Movie Popularity Prediction

GR 5291 Advanced Data Analysis

Group 8 Final Project Report

Mengqi Chen (mc4396), Sitong Chen (sc4283),
Xiyao Chen (xc2417), Xiaochen Fan (xf2170),
Jike Fang (jf3123), Ning Fu (nf2422),
Tao Han (th2710), Sijie Jin (sj2839),
Xianghong Luo (xl2723), Yan Qin (yq2232),
Yuhao Tie (yt2587), Che Xu (cx2202),
Siyang Xue (sx2222), Yixiao Zhang (yz3304)

Columbia University

Fall 2018

Contents

Introduction

“What can we say about the success of a movie before it is released? Has any company found a consistent formula which can ensure the success of movies? Given that movie can still flop even with over 100 million producing cost, this question is much more important than ever to the industry. Aficionado of films might have different interests. According to their interest, can we predict which films will be highly rated and whether or not they are commercial success?”

This report aims to show producers what kind of movies are more likely to receive better reputations so that they will know more about their movies before releasing them to the public. Linear and Nonlinear models, and Neural Network have been implemented.

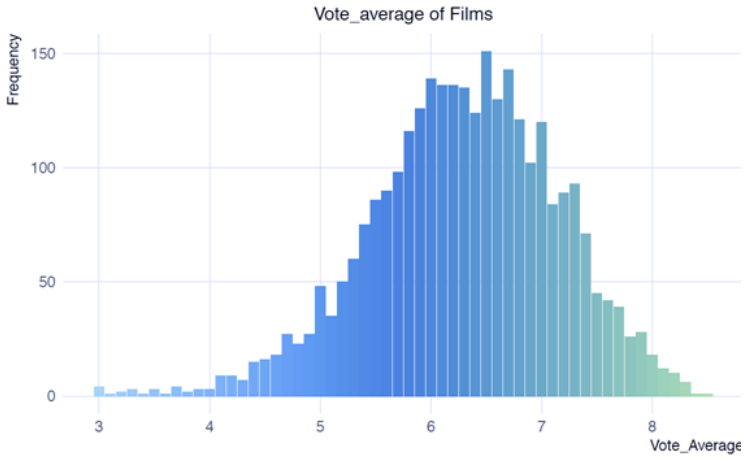
Data Source

The dataset in this report is from The Movie Database (TMDb) in accordance with their terms of use and downloaded from kaggle. The datasets include two main dataframes, named separately tmdb_5000_credits (credit)' and 'tmdb_5000_movies (movies)'. Both have 4803 lines. There are four variables in the credit: movie_id, title, cast, crew, in which the cast is the actors of the movie including cast_id, gender, character, id, name, credit_id, Likewise, the crew contains other stuffs and has subsequent variables: credit_id, department, job, name, id.

There are 20 variables in the movies including budget, genres, homepage, id, keywords, original_language, original_title, overview, popularity, production_companies, production_countries, release_date, revenue, runtime, spoken_languages, status, tagline, title, vote_average, vote_count. In details, budget and revenue are concerning money, but do not specify in currency; genres is to classify the movie such as drama or fiction, and one movie could wear multiple tags; even in production_companies and production_countries, one movie could be more than one companies or countries; original_title and title are the same but might in different languages; popularity and vote_averge are scores; vote_count is how many people participated in this movie's vote; keywords, titles, overview are words and sentences; status is divided into released and rumored. The dataset is originally extracted from company database and some variables contain sub-variables, code, which will be discussed further in exploratory data analysis.

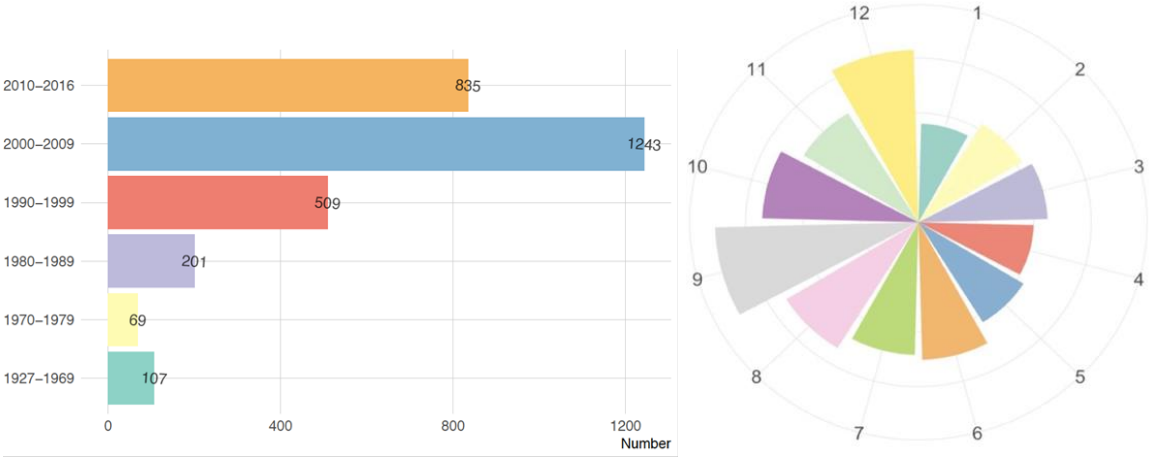
Exploratory Data Analysis

Due to currency inconsistency and missing values on "budget" and "revenue", “vote_average”- average rating that a movie received is used as the response variable instead of revenue and it follows a normal distribution. Plot 3-1 shows the distribution of the response variable, “vote_average”. Initial data cleaning drops off missing values and ‘0’, and puts the words description aside.

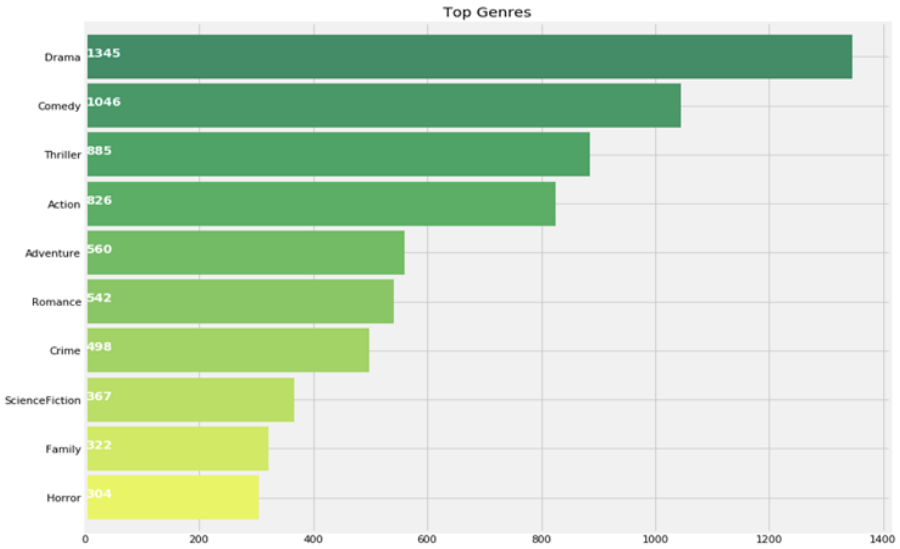


Plot 3-1

Explanatory variables generally are all numeric type including vote_count, runtime, year, month, genres (19 multi-hot variables), major languages (15 multi-hot variables), min_language, majority_studios, and minority_studios. The time span of the data is from 1927 to 2016. Plot 3-2 is the number of movies in different time slots. Since 90s there was an increasing number of movies thriving in the market. Releasing of movies differs either; summer session and December have more new movies, and January is the recession. The Genres (Plot 3-3) includes 19 categories in total and one movie has multiple tags; drama ranks 1st following by comedy, thriller, action. This might indicate public’s general preference.

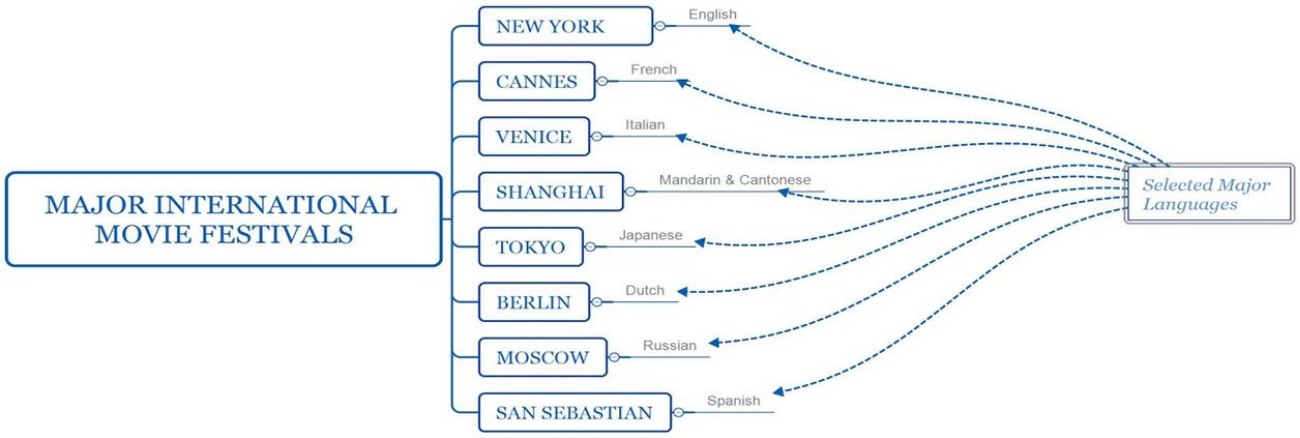


Plot 3-2



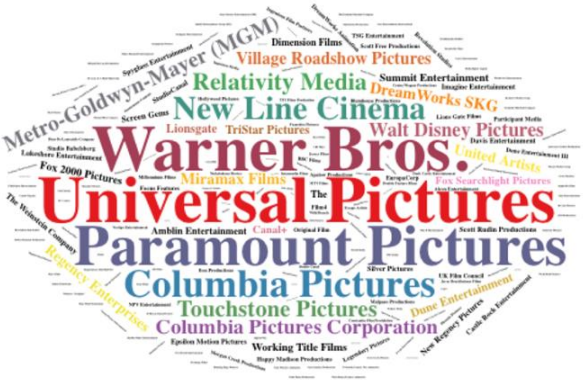
Plot 3-3

There are 55 languages in spoken_language. This language matrix exhibits quite sparsity since some of the languages are rarely used. Therefore the occurrences of languages spoken in certain movies are calculated, and set the 3rd quartile, which is 28 times, as our threshold to split the data into majority languages and minority languages. So there are 15 major spoken languages including English, Spanish, French, Mandarin, Japanese and etc, and the rest 40 minor spoken languages are compressed into one variable called min_language. Besides, our selected major spoken languages have covered most of the official spoken languages in the International Film Festivals (Plot 3-4).

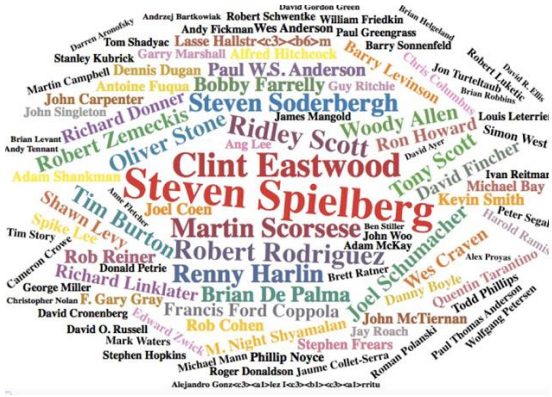


Plot 3-4

Likewise, there are over 3000 companies in movie industry but only several big ones such as Paramount, Universal Pictures. These top 6 companies occupy over half of movie market. Majority studio is created to represent the number of top 6 companies' movie production. Minority studio represents the number of small companies in a movie production. Plot 3-5 is the studio names word cloud ranked by the number of movies produced. The Director is a crucial component to a movie's success. Some of them are well-known and productive. Plot 3-6 is the director's name ranked by productivity.



Plot 3-5



Plot 3-6

After the data cleaning and exploratory data analysis, there are 41 predictors and 1 response in total. Before implementing models, collinearity and distributions have been checked. Since none of the predictors have VIF greater than 10, we conclude that no collinearity exists. Moreover, the data has been randomly set into train (75%) and test (25%), which are separately used to train models and evaluate their performances.

Model Implementation

Part I: Linear regression models

Starting with linear regression model (OLS), we adopt two selection methods:

1) Stepwise model selection
2) LASSO model selection

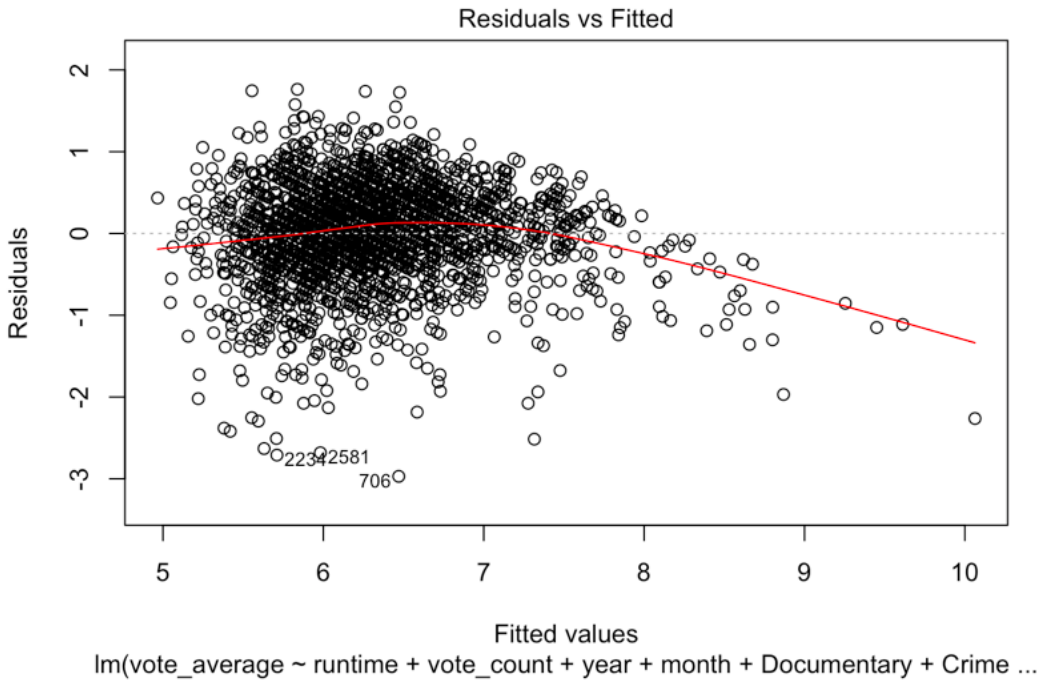
Stepwise model selection

We first use stepwise selection by AIC as the selection criteria from both directions to select our linear regression model, and the model is shown as below:

vote_average ~ runtime + vote_count + year + month + Documentary + Crime + Foreign + Adventure + Action + Comedy + Science.Fiction + Fantasy + Drama + Animation + Family + Horror + L4 (Russian) + L9 (Dutch) + L15 (English) + L50 (Cantonese) + L53 (Thai) + min_language + majority_studios + minority_studios

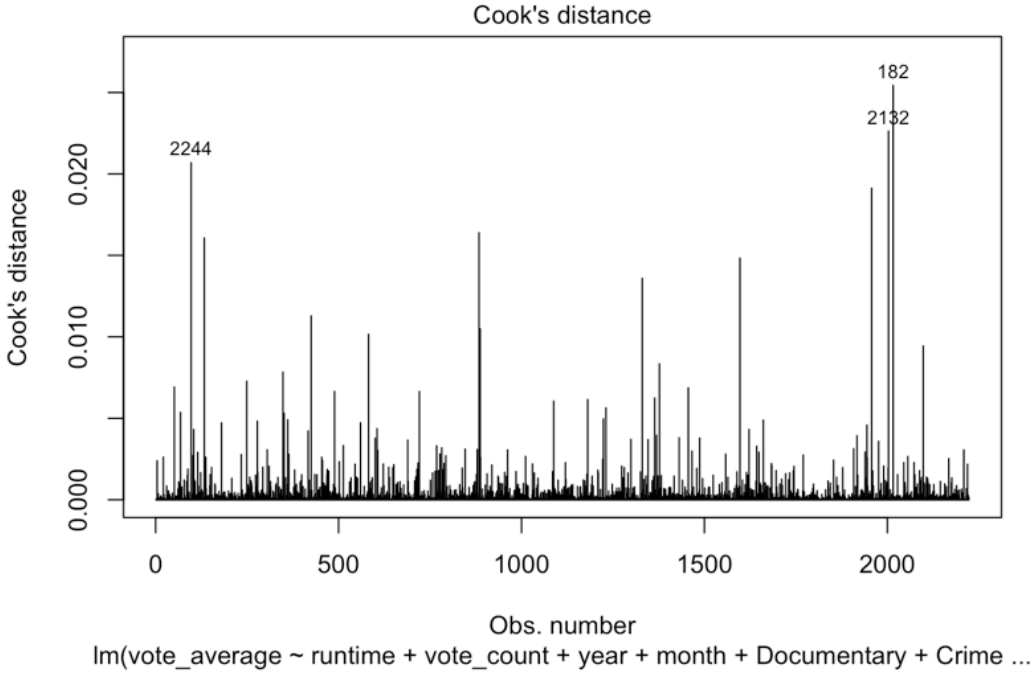
The final model suggests to include some specific genres and languages as the explanatory variables. This could be interpreted as the following genres: “Documentary”, “Crime”, “Adventure”, “Action”, “Comedy”, “Science Fiction”, “Fantasy”, “Drama”, “Animation”, “Family”, and “Horror”, together with the following languages: “English”, “Dutch”, “Russian”, “Cantonese”, and “Thai”, are most associated with the dependent variable, “vote_average”. In other words, these selected genres and languages are either most attractive or unattractive to the audiences who have been participated and voted to the movie survey.

We then check the validity of the underlying assumption, and we figure out that linearity and uncorrelated error assumptions are reasonable, but neither normality nor homoscedasticity is valid. For instance, the residual vs. fitted values plot indicates that the smaller fitted values correspond to larger error variances. We note a “metaphone” shape of residuals (see Plot 4-1-1). In other words, the plot has a mild pattern of non-constant variance.



Plot 4-1-1

The p-values we get from both Breusch-Pagan test and Non-constant Error Variance test are less than 0.05, in which match the plot 4-1-1, and thus we reject the constant variance assumption. Moveover, we apply Bonferroni outlier test to look for any outliers in the model, and 706, 2234, 2581, 357 are detected. The Cook's distance plot suggests that 2244, 2132, and 182 are suspected to be highly influential points (Plot 4-1-2).

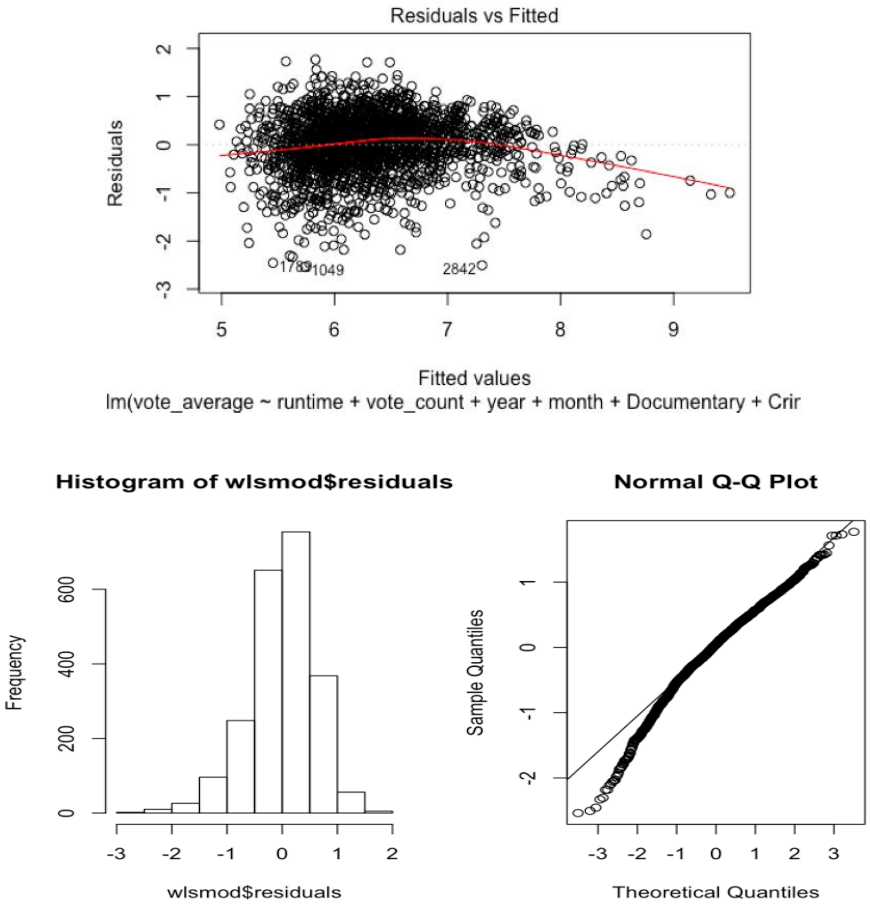


Plot 4-1-2

Since all of the outliers and influential points are the films that either made by small production companies or have extremely low values of the response variable “vote_average”, they do not represent majority of the films, therefore we can simply remove them for remedy. Since some of the underlying assumptions are violated in our OLS model, we take the next step to do some corrective measures and get three transformations, i.e. Log transformation, BoxCox transformation and Weighted Least Squares method respectively, to see which one would have a better performance while at the same time not violating the assumptions of regression models. The results from Log transformation and BoxCox transformation are very similar. Although both of them give us slightly higher R-squared and more normal plots, the problem of heteroskedasticity is not solved yet. No obvious improvements have achieved through either Log transformation or BoxCox transformation, therefore, we choose to try the weighted least squares regression, which solves the non-constant variance problem across all levels of the explanatory variables.

In order to work with the weighted linear regression, we treat each observation as more or less informative about the underlying relationship between independent and dependent variables. The points that are more informative are given more “weight”, while those that are less informative are given less “weight”. In theory, the weighted least squares method is based on the assumption that the weights are known exactly. However, the exact weights are almost never known in real applications, so estimated weights must be used instead. The effect of using estimated weights is difficult to assess, but experience indicates that small variations in the weights due to estimation do not often affect regression analysis or its interpretation. Thus, in terms of how to determine appropriate weights, we use the following procedure: first, store the residuals and the fitted values from the ordinary least squares (OLS) regression. Calculate the absolute values of the OLS residuals. Second, regress the absolute values of the OLS residuals versus the OLS fitted values and store the fitted values from this regression. Then, calculate the weights, which are equal to 1 over the square of the fitted values that are the fitted from the regression in the last step. Last, we refit the original regression model but use the weights in the weighted least squares (WLS) regression. One disadvantage of weighted least squares regression is that it is sensitive to the effects of outliers. If potential outliers are not investigated and dealt with appropriately, they are likely to have a negative impact on our weighted least squares analysis. Therefore, we remove outliers and influential points that are detected in non-transformed model.

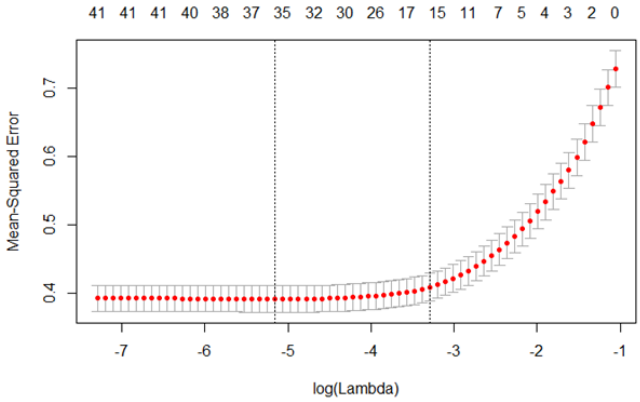
The result of weighted least squares regression is quite satisfying because the assumption of equal variance is finally justified, and all the other underlying assumptions are met as well (Plot 4-1-3). So we use the weighted least squares regression as our ultimate transformed model in corresponding to stepwise regression model.



Plot 4-1-3

LASSO Model Selection

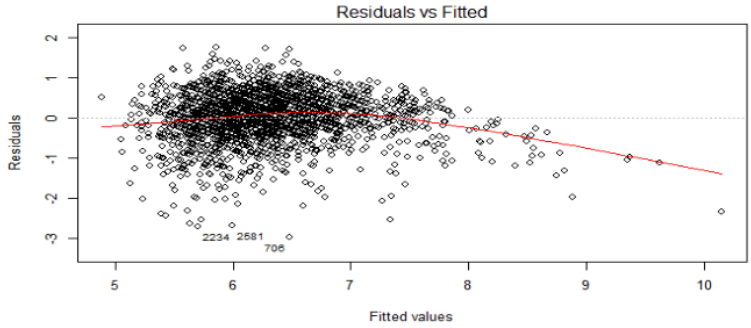
LASSO model selection method is invoked because the dimension of data is wished to be reduced further. However, LASSO L1 regularization selects 12 additional variables compared to stepwise method. In other words, we include 36 predictors in this model to get the smallest MSE. Plot 4-1-4 shows the MSE under different numbers of predictors.



Plot 4-1-4

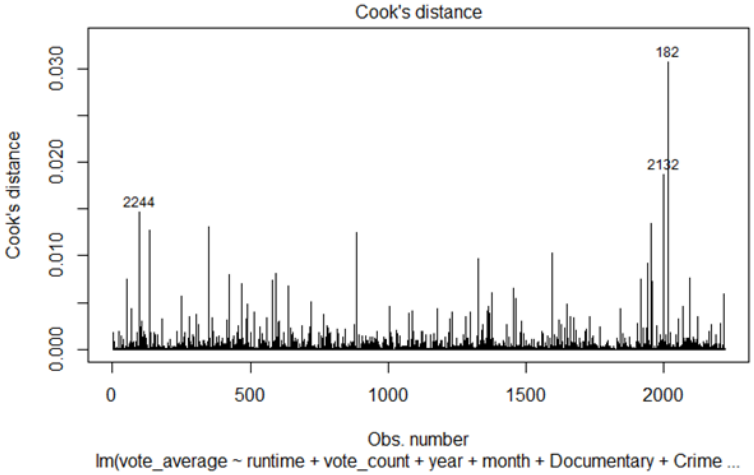
The model is shown below:
vote_average~ runtime + vote_count + year + month + Documentary + Crime + Foreign + War + Adventure + Western + Music + Mystery + Action + Comedy + Science.Fiction + Romance + Fantasy + Drama + Animation + Family + Horror + L4 (Russian) + L6 (Polish) + L9 (Dutch) + L10 (Latin) + L12 (Portuguese) + L13 (Spanish) + L15 (English) + L33 (French) + L50 (Cantonese) + L52 (Japanese) + L53 (Thai) + L54 (Czech) + min_language + majority_studios + minority_studios

The underlying assumptions of linear regression are checked for this model.The linearity assumption holds and we get R-squared value equal to 0.48. The Durbin-Watson test has p-value 0.604, therefore the uncorrelated errors assumption holds.The p-value we get from Shapiro-test is less than 0.05 and we reject normality assumption. The p-value we get from both Breusch-Pagan test and Non-constant Error Variance test are less than 0.05, in which match plot 4-1-5, and thus we reject the constant variance assumption.



Plot 4-1-5

We apply Bonferroni outlier test to look for any outliers in the model, and 706, 2234, 2581, 357 are detected. The Cook's distance plot (Plot 4-1-6) suggests that 2244, 2132, and 182 are suspected to be highly influential points.



Plot 4-1-6

To solve the failed assumptions shown above, we try WLS to modify our model as before. After applying weighted least squares, all assumptions hold.

In conclusion, weighted least squares is utilized as the remedial measures for linear regression models under both selection criterions.

Ridge Regression

Ridge regression is similar with LASSO in several aspects, however, ridge regression could not perform the function of variable selection. Since ridge regression is also considered as linear regression, the underlying assumptions of linear regression are checked and the conclusions are same as the results of LASSO.

The difference of OLS and ridge regression is compared first, based on the model including all 41 variables. Comparing test error of OLS and ridge, OLS has a better R-squared; however, ridge has a better result when considering the values of coefficients.

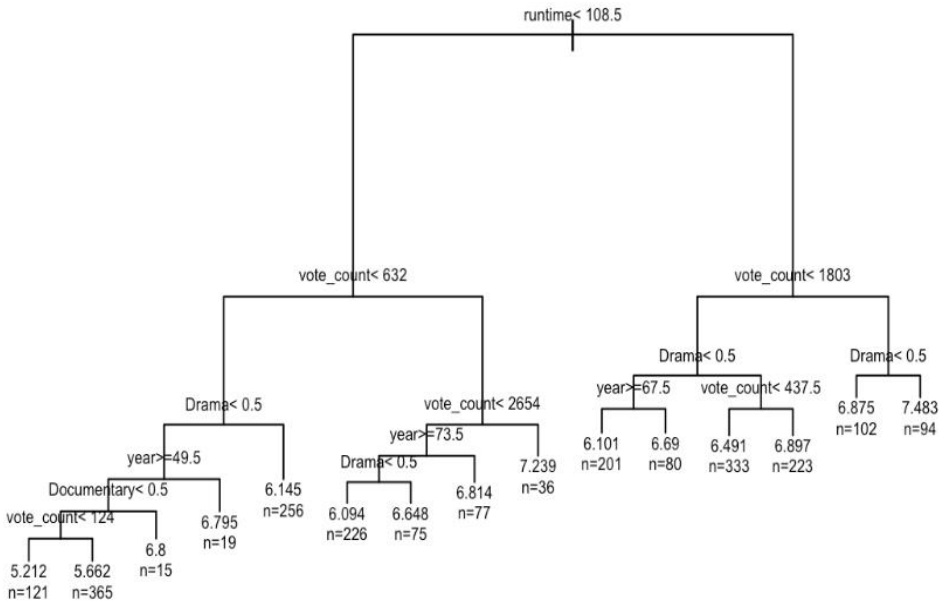
Since ridge model cannot select variables, we use ridge model based on the 36 predictors selected by LASSO. The specific values of test errors will be mentioned in the conclusion.

Part II Nonlinear models

We explore non-linear models in this section: tree model, random forest model and XGBoost model. Non-linear models are good for fitting complex data structure, though are subject to overfitting problems. We divide data set into training set and test set to reduce the over fitting problems.

Tree Model

Tree model is a basic non-linear method, with three main parameters: depth, split, and CP. Depth is tree depth. Split gives the minimum number of observations in a node before attempting a split, and a split must decrease the overall lack of fit by a factor CP. In our model, we set parameters to be depth = 8, split = 20, and CP = 0.01.



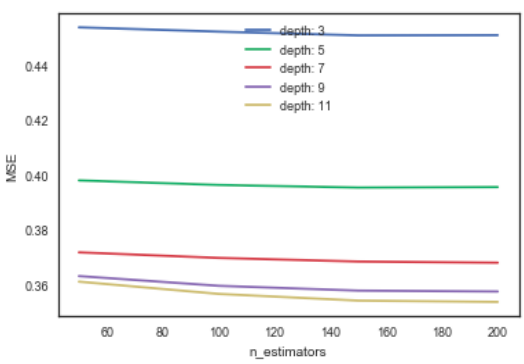
Plot 4-2-1

The tree structure (Plot 4-2-1) shows that runtime, vote count and drama are three most important factors in the tree analysis. The tree model is implemented in R with rpart library. The model is trained and tested in separate data set, and we get test error 9.6%.

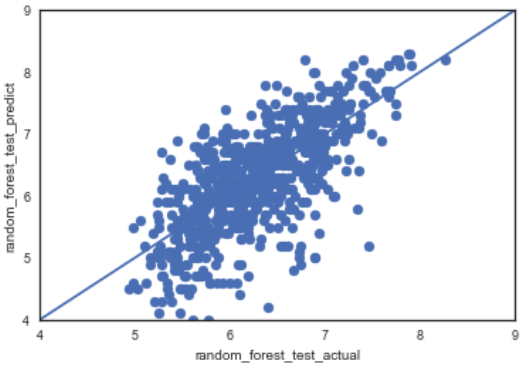
Random Forest

Random forest builds multiple decision trees and merges them together to obtain a more accurate and stable prediction. It is a sophisticated modeling technique and much more robust than a single decision tree. It aggregates many decision trees to limit overfitting as well as error due to bias and therefore yields useful results.

To implement random forest on our dataset, we first use GridSearchCV in Python on the training dataset to select the best hyperparameters for the model. The plot 4-2-2 below shows the mean squared error under different max depths (restricts the depth of each individual tree to prevent overfitting) and number of estimators (the number of trees in the forest). From the result, we choose max depth equal 11, number of estimators equal 200 to fit our model. Using these parameters, we can see from plot 4-2-3 the relationship of true response values and predicted ones.

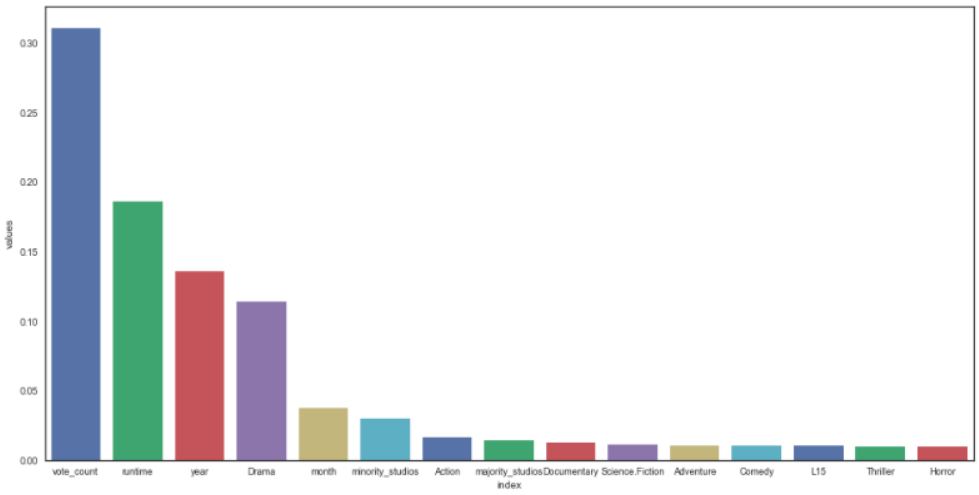


Plot 4-2-2



Plot 4-2-3

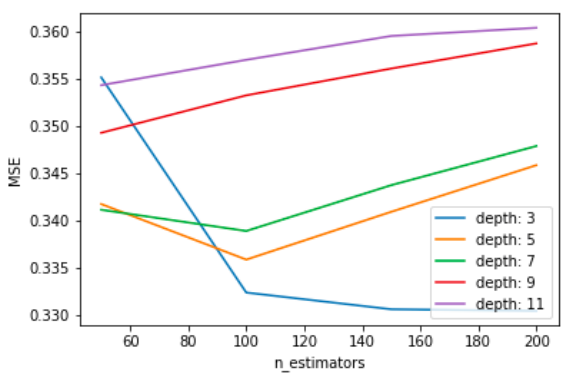
We also explore feature importance by implementing random forest. The plot 4-2-4 shows the 15 most important features that were selected by random forest model. We can see that vote count (the count of votes received) has the highest feature importance in the model. Here, L15, the fifth column from the right, means English.



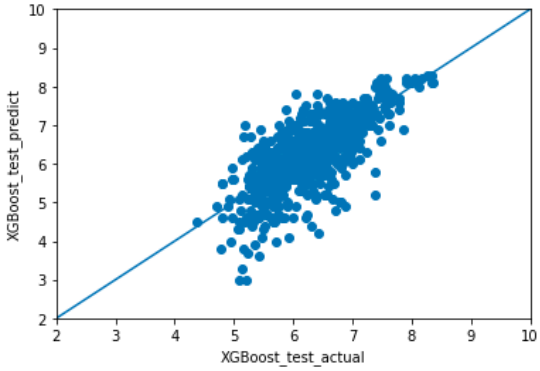
Plot 4-2-4

XGBoost

We also perform XGBoost on our dataset. XGBoost is a gradient boosting method with an additional custom regularization term in the objective function. To implement XGBoost, first we tune our model on the training dataset using a five-fold cross validation. The plot 4-2-5 below shows the mean squared error under different max depths and number of estimators. From this result we choose max depth to be 3, and number of estimators to be 200 as our best XGBoost model. Using these parameters, we can see from plot 4-2-6 the relationship of true response values and predicted ones.

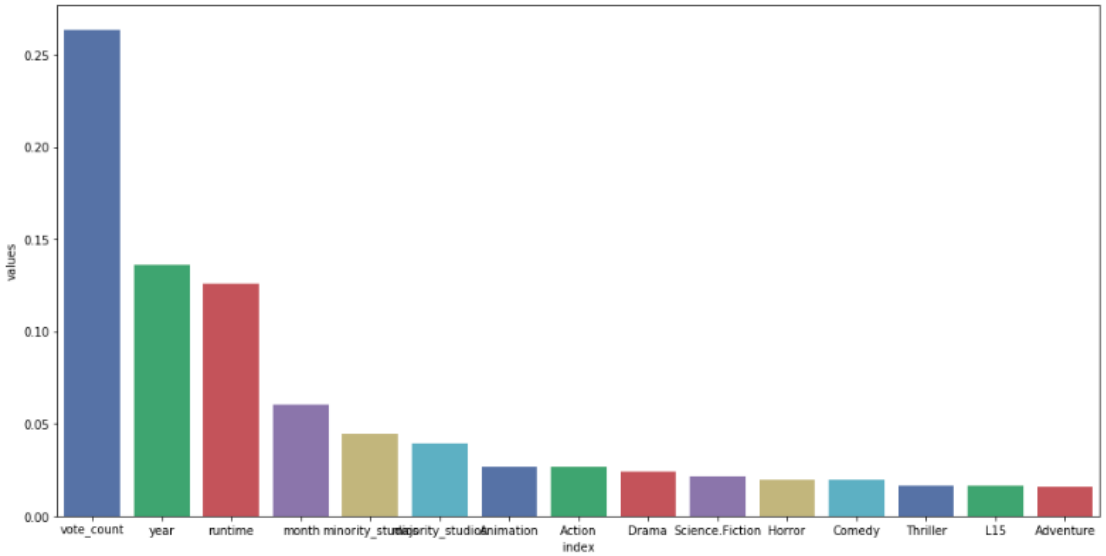


Plot 4-2-5



Plot 4-2-6

The plot 4-2-7 shows important features selected by XGBoost. We can see vote-count, year, runtime, month, studios etc. are important features in this prediction. This result is similar with the one selected by Random Forest.



Plot 4-2-7

Part III Deep Learning Model

Neural Network

A simple feedforward Neural Network exploits the flexibility of the nonparametric framework and its multi-layered construction complicates its representation of the dataset that might in some situation, where overfitting does not detract from its generalization, outperform other ML methods such as SVM, Random Forest, etc.

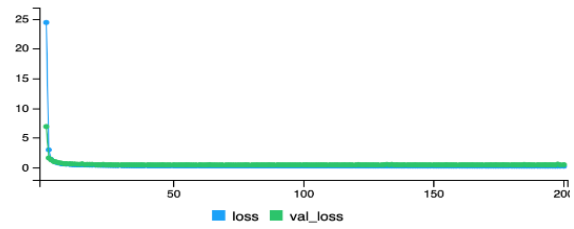
Model

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 16)	672
dropout_1 (Dropout)	(None, 16)	0
dense_2 (Dense)	(None, 16)	272
dropout_2 (Dropout)	(None, 16)	0
dense_3 (Dense)	(None, 1)	17

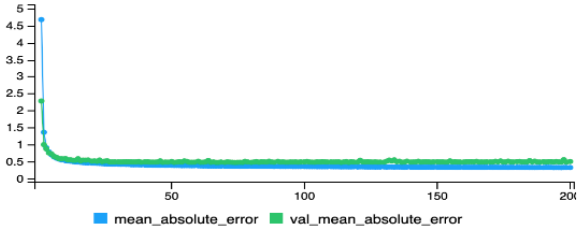
Total params: 961
Trainable params: 961
Non-trainable params: 0

Table 4-3-1

The Neural Network model (note the model architecture displayed above in table 4-3-1) we have used for our project consists of 2 hidden dense layers with 16 neurons each and ends with a output layer with an identity function as its activation, as required by a Regression problem. We divide the training set into 4 subsets and we deem cross validation appropriate for hyperparameter tuning in this instance due to the scanty observations we have with missing data eliminated. We feed into the model batches of 16 randomly chosen observations and train the model with stochastic gradient descent and back propagation algorithm for 200 epochs without regularizing. The two plots below show the results without the dropout layers under MSE loss function(plot 4-3-1) and MAE loss function(plot 4-3-2). The blue lines are training loss and green lines are validation loss.

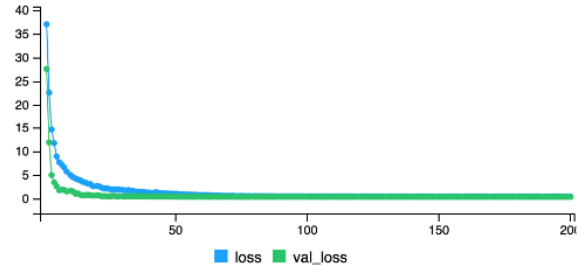


Plot 4-3-1

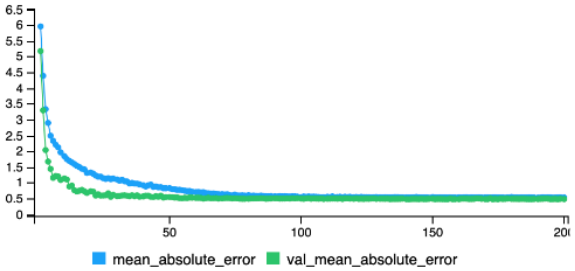


Plot 4-3-2

We have, at the beginning of the training phase, observe a drastic decrease in MSE loss, which is immediately followed by a long period of time when the model stop improving (note the graphs displaying two different kinds of loss above). We therefore have concluded that this was a classical case of overfitting. We thereby add the dropout layers that randomly set 50 percent of the outputs of the previous layer to zero and scale up the remaining values to compensate for the loss of information. The two plots below show the results with the dropout layers under MSE loss function(plot 4-3-3) and MAE loss function(plot 4-3-4). The blue lines are training loss and green lines are validation loss.



Plot 4-3-3



Plot 4-3-4

The result is slightly better than the non-regularized version with its smoother decreases in loss (note the graphs displayed above), which still stop improving after around 30 epochs, so we decide to stop the training at 50 epochs. We have also tried to incorporate L1 and L2 penalty terms into our loss function and to use different activation functions, but they perform the same, if not worse, comparing to the dropout layer and rectify linear activation function, so we leave them untouched.

Discussion

MAPE represents mean absolute percentage error which is calculated by:

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

We use MAPE as out test error measurement because it is intuitive and easy to understand. Table 5-1 shows the test errors of our models. We conclude that our best model for this data is XGBoost since it has the smallest test error, 7.83%.

Models	**Test Error (in %)**
Weighted Least Square (AIC)	8.37%
Weighted Least Square (LASSO)	13.43%
Ridge	8.41%
Tree Model	9.60%
Random Forest	8.21%
Xgboost	7.83%
Neural Network	8.21%

Table 5-1

We make some explanation on the results as follows:

Linear Model

As we discussed earlier, weighted least square is applied on our linear models to correct for heteroskedasticity. On average, linear models don't perform very well for this data. One possible reason is that we shouldn't put heavy penalty on the feature selection because most of our features belong to the same category. For example, the 19 genre variables should be considered as one variable which represents the genre of a movie. For the same reason this also applies to the 15 majority language variables. We would rather include all features (41 variables) than partial of them in linear models. Since R squared isn't quite large, we suspect that nonlinear model may adequately fit this data.

Nonlinear Model

For nonlinear models, we implement tree model, random forest, and XGboost. We can see XGBoost performs well in our dataset because of the following advantages. First, it can learn complex non-linear decision boundaries through boosting. Second, it is optimized for sparse inputs. Finally, it has an additional custom regularization term in the objective function, which can prevent overfitting.

Neural Network

We train the final model on the whole training set and test it on the test set but find that the MAPE to be comparable to that of Simple Linear Regression (MAPE=0.086), which is used as baseline model. We are understandably little frustrated by this result and at first again attribute it to overfitting, but the situation remains unimproved even if we try to reduce the number of hidden layers and neurons in each layer. Another possible explanation might be that the internal structure of the dataset is simple enough to be captured by a simpler model. The result therefore cautions against using deep learning models as some kind of panacea that overcomes all difficulties faced by shallower models without taking into account the context in which it is implemented.

Conclusion

In this project, we aim to show movie producers what kind of movies are more likely to receive better reputations, in order to help them know more about their movies before releasing them to the public.

To analyze what kind of movies gain higher rating, we do exploratory data analysis on a Movie Database (TMDb) dataset from Kaggle, and choose useful variables to build our model. We implement WLS as linear model, tree model, random forest, XGboost as nonlinear model, and neural network model as deep learning model on the dataset. Eventually we conclude that XGboost performs the best in this dataset.

From our analysis result, we conclude that releasing year and month, movie runtime, studio, genre and language are all significant features that give impact on a movie's rating. For example, we can see the scale of studio influence a lot, big studios such as Universal Pictures, Warner Bros., or Walt Disney may tend to produce movies with high reputation. Genre also impacts a lot; genres such as "Action", "Drama", "Science Fiction", "Horror" and "Comedy" seem to mean a lot to receive good reviews. Moreover, "English"has an important influence on movies' rating as the spoken language. A brief explanation for this may be movies released in English-spoken countries can receive larger market potential.

In particular, for linear models, movies with the genres in "Drama", "Crime" and "Animation" are more popular than others. Some genres, such as "Foreign", have negative coefficients, which indicates less attraction to the audiences. While, a combination of several genres in one movie will result in a "genre" that could meet the demand of a wide range of moviegoers. This is intuitive as most successful commercial movies are comprised of various elements.

Appendix

This is our GitHub link: https://github.com/GR5291Group8/GR5291Group8
It include our datasets, codes, figures, and outputs there for reference.