Predictive Model for Movie Rating

Final Report of DSC190-WI22

Angela Wang
University of California, San Diego
anw008@ucsd.edu

Xuzhe Zhi
University of California, San Diego
xzhi@ucsd.edu

Introduction

Rating is commonly used as an indicator of the quality of goods. In this era of information explosion and bloom of online shopping, people refer to rating a lot when selecting products. Similarly, ratings of a movie are related to its long-term reputation, popularity, and profit. In this project, we will be building a model to predict the rating of a movie based on its information.

Github: https://github.com/Angela-Wang111/DSC190_WI22_Final_Project_Movies.git (Refer to DSC190_final_project_cleaned.ipynb, FInalProject_additional_EDA.ipynb. Working demo: demo.ipynb)

1 Dataset

1.1 Identify dataset

Movie Dataset (stored as 'movies.csv') is from the IMDb website. IMDb stands for Internet Movie Database. Dataset in this project is obtained from IMDb API as a JSON file. It is converted to csv file by pandas package. Table comprises 26 million ratings on 45,000 movies from 27,000 users. Each row of the dataset is a movie object, and each column contains some information about the corresponding movie from the IMDb.

To better understand the data and prepare for feature selection, we investigate the IMDb website and its data generating process. The IMDb Website operated independently from 1990 to 1998. And it has been a subsidiary of Amazon.com, whose business also covers the production and selling of movies, since 1998. According to the introduction of the website, all ratings and basic information of the movie is 'registered and logged-in users can submit new material and suggest edits to existing entries. Most of the site's data has been provided by these volunteers[1]. The website included 83 million registered users and 8.7 million movies and TV episodes as of Dec 2021 according to the public statistics[2]. Other guidelines can also refer to the Contributor's Charter from the IMDb website.

1.2 EDA

Basic Statistics

This movies dataset (movie.csv) covers 45436 movies released between 1874 to late 2020 and covers the record of 5 million votes.
Columns of the dataset covers the basic production information about each movie (e.g. production company, budget), Information about the releasing movie (e.g. title, overview, genres, runtime, languages), and data summarizing the reflection of general public and IMDb registered users (e.g. revenue, popularity, vote_average, vote_count).

Data Cleaning

Columns drop: After basic exploration, movie identifiers, outside links and images columns of the dataset will be excluded. Columns that include the website-generated movie identifier will be removed since identifiers will not be used to discover features about the data. This project will be based on textual, categorical, and quantitative data. Columns containing links to images (posters), websites etc. will not be included.
Row drop: A few rows will not be included in future analysis. By identifying unique values in the `adult` column which should only contain true and false, we discovered a few rows that are off the place. Since the missing information can't be retrieved and the data in these columns is not outliers or special cases, we decided to just drop the data for these movies. Dropping them won't affect neighboring rows.

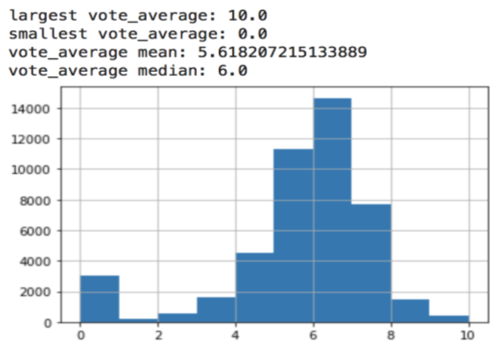
Special format cleaning: Among all the records converted from JSON file to pandas data frame, `belongs_to_collection`, `genres`, `production_companies`,`production_countries`, and `spoken_languages` columns contain the special string values. An example of one value in these columns from the `genres` column is “"[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}]"”. The nature of these strings should be a list of dictionaries. In each dictionary, the id is less informative than names. Considering this information would naturally be provided as a list of names, we decided to extract the names from each string using regex and store them in a list for future analysis. The cleaned version of the example given above will be a list of ['Animation', 'Comedy'].

Findings in EDA

[1] urldefense.proofpoint.com
[2] Press Room Statistic, IMDb.com

We explored most of the columns in the dataset. Below are some columns we closely investigated and worth discussing.

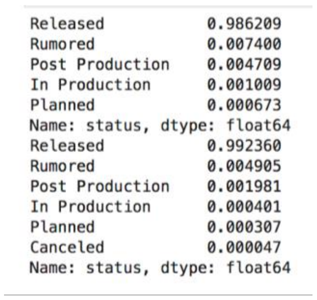
vote_average is the average IMDb rating of each movie based on review records of registered IMDb users on a scale of 1 to 10. Information in this column is predictive value for this project. IMDb ratings will exist for each released movie (movie shown at least once publicly[3]). Majority of the data is normally distributed and centered around 6. Based on the histogram, existing 0s indicates there is insufficient data to calculate the average (Figure 1).



```
largest vote_average: 10.0
smallest vote_average: 0.0
vote_average mean: 5.618207215133889
vote_average median: 6.0
```

Figure 1: Frequency of vote average (bin_width=1)

Non-released movies should have no rating. However, there is no clear difference in distribution of status among movies of zero and nonzero ratings. Therefore, the zero may be related to the popularity of a movie etc and zeros will be considered as a meaning rating in future analysis. Therefore, movies with 0 rating average are kept.



```
Released           0.986209
Rumored            0.007400
Post Production    0.004709
In Production      0.001009
Planned            0.000673
Name: status, dtype: float64
Released           0.992360
Rumored            0.004905
Post Production    0.001981
In Production      0.000401
Planned            0.000307
Canceled           0.000047
Name: status, dtype: float64
```

Figure 2: Table or normalized frequency of each status for movie in & not in collection

adult feauture markes if a movie is adult only: Among all the movies, only 9 movies are classified as adult movies and the 45454 other movies are not. Considering that the group is extremely unbalanced, and the classification is based on the consistency standard of different countries, this feature will be excluded in future tasks. However, based on this feature, we do see that our data is mainly of non-adult restricted movies. This will be taken into consideration for future generalization of the model.

status is the releasing status of a movie (e.g. released, canceled): similar to adult features, the grouping of status features is also very imbalanced. In the dataset, 45014 out of 45379 movies have "Released"' status and 365 movies of other various statuses. There are no significant differences between the distribution of vote_average in each group of movies. (histogram figure 1.b.1, blue for movies with Released status and orange for movies of other status) Therefore, we decided to exclude this feature in future tasks.

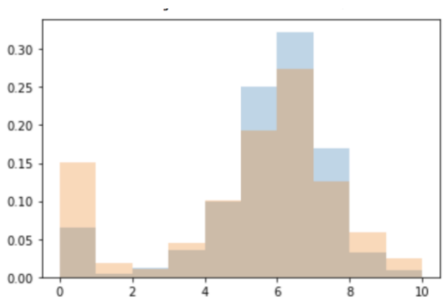


Figure 3 Distribution of IMDb rating scores for released moves (blue) and non-released movies (orange)

Belongs_to_collection this column contains many empty lists, which indicates that a large amount of data doesn't belong to any collection. However, there are only a few movies that belong to the same collection. If we maintain the original collection names, the majority of the future data will be missing this feature, which will negatively affect the generalizability of the result of exploration and prediction models.

Since this column has many empty values and many unique values, we decided to convert it into a boolean feature. Empty values will be converted into False indicating the movie doesn't belong to any known collection and non-empty values will be converted into True indicating it belongs to some collection.

Belonging to collection or not affect the correlation between released time (in years) and average rating (Figure 4). For movies that are not in collection, the release date has a weak positive relationship with IMDb ratings (r=0.0299). For movies belonging to a collection, the release date has a weak negative relationship with IMDb ratings (r=-0.0558).

[3] https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV#

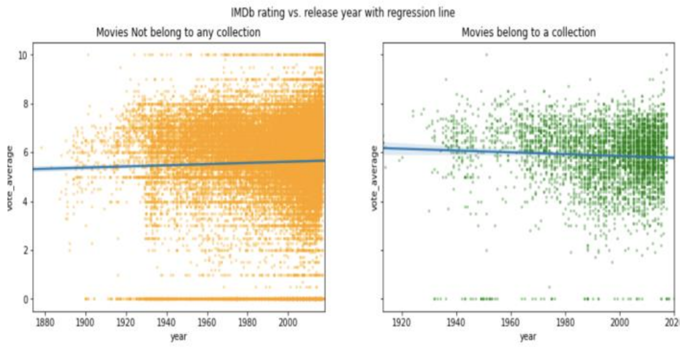


Figure 4: Scatter Plot with regression line for movies not in collection (left) and in collection (right)

Released data contains information about release year, month, day as a single string in format of "YYYY-MM-DD". While there is no obvious variability between ratings of movies released at different days or months, it does change with the release year (Figure 5). The Release date can be informative for predicting vote_average. The variability in data of earlier years might be because there are less recorded / released movies in those eras.

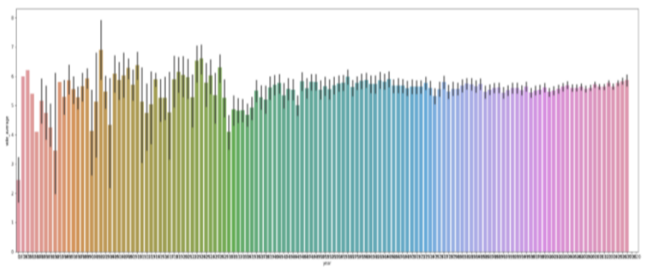


Figure 5: Release year vs IMDb rating

Genres: After cleaning, genres column contains the list of genres (e.g. animation, Crime) each movie is labeled with. All existing genres in the dataset cover a sufficient number of movies (num_movies>100).

```
<AxesSubplot:>
```

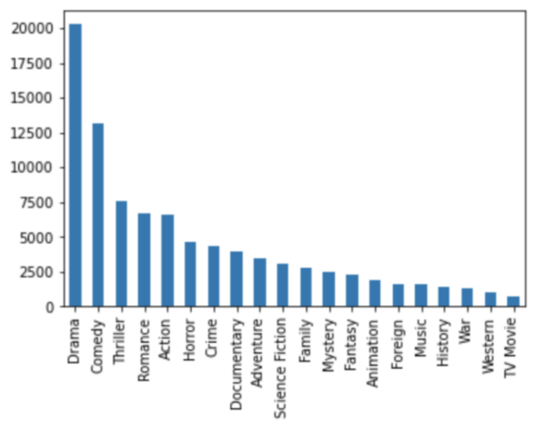


Figure 6: Count of movies labeling with each genre

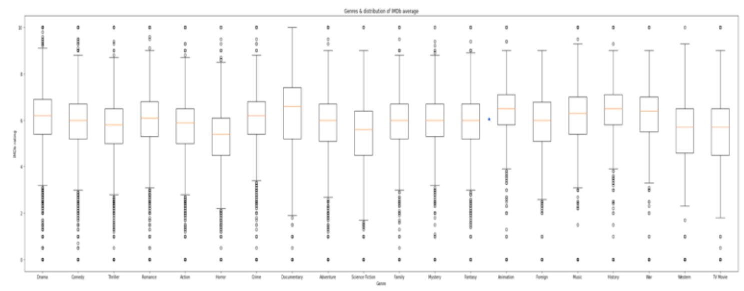


Figure 7: Distribution of IMDb ratings of movies within each genre

Based on the boxplot, movies containing different genres have different distribution of ratings. Movies containing history genres of Documentary genre tend to have higher ratings on IMDb. Movies labeled Horror or western tend to have lower ratings.
Also, outliers shows that there are some movies in each genre that earn extremely high ratings. Looking into the data, these movies tend to be less popular. The high scores might be the result of ratings only from fans.
Boxplot of ratings for movies containing each genre (ratings vs genre)
Production companies contain a list of company names after cleaning. Besides the actual company names, the count of number of companies participating in a movie production may also correlate with the rating of a movie.
Companies Count vs rate with regression

Companies and Genres: Warner Bro. and Metro-Goldwyn-Mayer (MGM) each are involved in the production of more than a thousand movies in this dataset. Each company has their own preference in movie production. The distribution of IMDb ratings of different movie genres are different from the distribution generated from the whole dataset. For example, the median of rating of most genres for movies produced by Warner bro tend to be higher than the median of all movies. Production companies and genres may together be correlated with rating of movies.

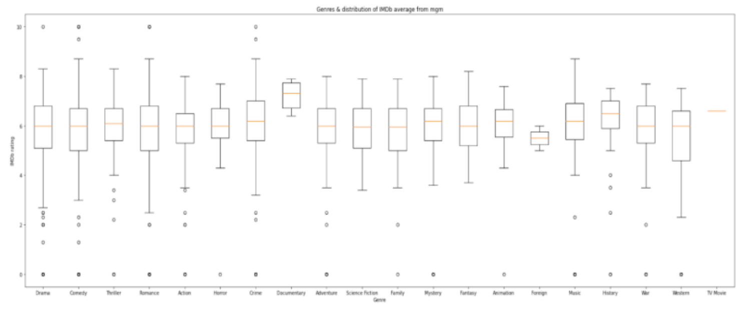


Figure 8: Distribution of IMDb ratings of movies within each genre for movies produced /co-produced by Metro-Goldwyn-Mayer (MGM)

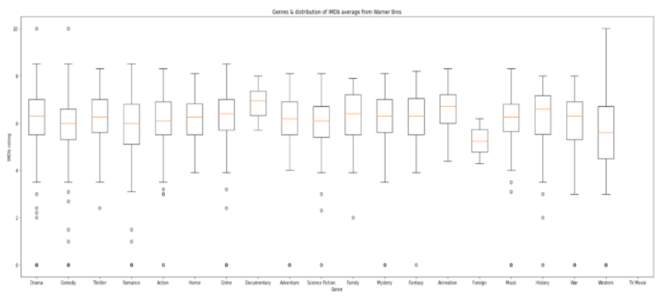


Figure 9: Distribution of IMDb ratings of movies within each genre for movies produced /co-produced by Warner Bro

Popularity and vote_count: Popularity is generated by the IMDb website. Movies with lower popularity score (popularity<10) have various ratings. Movies of Median popularity (10<popularity<50) tend to have ratings maximum

Popularity vs rating: Based on the EDA result, Quantitative features each has a clear correlation with IMDb rating is included in the baseline model. (eg. number of production companies). And we will possibly test if some combination of features could improve the performance of the model (eg. genres + production companies)

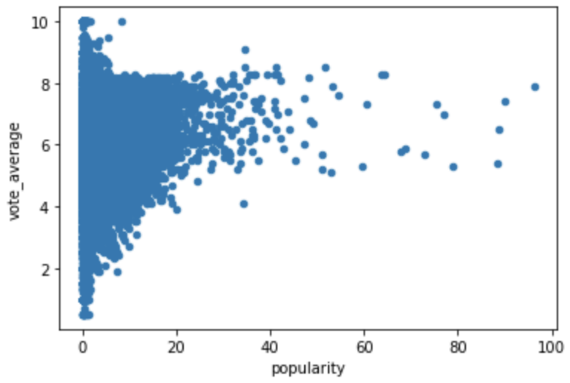


Figure 10: Popularity vs rating (exclude extreme outliers with more than 100 popularities)

2. Predictive Task

2.1 Predictive task

After basic cleaning and engineering, **given the production and releasing information of a movie, predict the vote_average of the given movie**. This is a regression prediction problem.

2.2 Evaluation

Root Mean square deviation (RMSE). For easier interpretation, RMSE will be calculated based on the original unit of measurement. RMSE is a common evaluation metric. It penalizes larger errors harsher than small errors. Since vote_avarage is between 0 to 10 and approximately normal distribution, we expect the model to make the prediction within a limited range and produce less outliers. And RMSE is more sensitive to predictions that are further from the center and can be calculated efficiently[4]. Therefore, in the following tasks, we will evaluate the model performance based on RMSE.

Performance will be dependent on the average RMSE score applying the model to a similar set of testing data.

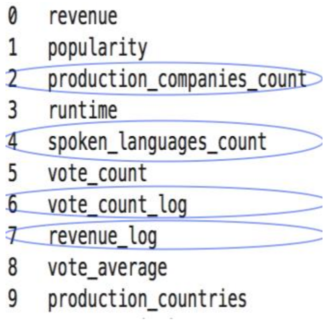
Generalization

To ensure the generalizability of the model, movies data is splited into two tables, US movies and NonUS movies, by whether the production countries of the movie include US. If the model of feature selection engineering and hyperparameter based US dataset performs well on non-US data, our model can predict movies from different countries and regions and doesn't overfit to the trend in US movies and ratings.

2.3 Baseline

Baseline dataset: US movies

Baseline involves features: circled ones are extracted during data cleaning and basic feature engineering.



```
0   revenue
1   popularity
2   production_companies_count
3   runtime
4   spoken_languages_count
5   vote_count
6   vote_count_log
7   revenue_log
8   vote_average
9   production_countries
```

Figure 11: list of baseline features

4 models are used as baseline: Linear regression model¶, Random Forest Regressor, XGB Regressor, and LightGBM.

Linear regression model assumes linear relationship between features and rating. Less likely to overfit. Good comparison for other more advanced models

Random Forest Regressor is a supervised learning algorithm that uses ensemble learning methods for regression performance in lots of tasks. The Random Forest Regressor model can capture some non-linear relationship between features and vote_average. It is also able to work well with both categorical and continuous variables and automatically handle missing values (many of which may be filled with zero in data cleaning and feature engineering)[5].

XGBRegressor: Extreme Gradient Boosting regression model ensembles many weak learners and can reach good accuracy efficiently. And it works well in data with a relatively smaller size[6].

[4]towardsdatascience.com
[5]theprofessionalspoint.blogspot.com
[6] datascience.foundation/datatalk

LightGBM: LightGBM is also based on Gradient Boosting like the XGB regressor but works more efficiently, especially for larger datasets[7].

Baseline models are trained on the same training set of the same features and the performance is evaluated using RMSE for predicting the same set of test data. All hyper-parameters are default except a few parameters are setted to avoid extreme overfitting and ensure efficient execution.(eg. Max_depth, n_estimator)

Baseline Result

LightGBM, XGBRegressor and RandomForestRegressor (average RMSE <1.3)perform significantly better than linear regression (RMSE>1.7) on testing set in a controlled setting.

Among the 3 advanced models, XGBRegressor and RandomForestRegressor perform much faster and can be trained more efficiently. Therefore, XGBRegressor and RandomForestRegressor are the models selected to be further optimized.

3. Model

Initial Setting

Dataset: US movies

Initial features Features: same as features used in baselines

XGBRegressor and RandomForestRegressor are the model packages we choose to further optimize.

Optimization of the model is based on an increase in RMSE scores. All operations that increase RMSE scores over the threshold (difference RMSE=-0.02) will be considered as a significant improvement in performance.

3.1 Basic Hyper parameter tuning

Since both prediction models have a risk of overfitting, we performed basic hyperparameter tuning to reach locally optimized performance on the testing set for each model.

We use cross validation results to select proper hyperparameters and test through some combination of features manually.

With max_depth = 10, n_estimators = 600, the performance of the Random Forest Regressor model improves significantly. The RMSE scores were reduced to RMSE = 1.197 on the testing set.

With n_estimators = 50, max_depth = 10, learning_rate = 0.1, the performance of the XGboosting regressor model improves slightly. The RMSE scores were reduced to RMSE = 1.213 on the testing set.

3.2 Feature Engineering

Quantitative Data

For Quantitative Data, we investigate the missing values and zero values. For features with missing values, we replaced Nan values with zero. (e.g., budget, release year) or mean values (e.g. runtime).

One hot encoding

Features with values in form of a list (e.g. genres, production_companies) are one hot encoded. Since all the categories in the lists are not ordinary and each movie could belong to more than one category, we choose to use one-hot encoding instead of labeling encoding and other encoding method. To prevent overfitting and ensure the efficiency, accuracy of the one-hot encoding of each feature, only common categories (categories that cover more than 100 movies) will be kept.

Since the values exist as lists, we use sklearn.preprocessing.MultiLabelBinarizer [8] to perform the encoding.

TFIDF vector of textual Data

Textual Data like title are quantified into TF IDF matrix, with each cell containing TFIDF score of the word in corresponding movie title. To prevent overfitting, only words with sufficient document frequency are kept. The threshold is set to DF =100 so the minimum category size is consistent with other features.

3.3 Model optimzation

Model optimization is done through feature selection and enginnering. Feature selection and engineering will be performed independently for the Random Forest regressor (max_depth = 10, n_estimators = 600) and XGboosting regressor (n_estimators = 50, max_depth = 10, learning_rate = 0.1). Performance of the model with each additional feature is compared with the performance baseline. The order we tested the features is based on the EDA result and our intuition.

Random Forest regressor (max_depth = 10, n_estimators = 600)

Typical RMSE score of models with each feature or groups of features is compared with the performance with Random Forest regressor (max_depth = 10, n_estimators = 600) using the baseline features.

Among individual features, adding one-hot encoded genres or Release year increases the typical performance of the model on the testing set significantly (increase_RMSE > 0.02)

We then select some combination of features based on common knowledge and the result of a single feature selection. Considering people may prefer different genres that may correlate with time of the year or production and production companies (EDA), we tested the combination of genres with other features. Among the combination of features including Genres, Genres + production_companies (OneHot) improve the performance (change_RMSE = -0.09) and Genres + release year + release month improves the performance significantly (change_RMSE = -0.13)

Feature belongs_to_collection doesn't increase the performance of the model significantly on its own. However, among the

[7] https://neptune.ai/blog/xgboost-vs-lightgbm

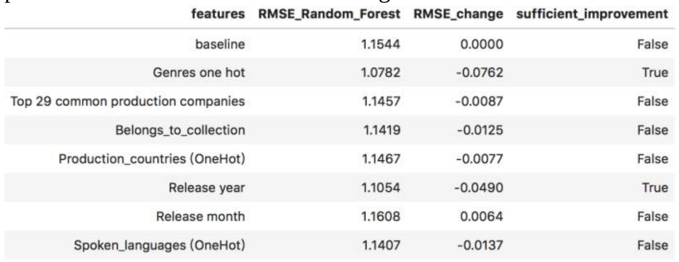
[8] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html

combination of features, Belongs_to_collection + release year improves the performance of the model RMSE score by 0.05.

The TF-IDF matrix extracted from titles of the movies failed to improve the performance of the model. With textual data, the performance of the model is reduced compared to without the textual data.

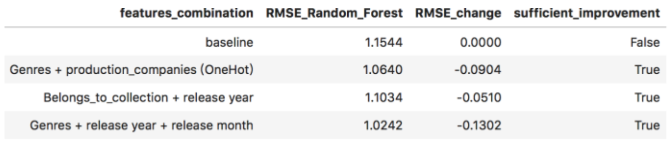
In conclusion, one-hot encoded all genres, one-hot encoded common production companies, belongs_to_collections and release year will be the features added to the baseline features that improve the performance of the model on the testing set significantly.

Training on selected and engineered features with tuned hyperparameters. The Random Forest Regressor reaches a performance of RMSE = 1.02 on testing sets of US data.



features	RMSE_Random_Forest	RMSE_change	sufficient_improvement
baseline	1.1544	0.0000	False
Genres one hot	1.0782	-0.0762	True
Top 29 common production companies	1.1457	-0.0087	False
Belongs_to_collection	1.1419	-0.0125	False
Production_countries (OneHot)	1.1467	-0.0077	False
Release year	1.1054	-0.0490	True
Release month	1.1608	0.0064	False
Spoken_languages (OneHot)	1.1407	-0.0137	False

Figure 12: Table of RMSE score change adding each individual features



features_combination	RMSE_Random_Forest	RMSE_change	sufficient_improvement
baseline	1.1544	0.0000	False
Genres + production_companies (OneHot)	1.0640	-0.0904	True
Belongs_to_collection + release year	1.1034	-0.0510	True
Genres + release year + release month	1.0242	-0.1302	True

Figure 13: Table of combination of features that improves the performance of model significantly

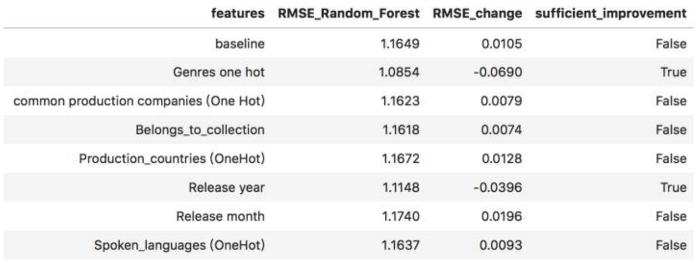
XGboosting regressor (n_estimators = 50, max_depth = 10, learning_rate = 0.1)

Similar to the optimization of the model based on Random Forest Regressor, we also test each feature or combination of features and compare the performance of the model to the model on baseline_features . Combinations of features are tested selectively based on the result of adding single features and common knowledge.

Feature one-hot encoding genres (change_RMSE =-0.07) and release year (change_RMSE =-0.04) improve the performance of the model above the threshold (change_RMSE = -0.02).

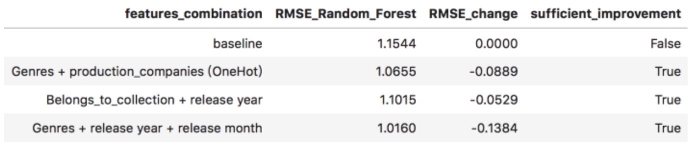
Among all combinations of features, Genres + production_companies (OneHot), Belongs_to_collection + release year and Genres + release year + release month all improve the performance significantly. These engineered features will be included in this model.

The TF-IDF matrix extracted from titles of the movies also failed to improve the performance of this model significantly. With textual data, the performance of the model is reduced compared to without the textual data.



features	RMSE_Random_Forest	RMSE_change	sufficient_improvement
baseline	1.1649	0.0105	False
Genres one hot	1.0854	-0.0690	True
common production companies (One Hot)	1.1623	0.0079	False
Belongs_to_collection	1.1618	0.0074	False
Production_countries (OneHot)	1.1672	0.0128	False
Release year	1.1148	-0.0396	True
Release month	1.1740	0.0196	False
Spoken_languages (OneHot)	1.1637	0.0093	False

Figure 14: Table of RMSE score change adding each individual feature for XGB regressor model



features_combination	RMSE_Random_Forest	RMSE_change	sufficient_improvement
baseline	1.1544	0.0000	False
Genres + production_companies (OneHot)	1.0655	-0.0889	True
Belongs_to_collection + release year	1.1015	-0.0529	True
Genres + release year + release month	1.0160	-0.1384	True

Figure 15: Table of RMSE score change adding each combination of features for XGB regressor model

Training on selected and engineered features with tuned hyperparameters. The XGBoost regressor achieves a performance of RMSE = 1.00 on testing sets of US data.

3.4 Final Optimization Result

The combination of features that optimizes the performance of each model are the same. On US data, both model performance about equally well. Then we applied each optimized mode with feature engineering and hyperparameters onto Non US data.

On non-US movies data, the RMSE of the model of Random Forest regressor is 1.14 compared to the 1.11 of the model of XGBoosting. The difference in performance score is larger than the Significant difference threshold we designed as 0.02. Therefore, we choose to propose an XGboosting regressor (n_estimators = 50, max_depth = 10, learning_rate = 0.1) trained on baseline features along with Genres, production_companies (OneHot), Belongs_to_collection boolean, release year and release month features as our final model in predicting IMDb rating of a movie.

4. Literature

Predict Movie Rating by Chuan Sun

https://nycdatascience.com/blog/student-works/web-scraping/movie-rating-prediction/

Sun's project intends to predict movie ratings. In the project, Chuan Sun generated the data by scrapping the movie information from the IMDb website. He then uses this information to predict the IMDb rating of movies. Highlights in his model is that he generates the posters for movies and does facial detection. He also uses the data from facebook to generate the popularity of the directors and actors of the movies. Overall, there is more complete information and less missing data in his dataset.

He also uses the Random Forest Regressor in his model and discovered that the features he generated from facial recognition and facebooks are helpful for his model performance, along with some original features like duration of the movie and budget.
Sun starts with a similar dataset as us and a similar baseline prediction model. From his project, we learned the possibility of combining datasets like Sun did for facebook data and movie dataset. Also, we will investigate the possibility of filling in the nan values in our dataset using more data sources from the original IMDb website.

The Story of Film by ROUNAK BANIK
https://www.kaggle.com/rounakbanik/the-story-of-film/notebook
Our project is inspired by "The Story of Film" project on Kaggle. Our project started with the same movies dataset as Banik's project. Banik built a regression model to predict the revenue of movies. Under different predictive tasks, Banik and us went through different Data Cleaning, EDA and feature engineering processes. Banik focuses on EDA in his project. For example, he closely investigates the relationship between revenue and release year, month, day and day of week with clear visualization and analysis.
Applying this pipeline of feature preprocessing and Gradient Boosting Regressor in our predictive task, the average RMSE score of the trained model on the testing set is 1.140. The performance is about the same as the baseline model trained by the baseline feature (typical RMSE = 1.15) but worse than performance of our optimized model trained on engineered features (Typical RMSE = 1.00). The result proves that our feature engineering and model selection can perform better in predicting ratings and the feature we engineered could better capture the useful pattern in the database.

5. Result:

5.1 Comparison:

Both RandomForestRegressor and XGBRegressor outperformed the baseline models for both US data and Non-US data.
We split the dataset into two subsets: US data & non-US data based on whether the US is the only country in the production_countries feature. All feature engineering and model optimization done on US data, and the performance confirmed in the other dataset.
After trying single-feature addition and multiple combinations of features, we have improved the RMSE by around 16% from both the RandomForestRegressor and XGBRegressor models. We also tested the generalizability of our feature-selection by applying the same model and feature-engineering to the others data, and also got 11% and 9%. improvement for both the RandomForestRegressor and XGBRegressor models. After several different tries, we found that genres, release_year, and release_month are the most helpful features in addition to the baseline features. For the XGBRegressor, genres, production_companies, belongs_to_collection, release_year, and release_month are the most helpful feature combinations in addition to the baseline features. Our models can outperform the baseline because we have more useful features like genre and release_year, and use hyperparameter tuning to prevent overfit.
The gap of the difference in performance is significant, since we improved at least 9% of the RMSE for both datasets and models.

5.2 Effectiveness

Not all the features we tried improved the performance of our model. When adding each feature individually to the baseline model with controlled hyperparameters, features like production_companies and belongs_to_collecion are not significantly helpful by themselves. However, most feature combinations we added outperformed the baseline model.

5.3 Hyperparameter

In feature engineering, we manually set a few thresholds. During our preprocessing, we kept the minimum number of movies in any single class greater or equal to 100. The threshold could be changed to include more categories for OneHot or more words for textual analysis. During feature selection, an RMSE improvement of >= 0.02 is set as being significant.
For the Random Forest Regresor, we used n_jobs = -1 to improve runtime, and max_depth = 10, and n_estimators = 600 to prevent overfit. The numbers are obtained from Grid Search CV. For XGB Regressor, we manually tuned a couple parameters, and picked the combination with the smallest RMSE, which is n_estimators = 50, max_depth = 10, and learning_rate = 0.1. We also used n_jobs = -1 to improve runtime.

5.4 Major Takeaways

1. If features have similar format, it is usually more efficient to perform EDA and feature engineering together for a group of features.
2. Common features like genres, production companies, and release time are very helpful with predicting movie ratings.
3. The features that improved the performance of both models are the same. This might be because a certain combination of features provides many meaningful information towards rate prediction. For future improvement or project, we could consider using a faster and simpler model to filter through the features and quickly identify helpful ones to fit on more complex models.
4. We performed control experiments during model development and optimization. In each round, we set either the features, model type, hyperparameters as the independent variable and compare the change in RMSE scores. This process is educational and gives us more insight to the data analysis and model building process.

REFERENCES

[1] https://scikit-learn.org/stable/
[2] The Story of Film by ROUNAK BANIK https://www.kaggle.com/rounakbanik/the-story-of-film/notebook
[3] Predict IMDb movie Rating based on text data from review by Javaid Nabi https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958