Box Office Revenue Prediction

ABSTRACT

Predicting the gross of the movie based on various factors known at the start of filming

GROUP 3 - SHERLOCK

Adithya Selvaganapathy
A0186084X

Dinesh Kumar Agarwal Vijayakumar
A0186283W

Hemanand Moorthy
A0186104L

Mookkandi Sathan Karthikeyan
A0186448N

Spatika Narayanan
A0088416X

Swetha Narayanan
A0074604J

BT5153 :

TOPICS IN BUSINESS ANALYTICS

Table of Contents

1. INTRODUCTION

The global box office revenue is expected to reach $50 Billion by 2020. A movie needs to make three times its production budget to be deemed profitable, as marketing and promotional costs are not accounted for in the production budget. Approximately 2577 movies are released every year, out of which only 10% of the movies are deemed to be profitable.

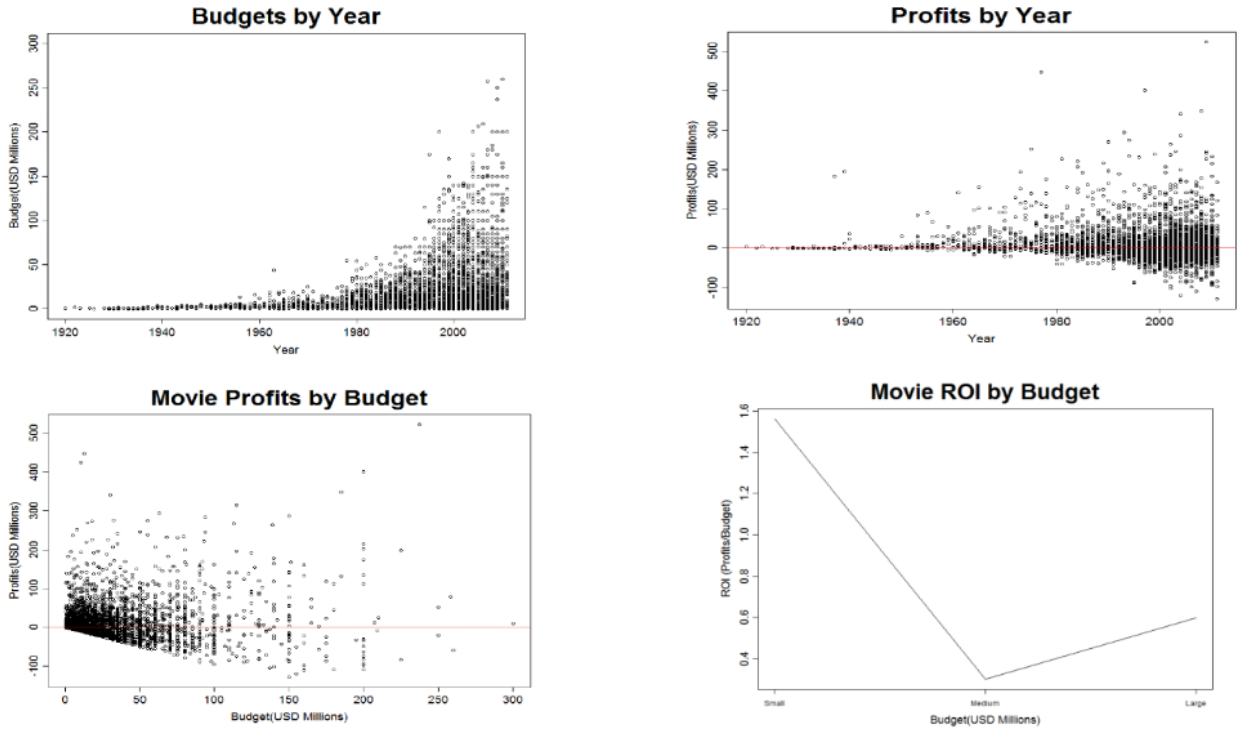


Fig 1.1 Graphs showing investments vs returns in movie making

As observed from the graphs above, even though the production costs are on the rise, the same cannot be said about the return of investment. This makes movie-making a risky business for the production houses.

The success of a movie primarily depends on two sets of inputs.

1. **Inputs known before the start of production**: Cast and crew, genre, storyline, release date, runtime.
2. **Inputs known only after movie release**: Success of promotional activities, user and critic reviews.

There are many other factors which can also be grouped under two of these categories. Since the success of the movie is dependent only on a finite set of indicators, machine learning models could be used to enable production houses to maximize the profits made.

Working on predicting the box office revenue using machine learning models is not new. For example, there had been Kaggle competitions which attempted to solve the same problem[1]. However, these approaches are limited as they include data from a singular source only. When we browse through the IMDb movie page (Appendix 1.1) we obtain information about the cast/crew and the genre of the movie, but this doesn't account for the popularity of the cast/crew, influence of reviews of a box office collection or the effectiveness of promotional activities.

Our approach is to not limit our model to a single source, instead scrape information regarding the movie from diverse sources to get a holistic view of the various factors influencing box office revenue. Our model is intended for usage by the production houses and consists of two components:

1. **Movie gross prediction:** Considering factors known before production and simulating the factors known after, arrive at a movie gross prediction.
2. **Finding similar movies:** Using the storyline of the movie, enable production houses to find similar movies to learn about how similar movies performed, how they were promoted, etc.

2. DATA COLLECTION

The four major sources used for extraction of data are MovieLens, IMDb, TMDb and YouTube, with each as the source for specific features, which affect the revenue of a movie. Appendix 2.1 has the list of inputs which we extract from these data sources.

2.1 MovieLens

MovieLens provides the master list of movies that it hosts along with the movie's respective IMDb id. IMDb id would be the primary key for the final dataset. Additionally, MovieLens also provides the title and genre of the movie.

2.2 IMDb – Internet Movie Database

Most data for the movie of interest is scraped from IMDb. Web scraping is done using 'BeautifulSoup'. The data that needs to be scraped from IMDb is categorized into:

JSON

Primary data about the movie such as Movie Name, Average Star Rating, No. of Text Reviews, Directors, Writers/Creators, Actors, Genres and Motion Picture Rating are available in JSON data format, making it the easiest to extract.

HTML
The content is embedded on the website with simple HTML tags. The extraction of a specific HTML content is done using string matching with wildcards. Additionally, the structural pattern followed in the website is leveraged to narrow down the search locations, thereby helping in optimizing the wildcard search time. Data such as Release Date, MetaScore, etc were extracted as HTML content.

Asynchronous Loaded Pages
Extraction of user reviews is complex as they are loaded onto the page asynchronously. The complexity inherent in the extraction of user reviews for a movie is a consequence of the huge number of reviews available for every movie. A single page in a website holds up to 20 user reviews. Further reviews are loaded on the page every time a user clicks the "Load More" feature, provided in the IMDb website. The scraping, however, employs redirection to a new website every time a 'load more' is clicked. Every redirection retrieves 20 review information from the IMDb database sorted in review popularity and is available to be extracted. Therefore, to obtain reviews that were posted until 10 days after the movie's release date requires us to parse through all the reviews until there are no more reviews.

Owing to the difficulty in extracting user review information, the original dataset that consisted of 17337 movies was pre-cleaned. 4531 movies that had complete information at every feature were selected to proceed towards review extraction.

2.3 TMDb – The Movie Database

TMDb was used to find the popularity of the cast and crew among the general audience through previous movies reach, social media activity etc. to calculate the popularity score for an actor. This information was extracted using the GET '/person/popularity' method in the TMDb API: https://developers.themoviedb.org/3/people/get-popular-people.

2.4 YouTube

We believed that YouTube acts as a good proxy to indicate audience interest in the movie (and success of promotional activities) prior to its release which will, in turn, be a good predictor for the performance of the movie.

API Data Extraction to extract relevant trailer link
We used the title of the movie to obtain the most relevant trailer for the movie. API keys generated for 'REST' call have a maximum quota per day, so we obtained as many as 60 API keys among 6 of us to generate data for the selected 4531 movies

API Data Extraction to extract stats
Using the trailer link, we performed another API call to obtain data on the number of views, likes and comments. The values were then normalized prior to feeding into model.

3. DATA VISUALIZATION

After collecting all the required data inputs, we visualized the data to understand and help with the model formulation.

3.1 Gross

Looking into Appendix 3.1, 3.2, 3.3 shows us that action and adventure movies make more revenue than musicals or documentaries. And more than 25% of the movies in our dataset are composed of data from five major production houses. Interestingly, Mandarin has the highest average gross per language which might be due to the fact that our dataset contains inputs only from extremely famous or internationally-renowned Mandarin movies.

3.2 Critics Rating

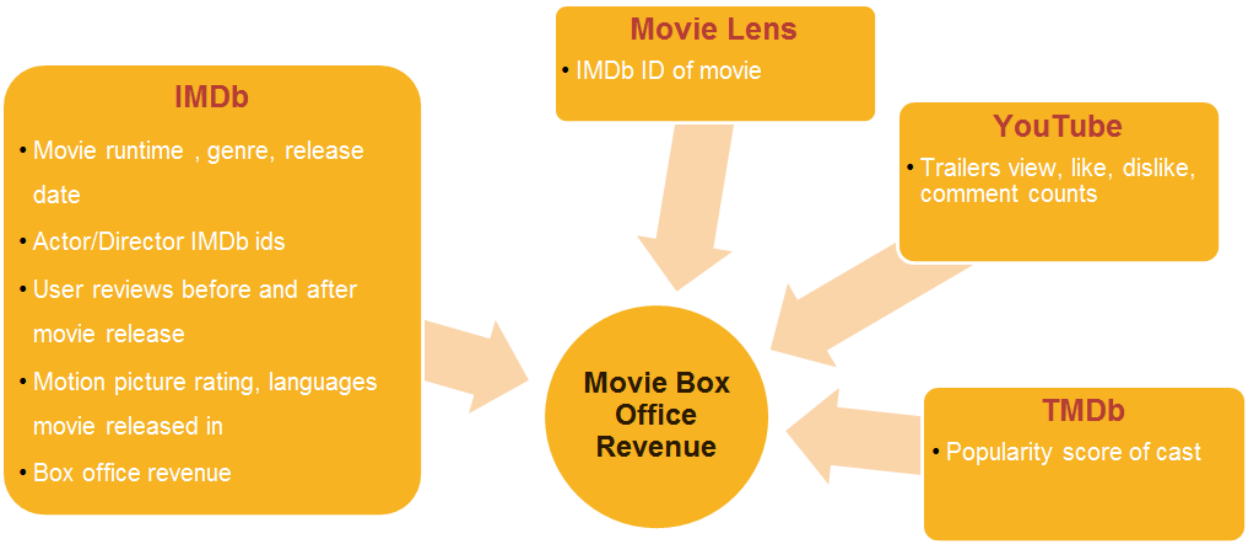
Looking into Appendix 3.4 shows us that though there are more movies released in the action genre than dramas, critics like, comment and talk more about movies in the drama genre than the rest.

3.3 Movies by Language and Country

Looking into Appendix 3.5, 3.6 shows that our dataset is skewed towards movies released in English (about 75 per cent of the data) and from the US (about 65 per cent of the data).

4. DATA MODEL

Our data model can be summarized as follows:



5. MACHINE LEARNING MODELS

5.1 Data Transformation and Feature Engineering

Once we aggregated the data from all our sources, we still had quite a few steps in the data cleaning and transformation process. At each step, we made sure to check for any induced NAs/NANs. We also engineered a few new features from the existing ones, and some of these were later found to be useful inputs to the model (Section 5.4). The steps are as below:

We dropped rows with NAs for categorical variables Language, Production House and Motion Picture Rating. This reduced our dataset from 4351 to 3909 observations. We had 42 rows with NAs in numerical variables (Number of User Reviews, Number of Critic Ratings) left which we dealt with using imputation as the first step in our modelling pipeline (Section 5.3).

Next, we used regex to extract 'number of languages' as another feature based on the list of languages a movie was shot in. Again, using regex, we parsed the duration string of the movie in the format "39min" or "1h30min" to integer duration (in minutes).

We also cut down the list of languages/countries/production houses to just the primary attribute in each case. Further, for these 3 columns, we re-classified them into top 5 labels + 'Others' avoid ending up with a large number of columns on performing one-hot encoding. For example, Production House would otherwise have ~2700 unique values, instead of just 6 (Universal Pictures, Columbia, Paramount Pictures, Warner Brothers, 20th Century Fox, and Others).

Using the release date, we extracted a binary feature to indicate whether the movie was released on a public holiday in the US or not. The hypothesis is that movies released at these times would be better crowd-pullers.

There were some movies which didn't have a rating put forth by the Motion Picture Association of America. We reclassified these according to MPAA standards [2] to fall under one of 5 categories: G, PG, PG-13, R, or NC-17.

The IMDb listings for the movie listed several genres for each movie, out of 23 possible unique genres. We parsed these genre lists using Sci-kit Learn's MultiLabelBinarizer to extract 23 one-hot genre columns. For example, the genre list for *The Avengers* was originally [Action, Adventure, Sci-Fi], so after this step, it would have a '1' under these 3 genres, but 0 for the rest.

Additionally, we used the text from the scraped IMDb reviews, to come up with 2 more features: average AFINN sentiment score using 'afinn' package, for reviews before release, and reviews after release.

5.1.1 Final Feature Set

With the categorical variables converted to their one-hot dummies, we ended up with 60 input features per observation. Appendix 5.1 lists the selected variables.

5.2 Transformations on Y Variable

Our dependent variable is cumulative gross worldwide revenue, a continuous variable. First, we had to parse the web-scraped strings to convert to float and convert all other currencies to USD. This involved some manual inspection as well for some unique cases where there was no currency mentioned, and the value wasn't in the default currency of USD. Finally, we converted the revenues to their inflation-adjusted values with respect to 2018 for more accurate comparison.

Next, since we wanted to predict the revenue bucket, not the continuous variable, we divided the revenue into categories by quintiles using pandas' qcut function. This also prevents the class-imbalance problem. So, we finally had 5 revenue buckets, ranging from very low to very high revenue, with around 781 observations each.

If we had divided categories by equal revenue brackets instead, it would've been easy to get a misleadingly high accuracy just by predicting the majority class, since an overwhelming majority of movies fall into the same range of mediocre revenue.

Final Revenue Ranges:

- **Very low:** 0 to 48,693.25 USD
- **Low:** 49145.84 to 389,168.30 USD
- **Medium:** 390,399.40 to 5,904,366 USD
- **High:** 5,917,917 to 67,486,060 USD
- **Very High:** 67,496,100 to 2,208,863,000 USD

5.3 Modelling Pipeline

We split our 3909 observations into train and test sets with 2931 and 977 observations respectively. For modelling, we had three steps in our pipeline: imputation for NAs in the numerical columns mentioned earlier, feature scaling or normalization, and finally, fitting the prediction model itself.

At each of these three steps, since there were various parameters we could try, we chose the final combinations for the best models using Randomized Search 5-fold cross-validation. The steps were chained using Sci-kit Learn's 'make_pipeline' method. Random seeds were set for non-deterministic operations for reproducibility of results.

For imputation (SimpleImputer), we could choose to either impute the NAs with the mean or the median. And for normalization (StandardScaler), we could control whether it would be performed with the mean (True/False) - i.e. data will be centered before scaling, and/or standard deviation (True/False) - i.e. data will be scaled to unit variance. This step was especially important for our k-Nearest Neighbors, Multilayer Perceptron (MLP) and Logistic Regression (L2 regularization) models, but we applied it to all models for consistency in comparison.

5.3.1 Models Fit

We compared the following models for multi-class classification, and for each one, the parameters we tuned are as below:

k-NN
We varied the number of neighbors: 50, 100, 150, 200, 250 - values below this were found to overfit the data in initial modelling experiments (i.e. train accuracy was much higher than test accuracy). Another parameter tuned was the algorithm used to find the nearest neighbors: either kd_tree, ball_tree, a brute force search or 'auto', which decides the most appropriate algorithm using the values passed. We also varied the weight function - either 'uniform' with all neighbors weighted equally, or 'distance', with neighbor closer to a point given more importance.

Logistic Regression
We varied the regularization parameter 'C', trying values 1, 0.75, 0.5. The best was found to be C = 1. We also experimented with various solvers: 'newton-cg', 'lbfgs', and 'sag'. While the other two led to convergence problems even on increasing the number of iterations, newton-cg was more stable.

Random Forest
We tuned the number of estimators (trees) used, trying values: 50, 100, 150, 200, 300, 500, and 600, the max tree depth (3, 5, 7, 9), the function to measure the quality of the split ('gini', 'entropy'), and the maximum number of features to consider at each split - either sqrt(n_features) or log2(n_features).

Multilayer Perceptron (Basic Feed-Forward Neural Network)
We varied the hidden layer sizes and/or number: (50,50,50), (50,100,50), (100), (200), (100, 200, 100), the activation function (tanh or ReLU), the optimization algorithm for backpropagation (Adam or SGD), and the L2 regularization penalty alpha (0.0001, 0.001, 0.05), the batch size (32, 64, 128, 256, 512), and whether to use a constant or adaptive learning rate.

Stacking
The level 0 models we used for the final ensemble were k-NN and random forest, the Level 1 meta-learner was a Logistic Regression model.

5.3.2 Results and Analysis

Table 5.1 shows a summary of our results and model comparison. Since our problem is multi-class classification and our classes are balanced, we chose accuracy and macro F1 on the test set as our comparison metrics. Macro F1 is more appropriate in this situation than micro F1 (summing individual true positives and false positives and then calculating the statistics) since there is no class-imbalance problem [3].

Our best-performing model was the random forest, with an accuracy of 55%. On further evaluation of this model, we noticed that there was better performance for 'high' and 'very high' revenue buckets, but the 3 closer-together revenue buckets (very low, low, medium) are harder to predict. The in-between one, 'low' is the toughest revenue bucket to correctly identify. (Refer Appendix 5.2). This is more easily visualized in the confusion matrix (Refer Appendix 5.3), where we see that most of the high and very high observations have been correctly labelled.

Model	Test Accuracy	Test Macro F1
Logistic Regression (newton-cg solver, C = 1)	0.5312	0.5287
k-NN (50 neighbors, kd_tree algorithm, uniform weight function)	0.4749	0.4782
Random Forest (600 estimators, max_depth 9, 'entropy' as a split criterion, 'log2' for max features)	0.5537	0.5448
Multilayer Perceptron (SGD with adaptive learning rate, hidden layers = (50, 100, 50), batch size 256, alpha 0.5, and tanh activation)	0.5240	0.5197
Ensemble methods (Stacking)	0.4227	0.3929

Table 5.1. Summary of Results

5.4 Key Insights

Refer Appendix - 5.4 for the feature importance plot

We found that post-release factors like the number of star ratings, number of user reviews and critic ratings which are conventionally good indicators of box office success (or at least the buzz surrounding the movie) figure prominently.

But we also saw that factors that can be determined before movie production starts: like the TMDb-based popularity score, the production house, genres, number of languages the movie will be released in, and duration have an impact on revenue as well. Within genres, action/adventure/family genres are also found to be significant, which makes sense intuitively too, due to their mass appeal.

Features that can be measured *before* release, but *after* the promotional activity has already started, such as the sentiment score of the pre-release reviews and YouTube statistics (view count, like count, comment count) are also found to be important.

5.5 Future Modelling Extensions

- K-means clustering on gross revenue to choose the gross category, instead of dividing revenue into buckets by the quintile. Or, we could have further granularity in the prediction by dividing into deciles (10 buckets).
- Imputation for categorical variables, instead of just removing the rows - for example, the strategy could be to replace by the most frequent category.
- We could also factor in the release location and the number of countries from the country list as additional engineered features. We could expand our search space for hyperparameter tuning to try a larger number of combinations for more extensively tuned models.

6. FINDING SIMILAR MOVIES

After predicting the revenue bucket in which a movie will fit in, we wanted to identify similar movies based on the storyline of the movie. The reasons for this are:

1) Based on the revenue made by the similar movies identified by the algorithm the production house can analyze the return on capital invested in the movie.
2) Based on the returns of successful similar movies, the production house can plan their marketing activities.
3) Cues can be taken from movies that failed to do well and identify the reasons why the movie failed to do well.

For identifying similar movies, content-based filtering was done where we compare the similarities of items based on the attributes of the item. In our case, the attributes of the movies were extracted. The reason for using content-based filtering is that since the model will be used before the movie release, other options like reviews will not be available for the filtering.

6.1 Methodology

For finding the similar movies the features we used were the plot of the movie, director, and top 3 actors from the movie. The first step in the process was to clean the text and remove the special characters and separators from the text.

The next step was feature extraction. Keywords were the features that were extracted from the plot of the story. For keyword extraction, we used NLTK package and RAKE (Rapid Automatic Keyword Extraction algorithm). RAKE is a domain-independent keyword extraction algorithm which tries to determine key phrases in a body of text by analyzing the frequency of word appearance and its co-occurrence with other words in the text. After keyword extraction from the plot, the director and actor names were appended to create a bag of words.

The final step was to create a vector which can be used to calculate cosine similarity between movies. We used Count Vectorizer instead of TF-IDF Vectorizer, as the latter removes the frequent words in the model which could affect the similarity between the movies. Using Count Vectorizer, a vector was created for each movie. Based on the count vector, a similarity matrix is constructed based on the cosine similarity score. The cosine similarity matrix was used to identify the top 10 movies which are similar to the movie which is given as input.

6.2 Web Framework for Visualization

After modelling, it is important to identify the visualization method presents the output to the end user. We chose to build a Python Flask application that creates a web page which can be hosted on a server. The web page takes movies as the input and would retrieve the top 10 similar movies from the cosine similarity matrix created in the previous step. Information of the top 10 movies such as release date, revenue, budget and plot of the movie is retrieved from the scraped data and is displayed along with the poster of the movie. JavaScript was used to dynamically display the output. (Refer Appendix 6.2 for screenshot)

7. TESTING THE MODEL

To test our model for real word movie revenue prediction, we tried to predict the revenue for the upcoming movie, *Avengers Endgame* for the following scenarios (which are known only post-release):

1. Highly positive reviews
2. Average reviews
3. Bad reviews

The results are presented as a screenshot (Appendix 7.1)

8. MODEL USAGE

1. Predicts the box office performance with a set of known input factors like actors, cast, crew, and genre
2. User reviews and the success of the promotional events cannot be predicted before the movie is made. So, the two features from the user reviews mentioned in Section 5 can be simulated by the producer to know how revenue varies. For example, the producer could fix the scores to their minimum possible values (highly negative user reviews) and repeat for maximum possible scores (highly positive user reviews), to generate the range of possible box office revenues. Similarly, the number of trailer views from YouTube trailers/teasers can be simulated.
3. Producers get to know movies with similar storylines. This would enable them to learn from the good/bad from those movies, their performance and they could budget accordingly

9. FURTHER IMPROVEMENTS

- Predicting the success/ failure of a movie is a qualitative problem rather than a quantitative problem.
- Even if the past set of directors, cast, and crew had given a great box office movie it is not necessary that the next movie with the same team would lead to similar performance
- Some factors like time of release, people's mindset during that time (example, calamities occurring during that time can hamper collection) and competition from other movies are some factors which we cannot address using our model.

10. CONCLUSION

This project is an attempt towards predicting the movie revenues using a combination of factors known as pre-release and simulating factors other factors (known post-release). Thus, this model can be used by the production houses to mitigate the risks associated with movie production.

REFERENCES

1. https://www.kaggle.com/tmdb/tmdb-movie-metadata/data
2. https://simple.m.wikipedia.org/wiki/Motion_Picture_Association_of_America_film_rating_system
3. https://medium.com/usf-msds/choosing-the-right-metric-for-evaluating-machine-learning-models-part-2-86d5649a5428

APPENDIX

1.1 An IMDb movie page

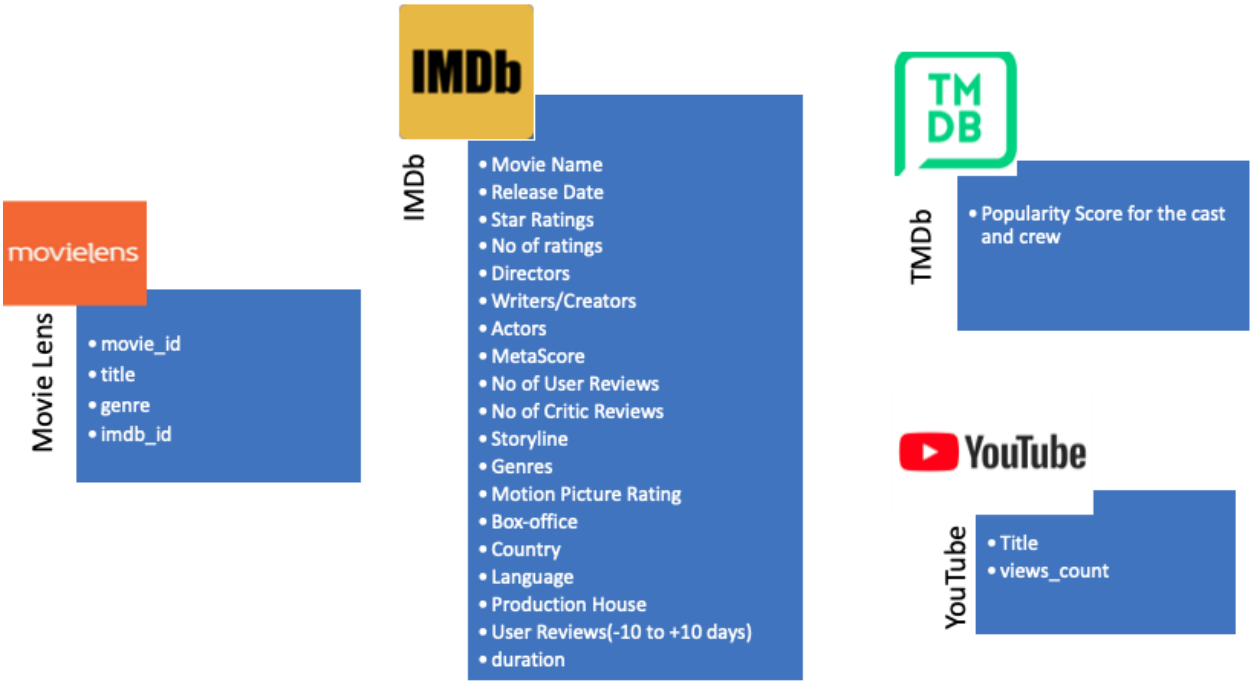
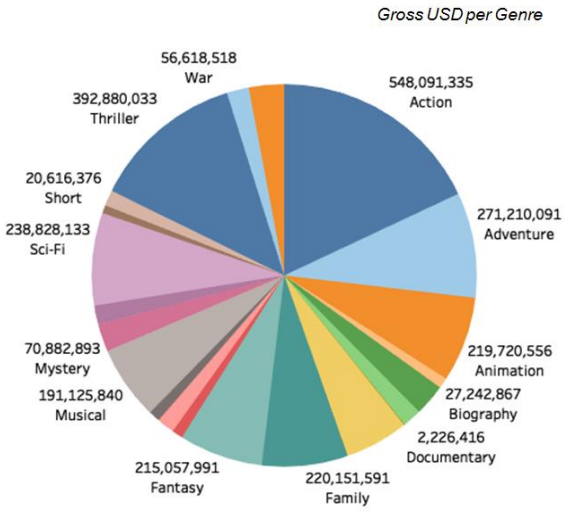
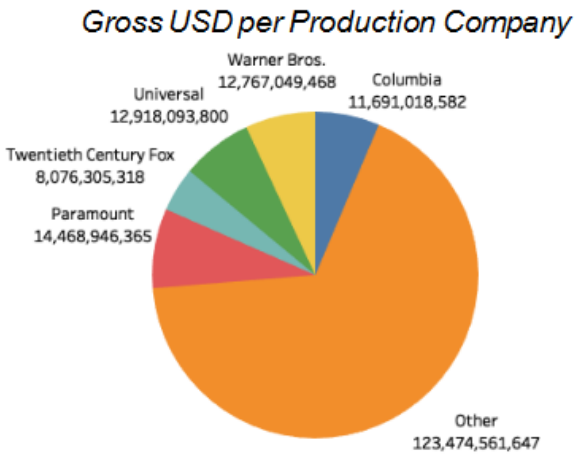


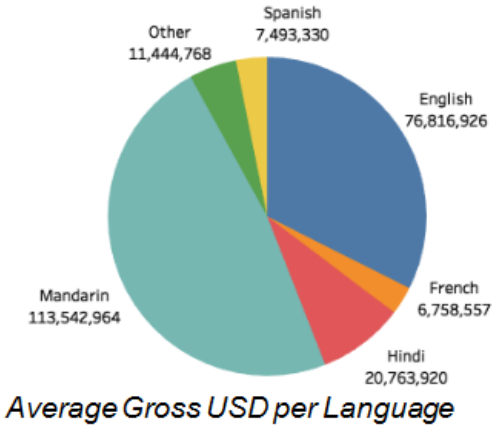
Fig: 2.1 Data collected



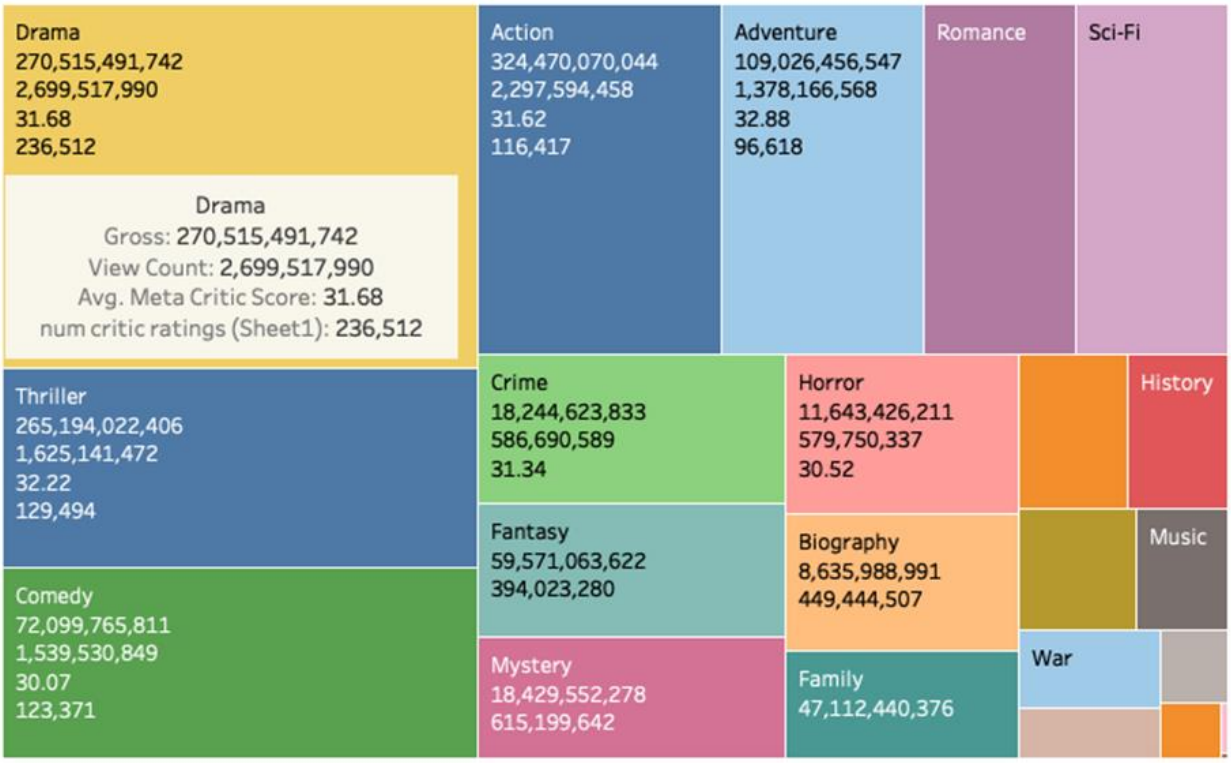
3.1 Gross per genre(in USD)



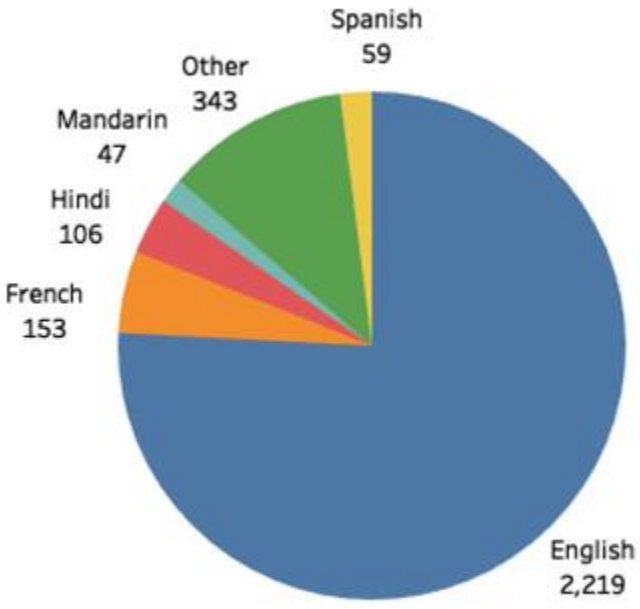
3.2 Gross per production company (in USD)



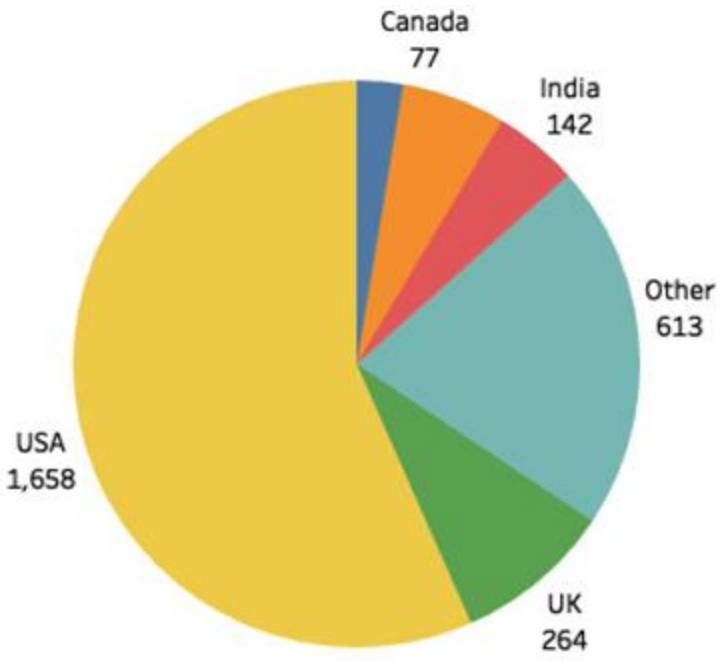
3.3 Average gross per language (in USD)



3.4 Critics info



3.5 Movies per language



3.6 Movies per country

	Numerical Features
1	Number of star ratings
2	Average star rating (out of 10)
3	Number of user (text) reviews
4	Number of critic ratings
5	Duration (minutes)
6	Holiday (binary)
7	TMDb Average Popularity Score
8	Number of languages
9	Text-based (IMDB user reviews): • Average AFINN score of pre-release reviews • Average AFINN score post-release reviews
10	YouTube-based: • View Count • Like Count • Dislike Count • Comment Count

	Categorical Features
1	**Primary Language:** English, French, Hindi, Mandarin, Spanish or Others
2	**Primary Country:** USA, UK, France, India, China, or Others
3	**Primary Production House:** Universal Pictures, Columbia Pictures Corporation, Paramount Pictures, Warner Brothers, Twentieth Century Fox, or Others
4	**Genres:** One or more of 23 unique genres
5	**MPAA Rating:** G, PG, PG-13, R, NC-17

5.1. List of numerical and categorical variables selected

	precision	recall	f1-score	support
very low	0.46	0.49	0.48	182
low	0.48	0.28	0.35	205
medium	0.51	0.47	0.49	198
high	0.79	0.82	0.81	187
very high	0.52	0.72	0.60	205
micro avg	0.55	0.55	0.55	977
macro avg	0.55	0.56	0.54	977
weighted avg	0.55	0.55	0.54	977

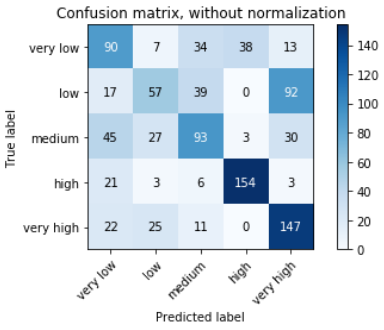


Fig 5.2, 5.3: Left - Classification Report; Right - Confusion Matrix

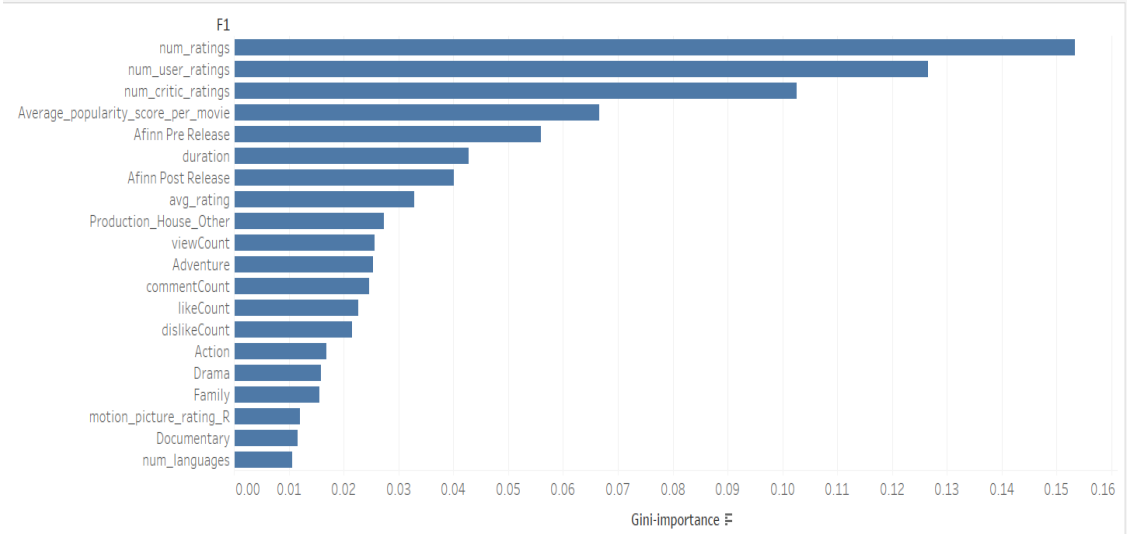
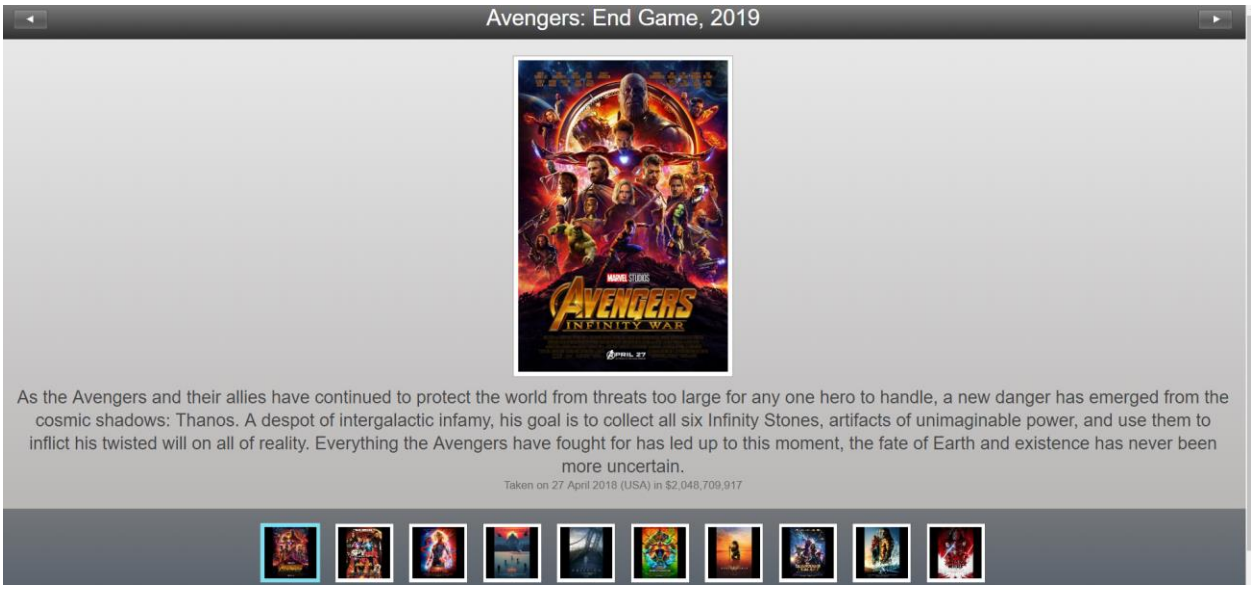
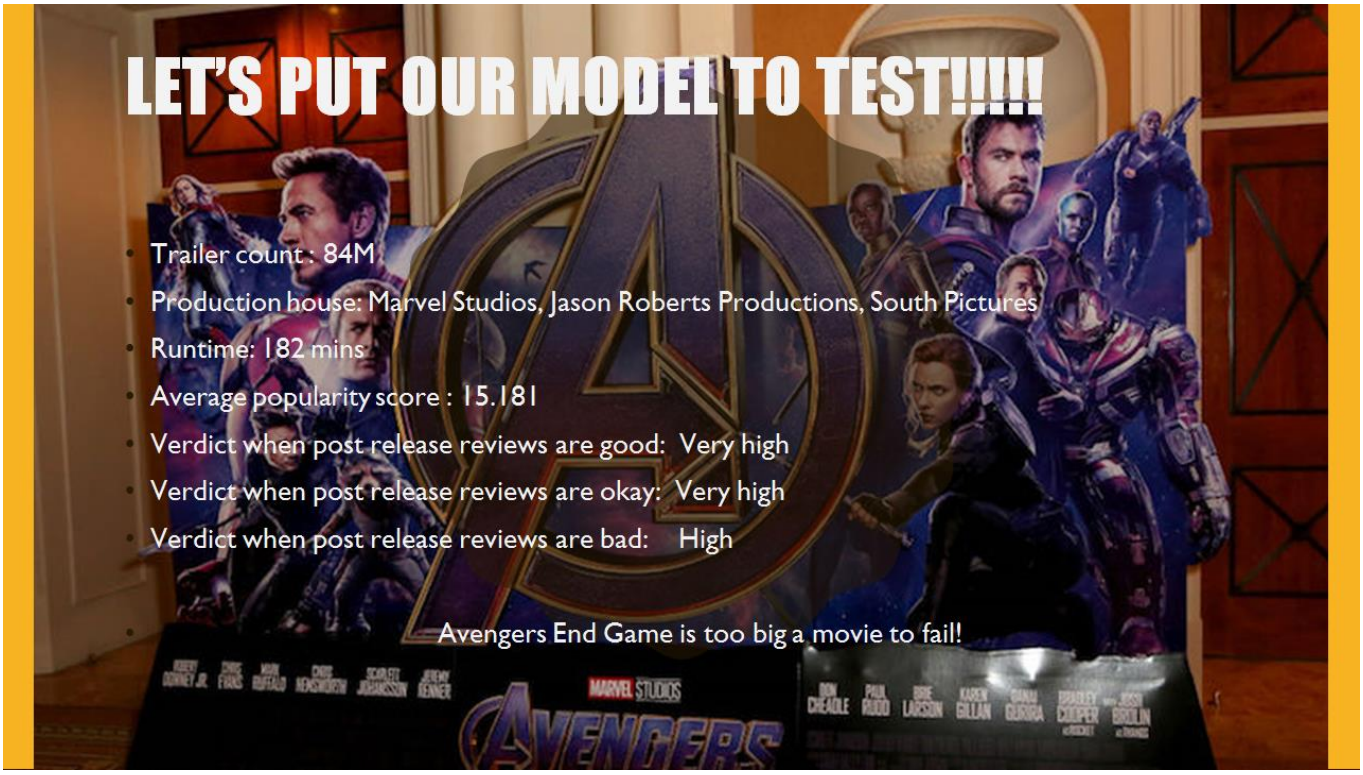


Fig 5.4: Feature Importance Plot from Random Forest Modelling



6.1 Recommended movies screenshot



7.1 Model Test