Movie Revenue Prediction and Recommender Model

Sophia Shan, Yuheng Lu, and Yuxiang Huang
New York University, js12623, yl7789, yh5047@nyu.edu

***Abstract* - The film industry, with its vast and dynamic market, presents a significant challenge in predicting the profitability of movies. The advent of advanced technology and data analytics has led to an explosion of data relevant to the industry's performance. Our study aims to forecast the financial success of films by employing machine learning models. We assembled a dataset with diverse movie attributes such as basic film information, IMDB scores, and Google Trends data from the release period. The dataset was used to train various machine learning models after rigorous preprocessing to ensure data quality. In addition to our predictive models, we developed a separate system using the Alternating Least Squares (ALS) recommender system, specifically designed to harness large-scale user-item interaction data for generating personalized film recommendations. The results from these models demonstrate a promising accuracy rate of 80% in predicting movie profitability. The integration of the ALS model provides a distinct avenue for exploring consumer preferences and enhancing our understanding of market dynamics without directly influencing the predictive accuracy of the financial success models. This research provides a framework for forecasting film revenues and offers valuable insights to stakeholders in the film industry, potentially informing their strategic decisions in this competitive market.**

Keywords – movies, machine learning, revenue prediction, classification, recommender system.

Ⅰ. Introduction

Predicting how well a movie will do at the box office is like cracking a challenging puzzle, mixing creativity with business savvy. Scholars and movie pros alike have been fascinated by this challenge. Now, with big data and machine learning, we are getting better at it. This model helps filmmakers, investors, and distributors get a clearer picture of what to expect.

But it's still a tricky game. A movie's success depends on many factors: the story, the cast, how it's marketed, and more. This study dives into this complexity using a powerful machine-learning approach. We aim to build several models that can accurately predict if a movie is profitable, giving the film industry a valuable tool for more intelligent decisions. Additionally, we have incorporated the Alternating Least Squares (ALS) recommender system, which operates independently to provide personalized movie recommendations.

In film industry analysis, predicting a movie's box office success is often treated as a regression problem, aiming to estimate the revenue a film will generate. However, it can be reframed as a classification problem to streamline this process, simplifying the prediction task. By categorizing movies based on whether their revenue exceeds their budget, a binary classification approach emerges: profitable or not. This simplification facilitates straightforward predictions regarding a movie's potential profitability, offering valuable insights for decision-making within the industry. The addition of the ALS model aims to explore a different facet of audience engagement, enriching our analytical framework without altering the core objectives of our financial success predictions.

To achieve this, we present several multifaceted predictive models considering the complex interdependencies contributing to a movie's financial success.

II. Methodology

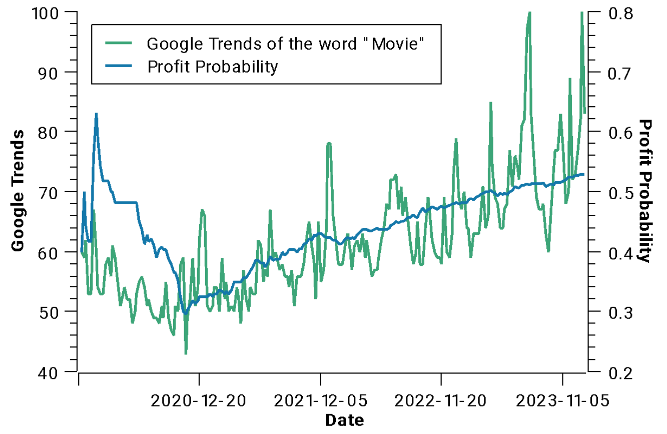
The project was completed in four phases.

- Data Extraction
- Data Processing
- Model Building
- Hyperparameter Tuning

A. Data Extraction

A significant part of the dataset required for the project was extracted from the global TMDB dataset using its APIs [1]. Following this, the OMDB API was used to extract the personal information of each movie's actors in the dataset. On top of that, the number of Google searches for the keyword "Movie" is also used as an indicator. The final dataset has features such as title, Budget, release date, revenue, runtime, average voting, popularity, genres, production companies, actors, and Google Trends.

To clarify, 1) Average Voting is the average of all the TMDB user ratings. 2) Popularity is a score calculated by TMDB based on unique page visitor views, number of ratings, number of times marked as favorites, and number of watched list additions on the TMDB website. 3) Google Trends is a website by Google that analyzes the popularity of top search queries in Google Search across various regions and languages. The website uses graphs to compare the search volume of different queries over time. After obtaining the weekly trend of the keyword "Movie" for each movie in the dataset, find the corresponding weekly hotness according to its release date and record it as a new column of features in the dataset.



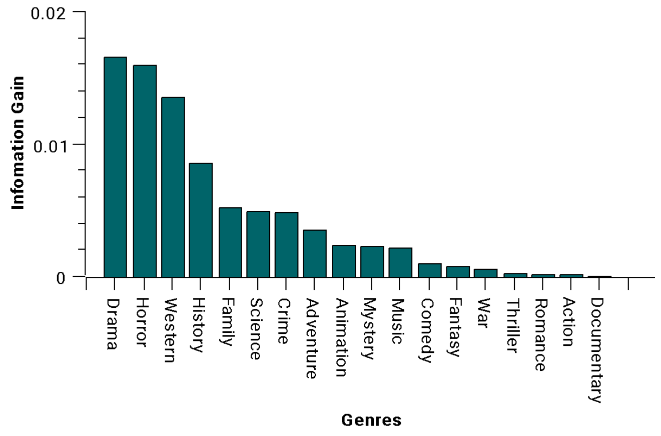
In the chart above, Profit Probability is defined as the cumulative number of profitable movies divided by the total number of movies since 2020. We can see that the probability of a profitable film positively correlates with the Google Trends of the word “Movie.”

We further performed a Granger causality test on Google trend and profitability. We achieved a significance score of 0.0022, thus rejecting the null hypothesis and concluding that Google trend does Granger cause profitability prediction.

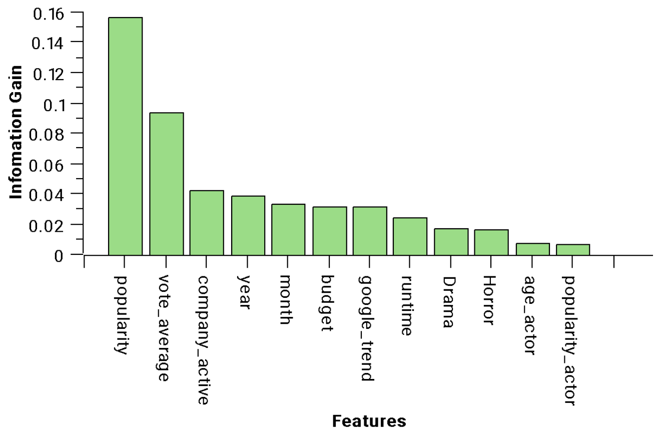
Finally, the dataset includes all movies released from January 2020 to December 2023. By removing outliers such as movies with zero revenue or a budget of less than $100,000, we end up with 425 movies in our final dataset.

B. *Data Processing*

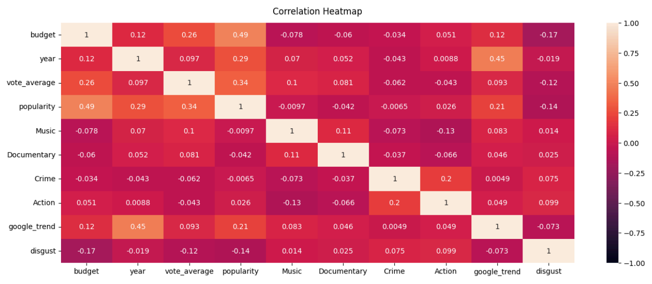
1) Use the “budget” and “revenue” columns to generate a new column for the feature “profitable,” which is a binomial value: 0 means loss, and 1 means profit.
2) Encoding. The feature “genres” is a categorical feature with 15 possible values. In this case, we perform One Hot Encoding and split the features into columns.
3) Feature Selection. After encoding, there are 15 different columns for genres, which will lead to the curse of dimensionality. So, we perform feature selection on these genres. By calculating their information gain, we notice that most of them can be removed. In this case, we only keep Drama and Horror features.



4) Actor data. The literal names of actors are not helpful during the prediction. We convert the feature into two columns: “average age” and “average popularity.” An actor's popularity is also a score calculated by the TMDB website.
5) Production companies data. We mark a movie as "produced by an active company" if its production company was involved in producing more than five movies in the last four years. This feature is experimental, and it is hoped that it will boost model accuracy.
6) Generate three different datasets. In addition to the previous steps, we copy the dataset three times to examine which one achieves the best performance. For the first copy, we simply perform Z-score normalization while conducting data binning before normalizing for the second copy. For the third copy, we perform PCA and keep only five principal components after normalization.
7) The following chart displays the information gains of all our features.



8) Inspired by Bollen et al.'s work predicting the stock market with mood [2], we utilized the overview section from the movie data. We performed sentiment analysis that categorizes the mood into joy, surprise, disgust, sadness, anger, and fear, with a score totaling 1 using pysentimiento, a pre-trained BERT model for social NLP tasks [3].
9) The correlation matrix table was constructed using the final selected features based on mutual information to study the linear correlation between features in the dataset before starting training and is shown as below. It was found that the highest correlation coefficient in the matrix was popularity and budget (0.49), followed by Google trend and year (0.45).



C. Model Building

The performance and predictability of various models on our dataset were evaluated using a systematic approach for model selection and validation. We divided the dataset into a training set and a test set using the train_test_split method from Scikit-learn. 70% of the data is used for training, and 30% is reserved for testing. This partitioning allows enough data to be available to train the models and evaluate their predictive performance on unseen data.

We selected different classifiers that represented different machine-learning algorithms. These were:

- K-Nearest Neighbors (KNN)
- Naive Bayes
- Decision Tree
- Neural Network (Multi-layer Perceptron classifier)
- Support Vector Classifier (SVC)
- Stochastic Gradient Descent (SGD) Classifier
- Random Forest
- Gradient Boosting
- Bagging Classifier with KNN

Each classifier was implemented using the corresponding Scikit-learn library class. The models were selected to include both single predictors and ensemble methods to evaluate several learning strategies on our dataset.

Each model's performance was assessed using 10–fold cross-validation to consistently evaluate various subsamples drawn from the training data. We calculated each model's metrics: Accuracy, Precision, Recall, F1 Score, F1 Macro and F1 Micro. These metrics were selected to provide an overview of all aspects that render models predictive capabilities, such as precision and recall, which are used to address any class imbalance issues where they arise.

In this regard, we take an all-encompassing approach to modeling training and assessment that informs the best-fitting choice of a model for movie revenue prediction. In this way, we intend, through stringent evaluation processes, to establish whether the selected model performs appropriately not only based on traditional metrics but also appeals more closely to practical demands and intricacies related to the prediction mission undergone.

The Random Forest model exhibited the highest overall accuracy (75.2%) among the evaluated classifiers. This superior performance, coupled with its substantial scores in precision (76.7%), recall (78.2%), and F1 score (77.3%), indicates its robustness in handling the predictive task at hand.

In contrast, the Naive Bayes classifier struggled significantly in recall despite its remarkably high precision. This discrepancy highlights the model's tendency to perform excellently in identifying true positives but at the cost of a higher false negative rate, undermining its overall effectiveness.

The Decision Tree model showed commendable balanced performance, with a test accuracy of 71.4%, coupled with consistent precision (73.3%), recall (75.0%), and F1 scores (73.7%). Its performance underscores the potential of decision trees to capture the complexity of the dataset through a hierarchical structure of decisions.

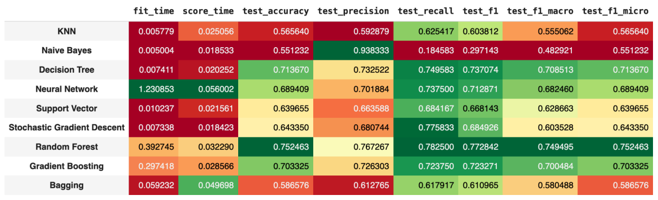
Furthermore, the Stochastic Gradient Descent classifier demonstrated a noteworthy recall score of 77.6%, the highest among all models, suggesting its ability to identify a higher ratio of positive instances correctly. However, its precision and overall accuracy (64.3%) lagged behind the ensemble models, indicating a higher false positive rate.

The Neural Network model delivered a solid F1 of 71.3% and showed competitive performance in precision and recall. The investment in computational resources for training neural networks is evidenced by their ability to learn complex, non-linear relationships within the data.

Support Vector Classifier and Gradient Boosting showed moderate performance. Still, they were surpassed by the ensemble methods, indicating the potential for further parameter tuning or the necessity for more complex models to capture the dataset's nuances.

While faster to train, the KNN and Bagging classifiers performed modestly compared to their more sophisticated counterparts. KNN's reliance on distance measures appeared less effective for this dataset, while Bagging showed a slight improvement but still needed to be more robust against more complex models.

This comprehensive study shows that classifiers varied drastically in their performance. It also tells us the importance of using the right metrics to gauge model performance. According to our analysis, ensemble methods like Random Forest perform well, supporting their strength and flexibility across various aspects of model assessment, such as accuracy, precision, recall, and F1 scores. Future work can investigate parameter optimization for each model, especially those with high potential for improvement, such as Neural Network and Stochastic Gradient Descent classifiers. Finally, this could be extended by exploring other ensemble techniques or new neural network architectures to improve predictive performance even more.



	fit_time	score_time	test_accuracy	test_precision	test_recall	test_f1	test_f1_macro	test_f1_micro
KNN	0.005779	0.025056	0.565640	0.592879	0.625417	0.603812	0.555062	0.565640
Naive Bayes	0.005004	0.018533	0.551232	0.938333	0.184583	0.297143	0.482921	0.551232
Decision Tree	0.007411	0.020252	0.713670	0.732522	0.749583	0.737074	0.708513	0.713670
Neural Network	1.230853	0.056002	0.689409	0.701884	0.737500	0.712871	0.682460	0.689409
Support Vector	0.010237	0.021561	0.639655	0.663588	0.684167	0.668143	0.628663	0.639655
Stochastic Gradient Descent	0.007338	0.018423	0.643350	0.680744	0.775833	0.684926	0.603528	0.643350
Random Forest	0.392745	0.032290	0.752463	0.767267	0.782500	0.772842	0.749495	0.752463
Gradient Boosting	0.297418	0.028566	0.703325	0.726303	0.723750	0.723271	0.700484	0.703325
Bagging	0.059232	0.049698	0.586576	0.612765	0.617917	0.610965	0.580488	0.586576

D. Hyperparameter Tuning

In our predictive model development, we focused significantly on hyperparameter tuning to enhance the performance of different classifiers. This is important in machine learning because it adjusts the parameters of models to make accurate predictions. We took advantage of GridSearchCV, a comprehensive feature provided by Scikit-learn, to automate the selection of the best hyperparameters for our models.

Three distinct types of classifiers were used in this optimization process: Support Vector Classifier (SVC), Multi-layer Perceptron (MLP) Classifier, and Random Forest Classifier. We picked them specifically since they each represent different underlying mechanisms, linear models, neural networks, and ensemble methods, respectively.

These classifiers were fed into a pipeline, allowing us to switch dynamically between models within the GridSearchCV framework, boosting our hyperparameter tuning process efficiency. In addition, using this pipeline approach means that each classifier is fitted with its own best combination of hyperparameters, as identified through grid search.

The optimal values for the F1 score after performing a grid search on our training data, which is an equilibrium metric that considers both recall and precision rates, led us to the path where the most suitable settings would be found among all available options. The optimal parameters were a Random Forest Classifier with max depth of 15, max features of 7, min samples leaf of 3, min samples split of 5, and n estimators of 50, achieving an F1 score of 80.9%.

III. The Recommender System

We selected the Alternating Least Squares (ALS) model for the recommender model portion of our project. ALS is a collaborative filtering technique widely used in recommendation systems. The core premise of collaborative filtering is to predict a user's interests based on the preferences of many users. ALS explicitly addresses the common issues of scalability and sparsity of user-item interactions in recommendation systems [4]. We implemented the recommender system using Apache Spark's MLlib in PySpark. PySpark's MLlib is designed for scalability across multiple nodes, making it an ideal choice for handling large datasets commonly found in recommendation systems. The ALS model in PySpark enables us to leverage this scalability and efficiently process large volumes of user-item interactions.

How ALS Works

The ALS algorithm decomposes the original user-item interaction matrix into two lower-dimensional matrices: one representing latent factors associated with users and the other with items. These factor matrices approximate the original matrix, aiming to fill in the missing entries (unrated items).

1. **Initialization**: Randomly initialize the user and item matrices.
2. **Alternation**:
   a. **Fix the item matrix**: Solve for the user matrix by minimizing the squared error of observed ratings versus predicted ratings, regularized to avoid overfitting.
   b. **Fix the user matrix**: Similarly, solve for the item matrix.
3. **Convergence**: Repeat the alternation until the algorithm converges

Using ALS for Recommendations

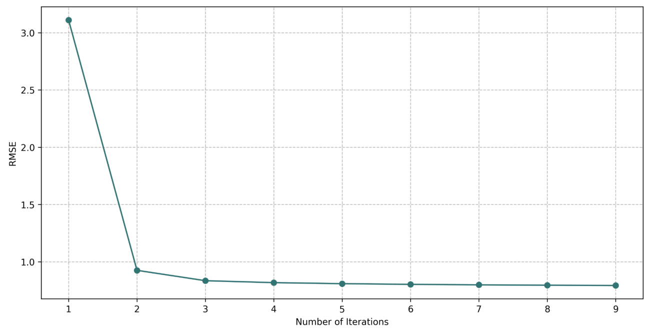
To use ALS for generating recommendations:

- **Training**: Load the user-item ratings into a Spark DataFrame, input the user-item rating matrix to the ALS model, and train it using historical data.
- **Prediction**: Once the model is trained, predict the missing entries in the matrix corresponding to the ratings a user might give to items they haven't yet rated.
- **Recommendation**: For a new user, predict their ratings for all items in the database using the learned item matrix. Sort these predicted ratings and recommend the top items.

Hyperparameter Tuning and Model Evaluation

In our project, we fine-tuned the ALS model by adjusting the following hyperparameters:

- **maxIter**: Maximum number of iterations for the algorithm to run. Our experiments showed that the model starts to converge after about three iterations.



- **rank**: The number of latent factors in the user and item matrices. We used grid search to experiment with different values and found that a rank of 20 provided the best performance.

- **regParam**: The regularization parameter helps prevent overfitting by penalizing large weights. A regularization parameter of 0.05 was optimal according to our grid search results.

We used the Root Mean Square Error (RMSE) to evaluate the model's performance, which measures the average magnitude of the prediction errors. A lower RMSE indicates a better fit of the model to the data. The best RMSE score achieved in our testing was 0.7907, indicating a relatively accurate model in predicting user ratings.

Addressing Common Challenges: Cold Start and Popularity Bias

ALS helps overcome two significant challenges in traditional recommender systems: the cold start problem and popularity bias.

- **Cold Start Problem**: Traditional systems need help with new users or items with few interactions. ALS mitigates this by utilizing user and item features to generate predictions even with minimal interaction data, thus providing reasonable recommendations from the start.
- **Popularity Bias**: Systems often favor popular items, overshadowing niche choices. ALS can adjust by incorporating user and item biases into the model, allowing it to recognize and recommend less common items based on individual preferences.

Overall, the ALS model has proven to be a robust and effective method for making personalized recommendations for our project. It is driven by its ability to handle large datasets and provide insights into user preferences and item characteristics.

IV. Conclusion

This paper has presented a comprehensive analysis of the factors influencing the profitability of movies and developed a robust predictive model using machine learning techniques. With its multifaceted nature, the film industry poses a significant challenge regarding financial forecasting. Our study has endeavored to address this challenge by leveraging a dataset encompassing many movie characteristics, including basic information, TMDB scores, and Google Trends data. Moreover, including the Alternating Least Squares (ALS) recommender system has enabled us to tap into a different dimension of data analysis by providing personalized recommendations based on user-item interactions, thus broadening the scope of our research.

Through various machine learning algorithms, we have successfully constructed a predictive model capable of forecasting movie profitability with an F1 score of 80.9%. This metric represents a substantial contribution to the field, offering more predictability in an industry traditionally characterized by uncertainty. The ALS model, operating in parallel, complements these efforts by enhancing user engagement through personalized content, illustrating the dual benefits of our analytical approach.

Our findings underscore the importance of a data-driven approach in the film industry, where integrating various data sources and machine learning algorithms can provide stakeholders with a strategic advantage. The predictive model developed in this study can serve as a decision-making tool for investors, producers, and distributors, enabling them to make more informed choices regarding film production and distribution.

However, it is essential to acknowledge the limitations of this study. The predictive model is based on historical data and may not fully account for unforeseen future changes in audience preferences or market conditions. Additionally, our model does not consider the qualitative aspects of filmmaking, such as the creative process and artistic merit, which can also significantly impact a movie's success. The ALS model, while beneficial, is similarly limited by its reliance on existing user data and might not capture the full spectrum of new viewer preferences.

In light of these considerations, future research could explore incorporating qualitative data and real-time audience feedback into the predictive model. Furthermore, the impact of emerging technologies, such as virtual reality and streaming platforms, on movie profitability could be investigated. These avenues of research have the potential to enhance further the accuracy and applicability of predictive models in the film industry.

References

[1] G. Velingkar, R. Varadarajan, S. Lanka and A. K. M, "Movie Box-Office Success Prediction Using Machine Learning," *2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T)*, Raipur, India, 2022, pp. 1-6, doi: 10.1109/ICPC2T53885.2022.9776798.

[2] Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of computational science*, *2*(1), 1-8.

[3] Pérez, J. M., Rajngewerc, M., Giudici, J. C., Furman, D. A., Luque, F., Alemany, L. A., & Martínez, M. V. (2021). pysentimiento: a python toolkit for opinion mining and social NLP tasks. *arXiv preprint arXiv:2106.09462*.

[4] Zhou, Y., Wilkinson, D., Schreiber, R., Pan, R. (2008). Large-Scale Parallel Collaborative Filtering for the Netflix Prize. In: Fleischer, R., Xu, J. (eds) Algorithmic Aspects in Information and Management. AAIM 2008. Lecture Notes in Computer Science, vol 5034. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-68880-8_32