# IDENTIFYING POTENTIAL FOR HIGH IMDB RATINGS IN NETFLIX

**Course No. :** DATA230
**Course Name**: Data Visualization

## PROJECT REPORT

**Student Name & ID (GROUP-4):**
**Arya Mehta (018292885)**
**Jane Heng (018321914)**
**Prajwal Dambalkar (018318196)**
**Vedika Sumbli (018305937)**

**Project Area: MACHINE LEARNING & DATA VISUALIZATION**

**Project Work carried out at:**

**SAN JOSE STATE UNIVERSITY**
**SAN JOSE, CA**

**May 2025**

# ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to everyone who supported us throughout the course of this project.

First, we extend our thanks to Prof. Guannan Liu for her invaluable guidance, encouragement, and constructive feedback, which greatly contributed to the successful completion of this work.

We also appreciate the resources and support provided by the Department of Applied Data Science which enabled us to conduct our research and complete this technical report effectively.

Lastly, we would like to thank our families and friends for their patience and encouragement throughout this journey.

This project has been a valuable learning experience for all four of us, and we are grateful for the opportunity to collaborate and grow as a team.

Signed,
Arya Mehta
Jane Heng
Prajwal Dambalkar
Vedika Sumbli

# ABSTRACT

This report presents *Netflix Show Popularity Prediction*, a machine learning (ML) initiative designed to address the challenge of forecasting the success of streaming content prior to its release. Unlike traditional approaches that depend on post-launch data such as IMDb ratings or audience feedback, this project focuses exclusively on leveraging metadata available during the pre-production and planning phases.

The primary objective is to develop a predictive model capable of identifying whether a Netflix show will achieve popularity, using inputs such as production budget, director's historical success, cast reputation, production company performance, prevailing genre trends, and planned release season. To simulate realistic production conditions, the dataset was enriched with these features, ensuring that only information available before launch was incorporated.

Data preprocessing involved systematic handling of missing values, categorical encoding and normalization of numerical variables. A custom *popularity score* was devised, using a weighted combination of key features, to label the top 40% of shows as "popular" and the remainder as "not popular." Notably, no IMDb or post-release metrics were included in target creation, effectively avoiding data leakage and ensuring the model's applicability to real-world forecasting scenarios.

For model selection, a Random Forest Classifier was employed due to its robustness in handling mixed data types, resilience against overfitting, and its capacity to provide interpretable feature importance metrics. Model evaluation was conducted using a combination of train-test splits and five-fold cross-validation, with performance assessed through accuracy, precision, recall, confusion matrices, and ROC-AUC scores.

The distinguishing feature of this project lies in its pre-release prediction framework. While conventional models rely on audience-derived metrics and therefore require post-launch data, the approach developed here operates solely on pre-release metadata, aligning closely with the decision-making processes employed by content producers and strategists. Moreover, while large language models (LLMs) offer impressive text analysis capabilities, they are inherently less interpretable, more resource-intensive, and less structured for numeric prediction tasks of this nature.

In conclusion, it demonstrates the potential of supervised machine learning to enhance content strategy in the entertainment industry. By focusing on structured, pre-release data and delivering a scalable, explainable solution, this project offers a valuable tool for production teams seeking to make data-informed decisions about which shows to greenlight, invest in, or prioritize for release.

## TABLE OF CONTENTS

| Section No. | Section Name | Page No. |
|---|---|---|
| 1 | Requirement Statement / Problem Statement | 5 |
| 2 | Project Scope | 5 |
| 3 | Methodology | 6 |
| 4 | Detailed Description of the project | 6 |
| 5 | Risks and Mitigations | 18 |
| 6 | Conclusions and Recommendations | 18 |
|  | References | 21 |
|  | Summary of how the feedback for Project Outline have been addressed | 21 |

## 1. Requirement Statement / Problem Statement

The project aims to develop a popularity prediction model for Netflix Movies and TV Shows using natural language processing (NLP) embeddings and machine learning. The system will classify movies into categories based on their popularity prediction score.
In the era of competition among various shows/movies over OTT platforms, it can be vital for a production company to understand the criteria of investment into a show. Conversely, it can aid the organization (Netflix) to assess various variables before releasing the show on their platform.

## 2. Project Scope

The objective of this project is to develop and implement a popularity prediction model that accurately detects and classifies movies into popular or unpopular through various features such as Genre, Cast, Director, Description etc. The project aims to leverage natural language processing (NLP) techniques and machine learning algorithms to automate the process of classification even before its release on the platform. The primary goal is to extract insights so that a promising assessment can be made. The project will focus on:

1. **Data Collection and Preprocessing**: Extracting relevant textual data from diverse sources, followed by preprocessing steps such as tokenization and text normalization to prepare the data for analysis.

2. **Classification**: Calculation of various numerical parameters that can contribute in making the prediction and implementing machine learning models to make the classification.

3. **Model Evaluation and Optimization:** Evaluating model performance using standard metrics (precision, recall, F1-score) and optimizing for accuracy, especially in handling ambiguities such as sarcasm or mixed sentiments.
4. **Visualization and Reporting**: Developing visualization to present popularity trends and distribution and key insights in a comprehensible format.

The outcome of this project will provide an efficient and scalable solution for real-time popularity prediction and analysis, enhancing organizational responsiveness to various features/variables.

## 3. Methodology

1. Enriched the dataset with pre-release metadata, including budget, director success, cast popularity, production company record, genre trends, release season, IMDB Scores etc.
2. Performed data preprocessing, i.e., filled missing values, applied OneHotEncoding to categorical variables and normalized numerical features.
3. Defined the target variable: calculated popularity score and labeled top 40% of shows as “popular”.
4. Trained a Random Forest Classifier and Logistic regression model to integrate preprocessing and modeling using scikit-learn’s Pipeline and ColumnTransformer
5. Evaluated model performance using train-test splits and five-fold cross-validation. Also, calculation of metrics including accuracy, precision, recall, and ROC-AUC was performed.

## 4. Detailed description of project

### 1. Literature review

The rapid expansion of online streaming platforms like Netflix, Amazon Prime Video, and Disney+ has sparked significant academic and industry interest in predicting audience preferences and content success. Prior research on movie and show popularity prediction has traditionally relied on post-release data, such as IMDb ratings, box office revenue, audience reviews, and social media sentiment. For instance, Asur and Huberman (2010) demonstrated the effectiveness of using social media chatter, particularly Twitter activity, to forecast box office outcomes. Similarly, research by Liu et al. (2016) focused on sentiment analysis of online reviews to predict a movie’s commercial success. While these models show high predictive accuracy, they inherently depend on data that only becomes available after a show’s release, making them unsuitable for pre-launch decision-making by studios and content strategists.

To address the limitations of post-release models, several researchers have explored the use of metadata and intrinsic features. Sharda and Delen (2006) introduced predictive models based solely on pre-release attributes like budget, cast, director, genre, and MPAA ratings to estimate box office performance. Their findings highlighted the potential of using structured metadata without relying on external audience feedback. Similarly, Lash and Zhao (2016) examined the role of production and distribution characteristics, noting that factors such as a director’s past success, genre popularity trends, and release timing play significant roles in a film’s commercial outcome. These studies emphasize the value of internal studio data in driving investment and release decisions, particularly when external validation data are unavailable or premature.

In the machine learning domain, the application of algorithms like Random Forest, Support Vector Machines, and neural networks for entertainment industry predictions has gained popularity. Breiman’s (2001) introduction of the Random Forest algorithm demonstrated its robustness, especially when handling complex, mixed-type datasets and preventing overfitting. Recent research, such as by Tsang and colleagues (2021), has applied ensemble models to predict Netflix viewing patterns using a combination of metadata and user interaction data, showcasing how ensemble approaches can outperform single classifiers. However, most of these advanced models either incorporate user data — which is unavailable pre-release — or focus on interpretability trade-offs, raising challenges when applied to production-phase predictions where explainability is crucial for decision-makers.

Finally, recent developments in artificial intelligence, particularly with large language models (LLMs) like GPT-3 and GPT-4, have opened new possibilities for text-based content prediction, script analysis, and audience modeling. While promising, these models come with interpretability challenges and higher computational demands, making them less practical for structured numeric tasks such as predicting show popularity based on pre-release attributes. This project, therefore, aligns with the body of literature advocating for interpretable, structured machine learning solutions that leverage only the metadata available to production teams prior to a show's release. By focusing on explainable models like Random Forest and carefully engineered features derived from historical success patterns, this work aims to provide a practical, scalable solution that complements existing research while addressing a critical gap: the ability to forecast success before audience engagement ever occurs.

## 2. Sources of primary and secondary data

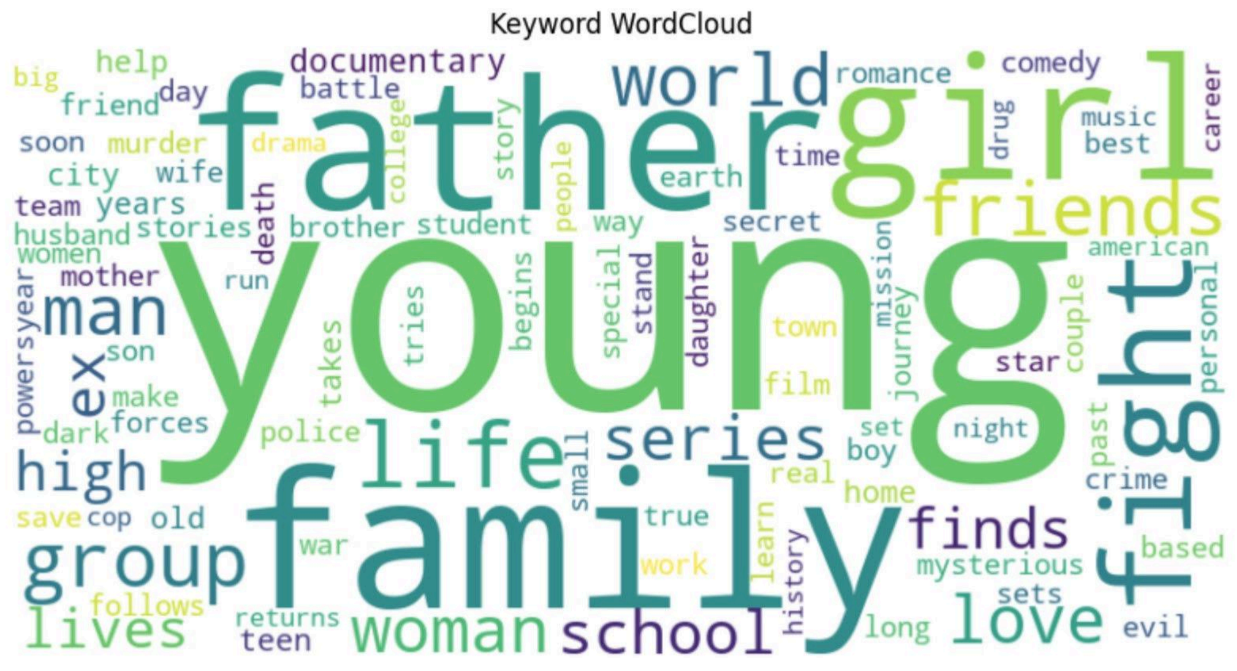
This study uses a dataset for the performance prediction which is available publicly on *https://www.kaggle.com/datasets/shivamb/netflix-shows.* It mentions the name of the show/movie along with its caste, year of release, genre, description, country of origin, production cost etc. Some new columns such as director popularity score, cast popularity score, genre trend, marketing cost etc are created for better evaluation of the performance score. IMDB scores along with votes as a dataset are used to correspond to the respective movies and shows which then aids in model assessment.

## 3. Data preprocessing before Popularity prediction

The dataset used for this study is purely qualitative. Hence, preprocessing the data before analyzing the scores is pivotal. The initial dataset consists of the name of the movie/shows and its full description along with its year of release, genre, type, cast name, director's name, production house etc. is given. All the rows with null values have been eliminated. Since the data is from various countries along with year range being from 1970-2020s, the dataset becomes highly skewed for analysis. The skewness is reduced by eliminating movies before the 2000s and the region of concentration is kept to be the United States. This has reduced the dataset to 50% of its original with around 4000 rows. New features such as cast popularity score, director popularity score, the budget of the movie along with the marketing budget are included for a better popularity score analysis.

## 4. Identification of keywords from Description using PorterStemmer

Stemming is the process of reducing a word to its base or root form, such that variants of the word are treated as the same word. PorterStemmer is an effective library to convert the words to their root form for analysis. It is available through the Natural Language Toolkit and is used with Stop words to avoid skewing the analysis with redundant information. Since it's fast and does not require training data, the Porter Stemmer is useful in applications where speed is important, such as real-time analysis. The stop words are aggregated in a  standard file in python but certain custom stop words are also added to the program to avoid the data skew. A word cloud is generated using the most commonly found words in the description..



5. **Identifying keywords signifying description**

The TF-IDF (Term Frequency - Inverse Document Frequency) vectorizer is used to convert the text data into a numerical feature to identify the frequency of specific words in the text. In performance prediction analysis tasks, TF-IDF can transform text reviews into a vector of weighted features, which can then be input into a classifier.

6. **Addition of new features**

The dataset is processed with various calculated features related to budgets, production, cast, and trends. Pandas and NumPy are used for data handling, requests and BeautifulSoup for web scraping (mainly to pull budget estimates from Wikipedia), and include fallback estimates based on industry standards if no online data is found.

Key functions include *get_show_budget()* to determine the estimated production budget, and *calculate_features()* which adds several derived columns: total budget (including per-episode scaling for TV shows), production company track record, director success, cast popularity (based on a detailed scoring system considering experience, genre diversity, ratings, and trend alignment), genre trend scores, release season (spring, summer, fall, winter), and marketing budget (typically 20–25% of production cost).

Numeric features are normalized using MinMaxScaler to bring them to a 0–1 scale, which is useful for machine learning or comparison tasks. The *process_netflix_data()* function applies the cast popularity calculations and adds extra metrics like actor experience, diversity, and versatility. An updated CSV file with all the new features is further used in the ML model.

## 7. ML Model Implementation

### A. Data Processing and Feature Engineering

The implementation begins with preparing the dataset from the Netflix IMDB clean CSV file. A critical preprocessing step involves transforming textual keyword data into a format suitable for machine learning algorithms. We developed a specialized function to convert string representations of keyword lists into space-separated strings, facilitating subsequent text feature extraction. This approach preserves thematic information while enabling numerical processing.

The resulting dataset comprised 1,982 content items split into training (n=1,585) and testing (n=397) sets, maintaining the original class distribution (approximately 80% non-popular and 20% popular content). This stratified splitting approach ensures that both training and testing datasets reflect the underlying class imbalance present in the complete dataset.

### B. Feature Selection and Representation

We incorporated two distinct categories of features:

1. **Numerical Features**: These include production metadata (budget, production company track record, director success rating, cast popularity, marketing budget), content characteristics (release year, duration in minutes/seasons), and categorical variables encoded as numerical values (content type, rating, release season).
2. **Text Features**: Keywords associated with content were processed using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization, with a maximum of 200 features extracted. This technique transforms the textual keyword data into a numerical representation that captures the relative importance of terms while controlling for their frequency across the dataset.

The feature engineering pipeline was implemented using scikit-learn's `ColumnTransformer`, enabling simultaneous application of different transformations to different feature types. This approach ensures consistent preprocessing across training and evaluation phases.

### C. Addressing Class Imbalance

With approximately 80% of instances belonging to the non-popular class, the dataset exhibited significant class imbalance. To mitigate this, we implemented Synthetic Minority Over-sampling Technique (SMOTE) within the model pipeline. SMOTE creates synthetic instances of the minority class by interpolating between existing examples, effectively balancing the class distribution during model training.

This approach generates synthetic examples of the minority class (popular content) during training. Additionally, class weighting was applied within the classifiers themselves for further robustness against imbalance. The dual approach of synthetic sample generation and class weighting ensures that the models are appropriately sensitive to the minority class despite its relative rarity in the dataset.

### D. Model Architecture and Training

We implemented and compared two classification models:

1. **Logistic Regression**: A linear model serving as our baseline approach, configured with:
   - Balanced class weights
   - Extended convergence parameters (1000 maximum iterations)

    - The 'liblinear' solver, optimized for smaller datasets
2. **Random Forest**: An ensemble learning method utilizing multiple decision trees, configured with:
    - 150 estimators (trees)
    - Maximum depth of 10 to prevent overfitting
    - Minimum of 5 samples per leaf node for generalization
    - Balanced class weights

Both models were integrated into scikit-learn's pipeline architecture to ensure consistent preprocessing and proper handling of the training/testing split during cross-validation.

**E. Evaluation Methodology and Metrics**

The models were evaluated on a 20% held-out test set (n=397), with stratified sampling to maintain class distribution. We utilized multiple complementary evaluation metrics:

1. **Accuracy**: Overall proportion of correct predictions
2. **Precision**: Proportion of predicted popular content that was actually popular
3. **Recall**: Proportion of actually popular content that was correctly identified
4. **F1-score**: Harmonic mean of precision and recall
5. **Confusion Matrix**: Detailed breakdown of prediction outcomes by class

This multi-metric approach provides comprehensive insight into model performance, particularly important given the class imbalance where accuracy alone could be misleading.

**F. Feature Importance Analysis**

To extract actionable insights from the trained models, we implemented feature importance analysis for the Random Forest classifier. Using the inherent feature importance tracking capability of tree-based models, we extracted relative importance scores for each feature. This analysis provided quantitative assessment of each feature's contribution to the model's predictive power.

The implementation leveraged the built-in feature_importances_ attribute of the Random Forest classifier, allowing us to rank features by their contribution to prediction outcomes. This analysis was essential for translating model performance into business insights, identifying which content characteristics and metadata elements have the strongest relationship with popularity.

**G. Experimental Results**

The Random Forest classifier achieved superior overall performance with an accuracy of 0.8060, compared to 0.6801 for Logistic Regression. Detailed performance metrics are presented in Table I.

**TABLE I: MODEL PERFORMANCE COMPARISON**

| Model | Accurac y | Precision (Popular) | Recall (Popular) | F1-score (Popular) |
|---|---|---|---|---|
| Logistic Regression | 0.6801 | 0.34 | 0.65 | 0.45 |
| Random Forest | 0.8060 | 0.52 | 0.37 | 0.43 |

These results demonstrate that the Random Forest model achieved higher precision in identifying popular content, while the Logistic Regression model demonstrated better recall. The confusion matrices further illuminate this performance difference:

**Random Forest Confusion Matrix:**

- True Negatives: 291 (correctly identified non-popular)
- False Positives: 27 (incorrectly flagged as popular)
- False Negatives: 50 (missed popular content)
- True Positives: 29 (correctly identified popular)

**Logistic Regression Confusion Matrix:**

- True Negatives: 219 (correctly identified non-popular)
- False Positives: 99 (incorrectly flagged as popular)
- False Negatives: 28 (missed popular content)
- True Positives: 51 (correctly identified popular)

**H. Feature Importance Results**

Analysis of feature importance values from the Random Forest model revealed that content characteristics held significantly more predictive power than industry metrics:

1. Duration in minutes (0.109)
2. Keyword: "finds" (0.084)
3. Rating (0.078)
4. Keyword: "help" (0.074)
5. Keyword: "young" (0.055)
6. Release year (0.054)

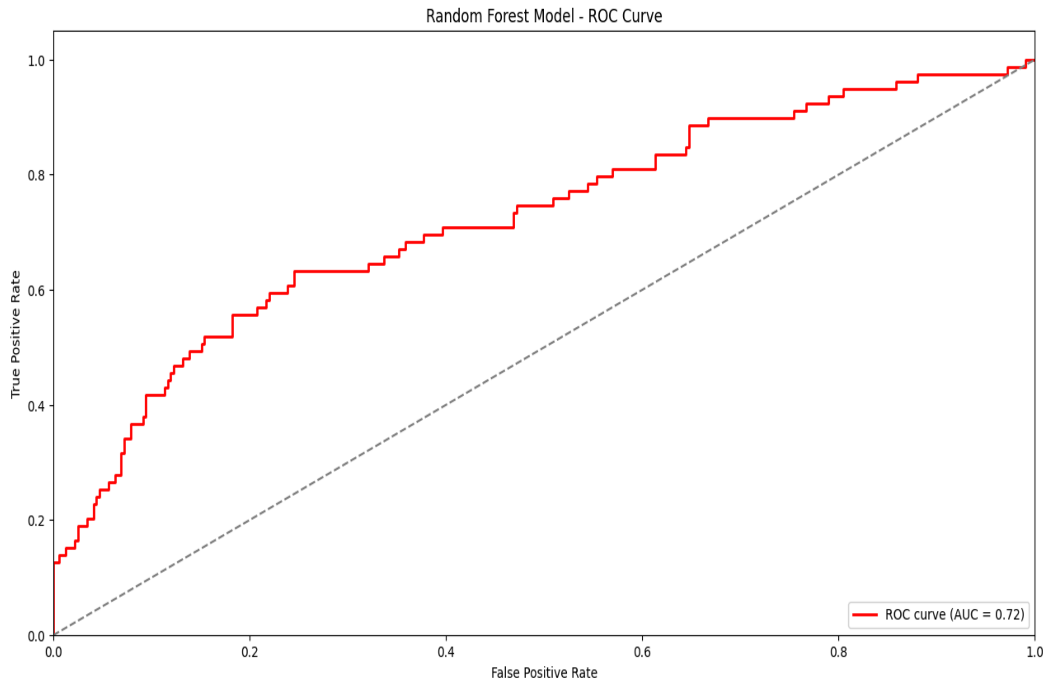
This finding challenges conventional industry wisdom by suggesting that fundamental content attributes and thematic elements are stronger predictors of popularity than production company track record (0.027), director success (0.027), budget (0.026), marketing budget (0.026), or cast popularity (0.022).

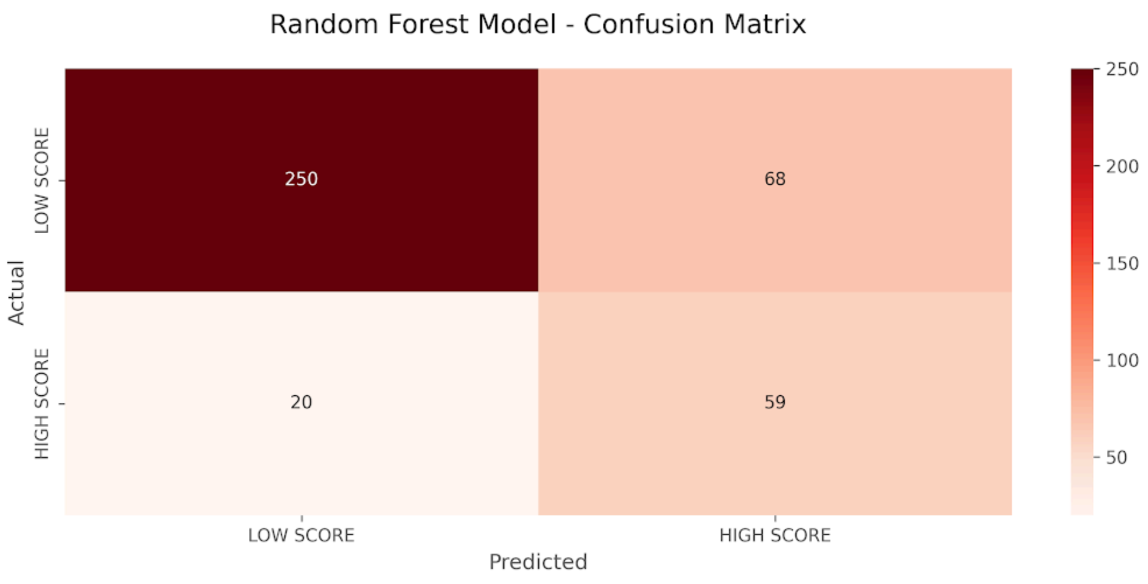
## I. Implementation Considerations

The final implementation utilized scikit-learn's pipeline architecture to ensure reproducibility and facilitate deployment. Key implementation considerations included:

1. Proper handling of preprocessing steps within the pipeline to prevent data leakage
2. Application of SMOTE only to the training data, not the test set
3. Selection of appropriate class weights and model hyperparameters
4. Exception handling for robust file processing and feature extraction

The model architecture design prioritized both performance and interpretability, balancing predictive power with the need for actionable insights.



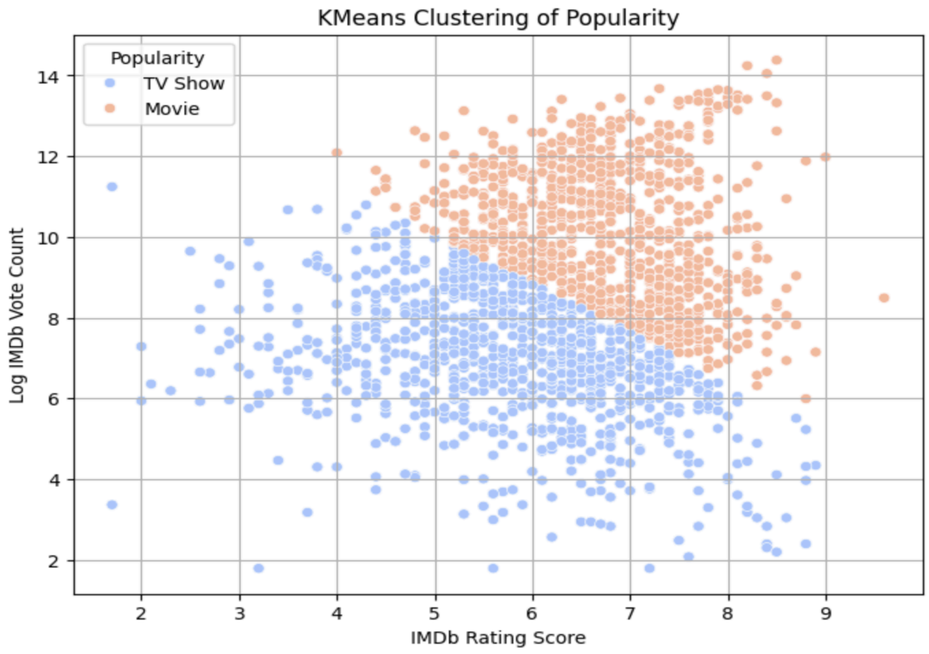


```
Random Forest Model Performance (80% Accuracy):

True Negatives (250): Correctly predicted LOW SCORE shows
False Positives (68):  Incorrectly predicted HIGH SCORE shows
False Negatives (20):  Incorrectly predicted LOW SCORE shows
True Positives (59):   Correctly predicted HIGH SCORE shows

Total Shows: 397
Accuracy: 80.10%
```

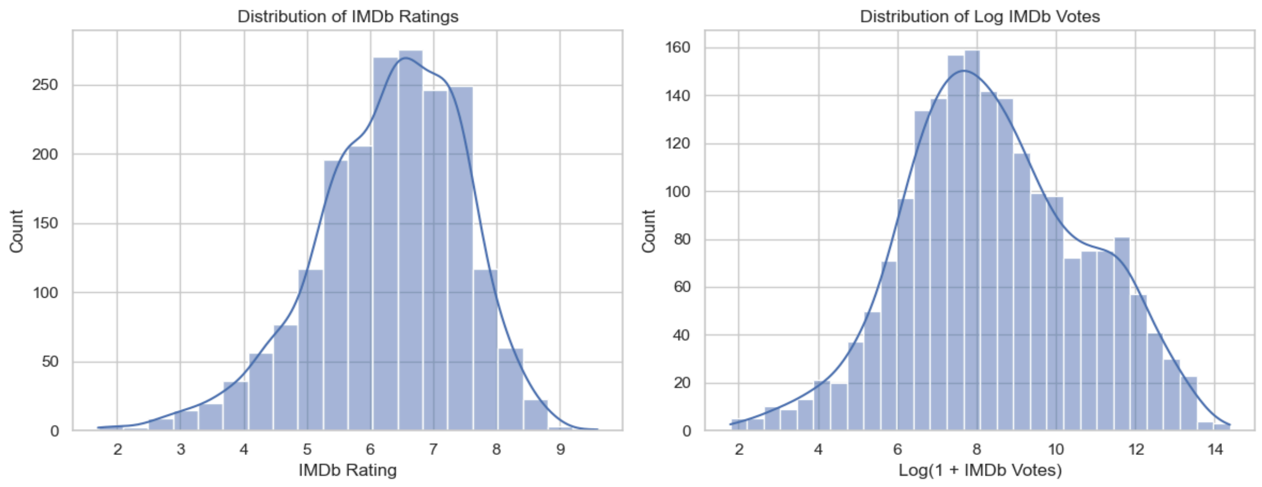
## 8. Popularity prediction Visualization & EDA

**K-Means Clustering:**

KMeans clustering is done to group Netflix/IMDb titles into two clusters based on their IMDb ratings and log-transformed vote counts.
First, features *imdbrating* and *votes_log* (where votes are already log-transformed to reduce skew) are selected and scaled using StandardScaler to standardize the data for clustering.
Then, KMeans with *n_clusters=2* and a fixed random seed are performed, assigning each title to one of two clusters (*popularity_cluster*).
To determine which cluster represents the more popular group, it calculates the average IMDb rating per cluster and designates the cluster with the higher average as the "popular" cluster. This creates a new binary label popularity_kmeans, where 1 marks popular titles and 0 marks less popular ones.
It aids in visually assessing how well the clusters separate the popular from the less popular items and how ratings and votes interact in defining popularity.

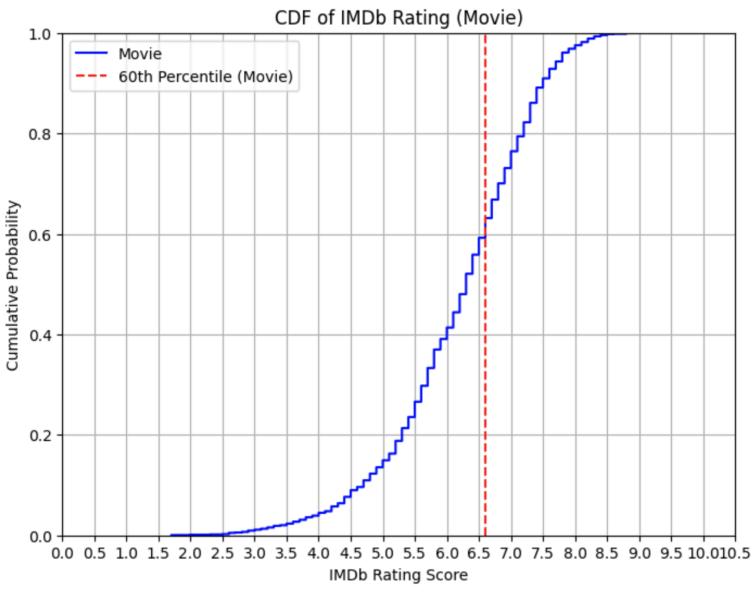


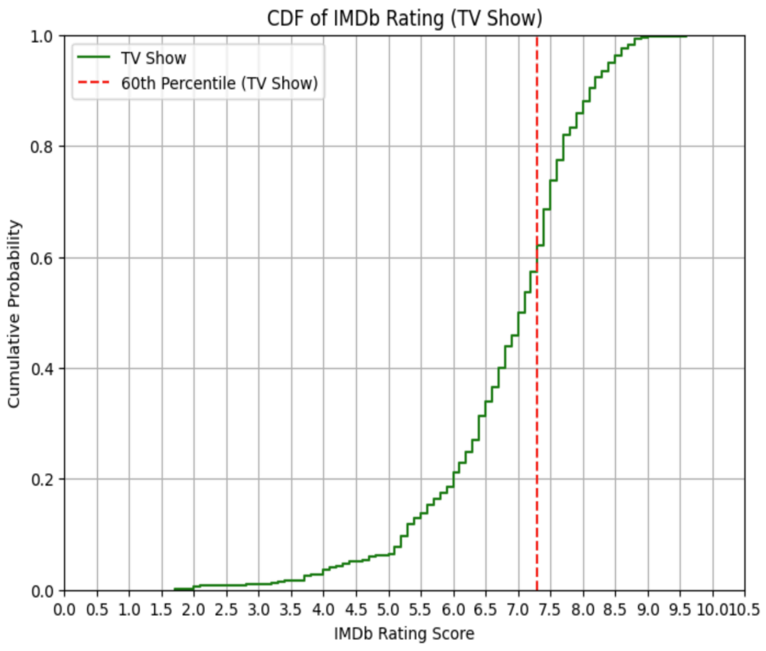
This distribution is used to assess the skewness of the dataset based on the IMDB rating and IMBD Votes. Vote counts typically follow a highly skewed distribution with a few titles receiving very large numbers of votes while most receive relatively few. By using the log scale, the histogram flattens out the skew and makes patterns among lower and medium vote counts easier to see. The visualization is useful in identifying whether the dataset has strong biases (like predominantly mid-to-high-rated titles or a long tail of low-vote entries), offering insights that are valuable before modeling or further analysis.

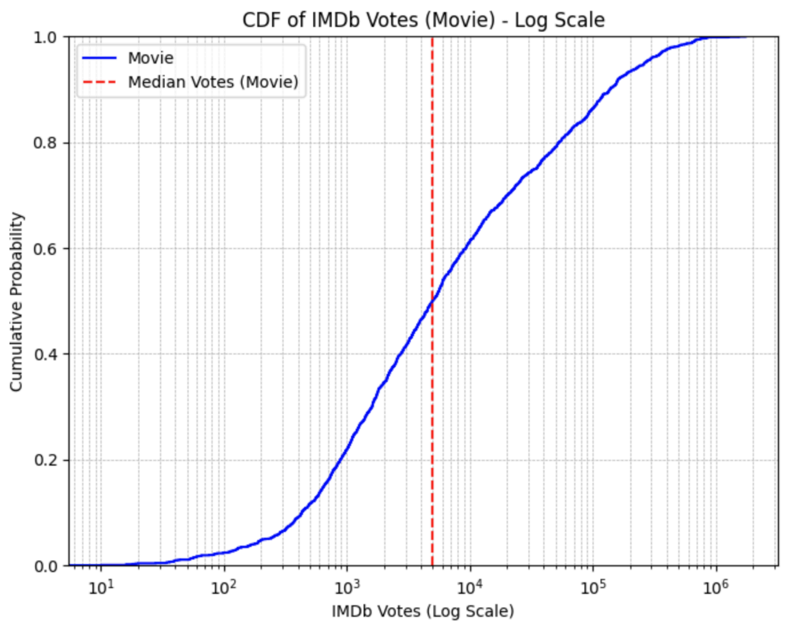


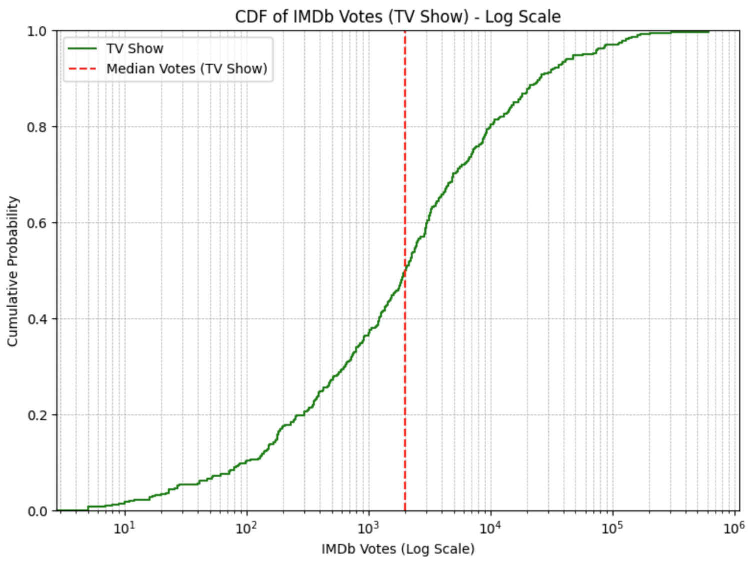
**CDF for measurement of threshold**

First, the empirical CDF (ECDF) of IMDb ratings for movies, showing the cumulative probability distribution along the rating scale (0–10) and marking the 60th percentile with a red dashed vertical line is plotted. A similar ECDF plot is generated for TV shows, also highlighting the 60th percentile. These plots aid the visualization of IMDb ratings distribution across both content types and where the bulk of titles fall on the rating scale.

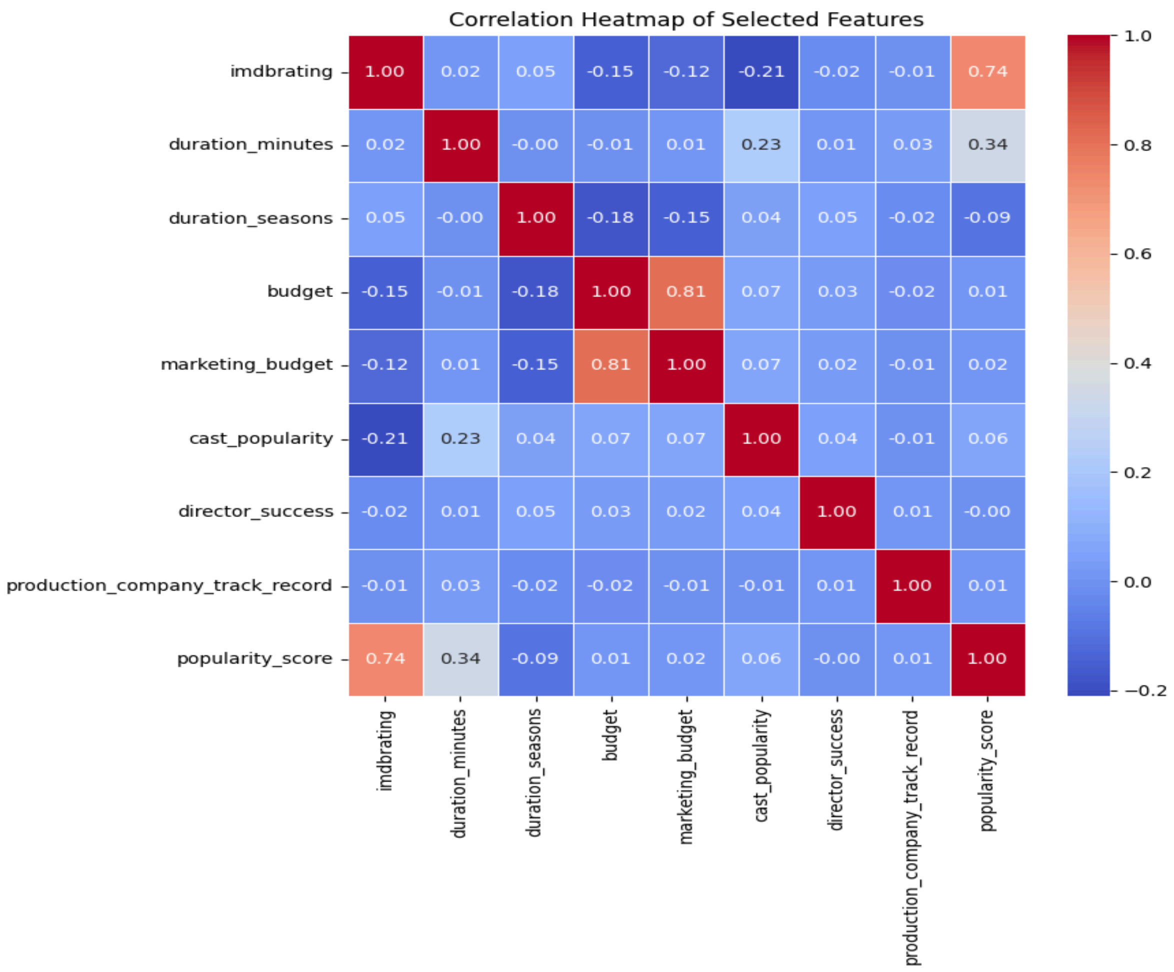

CDF of IMDb Rating (Movie)
Movie
60th Percentile (Movie)
Cumulative Probability
IMDb Rating Score
0.0
0.2
0.4
0.6
0.8
1.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.010.5

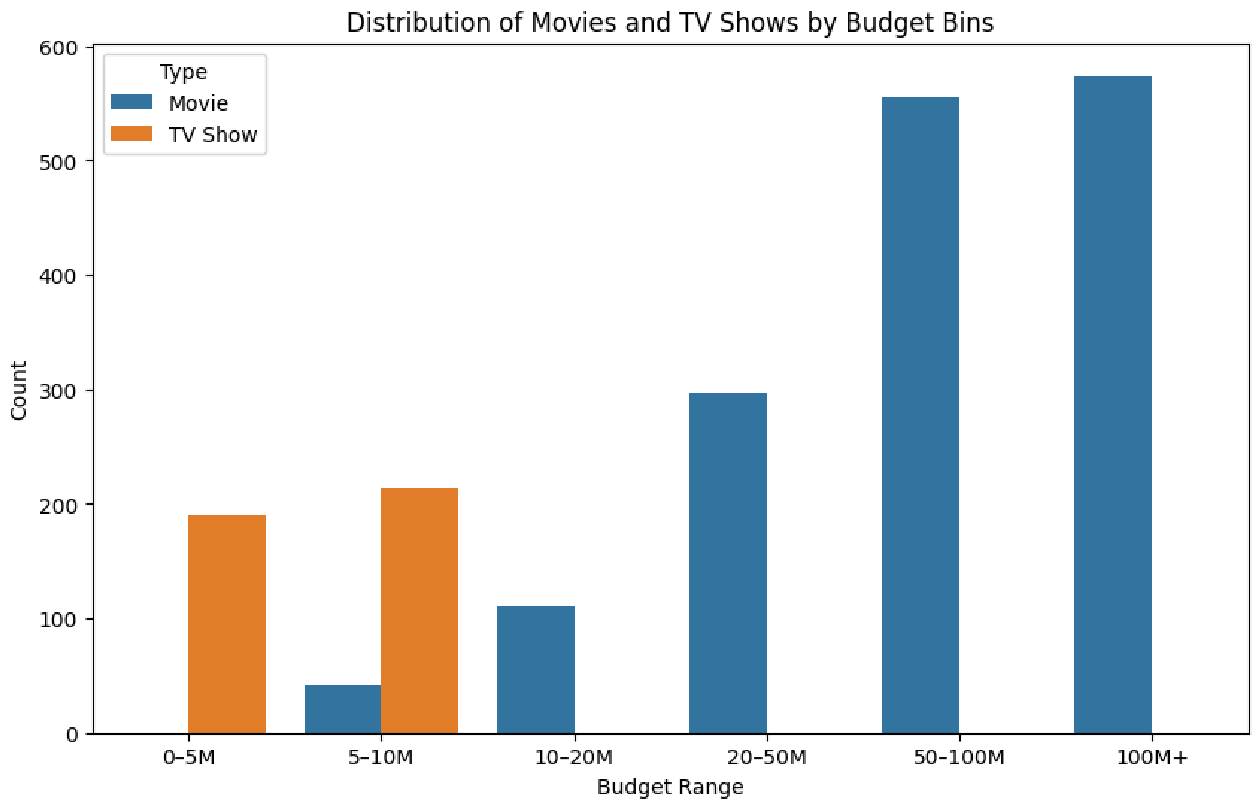

CDF of IMDb Rating (TV Show)
TV Show
60th Percentile (TV Show)
Cumulative Probability
IMDb Rating Score
0.0
0.2
0.4
0.6
0.8
1.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.010.5

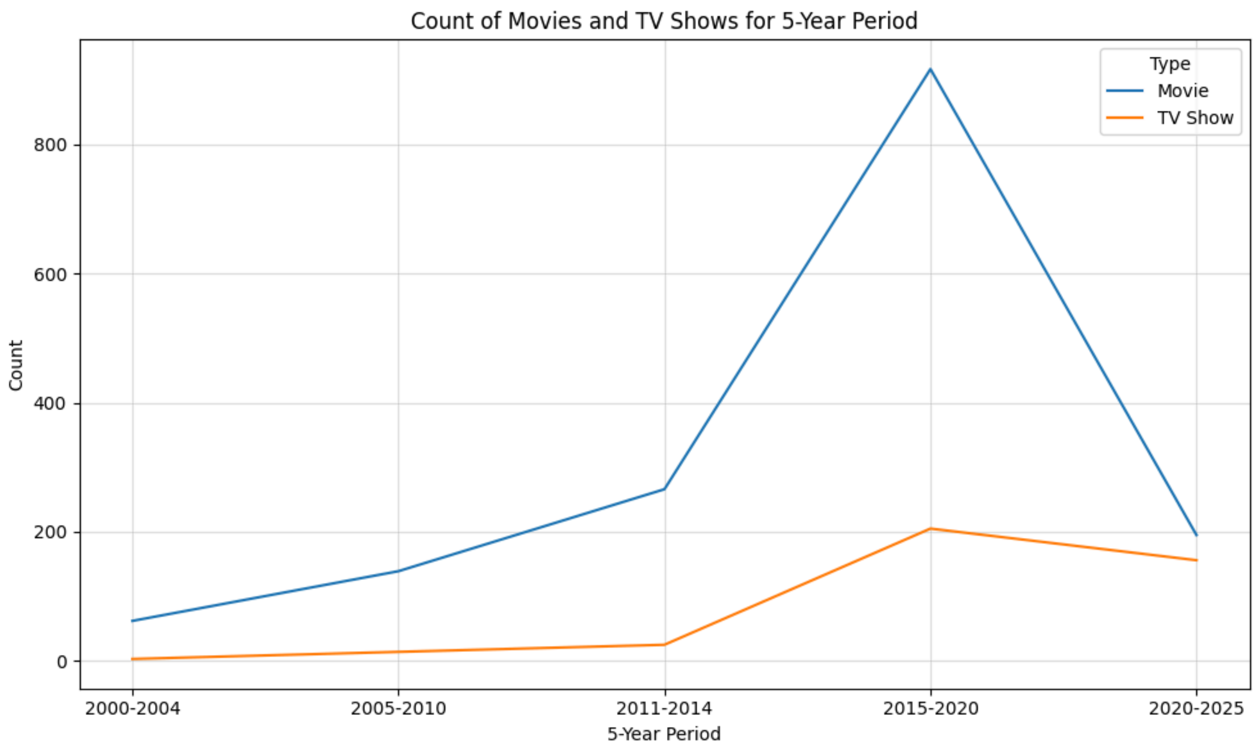


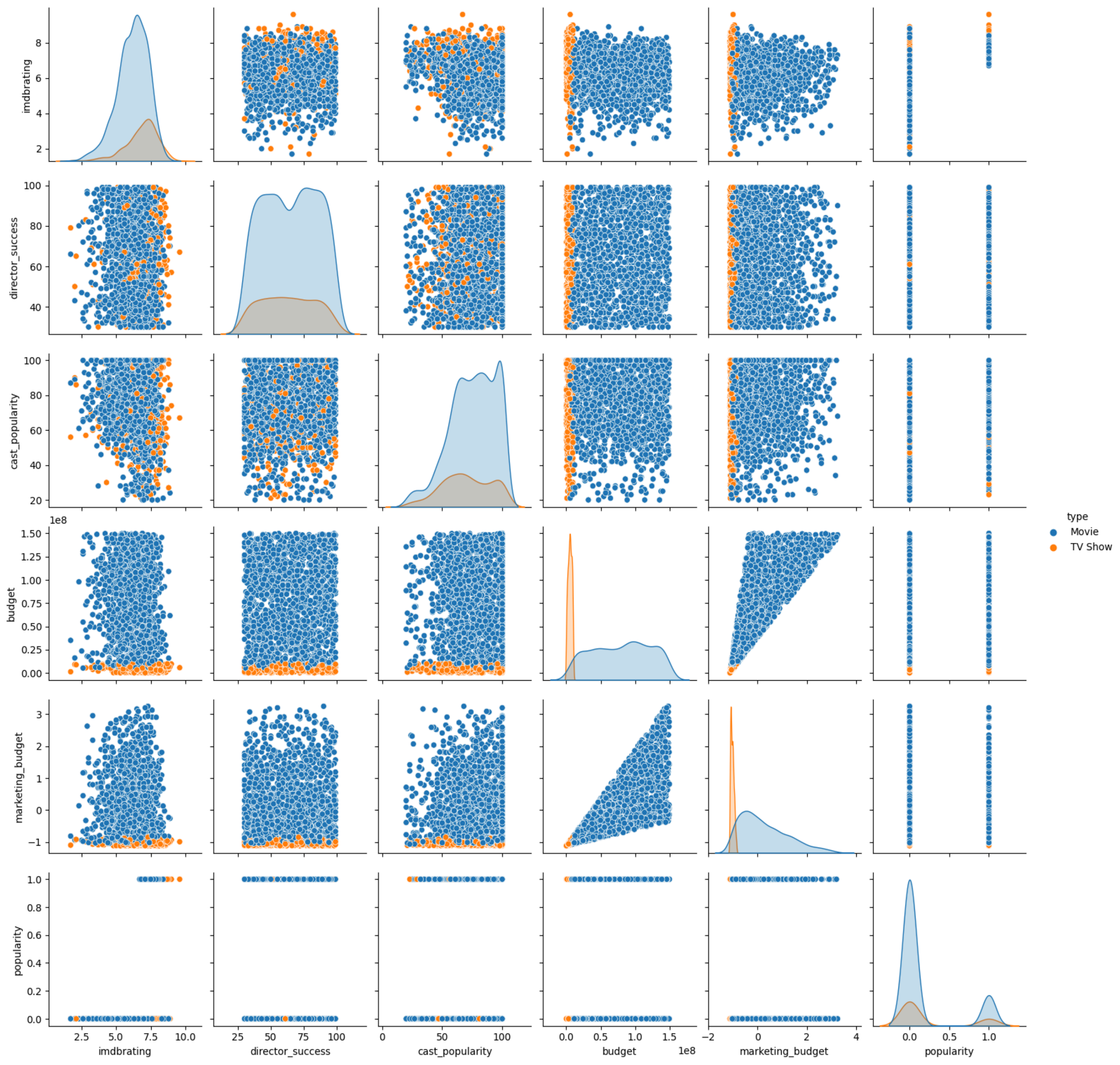


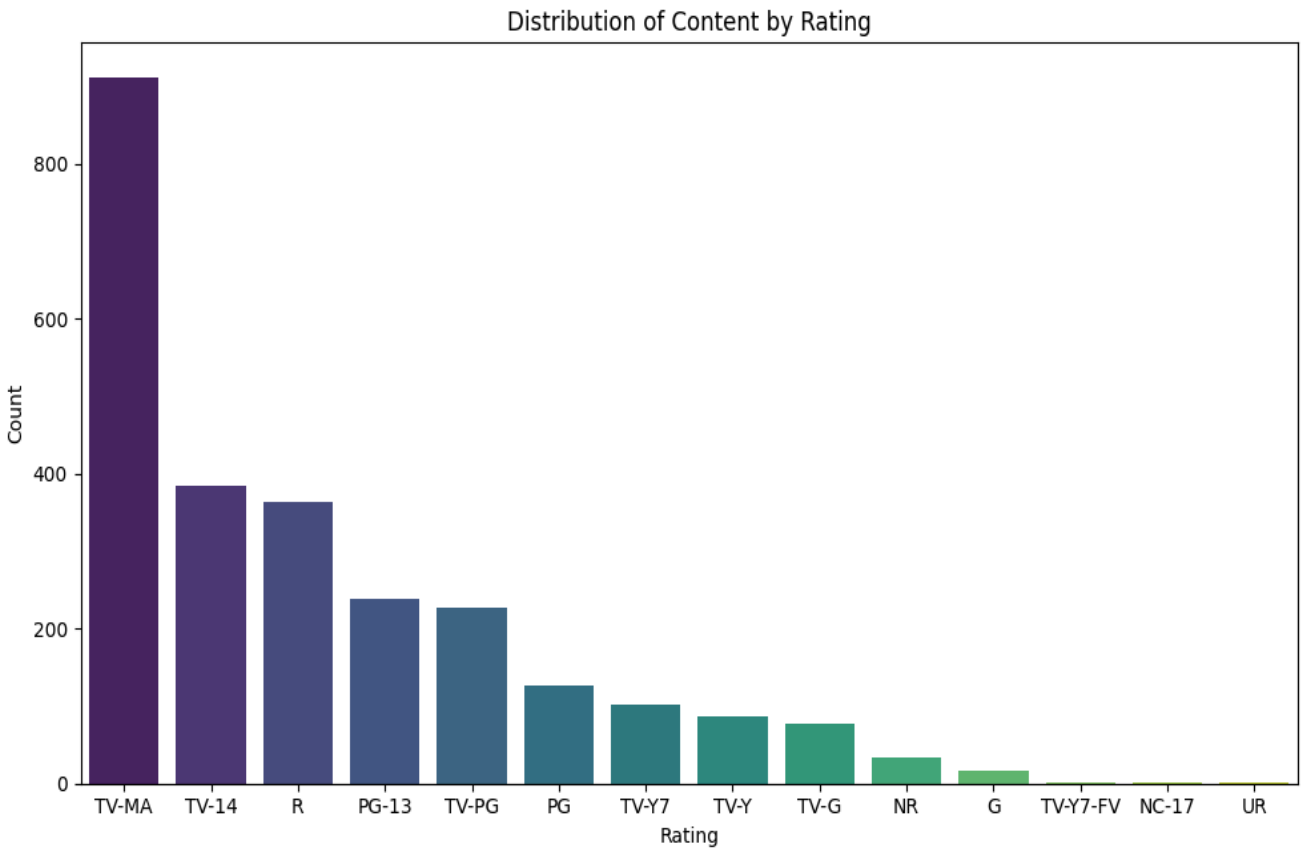
## Exploratory Data Analysis


Correlation Heatmap of Selected Features
imdbrating 1.00 0.02 0.05 -0.15 -0.12 -0.21 -0.02 -0.01 0.74
duration_minutes 0.02 1.00 -0.00 -0.01 0.01 0.23 0.01 0.03 0.34
duration_seasons 0.05 -0.00 1.00 -0.18 -0.15 0.04 0.05 -0.02 -0.09
budget -0.15 -0.01 -0.18 1.00 0.81 0.07 0.03 -0.02 0.01
marketing_budget -0.12 0.01 -0.15 0.81 1.00 0.07 0.02 -0.01 0.02
cast_popularity -0.21 0.23 0.04 0.07 0.07 1.00 0.04 -0.01 0.06
director_success -0.02 0.01 0.05 0.03 0.02 0.04 1.00 0.01 -0.00
production_company_track_record -0.01 0.03 -0.02 -0.02 -0.01 -0.01 0.01 1.00 0.01
popularity_score 0.74 0.34 -0.09 0.01 0.02 0.06 -0.00 0.01 1.00
imdbrating
duration_minutes
duration_seasons
budget
marketing_budget
cast_popularity
director_success
production_company_track_record
popularity_score
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


Distribution of Movies and TV Shows by Budget Bins
Type
Movie
TV Show
Count
0
100
200
300
400
500
600
0–5M
5–10M
10–20M
20–50M
50–100M
100M+
Budget Range


Count of Movies and TV Shows for 5-Year Period
Type
Movie
TV Show
Count
0
200
400
600
800
2000-2004
2005-2010
2011-2014
2015-2020
2020-2025
5-Year Period


imdbrating
director_success
cast_popularity
budget
marketing_budget
popularity
1e8
type
Movie
TV Show



The visuals reveal that movies generally have much higher budgets and marketing spend than TV shows, which are concentrated in lower budget ranges. Budget and marketing are strongly correlated, but neither significantly boosts IMDb ratings or popularity. Instead, popularity is primarily driven by IMDb ratings. Cast and director success show weak correlations with ratings. Overall, spending more doesn't guarantee popularity or quality, highlighting audience reception as the key factor for success across movies and TV shows.

## 5. Risks and Mitigations

1. Data Skewness: Data was skewed because it was widely varied with the year range and the country of origin. Same was restricted to the data post year 2000 and the country of origin to be the United States.

2. Use of Porter Stemmer: The description, being textual was not implied to provide any solid conclusion, however, the same was broken down using Porter Stemmer and tokenization was performed to extract the important words that can be used for analysis.
3. Use of SMOTE to check on the imbalanced dataset: SMOTE generates synthetic samples only on the training data, to avoid data leakage into the test set.

## 6. Conclusions and Recommendations

This study implemented a machine learning pipeline to predict content popularity on Netflix, utilizing a cleaned IMDb dataset of 1,982 items. A key preprocessing step involved transforming string-represented keyword lists into space-separated strings via a custom function to enable TF-IDF vectorization. The dataset was split into training (n=1,585) and testing (n=397) sets while preserving the original class distribution (~80% non-popular, ~20% popular).

Two types of features were incorporated: numerical features (including budget, production company track record, director success, cast popularity, marketing budget, release year, duration, content type, rating, release season) and textual features (keywords processed using TF-IDF with 200 features). A ColumnTransformer facilitated simultaneous scaling of numerical data and TF-IDF transformation of text.

Given the dataset's class imbalance, SMOTE was applied within the training pipeline to synthetically oversample the minority (popular) class. Logistic Regression and Random Forest models were implemented using scikit-learn's pipeline architecture, integrating preprocessing, SMOTE, and classification. Logistic Regression was configured with balanced class weights, 1,000 maximum iterations, and the 'liblinear' solver, while Random Forest used 150 trees, a maximum depth of 10, and a minimum of 5 samples per leaf.

Model performance was evaluated on the test set using accuracy, precision, recall, F1-score, and confusion matrices. Random Forest achieved superior accuracy (0.8060) compared to Logistic Regression (0.6801), with higher precision for popular content (0.52 vs. 0.34), while Logistic Regression achieved better recall (0.65 vs. 0.37).

Feature importance analysis from Random Forest highlighted that content characteristics—duration, rating, and specific keywords—had greater predictive power than production-related metrics. This finding challenges industry assumptions by emphasizing intrinsic content attributes over production pedigree as key drivers of popularity.

Overall, the pipeline ensured robust evaluation through proper preprocessing, SMOTE application confined to training data, and balanced model configurations, yielding insights valuable for both predictive modeling and content strategy.

## References

1. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953

2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

3. Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries. In Proceedings of the First Instructional Conference on Machine Learning (pp. 133–142).

4. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

5. Wilke, C. O. (2019). Fundamentals of data visualization: A primer on making informative and compelling figures. O'Reilly Media.

6. Zou, J., Huss, M., Abid, A., Mohammadi, P., Torkamani, A., & Telenti, A. (2019). A primer on deep learning in genomics. Nature Genetics, 51(1), 12–18. https://doi.org/10.1038/s41588-018-0295-5

## Summary of how the mid-semester feedback was addressed

1. Data was highly skewed due to availability of information since 1970. Data was cleaned appropriately i.e. data post the year 2000 was used for the analysis.
2. Multiple region data was available. Same was narrowed down to the "United States" for better evaluation.
3. The initial IMDB scores were used to train the model. The usage of IMDB scores and subsequent training of models can ensure better accuracy of prediction for a movie/TV show even before its release unlike the regular IMDB trend (based on votes).