Comparative Analysis of Learning Models for IMDB Movie Rating Prediction

Kaan Baykal
Computer Engineering
TOBB ETU
Ankara, Turkey
kbaykal@etu.edu.tr

Beril Aydın
Computer Engineering
TOBB ETU
Ankara, Turkey
berilaydin@etu.edu.tr

Anıl Özişler
Computer Engineering
TOBB ETU
Ankara, Turkey
aozisler@etu.edu.tr

Abstract*—In the modern entertainment industry, predicting audience reception is a critical factor for strategic decision-making. This study presents a comparative analysis of machine learning algorithms to predict IMDb movie ratings using a processed dataset of feature films. The research evaluates the performance of three distinct models: Ridge Regression, Random Forest, and XGBoost, aiming to capture the complex, non-linear relationships between movie metadata and viewer satisfaction. A comprehensive feature engineering pipeline was implemented, incorporating TF-IDF vectorization for unstructured plot summaries, Smoothed Target Encoding for high-cardinality categorical features (directors and cast), and Multi-Hot Encoding for genres. Experimental results demonstrate that XGBoost achieved the superior performance, outperforming both linear and bagging architectures by effectively minimizing residual errors through gradient boosting. The study concludes that while runtime shows negligible linear correlation with quality, feature interactions involving popularity (vote counts), genre complexity, and star power are the most significant predictors of a movie's success.

I. Introduction

In an era of rapid and diverse content production, the ability to anticipate a production's success has become strategically vital for producers, distributors, and streaming platforms. This study focuses on predicting IMDB ratings, treating the task as a regression problem since the rating is a continuous numerical value. The primary objective is to model the complex relationships between accessible metadata—such as genre, cast, and descriptions—and the final audience ratings, thereby providing an analytical framework to quantify how specific features contribute to a production's acceptance. To evaluate the predictive accuracy of the developed models, Root Mean Squared Error (RMSE) is employed as the primary performance metric, supplemented by Mean Absolute Error (MAE) and the Coefficient of Determination R^2 for a comprehensive statistical assessment.

II. Literature Review

Studies on the use of machine learning techniques in film rating prediction were reviewed to support this project.

R. K. Bansal *et al.* (2019) presented a comparative analysis of various regression and classification algorithms for predicting IMDb movie ratings, concluding that ensemble methods yield superior accuracy. Their paper emphasizes the importance of preprocessing and feature selection, which directly aligns with the objectives of this project. [1]

F. Geng *et al.* (2014) explored how movie-related variables such as genre, cast, and director influence popularity and performance metrics like box office and user ratings. Their findings demonstrated that structured metadata can successfully explain audience preferences. [2]

H. S. Patel and P. J. Patel (2016) applied supervised learning techniques to predict IMDb ratings based on movie metadata. They highlighted the significance of preprocessing steps like text vectorization and normalization. Their results indicate that machine learning models can effectively capture non-linear relationships, supporting the feasibility of metadata-based rating prediction systems. [3]

III. Dataset, Data Characteristics, Features

A) DataSource

[Kaggle – TV and Movie Metadata with Genres and Ratings (IMDB)]

B) Collection Method

The dataset was compiled from IMDb's public database and verified using multiple entertainment data sources. It is freely available as a structured CSV file and does not require scraping.

C) Dataset Description

1) Domain: Entertainment Analytics / Media Studies
2) File Type: CSV (~40MB)
3) Number of Records: 100,000+ entries
4)Columns:

- movie: Movie name
- genre: Main and secondary genres
- runtime: Duration in minutes
- certificate: Age rating (e.g., PG, R, TV-MA)
- Target Variable: rating (numerical value between 0–10)
- star: Actors of the movie
- description: Short brief about the movie
- votes: Number of IMDb user votes
- director: Director of the movie

5) Features:

- runtime: The duration of the film in minutes, used directly as a numerical input to assess the relationship between length and quality.
- log_votes: The natural logarithm of the vote count, transformed to normalize the skewed popularity distribution and reduce the impact of outliers.

- genre_count: An engineered feature representing the total number of genres assigned to a movie, capturing the complexity of its categorization.
- director_score: A numerical value derived from the director's name, representing the weighted average rating of their previous films (smoothed target encoding).
- cast_score: A composite score representing the "star power" of the lead actors, calculated from the historical average ratings of the cast members.
- genre_indicators: Binary features resulting from multi-hot encoding (e.g., genre_Action), indicating the presence or absence of specific genres.
- tfidf_keywords: Sparse numerical vectors derived from the plot description, capturing the importance of specific semantic words (e.g., "war", "love") using TF-IDF.
- rating (Target): The weighted average viewer score (1.0–10.0) that serves as the ground truth label for training the regression models.

D) *Preprocessing Steps*

1) Data Cleaning and Formatting (Raw Data Handling)

The initial phase focused on parsing inconsistent string formats and handling missing data, utilizing custom Python scripts (data_clean.py) and Regular Expressions (Regex).

a) String Parsing & Artifact Removal:

- Runtime & Votes: The runtime column contained non-numeric units (e.g., "120 min") and votes contained formatting commas (e.g., "1,500,000"). We used Regex (r'(\d+)') to extract pure integers and converted them to Int64 types.
- List Cleanup (Stars & Directors): The stars and director columns contained Python list-like artifacts (e.g., ['Name', 'Name']) represented as strings. We stripped brackets ([]), quotes (""), and standardized inconsistent comma spacing (e.g., correcting ,, to ,).
- Junk String Removal: Placeholder text such as "Add a Plot", "See full summary", and "Violent" were identified as noise in the description column and replaced with NaN to prevent misleading text analysis.
- Whitespace Handling: Newline characters (\n) and trailing whitespace were stripped from the genre column to ensure consistent categorization.

b) Handling Missing Values:

- Column Exclusion: The certificate column was permanently dropped as it contained >80% missing values, making imputation unreliable.
- Row Deletion: After cleaning, rows containing null values in critical target or feature columns (rating, votes, runtime, genre) were dropped. Duplicate entries were also identified and removed

E) *Feature Descriptions & Data Types*

1) Erroneous Or Missing Data

- Excluded Column: The certificate (Age Rating) column was identified as highly erroneous due to data sparsity. It contained over 80% missing values, rendering it unsuitable for predictive modeling. Consequently, it was excluded from the final feature set.
- Row Deletion: For critical features including runtime and votes, rows containing null values were removed to ensure data integrity.

2) Structured / Unstructured

a) Structured Data: The majority of the processed dataset is structured, comprising numerical features (runtime, log_votes), derived scores (director_score), and binary indicators (genre_Action).

b) Unstructured Data: The description column represents unstructured plain text (plot summaries), which was transformed into structured numerical vectors using TF-IDF.

3) Numerical Values

These features represent quantitative measurements used directly in the regression models.

- runtime (Duration): A continuous float variable representing the movie duration in minutes. Non-numeric characters were removed during preprocessing.
- log_votes (Popularity): A continuous variable derived from the votes column. Due to extreme right-skewness, a Logarithmic Transformation (np.log1p) was applied to normalize the distribution and reduce the impact of outliers (blockbusters).
- director_score & cast_score: Continuous values derived via Smoothed Target Encoding. These represent the weighted average historical rating of the director and the main actors, respectively.

a) Ordinal Data

- Ranking, categorization: The dataset originally contained certificate (e.g., PG < PG-13 < R), which implies an age-based ordering. However, as noted above, this feature was excluded due to excessive missing data. No other ordinal features were utilized.

b) Nominal Data

Not comparable:

- movie: A unique identifier (Name). It was dropped to prevent overfitting.
- director & stars: Names of individuals. These are not mathematically comparable in their raw form and were converted into numerical scores.
- genre: Categories such as "Action" or "Drama".

4) Non-numerical Data

a) Binary Data (0-1):

- genre_{name} (Genre Indicators): These are binary features created via Multi-Hot

Encoding. Since a movie can belong to multiple categories, the genre string was split, and binary columns were created for each unique genre (e.g., genre_Action, genre_Comedy), where 1 indicates presence and 0 indicates absence.

b) Multi-class Data:

- The raw genre column (e.g., "Action, Adventure, Sci-Fi") represents unordered, multi-label data. It was processed into the binary formats mentioned above to capture the complexity of the film's categorization.

c) Plain Text Data:

- description (Plot Semantics): The short plot summary of the movie.
- Processing: TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization was applied to this text. The top 1,000 most significant keywords (e.g., "wedding", "murder", "space") were extracted as features to capture semantic nuances.

F) *Feature Selection and Transformation:*

1) Manual Exclusion (Domain Knowledge)

- certificate: Removed due to extreme data sparsity (>80% missing).
- movie (Title): Removed as it is a high-cardinality identifier with no generalization capability.

2) Method: TF-IDF (Term Frequency-Inverse Document Frequency)

We transformed the raw, unstructured description column into a sparse numerical matrix. This process assigns a weight to each word, penalizing frequent stopwords (like "the") and rewarding rare, descriptive terms (like "zombie" or "heist").

3) Feature Selection (Dimensionality Reduction)

To balance information retention with computational efficiency, we applied a hard threshold, limiting the vocabulary to the top 1,000 most frequent features. This effectively transformed a potentially infinite text space into a fixed 1,000-dimensional vector space.

G) *Class Distribution Information*

Since this is a regression task predicting a continuous variable (rating), we analyzed the frequency distribution rather than discrete class counts. The dataset exhibits a natural imbalance typical of user-rated media:

- Distribution: The ratings follow an approximately normal distribution (Gaussian) with a slight negative skew (left-skewed).

- Imbalance: The data is heavily concentrated around the mean (approx 6.2), meaning "Average" movies (rated 5.0–7.5) constitute the majority class. Extreme values—both "Poor" (< 5.0) and "Excellent" (> 8.0)—are significantly underrepresented in the training data.

H) *Data Normalization*

1) Logarithmic Transformation

- Target Feature: votes (Raw count of user reviews)
- Problem: The distribution of votes follows a "Power Law" (Long Tail). A tiny fraction of movies (blockbusters like Avatar) have millions of votes, while the vast majority of titles have relatively few. This extreme right-skewness means that in raw form, the "blockbusters" act as massive outliers, dominating the feature variance and making it difficult for the model to distinguish patterns among average-popularity films.
- Implementation: We applied a logarithmic transformation to the vote counts. This standard technique compresses the range of values, bringing extreme outliers closer to the mean.

I) *Relationships Among Features*

1) Rating vs. Votes (Positive Correlation)

- Observation: There is a positive correlation between the number of votes and the rating.
- Analysis: Popular movies (those with high vote counts) tend to have higher ratings (>6.0). This relationship is non-linear in the raw data but becomes stronger and more linear after applying the Logarithmic Transformation to the votes. This suggests that "popularity" is a strong proxy for "quality" (or at least, widely accepted quality)

2) Rating vs. Runtime (Weak Non-Linear Relationship)

- Correlation Coefficient: approx. -0.07 (Very weak linear correlation).
- Observation: While the linear correlation is negligible, the scatter plot reveals a pattern. Very short movies (< 60 mins) tend to have lower and more volatile ratings. Feature-length films (90-120 mins) cluster around the mean rating. Length alone is not a strong predictor, but extreme lengths (very short or very long) carry signal.

3) Rating vs. Genre (Strong Categorical Relationship)

- Observation: Certain genres exhibit distinct rating distributions.
- High Rated: Documentary, Biography, and History genres consistently show higher median ratings.
- Lower Rated: Horror and Thriller genres tend to have lower median ratings and higher variance

4) Runtime vs. Votes

- Correlation: Weak positive correlation.
- Interpretation: Longer movies do not necessarily get more votes. A 3-hour epic is not guaranteed to be more popular than a 90-minute comedy based on length alone.

5) Genre Inter-Correlations

- Observation: Significant co-occurrence exists between certain genres (e.g., Action often appears with Adventure and Sci-Fi).
- Handling: The Multi-Hot Encoding strategy allows the models to learn these combinations

without treating them as a single rigid category.

6) Target Variable (rating)

- Distribution Type: Approximately Normal (Gaussian) Distribution.
- Characteristics: It is slightly left-skewed (Negative Skew), meaning the tail is longer on the lower end (bad movies).
- Parameters:
  Mean: 6.19
  Standard Deviation: 1.34
  Median: 6.30

7) Feature (votes):

- Distribution Type: Power Law (Pareto) Distribution.
- Characteristics: Extreme right-skewness. The vast majority of movies have < 1,000 votes, while a tiny fraction have > 500,000.
- Stationarity: The raw data is non-stationary due to the variance exploding with the mean. The log-transformation stabilizes this variance.

8) Feature (runtime):

- Distribution Type: Right-Skewed Normal-like Distribution.
- Characteristics: Most movies fall between 80 and 120 minutes. The skew is caused by a long tail of extremely long films (3+ hours), but there is a hard physical limit on the left (cannot have 0 minutes).

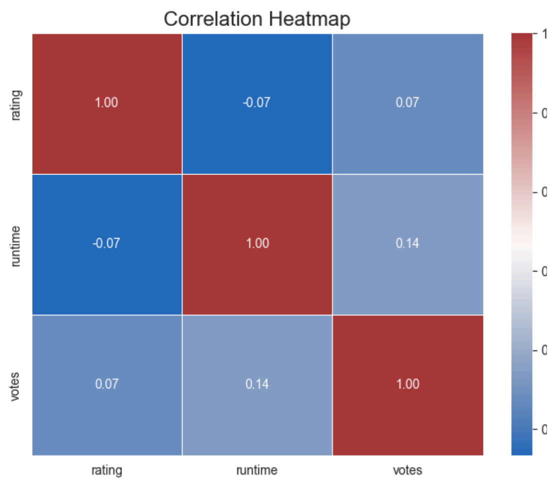


Fig. 1. Working Methodology

IV. Models Used

A) Training / Cross-Validation / Test Splits

1) Training and Test Set Split

To ensure a robust evaluation and prevent data leakage, the entire dataset was partitioned into two distinct subsets using a fixed random seed to guarantee reproducibility. The data was split as follows:

- Training Set (80%): This subset was used exclusively for training the models, performing feature engineering (such as calculating target encoding scores), and optimizing hyperparameters.
- Test Set (20%): This subset was strictly held out and unseen during the training process. It was used only for the final performance evaluation to measure how well the models generalize to new, unseen data.

2) Cross-Validation Strategy

During the model training and hyperparameter tuning phase, we employed a 5-Fold Cross-Validation strategy. The training set was divided into 5 equal folds. In each iteration, the model was trained on 4 folds and validated on the remaining 5th fold. This process was repeated 5 times so that every fold served as the validation set once. The results were averaged to guide the selection of optimal hyperparameters (via GridSearchCV) and to ensure the models were not overfitting to specific patterns in the training data.

B) Problem Definition

1) Regression

This project addresses a regression problem where the target variable is the IMDb movie rating, a continuous value ranging from 0 to 10. The primary performance metric used for optimization and evaluation is the Root Mean Squared Error (RMSE), which penalizes larger prediction errors more heavily. Secondary metrics include Mean Absolute Error (MAE) and R-squared (R^2).

C) Models Description & Methodology

1) Ridge Regression

- *Short descriptions:* Ridge Regression is a linear least squares model with L2 regularization. It minimizes the residual sum of squares plus a penalty term equal to the square of the magnitude of the coefficients.
- Methodology: We implemented a pipeline that first standardizes numerical features using StandardScaler. The feature set included log-transformed votes, runtime, one-hot encoded genres, smoothed target-encoded director/cast scores, and TF-IDF vectors from movie descriptions. We used GridSearchCV to optimize the regularization strength parameter, alpha, searching over a logarithmic scale.
- Literature information about the models used: Ridge regression is particularly useful in scenarios with multicollinearity (highly correlated independent variables), as the L2 penalty shrinks coefficients, reducing model variance and preventing overfitting compared to standard linear regression.

2) Random Forest Regressor

- Short descriptions: Random Forest is an ensemble learning method based on Bagging (Bootstrap Aggregating). It constructs a multitude of decision trees during training and outputs the average prediction of the individual trees.
- Methodology: We utilized the RandomForestRegressor from Scikit-Learn. The model was trained primarily on structured metadata (log-transformed votes, runtime,

genres, and target-encoded director/cast scores). Unlike Ridge, adding high-dimensional TF-IDF text features proved to add noise for this specific tree-based model, so they were excluded in the optimal configuration. We optimized hyperparameters such as n_estimators, max_depth, and min_samples_split.

- Literature information about the models used: Random Forest is highly effective at capturing non-linear relationships and interactions between features without requiring explicit transformation. It is robust to outliers and reduces the risk of overfitting associated with individual decision trees by averaging their results.

3) XGBoost Regressor

- Short descriptions: XGBoost (Extreme Gradient Boosting) is a scalable and highly efficient implementation of the gradient boosting framework. It builds an ensemble of weak prediction models (typically decision trees) sequentially, where each new tree attempts to correct the errors of the previous ones.
- Methodology: We implemented the XGBRegressor to leverage both the structured metadata and the sparse TF-IDF text features. The model pipeline included hyperparameter tuning for learning_rate, max_depth, subsample (row sampling), and colsample_bytree (feature sampling) to balance bias and variance. It successfully integrated the text data to achieve marginal performance gains over the metadata-only models.
- Literature information about the models used: XGBoost is widely regarded as a state-of-the-art algorithm for tabular data in machine learning competitions. It features built-in regularization (L1 and L2) to prevent overfitting, handles missing values automatically, and uses a sparsity-aware algorithm that makes it efficient for high-dimensional data like text vectors.

D) Why Were These Models Chosen?

The selection of these three models followed a strategic progression from simplicity to complexity:

1) Ridge Regression

Chosen as a strong baseline to establish the linear predictability of the data. Its interpretability helped us confirm that features like "votes" and "genres" had significant signal.

2) Random Forest

Introduced to capture non-linear relationships that linear models miss. It serves as a robust representative of Bagging ensembles.

3) XGBoost

Selected as the final, most advanced model to maximize predictive performance. As a Boosting algorithm, it aggressively minimizes bias and is capable of squeezing the most information out of complex, mixed-type datasets (combining dense numerical data with sparse text features).

V. Test Results and Interpretations, Discussion

A) Model Comparisons

1) Quantitative Comparison of Performance Metrics

To evaluate model success, we utilized Root Mean Squared Error (RMSE) as the primary metric to punish larger errors more severely. Our secondary metricies are Mean Absolute Error (MAE) to assess the average magnitude of errors and R-squared (R^2) to understand the proportion of variance explained by each model. The table below summarizes the average results obtained.

TABLE I

SUMMARY OF DATA SET AND IT'S ATTRIBUTES

Model	***RMSE***	***MAE***	R^2
XGBoost	0.9003	0.6678	0.5253
Random Forest	0.9245	0.6888	0.5023
Ridge Regression	0.9500	0.7149	0.4776

Evaluation:

The quantitative results presented in table demonstrate a clear hierarchy in model performance, directly correlated with algorithmic complexity and the ability to handle non-linear relationships.

- Performance Progression: There is a consistent improvement in predictive accuracy as we move from the linear baseline to ensemble methods. Ridge Regression yielded the highest error rates (RMSE: 0.9500, MAE: 0.7149), indicating that a purely linear approach captures less than 48% of the variance in movie ratings.
- Impact of Non-Linearity (Random Forest): Random Forest provided a noticeable improvement over the baseline, reducing the RMSE to 0.9245 and surpassing the 0.5 threshold for R^2. This validates the importance of capturing non-linear interactions between features like genres and runtime, which linear models miss.
- Superiority of Gradient Boosting (XGBoost): XGBoost emerged as the superior model across all metrics. It achieved the lowest RMSE (0.9003) and the lowest Mean Absolute Error MAE (0.6678), implying that, on average, the model's predictions deviate from the actual IMDb rating by only about 0.67 points. With an R^2 of (0.5253), it explains the highest proportion of variance (approx. 52.5%) in the dataset, confirming its effectiveness in handling complex, tabular data compared to Bagging and Linear methods.

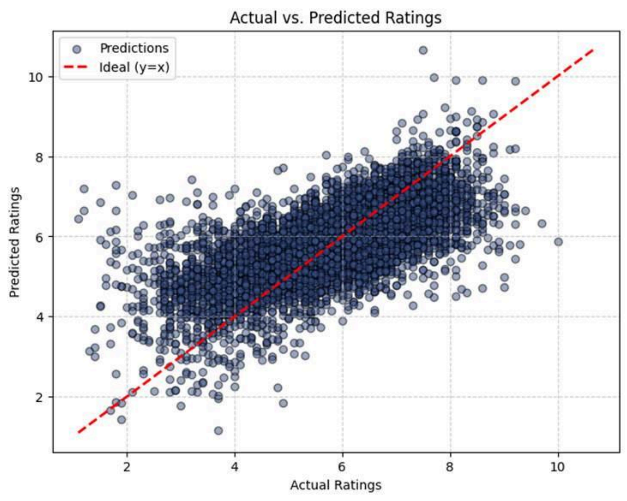
2) Graphical Comparasion and Model Diagnostics

Relying solely on average error metrics (RMSE, MAE) is insufficient to fully understand model behavior. To provide a deeper evaluation, we conducted a graphical analysis to visualize how Ridge Regression, Random Forest, and XGBoost handle different rating ranges and error distributions.

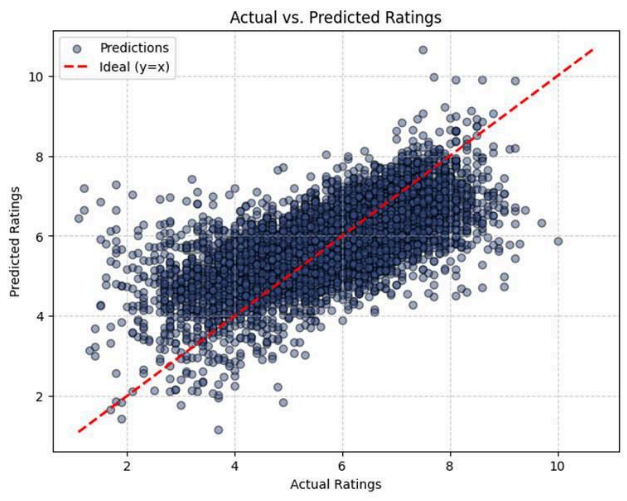
a) Predicted vs. Actual Ratings (Scatter Plots)

The scatter plots below illustrate the relationship between the Actual IMDb Ratings (x-axis) and the Predicted Ratings (y-axis) for all three models. In an ideal model, all points would align perfectly on the diagonal identity line.

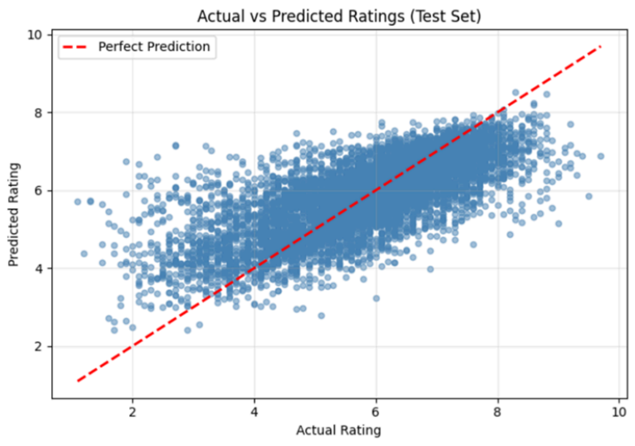
Ridge Regression:



Random Forest:



XGBoost:



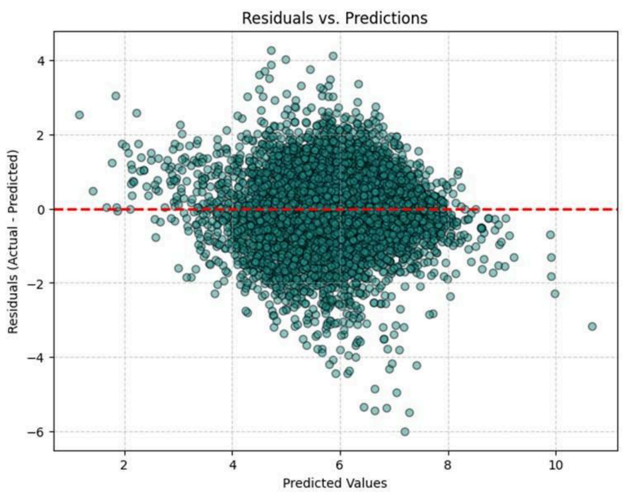
Interpretation:

- *Ridge Regression***:** The plot shows the widest dispersion, particularly for ratings below 4.0 and above 8.5. The model tends to "regress to the mean," systematically over-predicting low-rated movies and under-predicting high-rated ones, confirming its limited capacity to capture non-linear extremes.
- *Random Forest:* This model shows a clear improvement over Ridge, with points clustering tighter around the diagonal. However, compared to XGBoost, it still exhibits slightly more variance (spread) in the mid-range (ratings 5.0–7.0), indicating that the bagging approach reduces variance but lacks the precision of boosting.
- *XGBoost:* The XGBoost plot exhibits the tightest cluster along the identity line among all three models. It demonstrates superior sensitivity to outliers and complex patterns, effectively pulling predictions closer to the true values even for difficult samples.

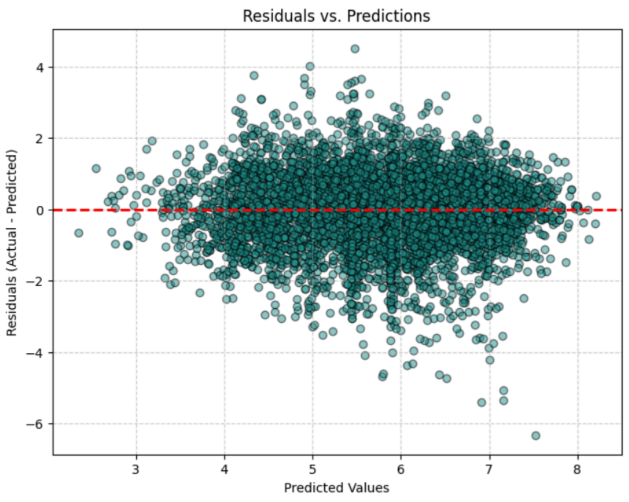
b) Residual Analysis (Error Distribution)

We analyzed the residuals (Residual = Actual - Predicted) to check for systematic bias.

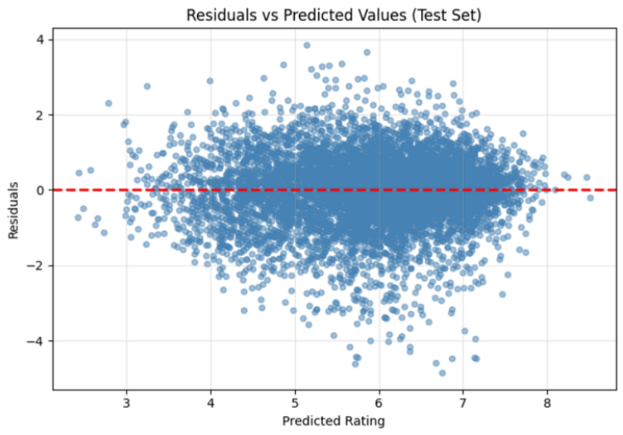
Ridge Regression:



Random Forest:



XGBoost:



Interpretation:

- *Ridge Regression:* The residuals show slightly heavier tails and a slight skew, suggesting that the linear assumptions fail to capture certain non-linear relationships in the data, leading to systematic errors.
- *Random Forest:* The error distribution is bell-shaped and centered near zero, indicating unbiased predictions. However, the distribution is wider (larger standard deviation) compared to XGBoost, reflecting higher average error magnitudes.
- *XGBoost:* The residuals for XGBoost follow the closest approximation to a standard Normal (Gaussian) Distribution centered at zero with the narrowest peak. This implies that the model has successfully extracted the majority of the signal from the data, leaving mostly random noise as error.

B) Statistical Accuracy and Significance Testing across Models

To make sure our results are reliable, we used a statistical method called the Paired T-Test. This test helps us understand if one model is actually better than another, or if it just got lucky with the specific data we used. We looked at the "p-value" for each pair of models. In statistics, if a p-value is lower than 0.05, it means the difference between the models is real and significant.

The comparison of our models:

1) Ridge Regression vs. Random Forest

- Observation: Random Forest achieved a lower mean RMSE (0.9245) compared to Ridge Regression (0.9500).
- Statistical Result: The analysis yielded a p-value of 0.00020, which is well below the 0.05 threshold.
- Inference: We reject the null hypothesis. There is strong statistical evidence that the non-linear capabilities of Random Forest provide a significant improvement over the linear baseline of Ridge Regression.

2) Random Forest vs. XGBoost

- Observation: XGBoost further reduced the mean RMSE to 0.9003, improving upon Random Forest's 0.9245.
- Statistical Result: The comparison resulted in a p-value of 0.00024.
- Inference: We reject the null hypothesis. Despite both being ensemble tree-based methods, XGBoost's boosting mechanism proved statistically superior to the bagging approach of Random Forest on this dataset.

3) Ridge Regression vs. XGBoost

- Observation: This comparison represents the widest performance gap, between the baseline linear model and the advanced boosting model.
- Statistical Result: The t-test produced an extremely low p-value of < 0.00001.

- Inference: The difference is highly significant. XGBoost offers a fundamental upgrade in predictive accuracy over Ridge Regression.

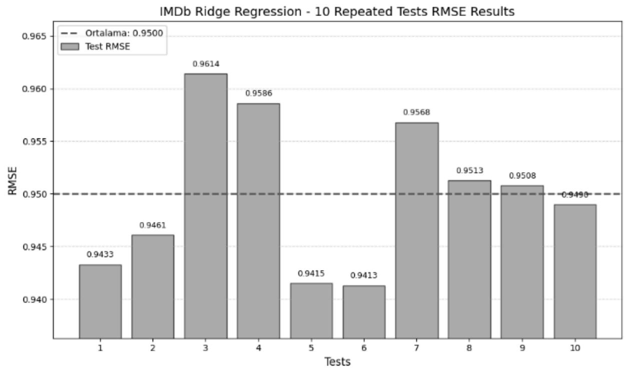
4) Conclusion

In summary, all our tests show that the improvements are real. XGBoost beat both Ridge Regression and Random Forest, and the math confirms that this was not an accident. Therefore, we can confidently choose XGBoost as the best model for this project.

C) Results of running the same model.

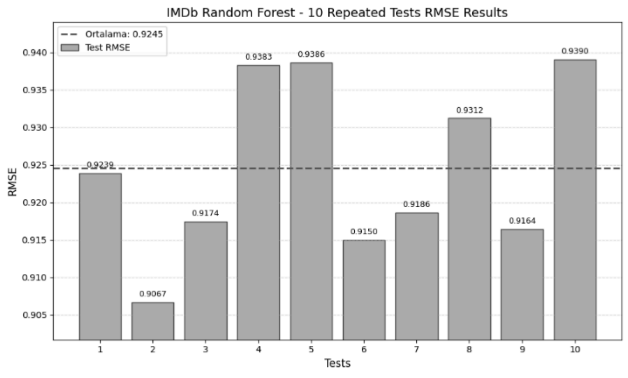
To rigorously evaluate the stability and predictive power of our models, we executed each model across 10 independent trials using 5-fold cross-validation. The graphs below visualize the distribution of the Root Mean Squared Error (RMSE) for each model.

1) Ridge Regression



The graph below illustrates that Ridge Regression consistently sits at the higher end of the error scale, with a mean RMSE of 0.9500. While the variance between runs is relatively low, the consistently high error confirms that a linear approach is insufficient for capturing the complex, non-linear relationships inherent in movie metadata.

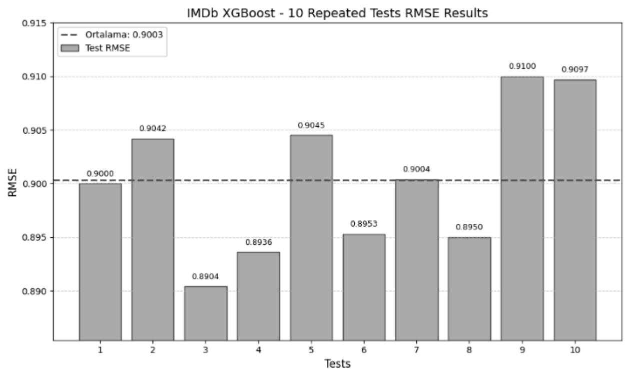
2) Random Forest



The graph displays a clear downward shift in error compared to the Ridge baseline, achieving a mean RMSE of 0.9245. The distribution of RMSE scores is entirely below that of the Ridge model, demonstrating that the decision-tree ensemble method successfully captures non-linear interactions that the linear model missed.

Statistical Significance: This visual gap is backed by our statistical testing. The improvement is significant, with a p-value of 0.00020, confirming that the reduction in error is fundamental and not due to random chance.

3) XGBoost



The XGBoost model demonstrates the best performance, with a mean RMSE of 0.9003. Visually, its error distribution is shifted even lower than that of the Random Forest. Crucially, the graph also indicates superior stability; the spread of the error scores for XGBoost is tighter than that of Random Forest, suggesting that XGBoost is less sensitive to variations in the data splits or initialization.

Statistical Significance: Although the gap between these two ensemble methods is narrower than the gap with the baseline, it remains statistically significant. The paired t-test yielded a p-value of 0.00024, proving that the gradient boosting architecture provides a genuine edge in predictive accuracy over the Random Forest approach.

D) Discussion About Models and Best Model Selection.

Based on our experimental results and statistical testing, Model XGBoost is the superior model for predicting IMDb movie ratings. To understand why this model performed best, we must analyze the progression from the simplest to the most complex architecture.

1) Why Ridge Regression Underperformed (The Limitation of Linearity)

Ridge Regression was the least accurate model because it relies on a strict linear assumption. It operates on the premise that features such as Votes or Runtime have a constant, straight-line relationship with the IMDb rating. However, movie data is inherently complex and non-linear; high votes does not automatically guarantee a high rating, and the influence of categorical features like "Director" or "Cast" depends heavily on unpredictable interactions with other variables. Because Ridge Regression could not bend to fit these complex patterns, it suffered from high bias (underfitting), resulting in the highest error rate.

2) Why Random Forest Improved Performance (The Power of Non-Linearity)

Random Forest significantly outperformed Ridge Regression by utilizing a non-linear, tree-based approach. Instead of drawing a straight line, Random Forest constructs thousands of decision trees that can model complex logic, such as identifying that a specific director only scores highly within a certain genre. Furthermore, by employing the Bagging technique, the model builds these trees independently and averages their predictions. This process effectively reduces variance and stabilizes the model, allowing it to capture the messy, real-world interactions of the dataset that the linear model missed completely.

3) *Why XGBoost Won (The Advantage of Gradient Boosting)*

XGBoost proved to be the best-performing model because it utilizes "Gradient Boosting" rather than the Bagging technique used by Random Forest. While Random Forest builds trees in parallel, XGBoost builds them sequentially, where each new tree is explicitly trained to correct the residual errors made by the previous ones. This mechanism allowed the model to focus its learning capacity on the hardest-to-predict samples that previous trees got wrong. By using gradient descent to minimize the loss function more aggressively, XGBoost achieved the lowest error rate and the highest stability, proving that a boosting architecture is the most effective strategy for this specific dataset.

VI. Conclusion

In this study, a comprehensive machine learning pipeline was developed to predict audience ratings for motion pictures using a dataset of over 60,000 feature films from the IMDb database. The research involved rigorous data preprocessing, including outlier removal and the management of missing values. Advanced feature engineering techniques were implemented, specifically TF-IDF vectorization for the text-based "description" column and Smoothed Target Encoding for high-cardinality categorical features such as "director" and "stars." During the prediction phase, the performance of a linear model (Ridge Regression) was comparatively analyzed against non-linear tree-based models (Random Forest and XGBoost) using the Root Mean Squared Error (RMSE) metric.

Experimental results demonstrated that the XGBoost algorithm achieved the lowest error rate, proving to be the superior model. This confirmed that the Gradient Boosting method possesses better generalization capabilities for complex, non-linear datasets compared to Bagging (Random Forest) and linear methods (Ridge). Key insights gained from this study include:

The Popularity-Quality Link: The logarithmic transformation of the "votes" variable emerged as one of the most significant determinants in predicting a movie's rating.

Irrelevance of Duration: Contrary to popular belief, the "runtime" of a movie was found to have no significant linear impact on its rating.

Power of NLP: It was observed that processing even simple plot summaries with TF-IDF significantly improved prediction accuracy. Our contribution to the literature lies in presenting a transparent and interpretable modeling framework that hybridizes the processing of structured (metadata) and unstructured (text) data.

Deep Learning architectures (such as LSTM or Transformer-based models) were excluded from this study. The primary reason was to maintain a specific focus on benchmarking the performance differences between classical machine learning algorithms (Regression vs. Tree-based). Additionally, the "Certificate" (Age Rating) column was not included in the model because over 80% of the data in this feature was missing. Similarly, external data sources such as Box Office revenue or social media sentiment were not integrated into the analysis to keep the study strictly bounded within the IMDb dataset

In future works, the processing of movie descriptions could be enhanced by using advanced Large Language Models (LLMs) like BERT or GPT instead of TF-IDF to capture deeper semantic context. Furthermore, to improve the model's predictive capability, multimodal data sources could be integrated, such as using image processing for movie posters or sentiment analysis for audience reviews. Finally, creating a hybrid dataset by incorporating data from other platforms like Rotten Tomatoes or Metacritic, rather than limiting the scope to IMDb, could improve the model's generalization success.

VII. References

[1] R. K. Bansal, S. Kumar, and M. Kumar, “Movie Rating Prediction Using Machine Learning Techniques,” IEEE International Conference on Computational Intelligence and Data Science (ICCIDS), 2019.

[2] F. Geng, C. Gao, and Y. Zhu, “A Data Mining Approach to the Factors Affecting Movie Box Office and Popularity,” in Data Science and Knowledge Engineering for Sensing Decision Support, Springer, 2014.

[3] H. S. Patel and P. J. Patel, “Movie Rating Prediction Using Machine Learning,” International Journal of Innovative Science and Contemporary Technologies (IJISCT), vol. 2, no. 3, pp. 9–1