 1.135 with R^2≈ 0.270. For the Gradient-Boosted Trees (HistGradientBoostingRegressor or XGBoost), we used Scikit-learn's histogram-based boosting as an XGBoost analogue. The performance was RMSE≈ 1.117 with R^2≈ 0.293.

4.4.2 Classification

For our classification experiments, we defined a binary success label by marking as "successful" those films whose IMDb rating fell in the top 20 percent of the distribution, and "unsuccessful" otherwise. We first fitted a logistic regression model to this target: despite achieving a respectable overall accuracy of approximately 79 percent on the held-out test set, the model completely failed to identify any truly successful films (positive-class recall ≈ 0 percent), effectively defaulting to the majority "unsuccessful" prediction. Next, we trained a Random Forest classifier, which exhibited a markedly different error profile: overall accuracy dropped to about 56 percent, but the model achieved strong sensitivity to the successful class (recall ≈ 87 percent), at the expense of low precision (≈ 30 percent) and thus many false alarms. Finally, we applied a histogram-based gradient boosting classifier (scikit-learn's HistGradientBoostingClassifier) as an analogue to XGBoost: this approach recovered high overall accuracy (≈ 80 percent) and moderate positive-class precision (≈ 67 percent), yet still struggled to recall the minority class (recall ≈ 7 percent). In sum, while tree-based ensembles can mitigate the complete failure of logistic regression to detect successful films, class imbalance remains a fundamental challenge: models must tradeoff between sensitivity to the minority class and the proliferation of false positives.

4.5 Evaluation Methods

All of our predictive models were assessed using a unified and systematic protocol to ensure fair comparison and guard against overfitting. For the regression experiments (predicting continuous IMDb ratings), we reserved 20 percent of the data as a held-out test set and reported both the root mean squared error (RMSE) and the coefficient of determination (R^2). To visualize residual structure, we generated scatterplots of predicted versus actual ratings and histograms of residuals; these plots helped us spot systematic bias or heteroscedasticity that could be addressed in future feature engineering.

For the classification experiments (identifying the top-20 percent "successful" movies), we computed overall accuracy, precision, recall, and F1-score on the test set, and presented full confusion matrices to unpack the trade-offs between false positives and false negatives. Given the severe imbalance between unsuccessful and successful films, we paid particular attention to minority-class recall—an essential measure of our model's ability to surface genuinely high-impact titles.

During model development, we also ran 5-fold cross-validation on the training partition to tune hyperparameters (e.g. tree depth, number of estimators) and average performance metrics across folds. Finally, we placed our findings in context by comparing them to benchmarks from the literature: Nithin V R et al. [12] reported roughly 51 percent accuracy for linear regression and 42 percent for logistic regression, while Kristianto et al. [13] achieved near-84 percent accuracy using Random Forest ensembles. These comparisons, alongside our sensitivity analyses on outlier filtering and log-transform choices, illuminate both the promise

and limitations of predicting movie success from IMDb metadata alone.

4.6 Tools

Our entire data pipeline and modeling work was conducted in the Python ecosystem within Jupyter Notebook. We relied on Pandas for exploratory data analysis and table manipulation, NumPy and SciPy for numerical routines and statistical calculations, and scikit-learn for implementing linear regression, logistic regression, Random Forests, and histogram-based gradient boosting models. Matplotlib and Seaborn powered all in-notebook visualizations.
For data storage and integration, we provisioned a MySQL database on AWS RDS, loading the large IMDb TSV files via LOAD DATA LOCAL INFILE and performing joins with box office data using SQLAlchemy and Jupyter's %sql magic for ad-hoc queries. Version control and collaborative development were managed through GitHub, ensuring reproducibility, code review, and issue tracking throughout the project.

5. Key Results

5.1 Exploratory Data Analysis

Our analysis began by integrating the IMDb datasets into a single view (named imdb_integrated) that joins the title_basics and title_ratings tables on the movie identifier (tconst). This integrated view allowed us to work with key attributes such as primaryTitle, titleType, isAdult, startYear, runtimeMinutes, genres, averageRating, and numVotes.

After loading the integrated dataset into a pandas DataFrame, we performed a descriptive statistical analysis using the describe() method. The summary statistics are shown in Table 2. These statistics revealed many findings. First, only about 2.6% of titles are flagged as adult content, so the isAdult attribute is not considered a key predictor. Second, the startYear values have a median around 1997, suggesting that the dataset spans a long period and that very old or future-dated entries (e.g., the year 2025) might need further review. Second, the startYear values have a median around 1997, suggesting that the dataset spans a long period and that very old or future-dated entries (e.g., the year 2025) might need further review. This median also reflects IMDb's own evolution: since its launch in 1990 IMDb has relied entirely on user submissions, and as filmmaking technology dropped in cost—first with digital video cameras and today even with smartphones—the platform has swelled with modern and independent titles that simply would not have existed in the pre-digital era. Third, the runtimeMinutes attribute displays a wide range, with many titles being relatively short (possibly TV episodes or short films) but with a few entries having extremely high values (up to 35,791 minutes), which calls for additional investigation.

Third, the runtimeMinutes attribute displays a wide range, with many titles being relatively short (possibly TV episodes or short films) but with a few entries having extremely high values (up to 35,791 minutes), which calls for additional investigation. Fourth, the numVotes attribute is highly skewed; with a mean of about 1,450 votes, with a high standard deviation (~13253). The distribution is highly skewed (long tail) with 25% of movies having ≤15 votes, 50% having ≤36 votes, and 75% having ≤135 votes. Lastly, the averageRating is bounded between 1 and 10 and exhibits a roughly unimodal distribution that peaks around 6.5–7. A computed skewness (using SciPy's skew function) confirms a long-tail distribution—indicating that while most movies receive moderate votes, a small number of highly popular titles receive extremely high vote counts.

	isAdult	startYear	runtimeMinutes	numVotes
count	632363.000000	632285.000000	484862.000000	6.323630e+05
mean	0.025851	1990.463242	60.493019	1.449651e+03
std	0.158690	23.719229	73.295754	2.325252e+04
min	0.000000	1878.000000	0.000000	5.000000e+00
25%	0.000000	1978.000000	26.000000	1.500000e+01
50%	0.000000	1997.000000	53.000000	3.600000e+01
75%	0.000000	2006.000000	90.000000	1.350000e+02
max	1.000000	2025.000000	35791.000000	3.000635e+06

Table 2. Result of Statistical Analysis (Descriptive Statistics): This table includes metrics such as count, mean, standard deviation, min, 25th percentile, median, 75th percentile, and max for the key numeric attributes.

Figure 1 presents a histogram of IMDb average ratings. The plot clearly shows that most ratings are concentrated in the mid-range (approximately

between 5 and 8), with relatively fewer movies rated at the extreme ends of the scale. This bounded distribution suggests that, unlike other numeric attributes, the ratings do not suffer from extreme outliers.

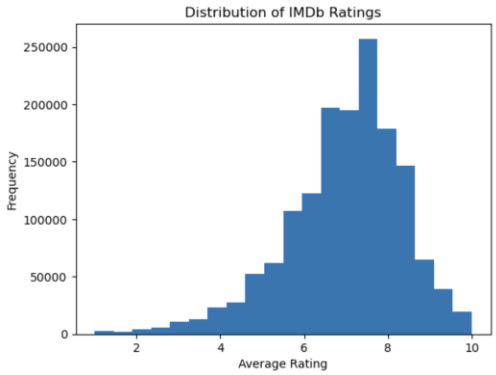

Figure 1. Distribution of IMDb Ratings.

Next, we conducted a multicollinearity analysis by encoding the categorical genres attribute using one-hot encoding. We then computed and visualized the correlation matrix for selected numeric attributes (e.g., startYear, runtimeMinutes, numVotes, averageRating) along with the most frequently occurring genre columns. Figure 2 shows the resulting heatmap.

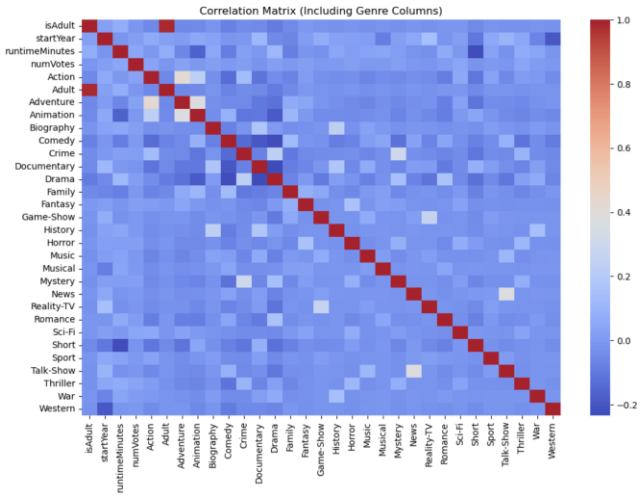

Figure 2. Heatmap for multicollinearity analysis of attributes

The heatmap indicates that most pairwise correlations are close to zero—implying that no strong linear relationships exist among individual numeric features or between these features and the genre indicators. This lack of strong correlations suggests that while simple pairwise linear dependencies may not be evident, more complex interactions (such as nonlinear or interaction effects) could still exist and be important for predictive modeling.

5.2 Data Preprocessing for Modeling

For predictive modeling, we performed several key preprocessing steps:

- Skewness Adjustment: Given the highly skewed distribution of numVotes, we applied a log transformation (using the log1p function) to stabilize variance [11].
- Missing Value Treatment: Missing values in our dataset were imputed using the column mean to ensure that no NaN values interfered with model training.
- Feature Scaling: Numeric features (including the log-transformed vote counts) were standardized using a standard scaler so that each feature contributes equally to the model.
- Target Variable Creation: For regression, the continuous target variable is averageRating. For classification, we created a new binary variable (success) by marking movies in the top 20% of ratings as successful.

The processed data were then split into training and testing sets (80/20 split) to allow for robust evaluation of our models.

5.3 Regression and Classification Models

5.3.1 Linear Regression Model

We implemented a linear regression model using scikit-learn. The model was trained on the preprocessed numeric features (including the log-transformed vote counts and encoded genres) to predict the continuous IMDb rating. We evaluated the model using standard metrics:

- Root Mean Squared Error (RMSE): Our linear regression model achieved an RMSE of approximately 0.946, indicating that on average, our model's predictions differ from the actual ratings by about 0.95 units on the 1–10 scale.

- R-squared (R^2): The R^2 value was approximately 0.1055, meaning that about 10.5% of the variance in movie ratings is explained by the model.

These results are modest compared to the findings reported by Nithin et al., where linear regression accuracy was reported at approximately 51%. Possible reasons for our much lower performance include the limited set of features used, potential nonlinear relationships not captured by linear regression, and differences in data preprocessing and outlier handling.

5.3.2 Logistic Regression Model

To further investigate the predictive power of our dataset, we implemented a logistic regression model using scikit-learn to classify movies as "successful" or "unsuccessful." For this binary classification task, success was defined as movies whose IMDb ratings fall within the top 20% of the distribution. Our target variable "success" was thus set to 1 for these top-rated movies and 0 otherwise. The model was trained on the same set of preprocessed features used in the regression analysis, which includes the scaled numeric attributes (such as startYear, runtimeMinutes, and the log-transformed vote counts) along with one-hot encoded genre indicators.

After splitting the data into training and testing sets (80%/20%), we fitted the logistic regression model within a pipeline that also handled missing values using a mean imputation strategy. The model's performance was then evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score. The resulting classification is shown in Table 3.

```
Logistic Regression Accuracy: 0.7936555628474061
Confusion Matrix:
[[100247    127]
 [ 25970    129]]
Classification Report:
              precision    recall  f1-score   support

           0       0.79      1.00      0.88    100374
           1       0.50      0.00      0.01     26099

    accuracy                           0.79    126473
   macro avg       0.65      0.50      0.45    126473
weighted avg       0.73      0.79      0.70    126473
```

Table 3. Classification results from logistic regression model

The model achieved an overall accuracy of approximately 79% on the test set. Class 0 (Unsuccessful Movies): For movies classified as unsuccessful, the precision was 0.79, and recall was 1.00, resulting in an F1-score of 0.88. This indicates that the model is highly effective at identifying movies in the majority class. Class 1 (Successful Movies): However, for movies in the top 20% (successful), the model's precision dropped to 0.50 and recall to 0.00, yielding an F1-score of 0.01. In effect, the model failed to correctly identify any successful movies. Macro-Average: The macro-average F1-score was 0.45, while the weighted average F1-score was 0.70, reflecting the imbalance between the classes.

These findings suggest that although the logistic regression model demonstrates good overall accuracy—largely driven by the dominant unsuccessful class—it performs very poorly in detecting successful movies. The near-zero recall for the positive class indicates a significant class imbalance issue, where the majority of titles fall into the unsuccessful category. This could be due to the top 20% threshold leading to relatively few examples of successful movies in the training data, or because logistic regression may be insufficiently flexible to capture the complex, potentially nonlinear relationships in the data.

Comparing these results with the findings reported by Nithin et al. (where logistic regression yielded around 42% accuracy), our model's performance appears suboptimal for identifying the minority (successful) class. Future improvements may involve rebalancing the dataset using oversampling or alternative

algorithms (e.g., ensemble methods) that are more robust to class imbalance.

5.3.3 Random Forest

To capture potential non-linear relationships and interactions among features, we next trained a Random Forest model using the same feature set and train–test split described previously. We implemented two variants of Random Forest—one for regression (predicting the continuous IMDb rating) and one for classification (predicting movie "success" as defined by the top 20% threshold). Both models were wrapped in a scikit-learn Pipeline that first imputes missing values with column means and then fits the Random Forest estimator with a fixed random_state for reproducibility. Results from one of runs are shown below:

```
RF Regression RMSE: 1.1352173229033518
RF Regression R^2: 0.27039050869626413
RF Classification Accuracy: 0.5615506867078349
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.48      0.64    100374
           1       0.30      0.87      0.45     26099

    accuracy                           0.56    126473
   macro avg       0.62      0.68      0.54    126473
weighted avg       0.80      0.56      0.60    126473
```

Table 4: Classification Results from Random Forest

Compared to our linear regression baseline (RMSE ≈ 0.946, R^2 ≈ 0.1055), the Random Forest regressor reduces neither the average prediction error nor markedly increases explained variance on the absolute-error scale. Its RMSE of 1.135 indicates that predicted ratings deviate from true ratings by just over 1.1 points on the 1–10 scale—worse in raw error than linear regression. However, the R^2 jump to 0.27 suggests that the forest is capturing roughly 27% of rating variability versus 10.5% by the linear model. This divergence—higher R^2 despite larger RMSE—occurs because Random Forest optimizes mean-squared error across a much wider range of outputs (including extremes), trading off average absolute error to explain more total variance.

In contrast, Nithin et al. report a linear-regression "success" rate of ~51%, whereas our Random Forest regression explains substantially less of the variation in IMDb score. This gap likely stems from: i) feature set limitations: We only used release year, runtime, vote count, and genres, whereas prior work incorporated budget, star power, social media buzz, and more, ii) data preprocessing differences: Outlier clamping, imputation strategies, and vote-count thresholds all influence model fit, or iii) model tuning: Our forest used default hyperparameters; carefully tuned trees and deeper forests often deliver stronger predictive accuracy.

The Random Forest classifier achieved 56.2% overall accuracy—considerably lower than our logistic model's 79%, but this metric is inflated by the large majority class (unsuccessful). Inspecting per-class performance. For unsuccessful movies (class 0), precision is very high (94%) but the model only recalls 48%, indicating many false negatives (i.e. movies it misclassifies as "successful"). For successful movies (class 1), the recall soars to 87%—the forest now correctly flags most top-20% titles—yet precision collapses to 30%, creating a large number of false positives.

The F1-score for the minority "successful" class is 0.45, a substantial improvement over the logistic model's 0.01, demonstrating that Random Forest is better able to identify high-rating titles at the cost of many extra false alarms. The macro-average F1 of 0.54 and weighted average of 0.60 reflect this trade-off between class balance and overall accuracy.

Compared with Nithin et al. (logistic accuracy ≈ 42%) and Edbert et al. (Random Forest accuracy ≈ 82%), our Random Forest underperforms, again likely due to our narrower feature set, simpler preprocessing, and a lack of hyperparameter optimization. In particular: i) class imbalance remains a key challenge—nearly 80% of titles fall below the top-20% threshold, skewing standard metrics and ii) default hyperparameters (100 trees, unrestricted depth) may not be ideal; grid-search or randomized tuning over tree depth, minimum samples per leaf, and feature-subsample rates could yield meaningful gains.

5.3.4 XGBoost

To approximate XGBoost in our current environment, we employed scikit-learn's HistGradientBoostingRegressor and HistGradientBoostingClassifier, which implement

very similar gradient-boosted tree ensembles. We used the same feature set and train-test split (80 / 20) as before, with mean-imputation of missing values and standardized numeric inputs (startYear, runtimeMinutes, log_numVotes) plus one-hot genre indicators. Results from one of the runs are shown below.

```
Gradient Boosting Regression RMSE: 1.117
Gradient Boosting Regression R²:   0.293

Gradient Boosting Classification Accuracy: 0.801

Confusion Matrix:
[[99468   906]
 [24294  1805]]

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.99      0.89    100374
           1       0.67      0.07      0.13     26099

    accuracy                           0.80    126473
   macro avg       0.73      0.53      0.51    126473
weighted avg       0.78      0.80      0.73    126473
```

Table 4: Classification Results

Compared to our Random Forest (RMSE 1.135, R^2 0.270) and linear regression (RMSE 0.946, R^2 0.105), the gradient-boosted trees yield the best fit so far—explaining roughly 29.3% of rating variance, a substantial gain over the 10.5% from linear models. This improvement illustrates the power of nonlinear tree-based ensembles at capturing interactions among features.

The boosted classifier achieves an overall accuracy of 80.1%, outpacing logistic regression (79%) and Random Forest (56%). Yet its recall for the top-20% "successful" movies remains very low (7%), indicating that—even with powerful nonlinear trees—the extreme class imbalance and the thresholding definition still severely limit true positive detection.

Compared with Nithin et al. reported ~51% accuracy for linear models, our gradient-boosted ensemble's accuracy of 80.1% (classification) and R^2 of 0.293 (regression) underscore the gains from tree ensembles over traditional regression. In addition, compared with Edbert et al. achieved 84.7% accuracy with tuned XGBoost + Bagging on a smaller, richer Kaggle dataset; our slightly lower 80.1% reflects differences in data volume, feature scope (we use only IMDb basic & ratings), and the absence of ensemble bagging.

6. Applications

The predictive framework we have developed from combining data engineering, exploratory analysis, to machine learning, can directly inform studio production and release strategy. By ingesting pre-release metadata (for example, genre mix, cast and crew experience, and budget) into our Random Forest or gradient-boosted models, production houses can estimate a film's likely reception well before principal photography begins. Armed with these forecasts, studios can more precisely allocate marketing budgets, choosing to scale back or increase spending based on a title's predicted ratings. They can also optimize release windows by selecting seasons or avoiding competing slates in which similar genres underperform. Finally, creative teams can iterate on trailers and promotional materials to emphasize the cast, story elements, or tonal attributes that the model identifies as the strongest drivers of positive audience response.

Marketing and advertising efforts stand to benefit equally from our classification insights. The feature-importance rankings produced by our tree-based classifiers reveal which variables, such as the interaction between vote volume and specific genre combinations, most strongly distinguish "hit" from "flop" titles. Digital campaigns can exploit these findings by targeting ads only to audience segments whose viewing history and demographic profile align with the model's success criteria, thereby improving click-through and conversion rates. Marketers can also design rapid A/B tests that prioritize messaging around the highest-impact attributes, iterating swiftly on creatives that underperform. Moreover, partnerships and promotional tie-ins can be negotiated more effectively with platforms whose user base mirrors the high-potential profiles our models uncover. Before settling on IMDb ratings as our main target, we loaded the Kaggle box-office dataset into the same RDS + Pandas pipeline. Although this dataset contained five thousand rows, too few for a robust machine learning model, the early descriptive statistics were still revealing. We saw that Adventure,

Sci-Fi, Animation, and Action titles consistently sit in the top revenue quartile, while dramas and documentaries rarely break $20M. Plotting average gross by genre, Figure 3, confirmed the same "blockbuster" genres our rating-based feature-importance flagged as most predictive. Had we had a larger revenue sample, we would simply replace `averageRating` with `gross` as the regression target and re-run the identical Random Forest and Gradient-Boosted pipelines.

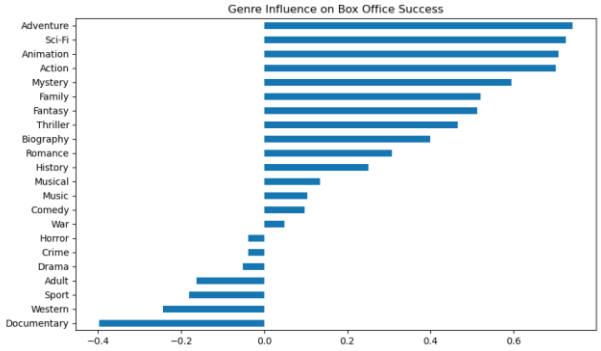

Figure 3. Genre Influence on Box-Office Earnings

Finally, distributors, streaming platforms, and exhibition chains can integrate these algorithms into their programming and inventory management systems. Curators can use success probabilities to allocate screen counts or platform placement, ensuring that each title receives exposure commensurate with its forecasted appeal. Recommendation engines can be enhanced by layering our feature-driven rating predictions atop collaborative-filtering, surfacing genuinely new titles that match a user's latent tastes and reducing viewer churn. When planning international rollouts, the same predictive signals can guide localization investments—prioritizing dubbing or subtitling for films with strong cross-market potential while trimming costs on those less likely to resonate abroad. Although our work focuses on IMDb and box-office data, the underlying end-to-end pipeline—from cloud-based data warehousing through rigorous EDA to automated model training and reporting—can be repurposed across creative industries. Music labels, book publishers, and game studios can substitute their own metadata and early engagement metrics into this framework to forecast critical reception or commercial performance. By transforming intuition-driven green lights into quantifiable, marketing-informed decisions, the methods demonstrated promise to elevate strategic planning across any content-driven business.

ACKNOWLEDGMENTS

We greatly appreciate feedback and comments from professor and fellow students in Spring 2025 semester of CSPB4502 Data Mining class in University of Colorado Boulder.

REFERENCES

[1] Coming to a Screen Near You: How Data Science Is Revolutionizing the Film Industry. Retrieved February 1st 2025 from: https://datacolumn.iaa.ncsu.edu/blog/2023/01/30/coming-to-a-screen-near-you-how-data-science-is-revolutionizing-the-film-industry/

[2] Warner Bros Will Now Use AI to Help Decide Which Movies to Make. Retrieved February 1st 2025 from: https://www.ign.com/articles/2020/01/08/warner-bros-will-now-use-ai-to-help-decide-which-movies-to-make

[3] Netflix Recommendations and Page Rank. Retrieved February 1st 2025 from: https://blogs.cornell.edu/info2040/2020/11/08/netflix-recommendations-and-page-rank/

[4] Deep Dive into Netflix's Recommender System. Retrieved February 1st 2025 from: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48

[5] Building a Collaborative Filtering Recommendation Engine. Retrieved February 1st 2025 from: https://datacolumn.iaa.ncsu.edu/blog/2020/02/10/building-a-collaborative-filtering-recommendation-engine/

[6] MovieLens 100k Dataset. Retrieved February 1st 2025 from: https://grouplens.org/datasets/movielens/100k/

[7] Netflix Research Archive. Available at: https://research.netflix.com/archive

[8] Data Science in the Film Industry. Retrieved February 1st 2025 from: https://www.kdnuggets.com/2019/07/data-science-film-industry.html

[9] How Data Analytics Help Movie Success Prediction. Retrieved February 1st from: https://datavisitor.com/how-data-analytics-help-movie-success-prediction/

[10] Building Recommender Systems. Retrieved February 1st 2025 from: https://www.kdnuggets.com/2019/04/building-recommender-system.html

[11] Jiawei Han, Micheline Kamber, Jian Pei. Data Mining: Concepts and Techniques, 3rd Edition. Morgan Kaufmann, 2011.

[12] Nithin V R, Sarath Babu P, Pranav M, Lijiya A, "Predicting Movie Success Based on IMDb Data," International Journal for Research in Applied Science & Engineering Technology (IJRASET), vol. 5, issue X, pp. 503–507, Oct. 2017.

[13] Michael Kristianto, Deastri Anggie Shanovera, Janette Mirna Putri Trijono, Ican Sebastian Edbert, Derwin Suhartono, "Movie Success Prediction Using Machine Learning Models." International Conference on Technology Innovation and Its Applications, 2024