



Film Industry Analysis Proposal

(Machine Learning Model to predict the movie's success)

21.11.2023

—

By Group 2

- Myo Thiha,
- Rakshya Lama Moktan

Introduction


The film industry faces an important challenge: predicting the success of films due to which millions are lost each year. Despite significant investment and a reputable cast, there is no guarantee of a film's profitability. The goal of this project is to address this challenge by leveraging comprehensive historical film data to analyze and predict key attributes that influence a film's success. Our goal is to provide valuable insights into the film industry and the factors that contribute to a film's audience and box office receipts.

Problem Statement

The central problem facing the film industry is the inherent unpredictability of a film's success. Even with significant investments and a highly capable cast, there is no guarantee of a profitable result. Our goal is to examine historical film data, discern underlying trends, and isolate essential attributes that influence film success. We then aim to leverage these identified characteristics to predict the film's potential performance, including audience and gross receipts.

Related Work

To address this challenge, we will build on existing research and related work, including studies such as "Box Office Forecasting" (BoxOfficePro) which provides insights into the box office, and "Streaming Platform" (Netflix) forecasts that help highlight trends in the entertainment industry.



Academic articles such as “Forecasting Box Office Revenues” (Ruus and Sharma, 2019) and “Predicting Movie Success” (Darapaneni et al., 2020) provide valuable insights on predicting movie success using metadata, reviews, and machine learning algorithms.

Dataset

For this project, we will use two main datasets:

1. TMDb Dataset:

This dataset provides metadata for 45,000 movies released before July 2017, from TMDb (Movie Database) and GroupLens. It includes details like cast, crew, plot keywords, budget, revenue, release date, language, production information, and user ratings.

2. IMDb Dataset:

IMDb Dataset provides information about movie features including IMDb Movie ID, Release Year, Certificate, Runtime performance, genre, ratings, description, director, stars, votes, and box office gross. Data. The inclusion of additional details like director and star IMDb IDs enhances the granularity and precision of the dataset.


Methodology

Our project will follow a structured methodology that includes:

Initial Preprocessing (Before EDA)

Merging of datasets

As the movie dataset of TMDb is scattered and fragmented into genres: each genre was in a separate CSV file. We had to merge them through outer join using movie_id present on both (though ‘tt’ had to be removed from TMDb id and so on. The datasets for TMDb and



IMDB were also segmented. We merged them through right join (as many movies were present on TMDb and then on IMDB, TMDb also had more more information. Then we removed some unwanted columns: "video", "poster_path", "homepage", "overview", "tagline", and "description" which are individual movie specific and do not have computational power. The shape of our initial unclean merged dataset was (78116, 14)].

Merging of columns


Once we merged the dataset we had a huge number of nan columns that needed to be addressed. There were also some redundant columns. For example: genres were replicated as we had genre columns in both IMDB and TMDb for the first part we had to make both the genres similar, for which we removed id from the IMDBbased genre column while the genre_y which is the TMDb genre column was split. After this we performed the following:

1. Conversion of IMDB genre from dictionary to list
2. Merging both IMDB and TMDb genre columns by keeping only the genres unique to both
3. Converting the genre list back to set
4. Dropping of unwanted redundant genre columns

We then further dropped more columns such as 'movie_id', 'tmdbId', 'tmdbId', 'imdbId', 'id', and 'original_title'. We also extracted the year from the release_date and combined it with the other release_year column which has been analyzed in the EDA. As a final step for the initial preprocessing we merged the final redundant columns movie_title and title and our final dataset was set for the EDA.

Feature Extrapolation

As our merged dataset is huge combined we decided to perform feature extrapolation for cleaner and less redundant data. We started with cleaning the belongs_to_collection columns which addresses whether the movie has a prequel or not. We applied a lambda collection to get only the name of the collection the movie belongs to along with its ID in a new column for each. We then preprocessed the language columns where we extracted the



3 languages the movie was produced in different columns(only 3 is an arbitrary number for easier computing). A similar approach was also implemented for production_companies and country_data where we extracted them to different columns (only 3 were also taken here).

We then proceeded to extrapolate the genre columns to separate columns for each unique genre of the movie where if present it was set to 1 else to 0. For example: if a movie has humor in its genre then the value for that specific movie record is set to 0. We also combined the revenue_x and revenue_y to revenue, runtime_x and runtime_y to runtime column using combine_first which combines two DataFrame objects and uses the value from the second DataFrame if the first has a NULL value.

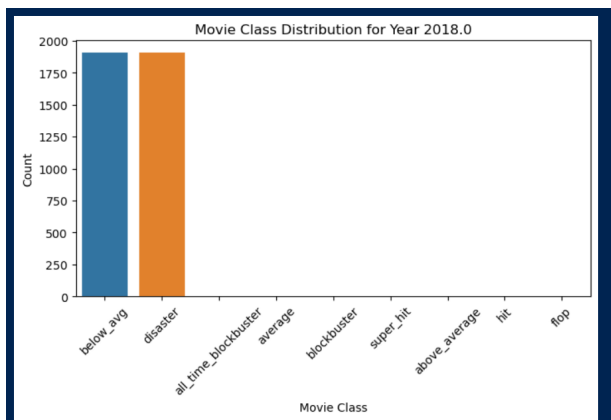
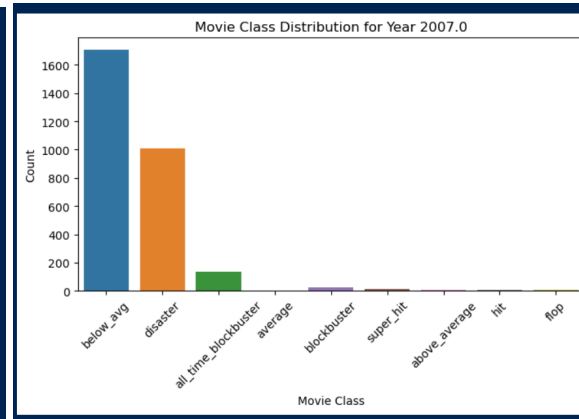
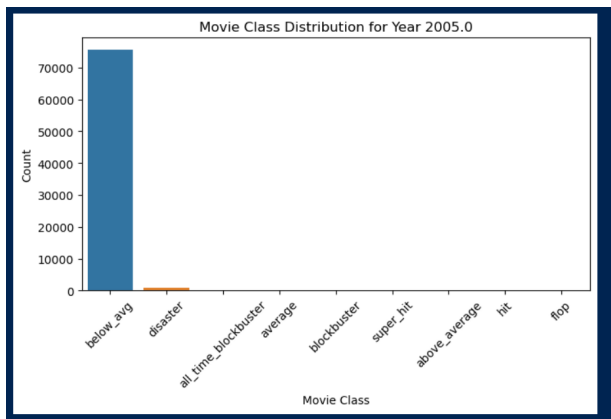
The final shape of the cleaned dataset was (1306179, 74)

Exploratory Data Analysis

Movie class distribution for each year

What is the movie success distribution for each year?

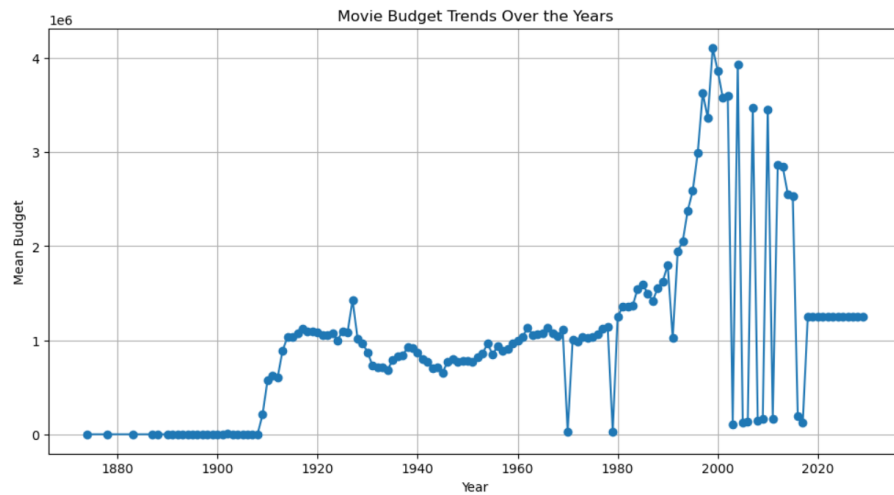
The film industry experienced fluctuations in movie classifications from 1985 to 2012, with most movies falling into the categories "Below Average" and "Disaster." Between 1994 and 1997, a majority of movies underperformed, while between 2007 and 2010, movies were well-observed across all classifications. Between 2012 and 2016, a mix of "Below Average," "Disaster," and "All-Time Blockbuster" movies was seen, with a balanced distribution between "Below Average" and "Disaster" from 2018 to 2023. Most movies grossed below average or disasters, with few all-time blockbusters and very few flops, hits, and averages. Some of the notable outputs are given below:



Yearly investment trend

What is the yearly investment trend in movies?

The investment across the years seems to grow significantly with the highest in the year around 2000 which decreased around 2020 which might be due to the outbreak of coronavirus. There was also a slight peak in the year around 1925 during major events such as the transition from silent films to "talkies" with synchronized sound, which was a revolutionary development in cinema and the prominent rise of Hollywood.



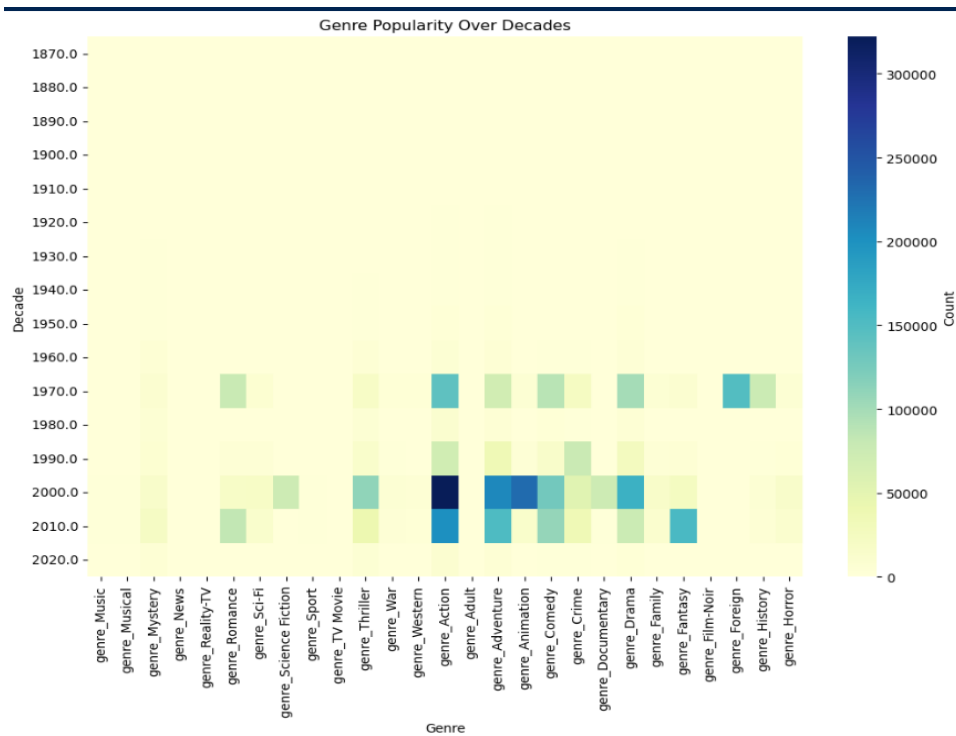
Popularity of genres over the years

Which genre is popular each year?

There seems to be a growing trend of Action movies from 1970 to 2010 onwards with the highest in 2000 when many Marvel and DC movies were released. The adventure and animation category also seems to be gaining momentum whereas the trend of Foreign and history seems to have decreased.

The top genres appear to be Action, Adventure, Animation, Fantasy, and Comedy

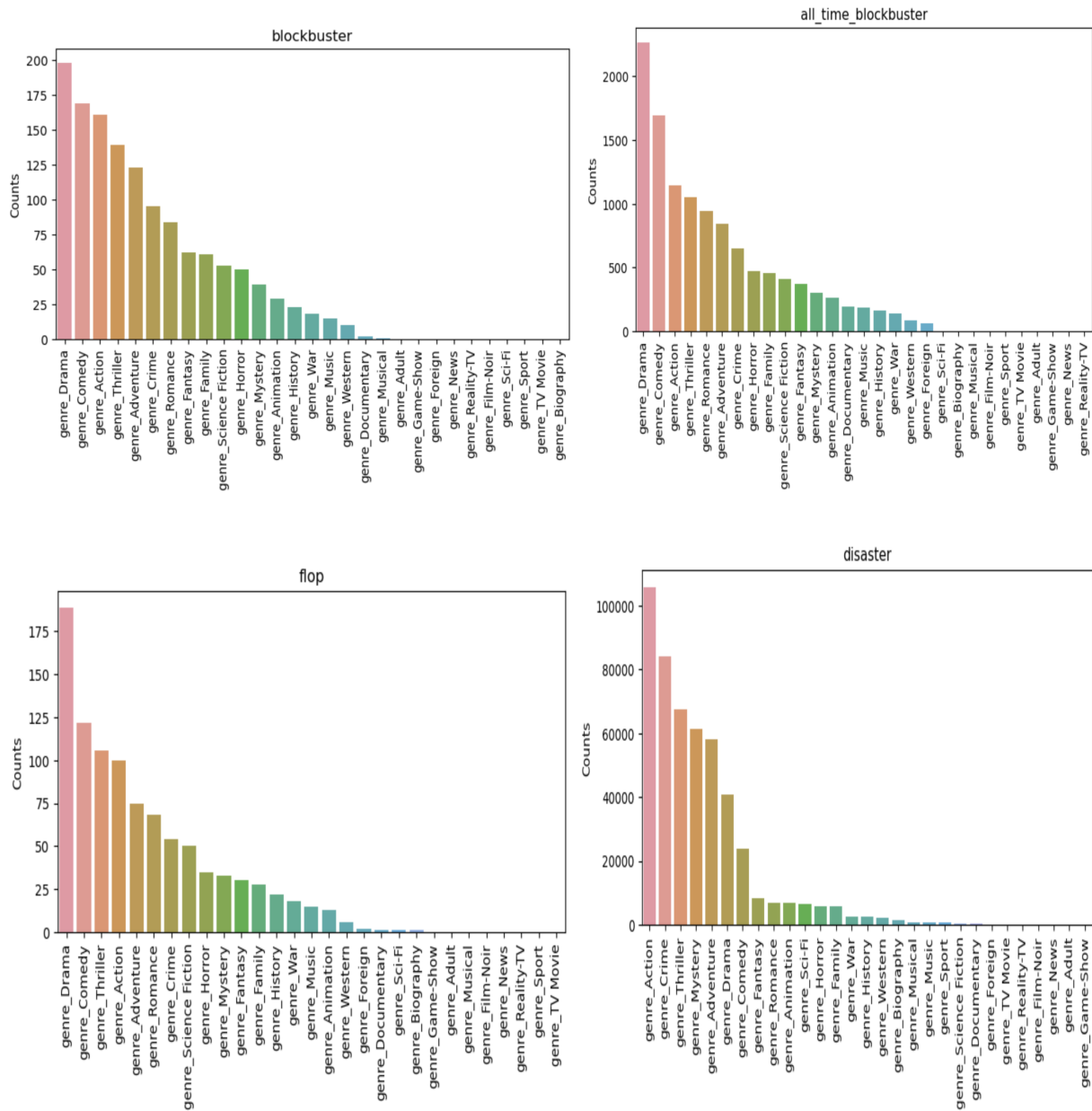
In summary, this project aims to provide a comprehensive analysis of the film industry, providing insight into the factors that determine a film's success. By leveraging two rich data sets and established methods, we hope to open up a wealth of knowledge about the complexities of the film industry.



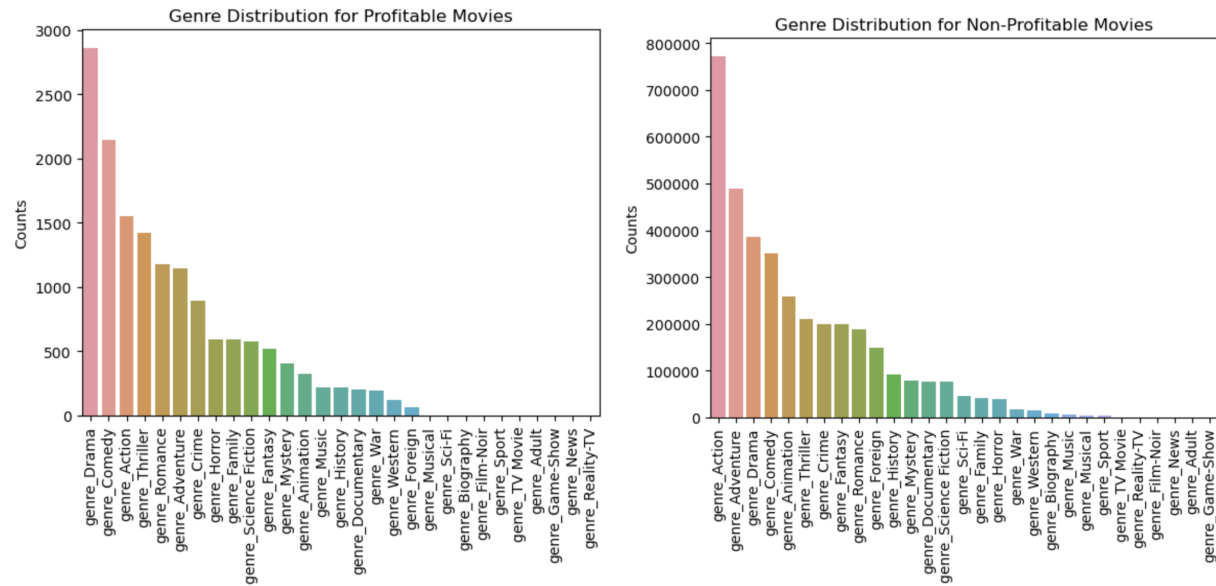
Role of Genre in Movie Success

How does the movie genre contribute to movie success?

We have also examined the genre distribution for movie categories to assess the influence of genres on the success of a film. Given the extensive range of movie categories available, we will exclusively present the data for the four extreme categories: "all-time blockbuster," "blockbuster," "flop," and "disaster."



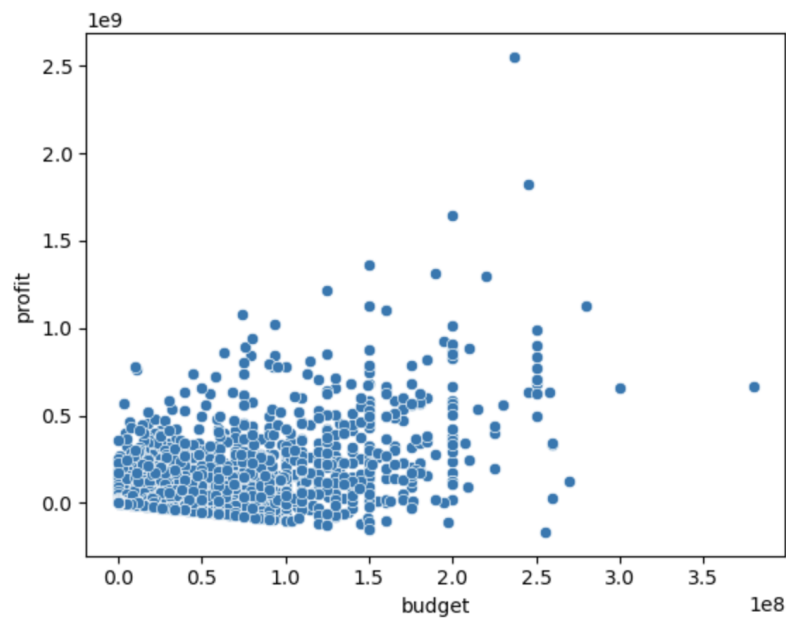
We also check the distribution of genres for movies that were generated and movies that weren't.



Budget vs Profit

How does the movie budget contribute to movie success?

We also tried to understand the relationship between the budget allocated and the profit generated. While it is acknowledged that budget is not the primary factor influencing a film's success, it is clear that the profits of some films will increase as their budgets increase. Therefore, it can be inferred that the budget size contributes partly to the overall profit.



Budget vs Movie Success

How does the movie budget contribute to movie success?

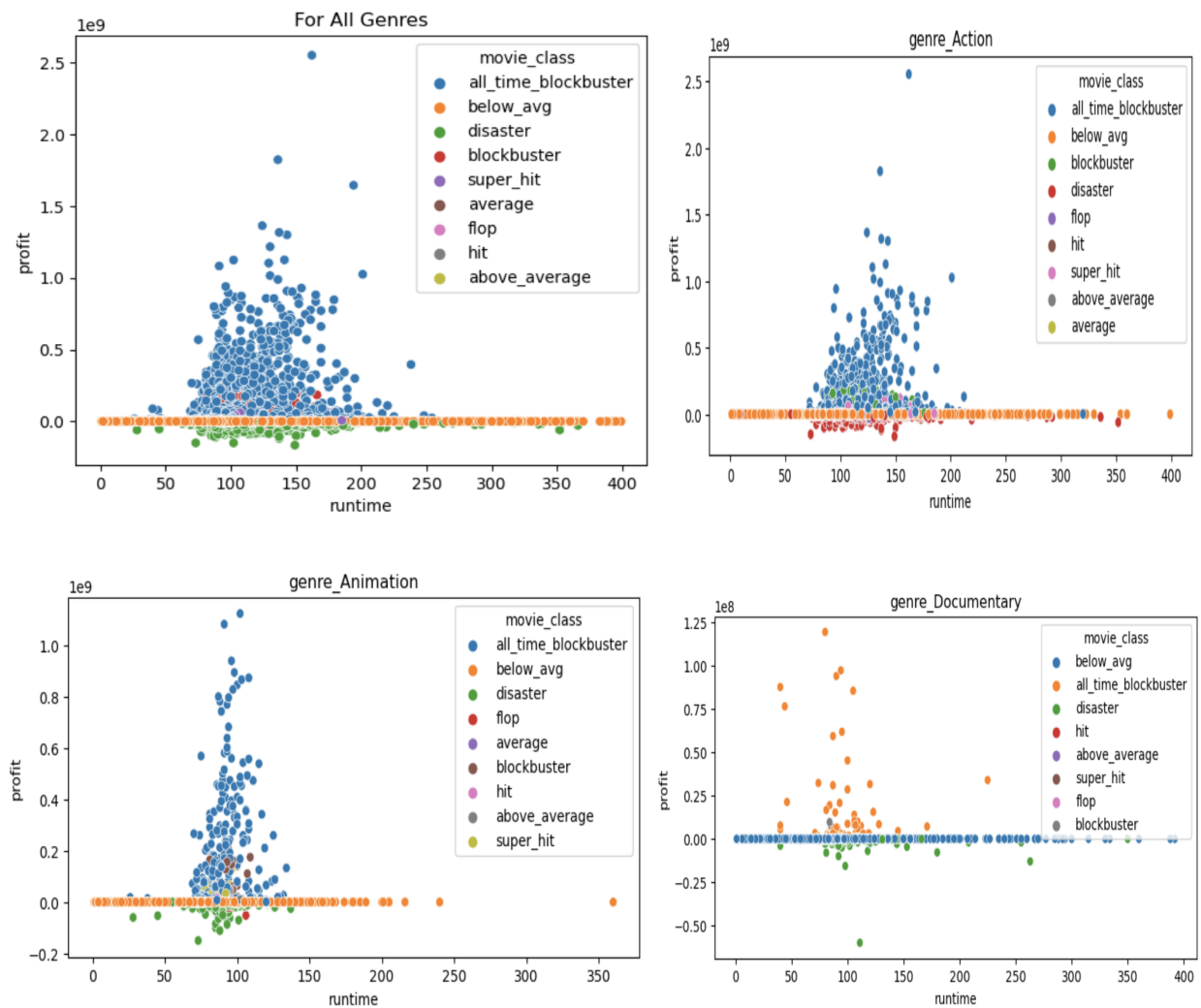
We have endeavored to calculate the budget associated with each film course. Based on the graphical presentation, it's clear that the majority of films classified as "below average" and "disasters" actually had relatively low budgets. This observation is consistent with the general view that producing a high-quality film often requires a significant financial investment.

	movie_class	budget_class	budget
0	above_average	high	52
1	above_average	low	20
2	above_average	mid	112
3	all_time_blockbuster	high	603
4	all_time_blockbuster	low	2730
5	all_time_blockbuster	mid	1257
6	average	high	35
7	average	low	19
8	average	mid	64
9	below_avg	high	46
10	below_avg	low	834393
11	below_avg	mid	116
12	blockbuster	high	165
13	blockbuster	low	63
14	blockbuster	mid	234
15	disaster	high	231
16	disaster	low	136206
17	disaster	mid	1949
18	flop	high	78
19	flop	low	49
20	flop	mid	214
21	hit	high	36
22	hit	low	32
23	hit	mid	83
24	super_hit	high	108
25	super_hit	low	48
26	super_hit	mid	178

Movie Runtime vs Profit

Does the movie runtime contribute to profit?

Next, we create a scatter plot between movie runtime and profit to see the relationship between movie runtimes and profits they made. We create scatter plots for each genre.

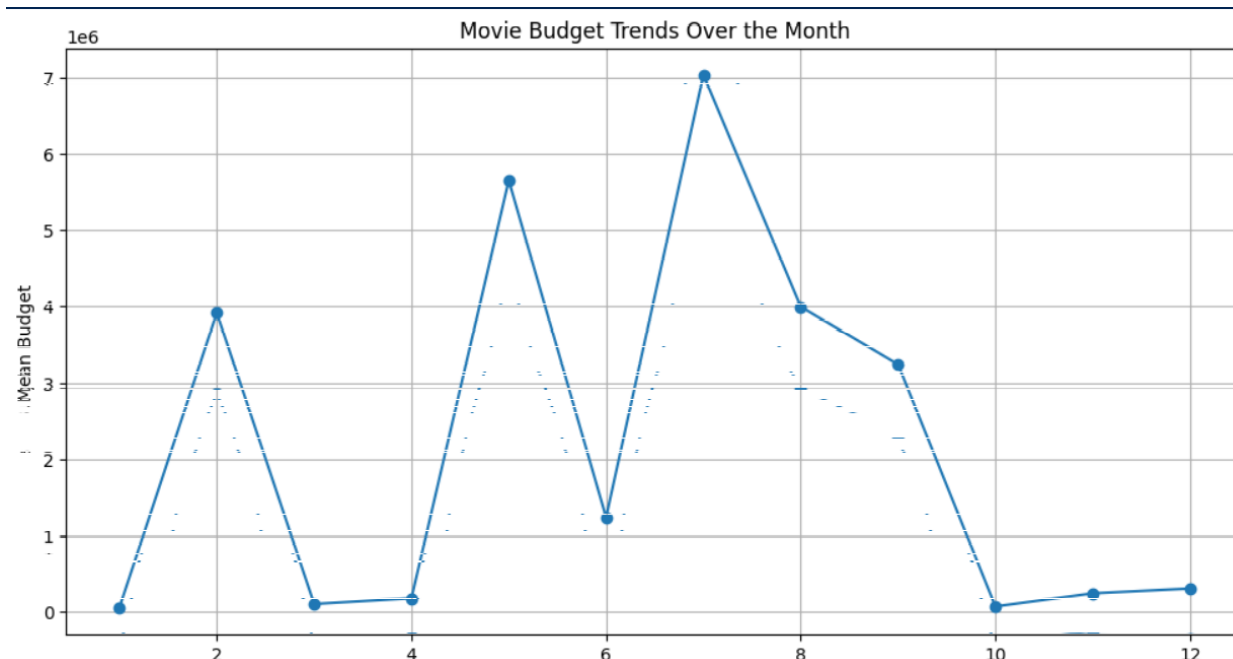


Deductions

- Overall, a movie's runtime should be within a certain range.
- For some genres, the runtimes are highly related to the profit.

Analysis of the Movie Budget through the months

How is the movie budget through the months



According to the plot above, we can observe the movie budget trends through the months. The movie budget of the movies released in July seemed to have the biggest budget. While the ones in January seemed to have the least amount of budget (appears to be 0). This might be true because in January many people go on vacations and do not visit the movies as much while in July most schools and colleges have leisure time so more big-budget releases are targeted then.

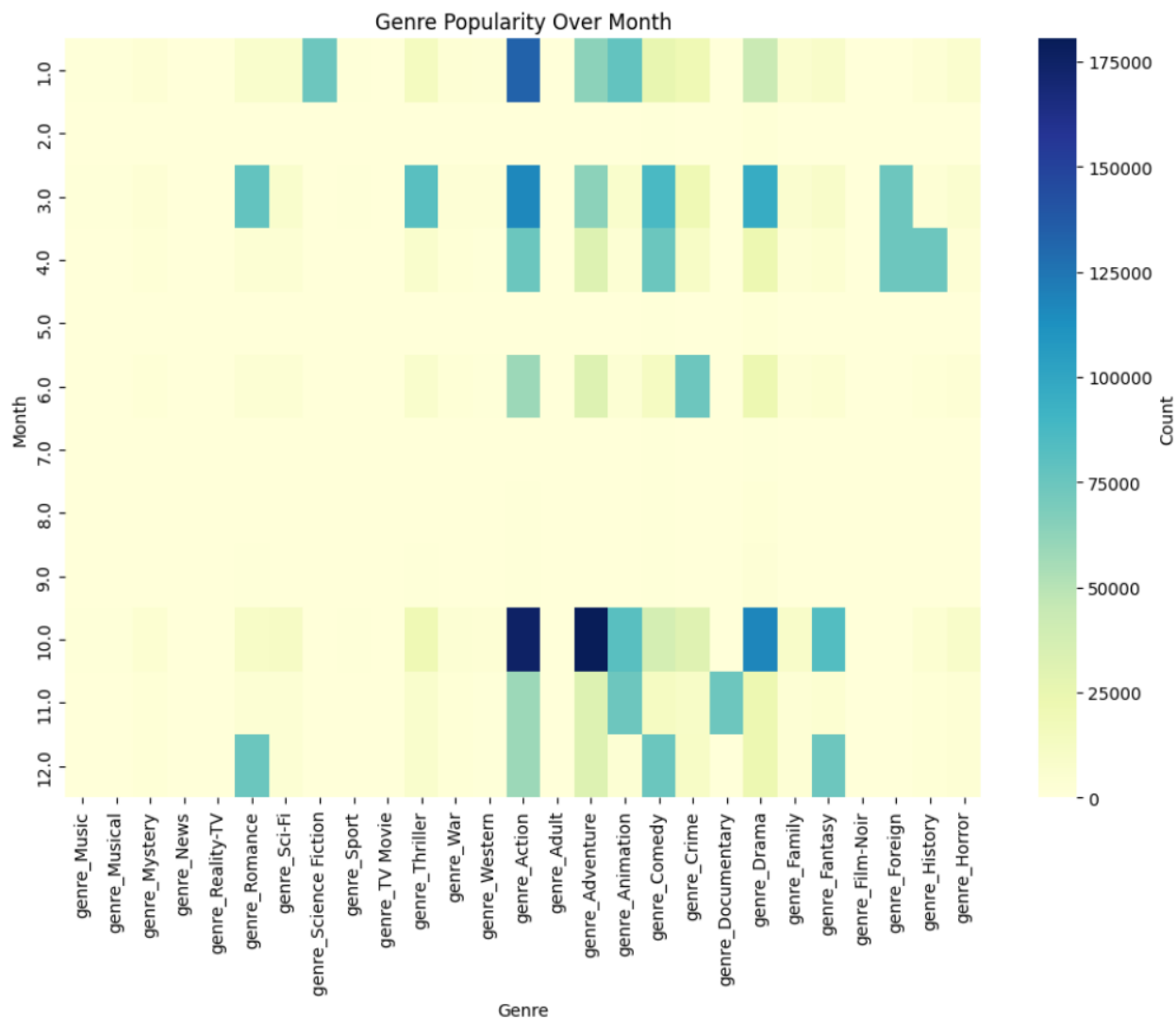
Popularity of genres across the months

Which genre is popular for each month?

There seems to be a specific movie genre that is popular for a particular month. Action along with Adventure genres seems to be the most popular, especially for October. Dramas are also popular in October. Action movies are also popular for January and March while Romance movies are the most popular for January.

Hence, the specific movie genres should target these release months (for example: Action movies should be targeted for October release along with Adventure.


This analysis provides ground for understanding the pattern between movie release months based on genre and their success probabilities.



Feature Engineering

Computation of Star Power

The success of movies depends on the cast and crew involved with it. For example: Tom Cruise is one actor in Hollywood who determines the success of the movie while a similar effect of Shahrukh Khan has been observed in Bollywood movies. Hence, decided to compute the star power and director power of the cast crew information provided to us.



For this, we used the star and revenue columns first we collected all the movie revenues of the star and summed them.

We then also used a similar approach for directors where we used the director power by summing all the revenues of his movies.

Finally, we had the columns star_power and director_power which we could use for our model building.

Classification of movie gross collection based on the reference

<https://balusboxoffice.quora.com/Classification-of-Movies-as-Hits-Superhits-Blockbusters-Flops>

We classified the movies based on their budget as follows:

- All time blockbuster
- Blockbuster
- Super hit
- Hit
- Above Average
- Average
- Below Average
- Flop
- Disaster

Extraction of year and month from Release Year

We also extracted the release year and month from the release date and created new columns. This helped in easier visualization of data and learning of new patterns. It also helped me understand the time effect on profit.

Data Pre-processing and Feature Selection

Feature Selection

Based on our EDA we have the following columns: 'budget', 'release_year', 'release_month', 'runtime', 'certificate', 'star_power', and 'director_power' as prominent in the prediction which is set as X. We also set our label ['movie_class'] as Y.

We then split the dataset into X_train, X_test, y_train, and y_test in the ratio of 0.8:0.2

Data Preprocessing

To handle null values in the dataset and address potential data leakage, a pipeline was implemented to enhance efficiency and enable parallel execution of tasks. The analysis involved using Kernel Density Estimation (KDE) plots to understand the imputation and scaling requirements of the columns.

For numeric transformers such as 'budget', 'director_power', and 'star_power', an Iterative Imputer was chosen. This decision was based on the recognition that these features depend on other movie-related information, like production company, genre, country of origin, release year, and duration. The Iterative Imputer was deemed appropriate for handling the interactive and dependent relationships between these columns, preventing the use of a simple imputer with mean or median values, which would be illogical given the uniqueness and independence of these features from movie to movie.

Additionally, it was observed that the distribution of 'budget', 'director_power', and 'star_power' was skewed, making Min-Max scaling suitable. Min-max scaling ensures that the values are transformed to a specific range, addressing the impact of skewed distributions.

For the columns 'release_month' and 'release_year', which exhibited distributions toward the mean, Standard Scaler was applied. Standard Scaler standardizes the features by removing the mean and scaling to unit variance, making it suitable for columns with distributions around the mean.

Addressing null values in categorical columns, specifically 'certificate,' and "Not Rated" was used to fill in missing values, creating a new unique column value. One-hot encoding was then applied to this column to handle the lack of ordinality among values and the relatively small number of unique values, ensuring minimal impact on computations.

In summary, the pipeline incorporates an Iterative Imputer for numeric transformers, Min-Max scaling for skewed distributions, a Standard Scaler for distributions around the mean, and appropriate handling of null values and categorical features through the use of unique values and one-hot encoding.

Modeling

Model Selection

For model selection, we used Cross-Validation to get the model best fit for our use case. We also used StratiedKfold to handle the class imbalances present in our data. Below are the observations we gathered:

Algorithm Name	Accuracy	F1 Macro Score	F1 Weighted Score
LogisticRegression	0.88	0.14	0.84
RandomForestClassifier	0.993 (best)	0.297 (best)	0.992 (best)
KNeighbour Classifier	0.989	0.268	0.988
Gradient Boosting Classifier	0.991	0.245	0.990

Discussions:

Logistics Regression:

Logistic Regression is a linear model commonly used for binary classification tasks. The lower F1 Macro Score suggests that the model may struggle with class imbalances or cannot capture complex relationships within the data. The high accuracy may be misleading, as it might be biased towards the majority class.

Random Forest Classifier:

Random Forest is an ensemble method known for its ability to handle complex relationships and outliers. The high accuracy and F1 Weighted Score indicate strong overall performance. However, the relatively low F1 Macro Score suggests that the model might struggle with certain minority classes or exhibit imbalanced class performance.

K Neighbors Classifier:

The K Neighbors Classifier relies on proximity-based decision-making. The high accuracy and F1 Weighted Score indicate good overall performance, but the lower F1 Macro Score suggests challenges with minority classes or imbalanced data. The model might struggle when classifying less prevalent categories.

Gradient Boosting Classifier:

Gradient Boosting is an ensemble method that builds trees sequentially, emphasizing misclassified instances. The high accuracy and F1 Weighted Score indicate strong overall performance. The lower F1 Macro Score suggests potential challenges in handling imbalanced classes or capturing nuanced relationships.

Possible Explanations for F1 Macro Score:

The F1 Macro Score is sensitive to the performance of minority classes. If the dataset is imbalanced, the models may struggle to generalize well to the minority classes, resulting in lower F1 Macro Scores.

It's possible that the features used for classification do not sufficiently capture the distinctions between different classes, leading to lower macro-level F1 scores.

In conclusion, while high accuracy and F1 Weighted Scores suggest overall good performance, the lower F1 Macro Scores indicate potential challenges with minority classes or class imbalances.

However, we found Random Forest to be the best fit and so we moved towards finding the best parameters for our already best model.

Hyperparameter Tuning


We then proceeded to Grid Search the best parameters for the Random Forest. We had the following parameters:

```
param_grid = {
    'classifier__n_estimators': [50, 100, 200],
    'classifier__max_depth': [None, 10, 20],
    'classifier__min_samples_split': [2, 5, 10],
    'classifier__min_samples_leaf': [1, 2, 4],
}
```

And observed the following results:

```
Cross-Validation Results:
[0.99352097 0.99353501 0.99355033 0.99381205 0.99384652 0.9938746
 0.99398695 0.99394226 0.99399716 0.9939678 0.99396907 0.99398184
 0.99392184 0.99398056 0.99398439 0.99399205 0.99399971 0.99402397
 0.99395758 0.99397801 0.99395631 0.99396014 0.99396907 0.9939729
 0.9939678 0.9939512 0.99398056 0.99366523 0.99363714 0.99363076
 0.99357714 0.99364736 0.99368438 0.99372779 0.99367672 0.99364353
 0.99361289 0.99358097 0.99356437 0.99356948 0.99360523 0.99359629
 0.9935631 0.99356437 0.99358863 0.99345969 0.99353246 0.99348905
 0.99348777 0.99357586 0.99350692 0.99360012 0.99353246 0.99352735
 0.99380183 0.993798 0.99386056 0.99397035 0.99396907 0.99399333
 0.99400227 0.99400482 0.99401376 0.99397673 0.99397035 0.9940112
 0.99398695 0.99399205 0.99401631 0.99399078 0.99401631 0.99399078
 0.99398439 0.99399588 0.99398184 0.99394737 0.9939678 0.99398184
 0.99398184 0.99395886 0.99398056]
```

Our best params were:



max_depth: None (indicating that nodes are expanded until they contain less than min_samples_split samples)

min_samples_leaf: 2 (the minimum number of samples required to be at a leaf node)

min_samples_split: 10 (the minimum number of samples required to split an internal node)

n_estimators: 200 (the number of trees in the forest)

The best CV score was 0.994 which indicated a high level of performance on the validation sets.

Interpretation:

The consistently high cross-validation scores across different folds suggest that the Random Forest model is robust and performs well on various subsets of the training data.

The chosen hyperparameters, as indicated by the "Best Parameter Combination," seem to be effective for the given task, with the model achieving a very high overall cross-validated score of 0.994.

ANN

We also tried defining, compiling, and training a neural network model using the Keras library for a classification task. The classification task involves predicting one of nine classes. For which a sequential neural network model is defined with several dense layers and a softmax output layer was defined. The model was compiled using the Adam optimizer, categorical cross-entropy loss, and additional metrics such as accuracy and TensorFlow's recall metric. The training involved 100 epochs with a batch size of 204800.

Results and Discussion

We used classification reports as an evaluation of the performance of two different models—Artificial Neural Network (ANN) and Random Forest—on a multiclass classification task.

Artificial Neural Network (ANN)

Overall Performance

Accuracy: 96%

Weighted Average F1-Score: 98%

Macro Average F1-Score: 27%

Class-Specific Metrics

Class 0 ("0"):

Low precision (2%) and recall (16%).

Low F1-score (3%).

Class 1 ("1"):

Precision (8%) and recall (54%) are relatively low.

F1-score (15%) is also modest.

Class 2 ("2"):

Precision, recall, and F1-score are all very low (0%).

Class 3 ("3"):

High precision (100%) and recall (97%).

High F1-score (98%)



Class 4 ("4"):

Precision (7%) and recall (24%) are both low.

F1-score (11%) is relatively low.

Class 5 ("5"):

High precision (100%) and recall (97%).

High F1-score (98%).

Class 6 ("6"):

Precision (5%) and recall (26%) are relatively low.

F1-score (9%) is modest.

Class 7 ("7"):

Low precision (1%) and recall (3%).

Very low F1-score (1%).


Class 8 ("8"):

Precision (4%) and recall (19%) are both low.

F1-score (6%) is relatively low.

Summary

The ANN model demonstrates high overall accuracy and weighted F1-score, primarily driven by excellent performance in the majority class (Class 3). However, the model



struggles with minority classes (Classes 0, 2, 4, 6, 7, 8) as evidenced by low precision, recall, and F1 scores for these classes.

Random Forest (with Best Parameters)

Overall Performance

Accuracy: 99%

Weighted Average F1-Score: 99%

Macro Average F1-Score: 26%

Class-Specific Metrics

Class 0 ("all time blockbuster"):

Precision, recall, and F1-score are all very low (0%).

Class 1 ("blockbuster"):

Precision (43%) and recall (24%) are moderate.

F1-score (30%) is relatively higher compared to ANN.

Class 2 ("super hit"):

Precision, recall, and F1-score are all very low (0%).

Class 3 ("hit"):

High precision (100%) and recall (100%).



High F1-score (100%).

Class 4 ("above average"):

Precision, recall, and F1-score are all very low (0%).

Class 5 ("average"):

High precision (98%) and recall (99%).

High F1-score (99%).

Class 6 ("below avg"):

Precision (50%) is relatively higher.

Recall (1%) and F1-score (3%) are low.

Class 7 ("flop"):

Precision, recall, and F1-score are all very low (0%).


Class 8 ("disaster"):

Precision, recall, and F1-score are all very low (0%).

Summary

The Random Forest model outperforms the ANN across most metrics. It achieves higher precision, recall, and F1 scores for several classes, resulting in excellent overall accuracy and weighted F1-score. However, like the ANN, it struggles with minority classes (Classes 0, 2, 4, 6, 7, 8).

Comparison of Model



The observed performance differences between the Artificial Neural Network (ANN) and Random Forest models can be attributed to various factors, including the inherent characteristics of the models, the nature of the dataset, and the specific challenges posed by the classification task. Here are potential reasons for the observed results:

Artificial Neural Network (ANN):

Complexity and Non-Linearity:

ANNs, especially deep architectures, are highly flexible and can capture intricate relationships within the data. However, this flexibility may lead to overfitting, particularly when dealing with imbalanced classes or noisy features.

Sensitivity to Initialization and Hyperparameters:

ANNs are sensitive to the choice of initial weights and hyperparameters. If not properly tuned, the model may struggle to generalize well to minority classes, leading to lower precision, recall, and F1 scores.

Class Imbalance:

The imbalanced distribution of classes in the dataset can adversely affect the training of ANNs. The model might prioritize accuracy by focusing on the majority class, neglecting the minority classes.

Limited Representational Capacity:

The chosen architecture and size of the ANN might not have sufficient representational capacity to effectively learn complex patterns in minority classes, resulting in poor performance.

Random Forest:

Ensemble Advantage:

Random Forests are ensemble methods that aggregate predictions from multiple decision trees. This ensemble approach often provides robustness against overfitting and tends to generalize well to different classes.

Handling Imbalanced Data:

Random Forests inherently handle imbalanced datasets by considering different subsets of features and samples in each tree. This can mitigate the impact of imbalanced classes on the model's performance.

Class Weighting:


The use of class weights, particularly in the Random Forest model, helps address class imbalance by assigning higher weights to minority classes during training. This allows the model to pay more attention to under-represented classes.

Tree-Based Structure:

The tree-based structure of Random Forests allows them to capture non-linear relationships in the data effectively. Each decision tree contributes to the overall decision, helping the model make nuanced predictions.

Recommendations:

Hyperparameter Tuning for ANN:



Fine-tuning the architecture and hyperparameters of the ANN to optimize its performance, especially considering the class imbalance.

Data Augmentation or Resampling:

Exploring techniques such as data augmentation or resampling to balance class distribution, aiding both models in learning patterns from minority classes.

Feature Importance Analysis:

Conducting a feature importance analysis to identify influential features. This can guide feature engineering efforts and improve the models' understanding of the data.

Ensemble Strategies:

Considering combining predictions from both models using ensemble strategies. This can leverage the strengths of each model and potentially enhance overall performance.

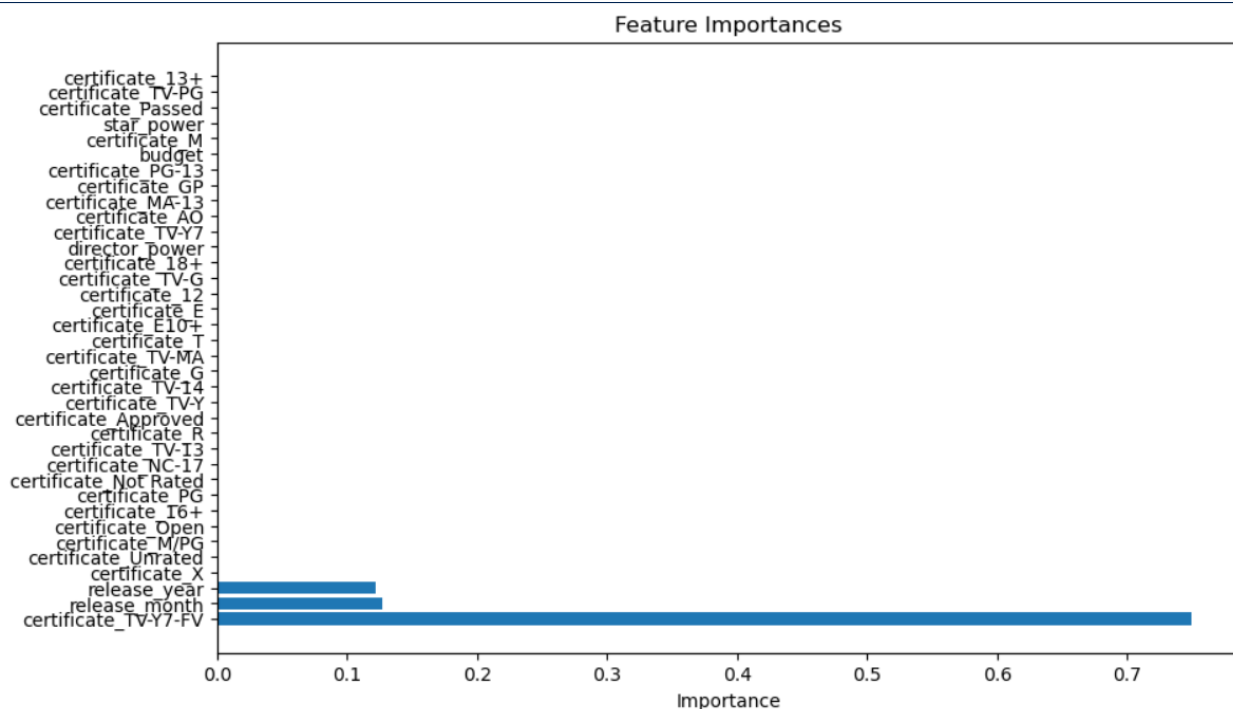
Threshold Adjustment:

Experiment with adjusting classification thresholds to balance precision and recall based on the specific goals of the classification task.

Cross-Validation and Robust Evaluation:

Ensure that model evaluation is robust through techniques like cross-validation to provide a more comprehensive assessment of performance.

By addressing these considerations and iteratively refining the models, it's possible to enhance their capabilities and achieve better performance, particularly in minority classes.





NLP for Movie Plot Analysis:

Implement Natural Language Processing (NLP) techniques to analyze movie plots. This could involve sentiment analysis, topic modeling, or even building a text-based model to extract valuable information from plot summaries. Insights from NLP could contribute to a better understanding of audience preferences and predict movie success.

Data Collection for Rare Classes:

Collect additional data, specifically targeting rare classes in your classification task. This could involve actively seeking out and including examples of movies that fall into rare classes. Increased representation of rare classes in the dataset will contribute to more robust model training and better generalization.


Temporal Analysis for Release Month:

Explore temporal analysis specifically related to release months. This could involve identifying patterns or trends in movie success based on the time of year. Understanding seasonal effects on audience preferences can contribute to more informed decision-making in the film industry.

Collaboration with Industry Experts:

Collaborate with industry experts, such as film critics, producers, or marketing professionals, to gather insights into the factors influencing movie success. This collaboration can provide valuable domain knowledge that can guide feature engineering and model improvement.

User Feedback and Iterative Improvement:



Implement mechanisms for collecting user feedback on movie recommendations or predictions. This feedback loop can be used for iterative model improvement, ensuring that the system continuously adapts to changing user preferences and industry dynamics.

Explainability for Recommendations:

Incorporate explainability into the recommendation system. Users are more likely to trust and engage with a system that provides transparent explanations for its recommendations. Techniques such as SHAP values or attention mechanisms can be useful for this purpose.

Continuous Model Monitoring:

Establish a system for continuous monitoring of model performance, especially as new data becomes available. Periodically retrain the model to ensure it remains relevant and effective in capturing evolving patterns in the movie industry.


Ensemble Modeling for Recommendations:

Consider building ensemble models for recommendation by combining predictions from multiple models. This can enhance the diversity and accuracy of recommendations, especially when dealing with different types of features (numerical, categorical, text).

Evaluation Metrics for Recommendation System:

Define and use appropriate evaluation metrics for the recommendation system. Metrics such as precision, recall, and Mean Average Precision (MAP) are common for recommendation systems. Customize the metrics based on the specific goals and characteristics of your application.

User Personalization:



Extend the recommendation system to incorporate user personalization. This could involve building user profiles based on historical preferences and adapting recommendations accordingly.

References

Dataset link:

IMDb Movies Dataset

<https://www.kaggle.com/datasets/rajugc/imdb-movies-dataset-based-on-genre>

TMDB Dataset

<https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset>