GR 5291 Advanced Data Analysis Project

Prediction of Movie Box Office

Yifei Lang (yl3365)

Ke Lu (kl2839)

Chenlu Ji (cj2452)

Zhaoyan Jin (zj2186)

Jiayi Wang (jw3316)

Chi Zhang (cz2388)

Xuan Zhang (xz2463)

Tianying Yu (ty2299)

1. Introduction

Short weekend after long days working, you are going to a movie with friends. You may have a lot of questions before finally decide the one. Like what are showing recently? It seems a lot of people go for this one. Another movie that co-workers have mentioned sounds better than the trailer I have watched. Although it might be tough, the choice is limited to one per time. Only after you have watched the full movie, you can tell whether deserves the ticket or not.

During this decision-making process, a moviegoer have thought of a large majority of factors when discovering his movie option. As this process get repeated millions of time, the movie chosen system are created and a particular movie's box office get generated.

By understanding how this process run and what specific elements moviegoers care most, we can get insights into their preference.

2. Objectives

Since our goal is to provide a reference for the movie companies, film distributors, as well as movie theaters to have an estimation about the revenue of a specific movie. Obviously, the most difficult part will be giving a exact estimation of the numerical revenue. So we decided to separate the goal into three parts:

1. Profit or Loss: there are plenty of movies produced every year, and some of them can make high profit with low budget, some of them may face a loss even they spent a lot of money on the movies. So, based on the information of budget, actors, directors, movie genres, etc. it will be helpful if we can provide a prediction about whether the movie will earn or loss money.
2. Higher or Lower Than Annual Average Revenue: except for the Profit or Loss, the revenue of a movie can represent a director's or some actor's box office appeal, a market test for some theme, as well as the reputation of a movie company. Thus, if we can predict whether the box office of a newly-released movie can exceed the Annual Average Revenue or not, it will be a powerful tool for the movie companies to make a decision about the issue of a movie.
3. Box Office: this is our original goal for this project. If we can figure out a numerical prediction of the box office before a movie released or even made, then we can provide a solution to the movie companies. Within the solution, we can offer them some strategies to increase their profit, or the latest information to help them make choice.

3. Materials and Methods

a) Data Source

There are a lot of websites focusing on the data collection and comments of the released movies. Since we were trying hard to collect as much data as possible, we compared tens of most famous movie websites and decided to scratch data from the websites with largest number of past movies.

1. TMDB (The Movie Database): the movies here come from the last 100 years and countries all over the world. We scratch a group of information from 19453 movies, and the information includes Movie Title, Year, Status (Released, Post Production, In Production, Planned or Rumored), Language (27 kinds of languages in total), Runtime, Budget, Revenue, Cast (the first 6 actors of highest bill), Studio, Opening Theaters (number of the theaters), and Open Date. These data contained almost all of the basic statistics we can get of a movie.
2. Box Office Mojo: this website is similar as the TMDB, but they focus on different direction. The TMDB has all kinds of information, and the Box Office Mojo focuses on the box office. So we can get the most detailed data of the box office of a movie. At this time, we scratched 16677 movies' box office data from this website, including the Domestic Box Office (total), Foreign Box Office (total), First Week Domestic Box Office, Rank (first week), as well as the Genre.
3. Kaggle: this is a famous website for data analysts, there are a lot of data analysis competitions and sharing of datasets. Occasionally, we found an article talking about the movie recommendation, that article focuses on recommending movie to users. But it provides a whole new idea to us, which is we can add more kinds of information in our analysis like: Movie Facebook Like (number), Number of Reviews, Director Facebook Follower (number), Actor Facebook Follower (number), IMDB Score and the Number of Voters for Score.

Since we obtained the data from three different websites, there were a lot of differences among the datasets, some of the movies existed only in one dataset, some of the values were missing. So we spent a lot of time on cleaning the data to make it easier and more clear to analyze. Finally, we got a dataset with 2833 movies and 41 variables, and 15 of the variables are numerical and the other 26 variables are categorical. And in order to do the following analysis, we changed most of the categorical variables into dummy variables. This method increase the number of variables rapidly, but can give us a reasonable way to explain the results.

b) Analytical Plans

In this project, we will discuss what kinds of factors are essential in this decision-making process, how great the impacts are and finally suggest three different ways to predict movie box office.

In this project, we combined the data scraped from TMDB, Movie Box Office and the data available on Kaggle to predict the newly-released movie box office.

We first did some exploratory analysis to check some properties of our variables within the data such as normality, univariate, multi-collinearity. We also did time series analysis on average movie revenue and budget. The best models we fitted on them were ARIMA. However, we failed to find cointegration between these two variables. We then fitted decision tree model on merged data on whether movie is profit or loss, below or above average revenue, profit range of movie box office respectively. Besides that, random forest model also suggested a good fit. Furthermore, we constructed neural network model, which returns the best fit with 80% successful rate on first two factors and 30% on the third one.

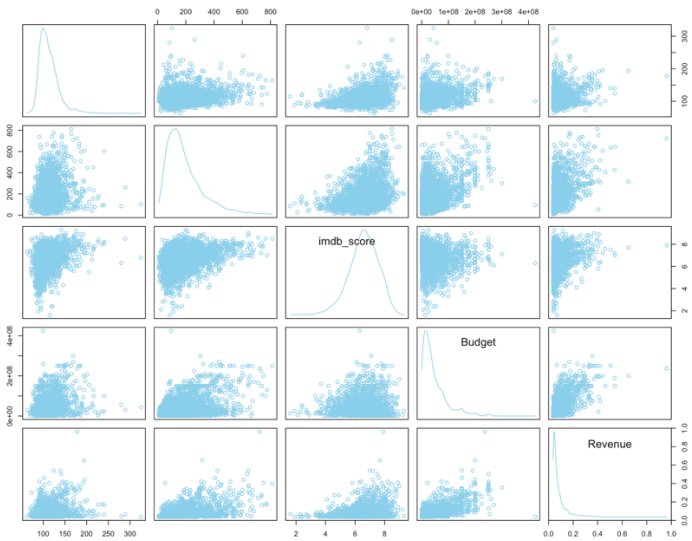
In the cluster analysis part, we ranked actors from 1 to 6 based on the number of movies they showed, movie budget, and the average information in the specific year, making it more convenient for further analysis as well as model application.

4. Results

4.1 Exploratory Data Analysis

Missing data point check

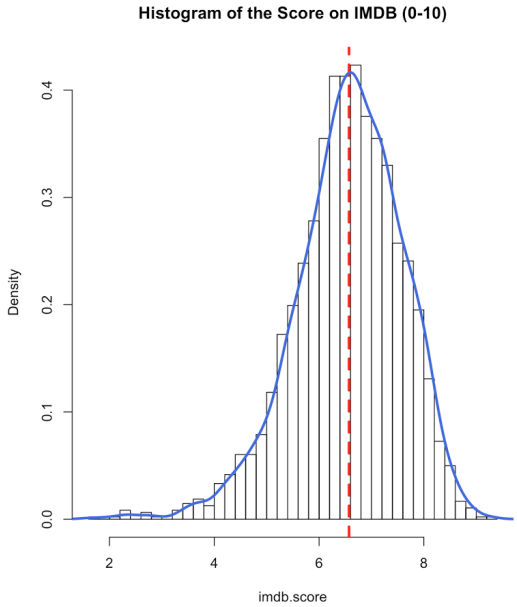
After merge and clean the data, we can focus on some EDA of the data to find basic information and interesting phenomena within the data. The first thing we can do is to check whether there is any missing data within the dataset.

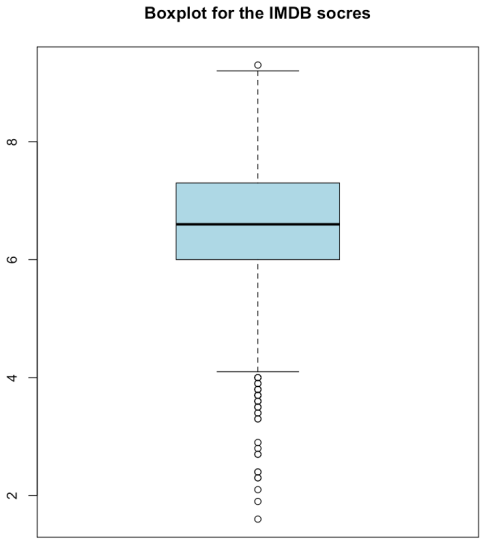


From the plot above, we can make sure that our dataset has no missing point. Because there is no “red point” within the scatter plot matrix.

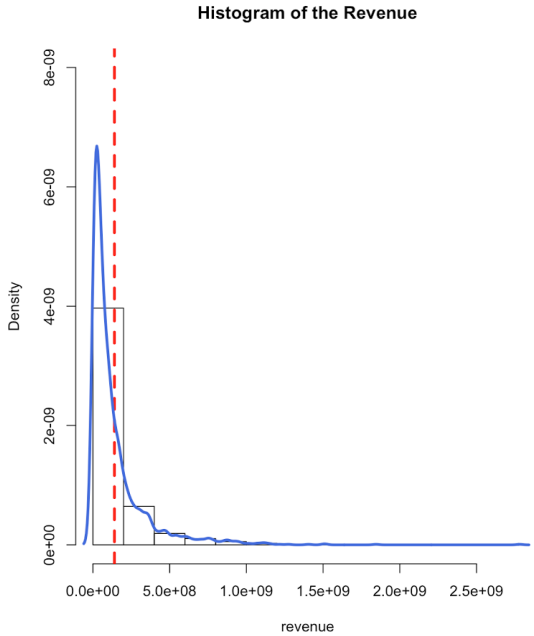
Histograms and box plots

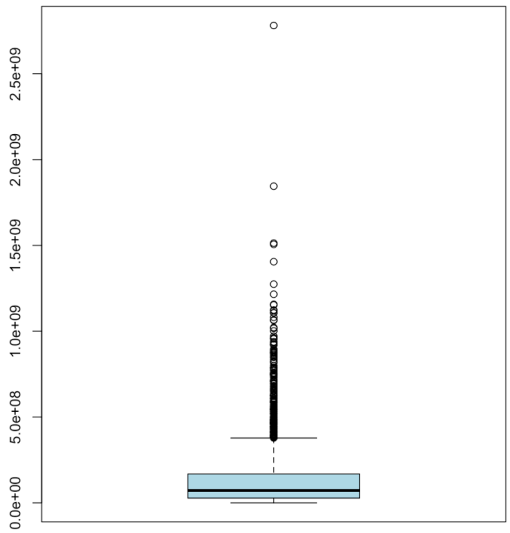
We can make histograms for some variables to see the distribution as well as the box plot to find out the potential outliers.



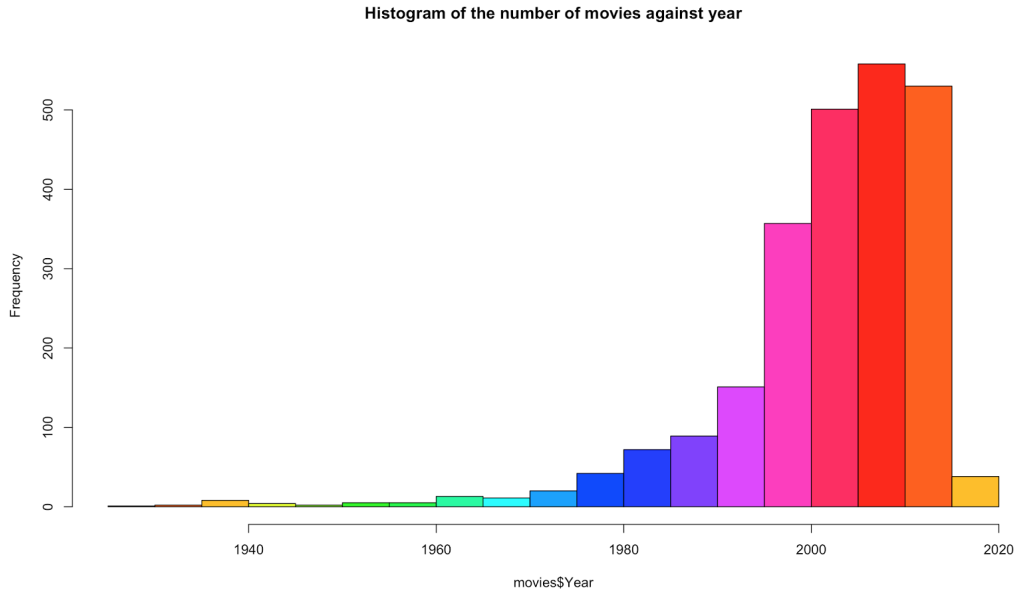


We can find that the score has left skew distribution but close to normal distribution. From the box plot we can also see that there are many outliers in the low score region. If we check the Revenue:





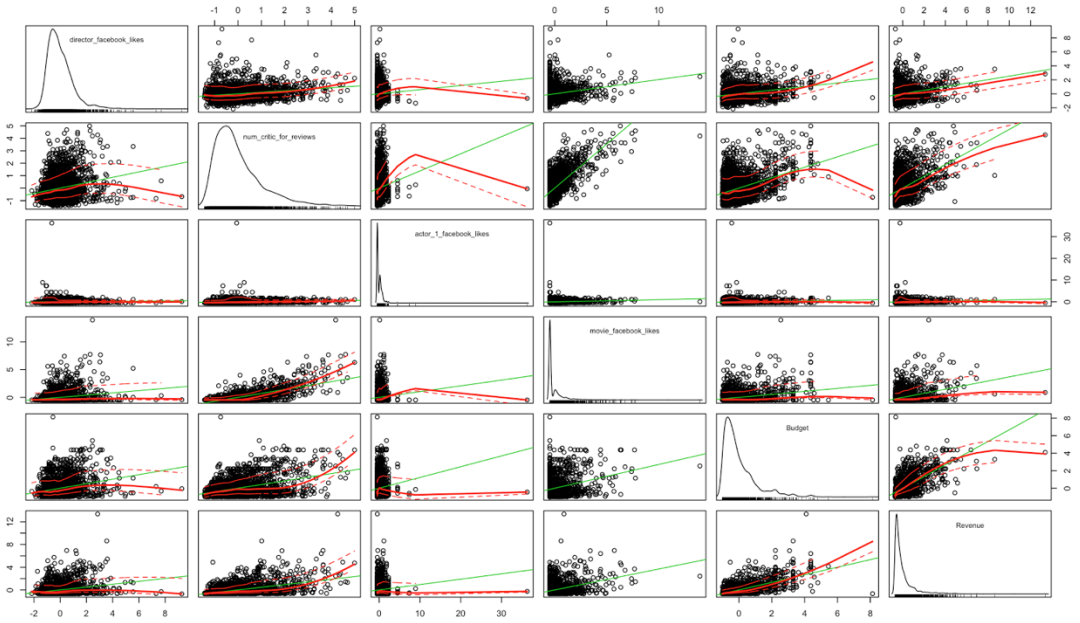
We can find that the Revenue is highly right skewed, and have a lot of outliers in high level revenue. Then it may be useful to take a look at the number of movies released during the period we studied in.



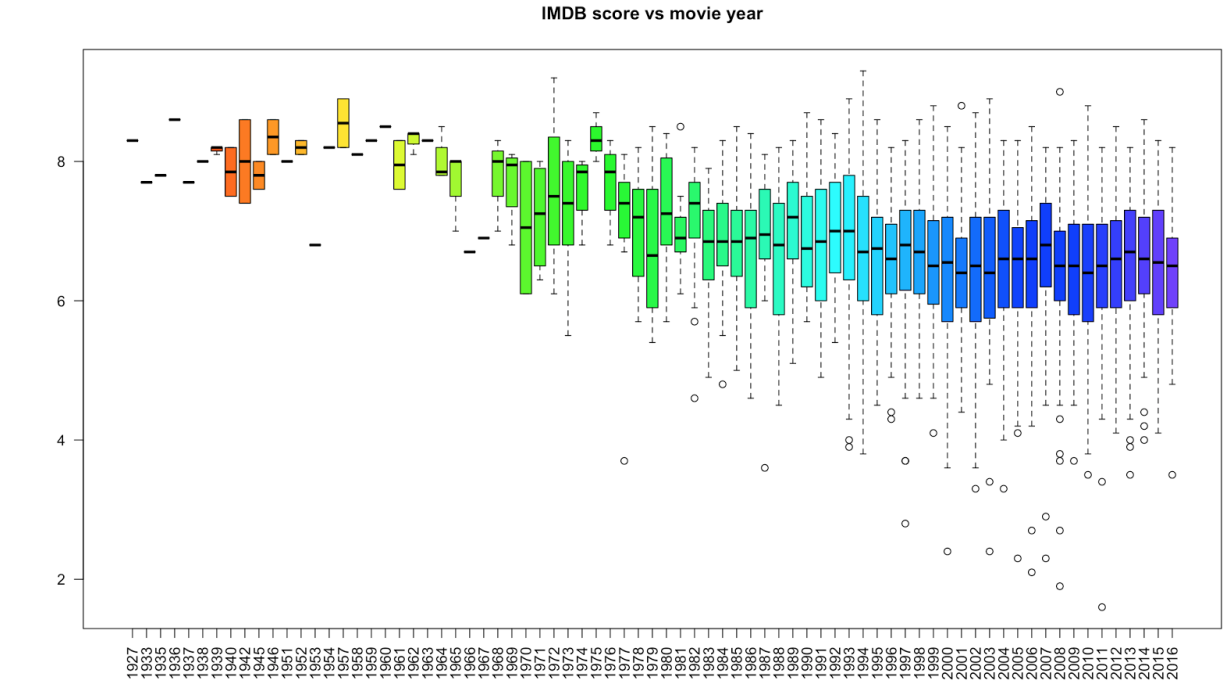
It is obvious that the number of movies that released experienced an explosive growth during recent 30 years. Then what are the relationships between the Revenue and other variables? We can make more analysis.

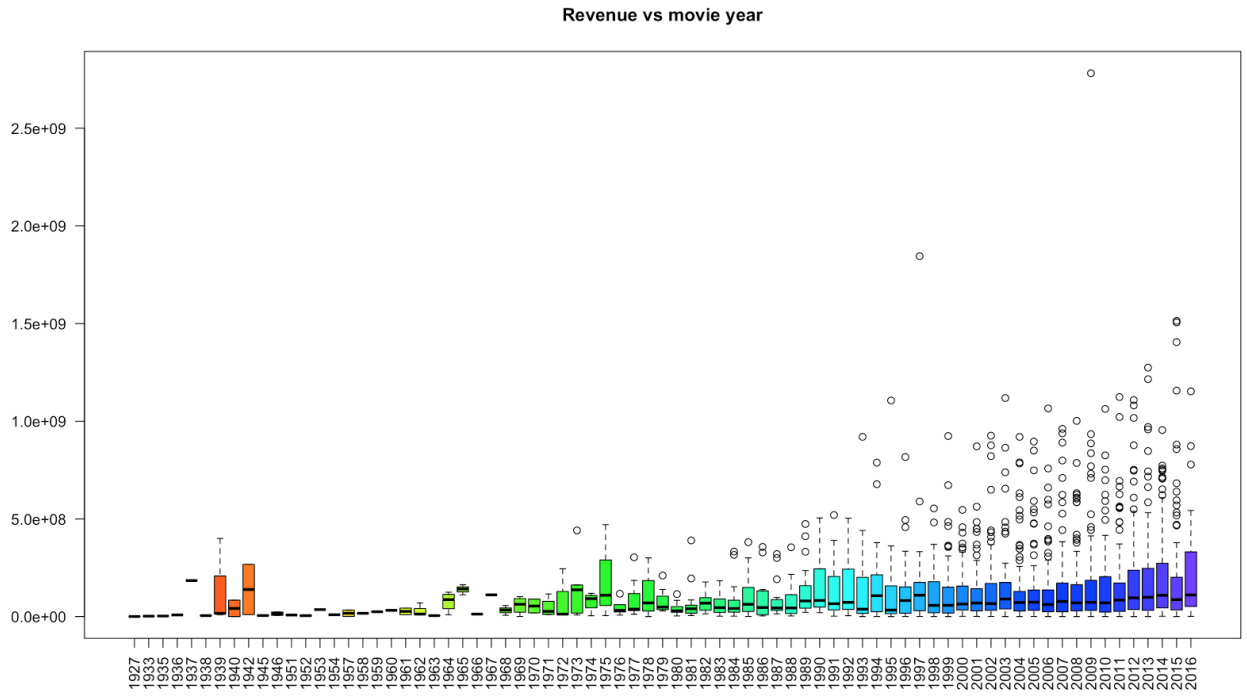
Scatterplot matrix and Comparison plots

First we can take a look at the scatterplot matrix among all numerical variables, since if multicollinearity exists among the variables, then we delete some of the collinear variables.



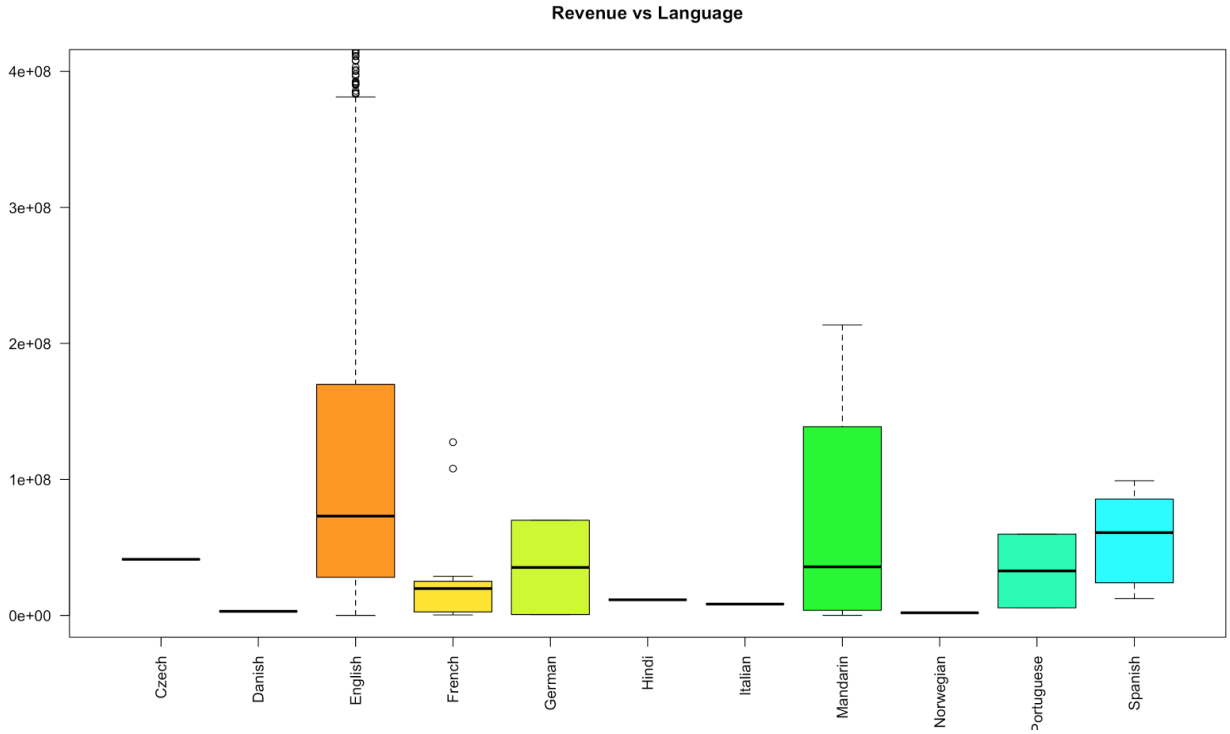
We can find the “number of critics” and “facebook likes for movie” are positively correlated, and the “Budget” and “Revenue” are correlated too. Next, we can make some comparison plots.





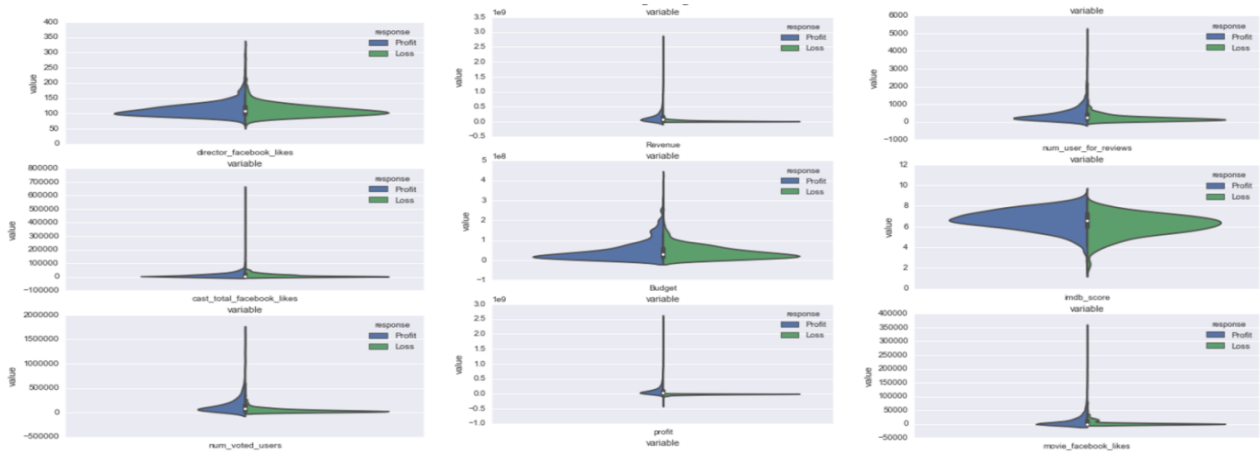
Here the average scores for the movies in different year are almost the same, but there are a lot of movies with much lower scores during recent years. This may due to the high quantity of movie released every year and the improvement of customers’ appreciation level. On the other hand, the revenues of recent years are

increasing rapidly, this tendency can be shown from the higher 3rd quantile and more outliers.



Here we can find most of the movies are made by English and Mandarin, which implies that American and China may be the main country of producing movies. And most of the extreme high revenue movies are made by English.

Univariate Characteristics



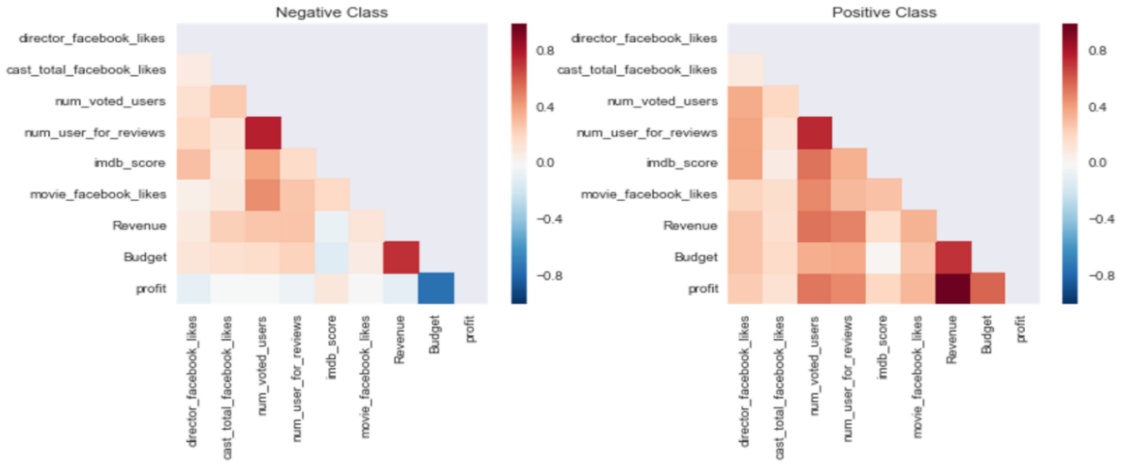
In order to understand better the predictive power of single features, we compare the univariate distributions of the most important features (director's Facebook likes, revenue, number of users for review, cast's total Facebook likes, budget, IMDB score, number of voted users, profit, movie's Facebook likes).

Result: By drawing violin plots, we have comprehensive sense about how different variables distributed

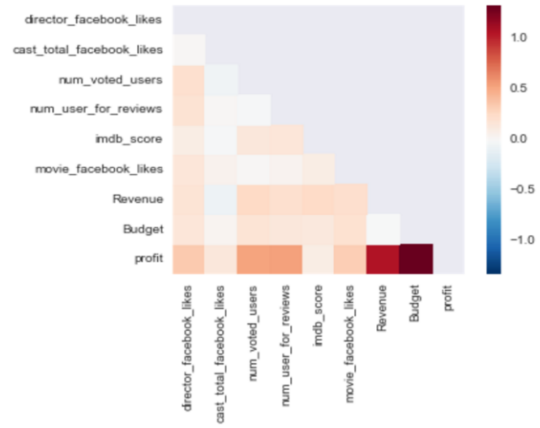
between positive and negative sample. Also, we see that for many of the distribution there is no significant difference between the positive and negative samples. However, the distribution of 'Revenue" between the two group is quite different.

Correlation Heat map

In the previous section, we have seen differences between negative and positive samples for univariate characteristics. We go down the rabbit hole a little further and analyze covariance for the negative and positive samples separately. The darker the color is, the higher correlation it is between the two features.



Result: For both of the samples, the Budget is highly correlated to Revenue.

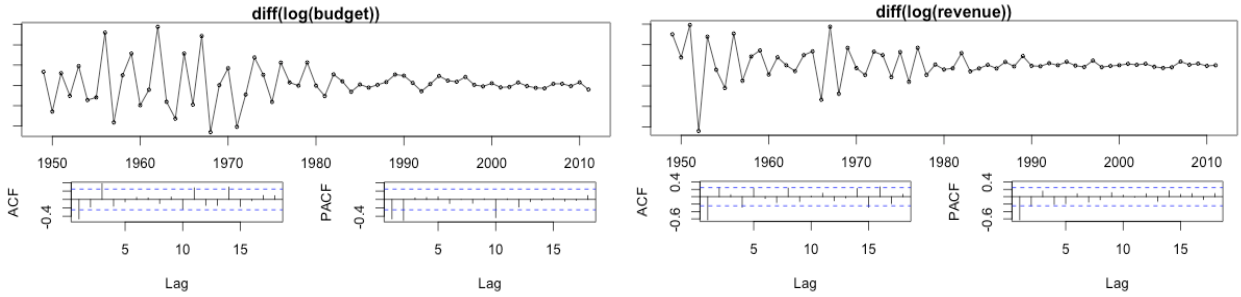


Result: The difference between the two matrices is sparse except for specific feature combinations.

By subtract the covariance of the two sample, we see the difference of the two. We can see that except for the Budget versus profit, the number of reviews in IMDB also have a significant influence on whether the movie is profitable or not.

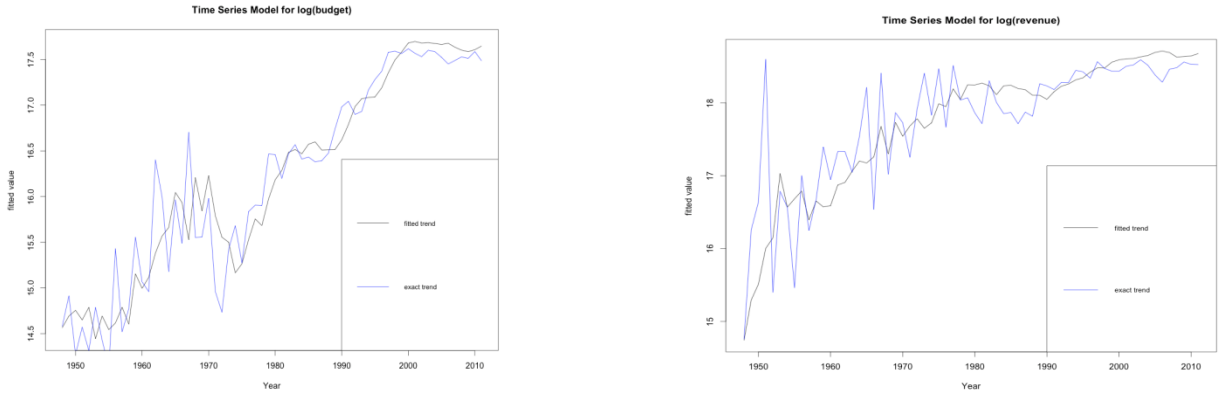
4.2 Time Series Analysis

As historical events, can draw big influence on movies' We averaged the movie budget and movie revenue by year and tried to fit time series model in order to get insights of movie box office.

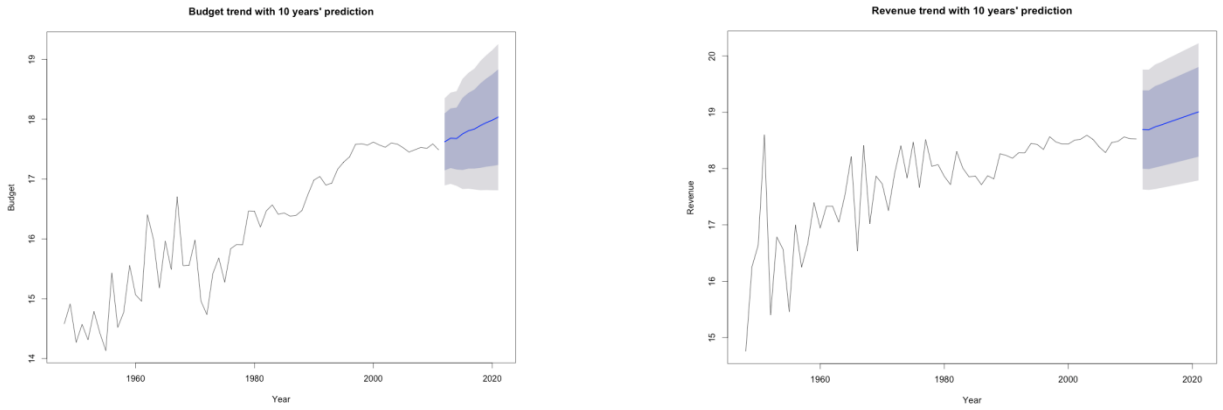


After analyzing the ACF and the PACF of log(budget) and log(revenue), we can see that both of the lines are between the blue bounds, indicating that time series model is a good candidate. Hence, we fitted SARIMA, ARIMA, ARMA models and picked the one with smallest AICC.

Accordingly, the best models are ARIMA(2,1,0) and ARIMA(1,1,1) respectively.



Based on the models fitted, we have also used already collected data of recent 5 years and compared them with the predicted value. The results show that our univariate time series models have a strong prediction power with a good fit.



When testing for the cointegration, both the null hypothesis r=0 and r<1 are rejected with 5% significance level.

4.3 Classification Models

In our project, we applied decision tree, random forest and feed forward neural network to predict following three models:

- ❖ Whether a movie is going to gain or lose money

  As for a movie, the most important thing a producer cares about is whether the movie will help him/her make the money. This model aims to help producer judge whether a movie is profitable beforehand.

- ❖ Whether a movie's revenue is above or below the average revenue level

  From the view of audience, whether a movie's revenue is above or below the annual average is an important criterion to judge a movie's quality, this model aims to help people roughly rate a movie beforehand.

- ❖ Which revenue range does the movie lies in.

  This model aims to give a specific revenue range for a movie before hand. We do this instead of predicting a specific number because it is more reasonable to assess a movie's revenue in a range. Besides, it is easier to define and calculate the accuracy of the model.

Decision Tree Model

- Data Processing

  In order to achieve the prediction goal, we need to train classification models using historical data. The baseline model we selected is decision tree model.

  We first did further data processing based on the merged dataset of 2409 observations. For the label variable revenue, we split it into 9 ranges following the table below.

1	2	3	4	5	6	7	8	9
< $1M	$1M - $10M	$10M - $20M	$20M - $40M	$40M - $65M	$65M - $100M	$100M - $150M	$150M - $200M	> $200M

  The predictors are processed differently according to their different data type.

  a. For the numerical variables, such as 'director_facebook_likes', 'director_facebook_likes' and 'movie_facebook_likes', we kept their original form.

  b. For special numerical variables such as 'Annual Average Revenue' and 'Budget', we transform it into 9

levels in the same way as revenue.

c. For categorical variables that has only one value in each cell, e.g., 'studio' and 'director_name', we transform them into dummy variables.

d. For categorical variables that has multiple values in each cell, e.g., 'genres' and 'actors', we create many bags of words as feature names and assign 1 if the value is included in the corresponding bag of words, otherwise, 0.

As an illustration for the last two transformations. We use some tables to show the processing. For process c, the original data is in the following form, although the real dataset is much larger than it.

Obs	1	2	3	4	5
Studio	BV	Fox	Uni.	Uni.	BCV

After data processing, the variables' names and values are shown below.

Obs	**BV**	**Fox**	**Uni.**	**WB**	**LG/S**
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	1	0	0
5	1	0	0	0	0

For process d, the original data is in the following form.

Obs	**Genre 1**	**Genre 2**	**Genre 3**	**Genre 4**	**Genre 5**	**Gener 6**
1	Action	Adventure	Sci-Fi	NA	NA	NA
2	Action	Adventure	Sci-Fi	NA	NA	NA
3	Action	Adventure	Fantasy	NA	NA	NA
4	Mystery	Thriller	NA	NA	NA	NA
5	Action	Adventure	Sci-Fi	Thriller	Crime	Mystery

After transformation, the columns corresponding to Genres changed to the following form.

Obs	Action	Adventure	Sci-Fi	Fantasy	Mystery	Thriller	Crime	...
1	1	1	1	0	0	0	0	0
2	1	1	1	0	0	0	0	0
3	1	1	0	1	1	0	0	0
4	0	0	0	0	1	1	0	0
5	1	1	1	0	0	1	1	0

As an overview, the predictors used in the following classification models are composed of

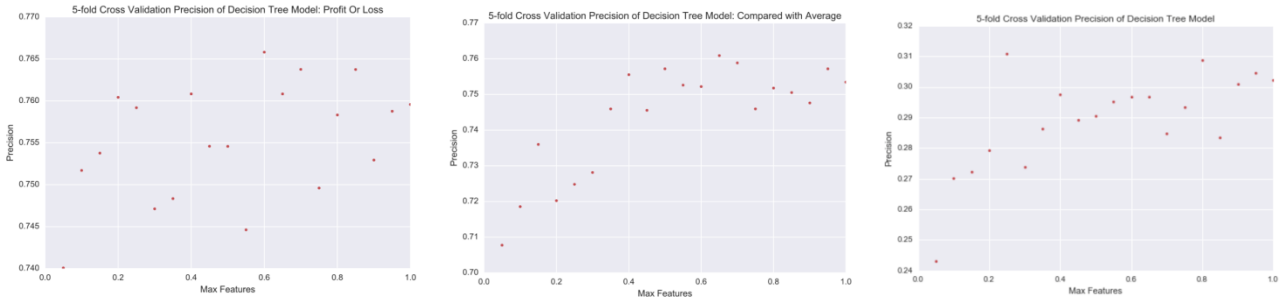
Name	Column indices	Column counts	Name	Column indices	Column counts
director_facebook_likes	1	1	movie_facebook_likes	11	1
num_critic_for_reviews	2	1	Year	12	1
actor_1_facebook_likes	3	1	Budget	13	1
actor_2_facebook_likes	4	1	director_name	14-1091	1078
actor_3_facebook_likes	5	1	genres	1092-1114	23
cast_total_facebook_likes	6	1	keywords	1115-6226	5112
num_voted_users	7	1	actor	6227-11819	5593
num_user_for_reviews	8	1	studio	11820-11918	99
imdb_score	9	1	rating	11820-11918	13
aspect_ratio	10	1	Language	11932-11942	11

- Model parameters

  We tried the decision tree model as a baseline of our prediction models. Decision tree is a popular model for classification in machine learning. It's convenient and the results are easy to interpret. We tuned the parameter max_features in the tree model. Max_featuresmeansthe number of features (as a percentage of total features) to consider when looking for the best split. We tried the range [5%, 100%] with the step of 5%. The splitting criterion is Gini Impurity.

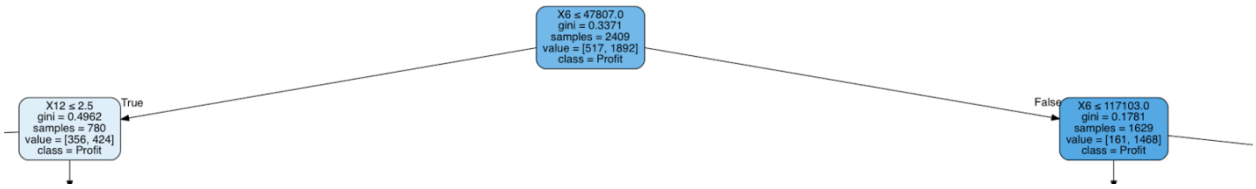
- Result

We used 5-fold Cross Validation Precision (CVP) as the criterion to evaluate the model accuracy and plotted CVP vs. max_features value for the three prediction targets.

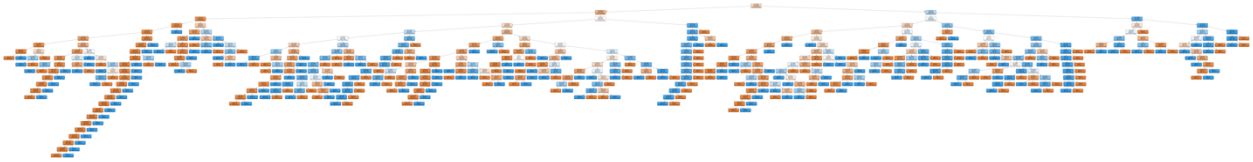


These three plots indicate that the optimal Max_Features is 60%. So we use this value to run the model again to do the predictions. The decision trees for the targets are given below. For visualization purpose, the trees are truncated.

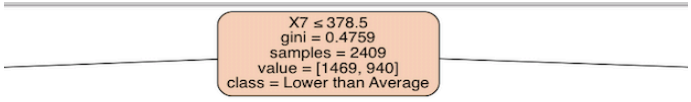
The following plot is the decision tree for prediction of Profit or Loss. Compared the feature index with precious table of all predictors, the first three splits are at features num_voted_users, budget, num_voted_users, which all have dark color in the correlation heat map mentioned before.



Then the tree for prediction of comparison with average is as follows.



Let's zoom in the first split. X7 represents num_user_for_reviews, dark colored in the heat map, too.



The relation between the important splitter and the heat map in EDA park suggests a consistency between

our model and the dataset characteristics.

Random Forest

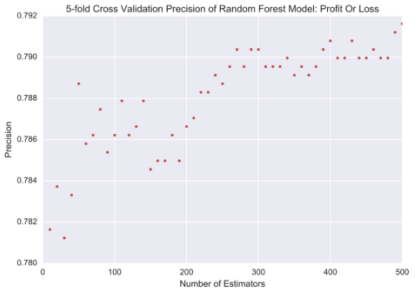
We also tried random forest random forest, which is expected to give higher precision than decision tree models. This is because random forest is an ensemble method, which combines the results from many weak learners.

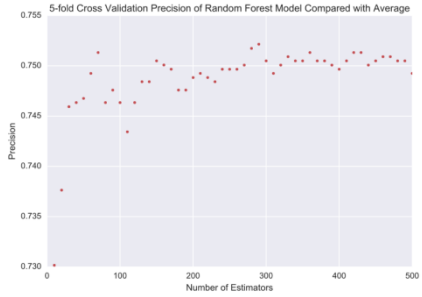
- Data and parameters

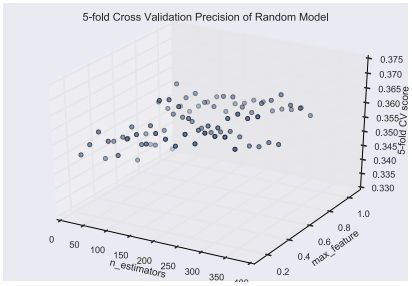
  The input features are the same as that for decision trees. However, besides the max_features in decision tree models, there is a new parameter need to be tuned, n_estimators. N_estimators means the number of estimator trees in the forest. We selected a range of [0, 500] with the step being 10. The criterion is still Gini Impurity and min_samples_split equals to 2, which means the minimum number of samples required to split at an internal node is 2.

- Result

  The following Cross Validation Precision plots suggest the minimum number of estimators should be at lease 200. A 3-D plot of CVP with respect to various number of estimators and various max_features is given for prediction of revenue, consistent with the conclusion of the other two plots.







As expected, our random forest model gives higher precisions compared with decision tree model. A further discussion of both models will be given in a later chapter.

Neural Network

To apply the feed forward Neural Network, we can divide the process into three parts as follow:

- Data manipulation

  In order to make the result more robust and with less variability, we need to change all factor features into binary dummy variables. Then normalize all numerical features. In equation:

$$\tilde{X} = \frac{X - \bar{X}}{sd(X)}$$

In which $\bar{X}$ is the mean and $sd(X)$ is the standard deviation of raw feature.

- Parameter optimization

  After data manipulation, we put data into the network for training. To get higher prediction accuracy, various parameters like number of hidden layers, number of nodes in every layer, stopping criteria, learning rate etc. should be tuned.

- Cross validation

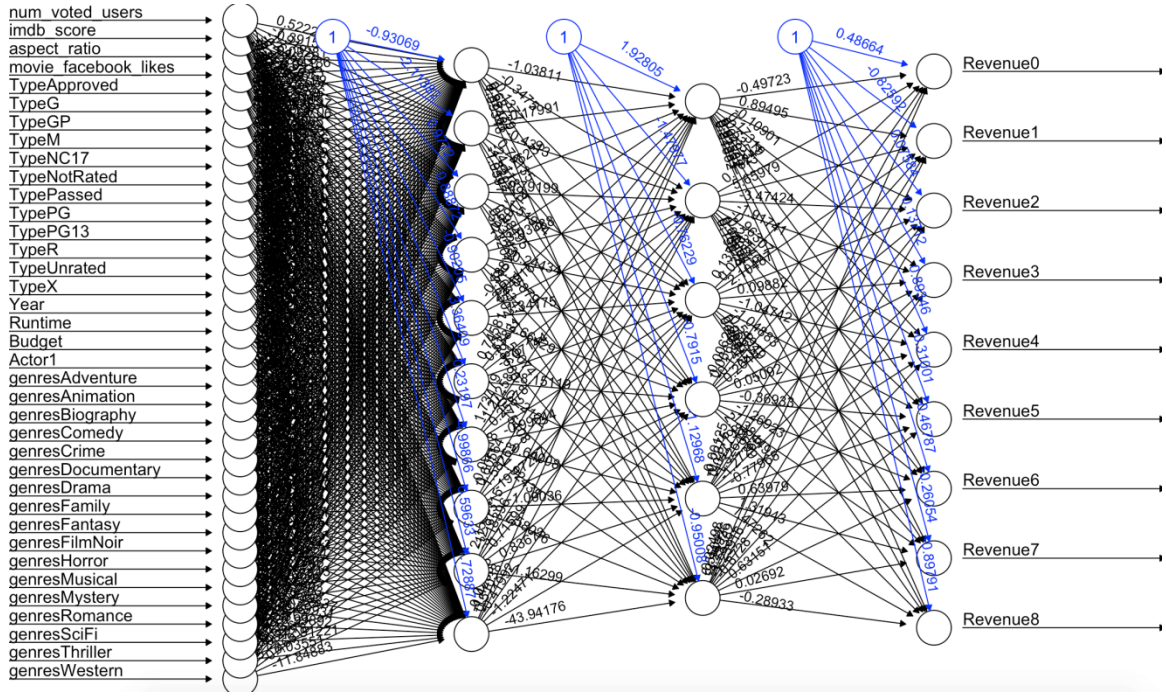
  After parameters are tuned, use cross validation to check the model accuracy, calculate the variance of our model and assess the result.

- Result

  In our case, we implemented neural network using 19 raw features including movies' genres, release year, runtime, budget and K-means clustered actors etc. to train the neural network.

  For the first model, we set two hidden layers with 3 nodes in each layer, threshold for stopping criteria as 1 and learning rate to be 0.1. The cross validation shows that our model's prediction accuracy can get up to 86% at its best and 83% in average. For the second model, we set up parameters mostly the same as the first model, except for changing the learning rate to be 0.05. This time the model's prediction accuracy can get up to 84% at its best and 81% in average. For the last model, we also set two hidden layers but with 10 nodes in first layer and 6 nodes in the second. Stopping threshold as 1.5 and learning rate as 0.5. Under this setting, we get 38% accuracy at best and 35% accuracy in average.

Neural network for model 3



4.4 Cluster Analysis

- Data

  Choice of actors couldhave significant impact on movie's performance, popularity, and revenue. TMDB data includes main cast names ordered by remuneration for each movie. However, using actor names directly not only significantly increases number of factors but also not resistant to changes in actors' performance. Thus, we performed cluster analysis to group the actors in such a way that actors in the same group are more similar to each other than to those in other clusters. All records in TMDB data are used so that an actor's performance is better captured over years. We aimed to group actors by their performance and popularity. Factors considered in this analysis includes an actor's rank, number of movies an actor participated in a year, and the budget of these participated movies;specifically, the factors are this year's rank, previous year's rank, the year before last year's rank,# of movies in a year, % high budget movie% med budget movie, and % low budget movie. Actors rank and movie budget bucket are not explicitly provided in the dataset. Movie's budget bucket is assigned by the percentile in all movies' budgets in a given year. And an actor's rank is the simple average of an actor's all-year rank adjusted by movie budget. See chart 1 for sample calculation.

Chart 1: Sample movie budget bucket and actor rank calculation for cluster analysis

Captain America: Civil War (2016)

Budget	$250,000,000.00
Number of movie in 2016	117
This movie has the highest budget in 2016.	
High budget	1
Med budget	0
Low budget	0

Actor	Rank by Bill	Adjusted Rank
Chris Evans	1	1
Robert Downey Jr.	2	2
Scarlett Johansson	3	3
Sebastian Stan	4	4
Anthony Mackie	5	5
Jeremy Renner	6	6

Chris Evans	rank	high budget	med budget	low budget		
2014	1	1	0	0		
2014	1.5	0	1	0		
2014	1.5	0	1	0		
2014	1.5	0	1	0		
2014	4.5	0	1	0		
2014	2	20%	80%	0%	Count	5
2015	4	1	0	0		
2015	4	100%	0%	0%	Count	1
2016	1	1	0	0		
2016	1	100%	0%	0%	Count	1

a. For each movie, rank actors 1to 6 by remuneration.

b. For each year, separate movies into high/med/low budget bucket by percentile (25% and 75%).

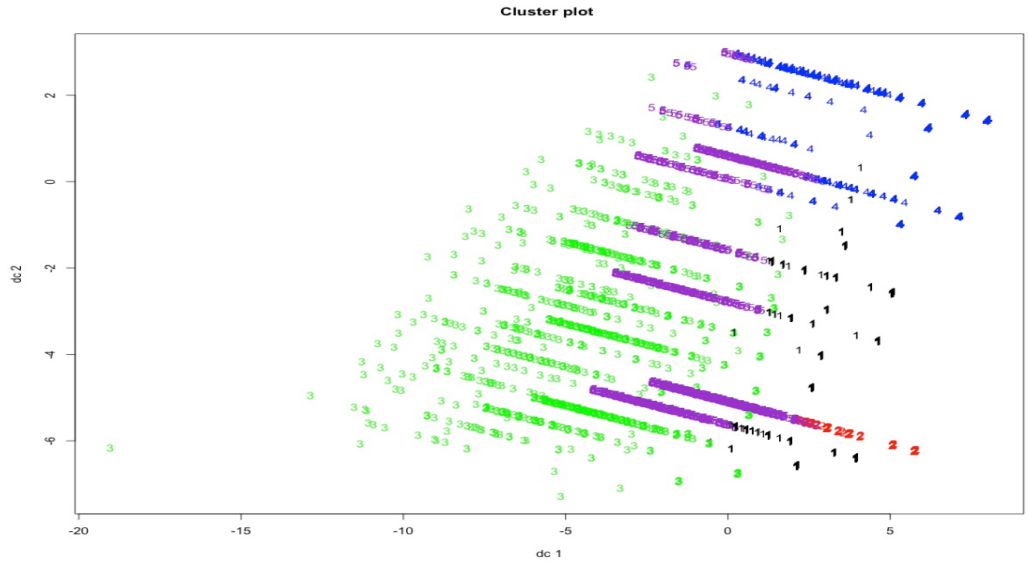
c. Adjust rank by budget:

$Adjusted\ Rank = High_b \cdot x + Med_b \cdot 1.5x + Low_b \cdot 2x,$ where x is original rank from step 1.

d. For each actor and each year, calculate average rank, count number of movies in a year, and calculate percentage of movies in high/med/low budget buckets.

Cluster methodology selected is 5-mean clustering with distance function Euclidean. Cluster results projected to 2 dimension is shown below in Graph 1. Actor clusters could be used as a factor in predicting movie's revenue replacing the original actor variable.

Graph 1: Actor cluster analysis



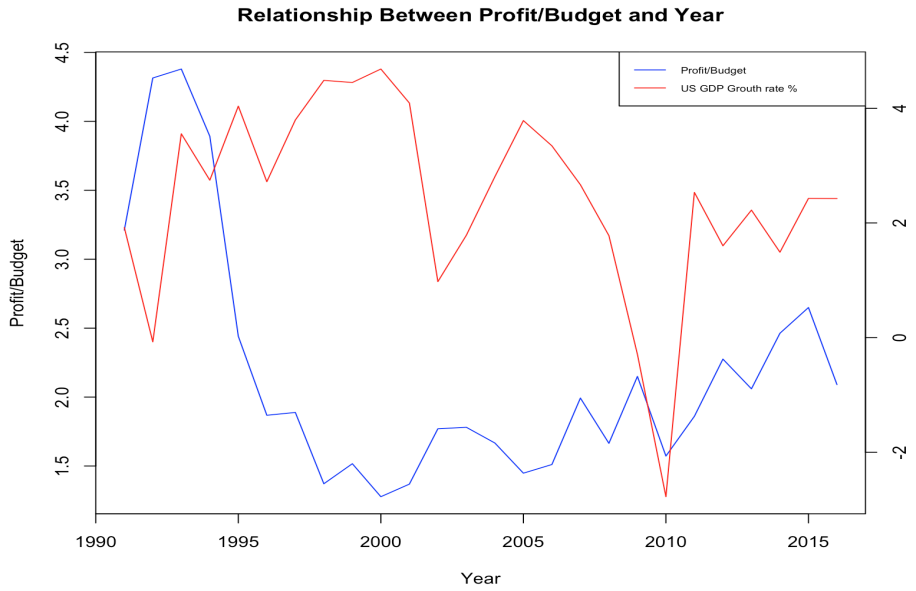
5 Sensitivity Analysis to Validate Any Assumptions

All of our classification models are validated through 5-fold cross validations. And for the time series model, we have used 5 different ways to test for the time series model residuals, including Ljung-Box, McLeod-Li, Turning points, Diff signs and Rank test. Both the results for budget model and revenue model show that the residuals are white noise. They are independent from each other as well as stationary.

6 Conclusion

6.1 Summary of Major Findings

- The variables “Budget” and “Revenue” are positive correlated. it means that high budget is helpful for a movie to have high revenue. And in recent years, the revenue increase as rapidly as budgets, this tendency is also an evidence to prove that here exists positive relationship between budget and revenue.
- The average scores for the movies are almost the same in different year, however, recently here appear some movies with very low scores, it means that people have higher appreciation level. And compare the movies with profit and loss, we can see the number of movies with profit and loss are similar, however, the revenue of them are totally different, the movies with loss would have much lower revenue than the movies with profit. This situation proves that audiences are intelligent to figure out good movies and bad ones.
- Most movies are made for United States and China because the number of English and Mandarin movies is larger than others obviously, however, the movies that have high revenue are mainly made by English, which means that United States has better technique to file a famous movie.
- Like “Lipstick Effect”, we find here exists “Movie Effect”. When facing an economic crisis, people will be more willing to buy less costly luxury goods. Instead of buying expensive fur coats, for example, people are more likely to watch movie. From the graph below, we can see in 1991, United States suffered economic recession, the profit/budget of movie increased a lot and almost got its peak. Then, in 2008, when economic crisis happened, we can see the profit/budget of movie increased obviously.



- At first, we used recent 5 years' data to fit time series model and found the predicted value locate on 80% Confidence Interval so this proves that time series model has good predicted performance.
- Then we used decision trees model, random forest and neural network, the precision table is as below:

	Decision Tree Model	Random Forest	Neural Network
Profit or Loss	76.6%	79.2%	86.0%
Revenue above or below average	76.1%	75.2%	84.0%
Profit Range (10 groups)	30.5%	36.9%	38%

From the table above, we can see Neural Network performed better than other models if we want to predict whether a movie would have profit or loss in the future and its revenue is above or below the average. It can prove us basic information about the box office of a movie, however, if we want to new more specific information such as profit range, random forest performed better. So we give audiences the freedom to select model when they have different goal to predict a movie.

6.2 Limitations of Study

There do exist some limitations in our project, which may be further explored in the future. The first is that some features for the movies in the future may not be valid, which may result in prediction failure. For example, we used actor as a feature, if there is a new movie star comes out that doesn't exist in our data set, this may greatly affect our model. Besides, in the model 3, we didn't consider the effect of inflation rate, which may also influence the prediction result.

7 References

TMDB: https://www.themoviedb.org/movie?page=1&language=en

Box Office Mojo: http://www.boxofficemojo.com/movies/alphabetical.htm?letter=A&p=.htm

Kaggle: https://www.kaggle.com/deepmatrix/imdb-5000-movie-dataset

NYC Data Science Academy: https://blog.nycdatascience.com/student-works/machine-learning/movie-rating-prediction/

8 Appendix

- Code for EDA

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
data1 = pd.read_csv("data_performance_cor.csv")
 X_neg, X_pos = data1[data1['response'] == 0].iloc[:, :-1], data1[data1['response']==1].iloc[:, :-1]
FIGSIZE = (13,4)
_, (ax1, ax2) = plt.subplots(1,2, figsize = FIGSIZE)
MIN_PERIODS = 100
triang_mask = np.zeros((X_pos.shape[1], X_pos.shape[1]))
triang_mask[np.triu_indices_from(triang_mask)] = True
ax1.set_title('Negative Class')
sns.heatmap(X_neg.corr(min_periods = MIN_PERIODS), mask = triang_mask, square=True,    ax = ax1, vmin = -1, vmax
= 1)
ax2.set_title('Positive Class')
sns.heatmap(X_pos.corr(min_periods = MIN_PERIODS), mask = triang_mask, square=True,    ax = ax2, vmin = -1, vmax
= 1)
sns.heatmap(X_pos.corr(min_periods = MIN_PERIODS) -X_neg.corr(min_periods = MIN_PERIODS),
                    mask = triang_mask, square=True)
feature_names                                                                                                    =
['director_facebook_likes','cast_total_facebook_likes','num_voted_users','num_user_for_reviews','imdb_score','movie_fa
cebook_likes','Revenue','Budget','profit']
numeric_cols = pd.read_csv("data_performance_cor_id.csv", nrows = 1).columns.values
imp_idxs = [np.argwhere(feature_name == numeric_cols)[0][0] for feature_name in feature_names]
data = pd.read_csv("data_performance_cor_id.csv",
                        index_col = 0, header = 0, usecols = [0, len(numeric_cols) - 1] + imp_idxs)
data = data[feature_names + ['response']]
BATCH_SIZE = 1
train_batch =[pd.melt(data[data.columns[batch: batch + BATCH_SIZE].append(np.array(['response']))],
                                id_vars = 'response', value_vars = feature_names[batch: batch + BATCH_SIZE])
                    for batch in list(range(0, data.shape[1] - 1, BATCH_SIZE))]
FIGSIZE = (7,30)
_, axs = plt.subplots(len(train_batch), figsize = FIGSIZE)
plt.suptitle('Univariate distributions')
```

```
for data, ax in zip(train_batch, axs):
    sns.violinplot(x = 'variable',    y = 'value', hue = 'response', data = data, ax = ax, split =True)
```

- Code for Time Series Analysis:

```
revenue.budget.time<-read.csv("year.budget.revenue.csv")
library(forecast)
library(TSA)
library(MTS)
budget<-ts(revenue.budget.time$Average.of.Budget[19:82],start = 1948,frequency = 1)
revenue<-ts(revenue.budget.time$Average.of.Revenue[19:82],start = 1948,frequency = 1)
budget.arima<-auto.arima(log(budget))
revenue.arima<-auto.arima(log(revenue))
budget.arima$residuals
revenue.arima$residuals
fit.budget<-Arima(log(budget),order = c(2,1,0),include.drift = TRUE)
fit.revenue<-Arima(log(revenue),order = c(1,1,1),include.drift = TRUE)
tsdisplay(diff(log(budget)))
tsdisplay(diff(log(revenue)))
plot(fitted(fit.budget),ylab="fitted value",xlab="Year",main="Time Series Model for log(budget)")
lines(log(budget),col="blue")
legend("bottomright",legend = c("fitted trend","exact trend"),lty=c(1,1),col = c("black","blue"),cex=0.8)
plot(forecast(fit.budget),xlab="Year",ylab="Budget",main="Budget trend with 10 years' prediction")
plot(fitted(fit.revenue),ylab="fitted value",xlab="Year",main="Time Series Model for log(revenue)")
lines(log(revenue),col="blue")
legend("bottomright",legend = c("fitted trend","exact trend"),lty=c(1,1),col = c("black","blue"),cex=0.8)
plot(forecast(fit.revenue),xlab="Year",ylab="Revenue",main="Revenue trend with 10 years' prediction")
library(urca)
test<-ca.jo(data.frame(budget,revenue),type="trace",ecdet="none",spec="longrun")
summary(test)
```

- Code for Classification (Tree and Random Forest)

```
matplotlib inline
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from collections import Counter
import pydotplus
from io import StringIO
```

```
from IPython.display import display, Image
from sklearn.externals.six import StringIO
import random
import seaborn as sns
import time

sns.set_style('whitegrid')
sns.set(color_codes=True)

# split into 9 levels
def MoneyToCate(input_list):
    out = [];
    for m in input_list:
        if m <= 1000000:
            out.append(0)
        elif m > 1000000 and m <= 10000000:
            out.append(1)
        elif m > 10000000 and m <= 20000000:
            out.append(2)
        elif m > 20000000 and m <= 40000000:
            out.append(3)
        elif m > 40000000 and m <= 65000000:
            out.append(4)
        elif m > 64000000 and m <= 100000000:
            out.append(5)
        elif m > 100000000 and m <= 150000000:
            out.append(6)
        elif m > 150000000 and m <= 200000000:
            out.append(7)
        else:
            out.append(8)
    return(out)

def Bag(input_list, bag_list):
    n = len(bag_list)
    output = []
    for obs in input_list:
        feature = [0]*n
        for word in bag_list:
            if word in obs:
                feature[bag_list.index(word)] = 1
        output.append(feature)
    return output
```

```
table1 = pd.read_excel("Data Merged( Kaggle TMDB Studio, no NAs).xlsx", index_col=0)

genre = np.matrix(table1[['genres.1', 'Unnamed: 11', 'Unnamed: 12', 'Unnamed: 13', 'Unnamed: 14', 'Unnamed: 15']]).tolist()
bag_genre = list(set([val for sublist in genre for val in sublist]))
bag_genre.remove(bag_genre[0])

keywords = np.matrix(table1[['Unnamed: 18', 'Unnamed: 19', 'Unnamed: 20', 'Unnamed: 21', 'Unnamed: 22']]).tolist()
bag_keywords = list(set([val for sublist in keywords for val in sublist]))
bag_keywords.remove(bag_keywords[0])

actor = np.matrix(table1[[8, 9, 10, 11, 12, 13]]).tolist()
bag_actor = list(set([val for sublist in actor for val in sublist]))
bag_actor.remove(bag_actor[0])

table1['Revenue'] = MoneyToCate(list(table1['Revenue']))
table1['Budget'] = MoneyToCate(list(table1['Budget']))
table1['Annual Average Revenue'] = MoneyToCate(list(table1['Annual Average Revenue']))
table1_director_name = pd.get_dummies(table1['director_name']).values
table1_genres = Bag(genre, bag_genre)
table1_keywords = Bag(keywords, bag_keywords)
table1_actor = Bag(actor, bag_actor)
table1_studio = pd.get_dummies(table1['studio']).values
table1_rating = pd.get_dummies(table1['rating']).values
table1_Language = pd.get_dummies(table1['Language']).values
table1 = table1.drop(['genres', 'genres.1', 'Unnamed: 11', 'Unnamed: 12', 'Unnamed: 13', 'Unnamed: 14', 'Unnamed: 15',
                      'plot_keywords', 'Annual Average Budget',
                      'Unnamed: 18', 'Unnamed: 19', 'Unnamed: 20', 'Unnamed: 21', 'Unnamed: 22', 'rating', 'Status',
'Runtime',
                      8, 9, 10, 11, 12, 13, 'director_name', 'studio', 'Language'], axis=1)
table2 = table1.drop('Revenue', axis=1)

Y = table1['Revenue']
X = np.column_stack([table2, table1_director_name, table1_genres, table1_keywords, table1_actor, table1_studio,
                     table1_rating, table1_Language])

# all genres and counts
Allgenres = list([val for sublist in genre for val in sublist])
Allgenres = Counter(Allgenres)
del Allgenres[list(Allgenres.keys())[0]]

clf = RandomForestClassifier(n_estimators=10)
clf = clf.fit(X[100:], Y[100:])
```

```
score_decision = []
for i in range(20):
    clf = tree.DecisionTreeClassifier(max_features=(i+1)/20, random_state=0)
    score = cross_val_score(clf, X, Y, cv=5).mean()
    score_decision.append(score)

max_feature = [(x+1)/20 for x in range(20)]
DecisionTree = pd.DataFrame()
DecisionTree["max_features"] = max_feature
DecisionTree["precision"] = score_decision
DecisionTree.to_csv("Decision Tree Precision.csv")

plt.plot(max_feature, score_decision, 'r.')
plt.xlabel('Max Features')
plt.ylabel('Precision')
plt.title('5-fold Cross Validation Precision of Decision Tree Model')
plt.savefig('DecisionTree.pdf')
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(n_estimators, max_feature, score_decision, 'r.')
ax.set_xlabel('n_estimators')
ax.set_ylabel('max_feature')
ax.set_zlabel('5-fold CV score')
plt.title('5-fold Cross Validation Precision of Random Model')
plt.savefig('RandomForest3d.pdf')
plt.show()

clf = RandomForestClassifier(n_estimators=250, min_samples_split=2, random_state=0)
scores = cross_val_score(clf, X, Y, cv=5)
scores.mean()

clf = ExtraTreesClassifier(n_estimators=220, max_depth=None,
      min_samples_split=2, random_state=0)
scores = cross_val_score(clf, X, Y, cv=5)
scores.mean()
```

- Code for Neural Network

```
#Model 1
library(neuralnet)
sign.neural<-read.csv("/Users/JJason/Documents/Study/summer project/Predict-Box-Office/data/revenue.sign.neural.csv")
```

```
sign.neural<-subset(sign.neural,select = -studio)
sign.neural$num_voted_users = (sign.neural$num_voted_users - mean(sign.neural$num_voted_users))/sd(sign.neural$num_voted_users)
sign.neural$num_user_for_reviews = (sign.neural$num_user_for_reviews - mean(sign.neural$num_user_for_reviews))/sd(sign.neural$num_user_for_reviews)
sign.neural$imdb_score = (sign.neural$imdb_score - mean(sign.neural$imdb_score))/sd(sign.neural$imdb_score)
sign.neural$aspect_ratio = (sign.neural$aspect_ratio - mean(sign.neural$aspect_ratio))/sd(sign.neural$aspect_ratio)
sign.neural$movie_facebook_likes = (sign.neural$movie_facebook_likes - mean(sign.neural$movie_facebook_likes))/sd(sign.neural$movie_facebook_likes)
sign.neural$Runtime = (sign.neural$Runtime - mean(sign.neural$Runtime))/sd(sign.neural$Runtime)
sign.neural$Budget = (sign.neural$Budget - mean(sign.neural$Budget))/sd(sign.neural$Budget)
sign.neural$budget.diff = (sign.neural$budget.diff - mean(sign.neural$budget.diff))/sd(sign.neural$budget.diff)
sign.neural$Year = sign.neural$Year - mean(sign.neural$Year)
index = sample(c(1:2408),1820)
train = sign.neural[index,]
validate = sign.neural[-index,]
train<-model.matrix(~Sign + genres + num_voted_users + imdb_score +
                    aspect_ratio + movie_facebook_likes + Type + Year + Runtime +
                    Budget + Actor1,data=train)
validate = model.matrix(~Sign + genres + num_voted_users + imdb_score +
                        aspect_ratio + movie_facebook_likes + Type + Year + Runtime +
                        Budget + Actor1,data=validate)
neuralnet.sign<-neuralnet(Sign ~genresAdventure + genresAnimation + genresBiography +
                          genresComedy + genresCrime + genresDocumentary + genresDrama +
                          genresFamily + genresFantasy + genresFilmNoir + genresHorror +
                          genresMusical + genresMystery + genresRomance + genresSciFi +
                          genresThriller + genresWestern + num_voted_users +
                          imdb_score + aspect_ratio + movie_facebook_likes + TypeApproved +
                          TypeG + TypeGP + TypeM + TypeNC17 + TypeNotRated + TypePassed +
                          TypePG + TypePG13 + TypeR + TypeUnrated + TypeX + Year +
                          Runtime + Budget + Actor1,data=train,hidden = c(3,3),lifesign ="full", threshold = 1,
learningrate = 0.05)
predict = neuralnet.sign$net.result[[1]]
predict[which(predict >= 0.5)] = 1
predict[which(predict < 0.5)] = 0
sum(train[,2] == predict)/1820
result = compute(neuralnet.sign, validate[,c(-1,-2)])$net.result
result[which(result >= 0.5)] = 1
result[which(result < 0.5)] = 0
sum(result == validate[,2])/588

#Model 2
revenue.neural<-read.csv("/Users/JJason/Documents/Study/summer project/Predict-Box-Office/data/revenue.neural.csv")
revenue.neural<-subset(revenue.neural,select = -studio)
```

```
revenue.neural$Revenue[which(revenue.neural$Revenue - revenue.neural$Budget >= 0)] = 1
revenue.neural$Revenue[which(revenue.neural$Revenue != 1)] = 0
revenue.neural$num_voted_users = (revenue.neural$num_voted_users -
mean(revenue.neural$num_voted_users))/sd(revenue.neural$num_voted_users)
revenue.neural$num_user_for_reviews = (revenue.neural$num_user_for_reviews -
mean(revenue.neural$num_user_for_reviews))/sd(revenue.neural$num_user_for_reviews)
revenue.neural$imdb_score = (revenue.neural$imdb_score -
mean(revenue.neural$imdb_score))/sd(revenue.neural$imdb_score)
revenue.neural$aspect_ratio = (revenue.neural$aspect_ratio -
mean(revenue.neural$aspect_ratio))/sd(revenue.neural$aspect_ratio)
revenue.neural$movie_facebook_likes = (revenue.neural$movie_facebook_likes -
mean(revenue.neural$movie_facebook_likes))/sd(revenue.neural$movie_facebook_likes)
revenue.neural$Runtime = (revenue.neural$Runtime - mean(revenue.neural$Runtime))/sd(revenue.neural$Runtime)
revenue.neural$Budget = (revenue.neural$Budget - mean(revenue.neural$Budget))/sd(revenue.neural$Budget)
revenue.neural$budget.diff = (revenue.neural$budget.diff -
mean(revenue.neural$budget.diff))/sd(revenue.neural$budget.diff)
revenue.neural$Year = revenue.neural$Year - mean(revenue.neural$Year)
index = sample(c(1:2408),1820)
train = revenue.neural[index,]
validate = revenue.neural[-index,]
train<-model.matrix(~Revenue + num_voted_users + imdb_score +
                    aspect_ratio + movie_facebook_likes + Type + Year + Runtime +
                    Budget + Actor1 + genres,data=train)
validate<-model.matrix(~Revenue + num_voted_users + imdb_score +
                       aspect_ratio + movie_facebook_likes + Type + Year + Runtime +
                       Budget + Actor1 + genres,data=validate)
neuralnet.revenue<-neuralnet(Revenue ~
                             num_voted_users +
                             imdb_score + aspect_ratio + movie_facebook_likes + TypeApproved +
                             TypeG + TypeGP + TypeM + TypeNC17 + TypeNotRated + TypePassed +
                             TypePG + TypePG13 + TypeR + TypeUnrated + TypeX + Year +
                             Runtime + Budget + Actor1 + genresAdventure + genresAnimation +
genresBiography +
                             genresComedy + genresCrime + genresDocumentary + genresDrama +
                             genresFamily + genresFantasy + genresFilmNoir + genresHorror +
                             genresMusical + genresMystery + genresRomance + genresSciFi +
                             genresThriller + genresWestern,data=train,hidden = c(3,3),lifesign
="full",threshold = 1,learningrate = 0.1)
predict = neuralnet.revenue$net.result[[1]]
predict[which(predict >= 0.5)] = 1
predict[which(predict < 0.5)] = 0
sum(train[,2] == predict)/1820
result = compute(neuralnet.revenue, validate[,c(-1,-2)])$net.result
result[which(result >= 0.5)] = 1
```

```
result[which(result < 0.5)] = 0
sum(result == validate[,2])/588
#Model 3
revenue.neural<-read.csv("/Users/JJason/Documents/Study/summer project/Predict-Box-Office/data/revenue.neural.csv")
revenue.neural<-subset(revenue.neural,select = -studio)
revenue.neural$Revenue0 = 0
revenue.neural$Revenue1 = 0
revenue.neural$Revenue2 = 0
revenue.neural$Revenue3 = 0
revenue.neural$Revenue4 = 0
revenue.neural$Revenue5 = 0
revenue.neural$Revenue6 = 0
revenue.neural$Revenue7 = 0
revenue.neural$Revenue8 = 0
for(m in 1:nrow(revenue.neural)){
   if (revenue.neural$Revenue[m] <= 1000000)
      revenue.neural$Revenue0[m] = 1
   else if (revenue.neural$Revenue[m] > 1000000 & revenue.neural$Revenue[m] <= 10000000)
      revenue.neural$Revenue1[m] = 1
   else if (revenue.neural$Revenue[m] > 10000000 & revenue.neural$Revenue[m] <= 20000000)
      revenue.neural$Revenue2[m] = 1
   else if (revenue.neural$Revenue[m] > 20000000 & revenue.neural$Revenue[m] <= 40000000)
      revenue.neural$Revenue3[m] = 1
   else if (revenue.neural$Revenue[m] > 40000000 & revenue.neural$Revenue[m] <= 65000000)
      revenue.neural$Revenue4[m] = 1
   else if (revenue.neural$Revenue[m] > 64000000 & revenue.neural$Revenue[m] <= 100000000)
      revenue.neural$Revenue5[m] = 1
   else if (revenue.neural$Revenue[m] > 100000000 & revenue.neural$Revenue[m] <= 150000000)
      revenue.neural$Revenue6[m] = 1
   else if (revenue.neural$Revenue[m] > 150000000 & revenue.neural$Revenue[m] <= 200000000)
      revenue.neural$Revenue7[m] = 1
   else
      revenue.neural$Revenue8[m] = 1
}
revenue.neural$num_voted_users = (revenue.neural$num_voted_users -
mean(revenue.neural$num_voted_users))/sd(revenue.neural$num_voted_users)
revenue.neural$num_user_for_reviews = (revenue.neural$num_user_for_reviews -
mean(revenue.neural$num_user_for_reviews))/sd(revenue.neural$num_user_for_reviews)
revenue.neural$imdb_score = (revenue.neural$imdb_score -
mean(revenue.neural$imdb_score))/sd(revenue.neural$imdb_score)
revenue.neural$aspect_ratio = (revenue.neural$aspect_ratio -
mean(revenue.neural$aspect_ratio))/sd(revenue.neural$aspect_ratio)
revenue.neural$movie_facebook_likes = (revenue.neural$movie_facebook_likes -
mean(revenue.neural$movie_facebook_likes))/sd(revenue.neural$movie_facebook_likes)
```

```
revenue.neural$Runtime = (revenue.neural$Runtime - mean(revenue.neural$Runtime))/sd(revenue.neural$Runtime)
revenue.neural$Budget = (revenue.neural$Budget - mean(revenue.neural$Budget))/sd(revenue.neural$Budget)
revenue.neural$budget.diff = (revenue.neural$budget.diff - mean(revenue.neural$budget.diff))/sd(revenue.neural$budget.diff)
revenue.neural$Year = revenue.neural$Year - mean(revenue.neural$Year)
index = sample(c(1:2408),1820)
train = revenue.neural[index,]
validate = revenue.neural[-index,]
train<-model.matrix(~Revenue0 + Revenue1 + Revenue2 + Revenue3 + Revenue4 + Revenue5 + Revenue6 + Revenue7 + Revenue8    + num_voted_users + imdb_score +
                                aspect_ratio + movie_facebook_likes + Type + Year + Runtime +
                                Budget + Actor1 + Actor2 + Actor3 + Actor4 +
                                Actor5 + Actor6 + genres,data=train)
validate<-model.matrix(~Revenue0 + Revenue1 + Revenue2 + Revenue3 + Revenue4 + Revenue5 + Revenue6 + Revenue7 + Revenue8    + num_voted_users + imdb_score +
                                    aspect_ratio + movie_facebook_likes + Type + Year + Runtime +
                                    Budget + Actor1 + genres,data=validate)

neuralnet.revenue<-neuralnet(Revenue0 + Revenue1 + Revenue2 + Revenue3 + Revenue4 + Revenue5 + Revenue6 + Revenue7 + Revenue8 ~
                                        num_voted_users +
                                        imdb_score + aspect_ratio + movie_facebook_likes + TypeApproved +
                                        TypeG + TypeGP + TypeM + TypeNC17 + TypeNotRated + TypePassed +
                                        TypePG + TypePG13 + TypeR + TypeUnrated + TypeX + Year +
                                        Runtime + Budget + Actor1 + genresAdventure + genresAnimation + genresBiography +
                                        genresComedy + genresCrime + genresDocumentary + genresDrama +
                                        genresFamily + genresFantasy + genresFilmNoir + genresHorror +
                                        genresMusical + genresMystery + genresRomance + genresSciFi +
                                        genresThriller + genresWestern,data=train,hidden = c(10,6),lifesign ="full",threshold = 1.5,learningrate = 0.5)
predict = neuralnet.revenue$net.result[[1]]
aa = apply(predict,1,max)
predict = predict/aa
predict[predict!=1] = 0
error = predict - train[,c(2:10)]
b = apply(abs(error),1,sum)
table(b)/1820
result = compute(neuralnet.revenue, validate[,-c(1:10)])$net.result
aa = apply(result,1,max)
result = result/aa
result[result!=1] = 0
error = result - validate[,c(2:10)]
b = apply(abs(error),1,sum)
```

```
table(b)/588
plot(neuralnet.revenue)
```

- Test of the Assumption (Time Series Model)

```
library(itsmr)
test(budget.arima$residuals)
test(revenue.arima$residuals)
```

- Code of the Cluster Analysis

```
# input data rank1.csv
# output data actor.rds
setwd("/Users/S7/Dropbox/5291 ADA/Project/data")

library(plyr)
library(sqldf)

rank=read.csv("rank1.csv",header=T)
rank1<-rank[,c(1:6)]

rank2<-rank[,c(7:12)]
rank3<-rank[,c(13:18)]
rank4<-rank[,c(19:24)]
rank5<-rank[,c(25:30)]
rank6<-rank[,c(31:36)]
names(rank1)
names(rank2)
names(rank3)
names(rank4)
names(rank5)
names(rank6)
setnames(rank2,old=c("title.year.1", "actor.1" ,  "rank.1"  ,  "high.1"  ,  "med.1"  ,  "low.1"),
         new=c("title.year" ,"actor"  ,  "rank"  ,  "high"  ,  "med"  ,  "low"))
setnames(rank3,old=c("title.year.2" ,"actor.2"  ,  "rank.2"  ,  "high.2"  ,  "med.2"  ,  "low.2"),
         new=c("title.year" ,"actor"  ,  "rank"  ,  "high"  ,  "med"  ,  "low"))
setnames(rank4,old=c( "title.year.3", "actor.3"  ,  "rank.3"  ,  "high.3"  ,  "med.3"  ,  "low.3"),
         new=c("title.year" ,"actor"  ,  "rank"  ,  "high"  ,  "med"  ,  "low"))
setnames(rank5,old=c("title.year.4","actor.4"  ,  "rank.4"  ,  "high.4"  ,  "med.4"  ,  "low.4" ),
         new=c("title.year" ,"actor"  ,  "rank"  ,  "high"  ,  "med"  ,  "low"))
setnames(rank6,old=c("title.year.5" ,"actor.5"  ,  "rank.5"  ,  "high.5"  ,  "med.5"  ,  "low.5"),
         new=c("title.year" ,"actor"  ,  "rank"  ,  "high"  ,  "med"  ,  "low"))

actor<-rbind(rank1,rank2, rank3, rank4, rank5, rank6)
names(actor)
```

```
setnames(actor,old=c("title.year"),new=c("year"))
actor1<-sqldf("select actor,year, count(*) as count, sum(rank)/count(*) as rank,sum(high) as high,sum(med) as med,sum(low) as low
                from actor
               group by actor, year")
actor1$high.rate<-actor1$high/actor1$count
actor1$med.rate<-actor1$med/actor1$count
actor1$low.rate<-actor1$low/actor1$count
actor2<-actor1[-c(1:86,27838:27925,52776:52816),]

save(actor2, file="actor.rds")

#load(file="actor.rds")

#input data actor.rds
#output data actor_cluster.rds

library(data.table)
library(cluster)
library(fpc)
#read in raw data
#setwd("/2 ADA/Project/data")
setwd("/Users/Caren TY/Dropbox/5291 ADA/Project/data")
load(file="actor.rds")

actor3<-actor2[,c(1,2,4)]
setnames(actor3,old="rank",new="ranklag1")
actor3$year=actor3$year+1

actor4<-actor2[,c(1,2,4)]
setnames(actor4,old="rank",new="ranklag2")
actor4$year=actor4$year+2

actor5<-merge(actor2,actor3,by=c("actor", "year"),all.x=TRUE)
save<-merge(actor5,actor4,by=c("actor", "year"),all.x=TRUE)

actor6<-save

actor6$delta1=(actor6$rank-actor6$ranklag1)/actor6$ranklag1
actor6$delta2=(actor6$ranklag1-actor6$ranklag2)/actor6$ranklag2
actor6$delta1[is.na(actor6$delta1)] <- .24
actor6$delta2[is.na(actor6$delta2)] <- .24
```

```
save2<-actor6
actor6<-save2

actor6$ranklag1[is.na(actor6$ranklag1)] <- actor6$rank[is.na(actor6$ranklag1)]/1.24
actor6$ranklag2[is.na(actor6$ranklag2)] <- actor6$ranklag1[is.na(actor6$ranklag2)]/1.24

actor7<-actor6[,-c(1,2,3,5,6,7)]
actor8<-scale(actor7)

# Determine number of clusters
#wss <- (nrow(actor8)-1)*sum(apply(actor8,2,var))
#for (i in 2:539988) wss[i] <- sum(kmeans(actor8, centers=i)$withinss)
#plot(1:539988, wss, type="b", xlab="Number of Clusters", ylab="Within groups sum of squares")

set.seed(4)
# K-Means Cluster Analysis
fit <- kmeans(actor8, 5) # 5 cluster solution
# get cluster means
aggregate(actor7,by=list(fit$cluster),FUN=mean)
# append cluster assignment
actor9 <- data.frame(actor6, fit$cluster)
a<-actor9[,c(4,8,9,10,11,12)]
plotcluster(a,actor9$fit.cluster, main = "Cluster plot")


#fit1 <- kmeans(actor8, 6)
#aggregate(actor7,by=list(fit1$cluster),FUN=mean)

#fit2 <- kmeans(actor8, 4)
#aggregate(actor7,by=list(fit2$cluster),FUN=mean)


#fit1 <- pam(actor8, 5) # 5 cluster solution
#library(fpc)
#fit <- pamk(actor8, krange=3:6) # 5 cluster solution

#d <- dist(actor8, method = "euclidean") # distance matrix
#fit.h <- hclust(d, method="ward")
#plot(fit) # display dendogram
#groups <- cutree(fit.h, k=5) # cut tree into 5 clusters
# draw dendogram with red borders around the 5 clusters
#rect.hclust(fit.h, k=5, border="red")
actor6[is.na(actor6)] <- 12
actor60<-actor6
```

```
for(i in 1:89999) {
   if (actor60$ranklag1[i]<-13){actor60$ranklag1[i]=actor60$ranklag2[i]}
}
for(i in 1:89999) {
   if (actor60$ranklag2[i]<-13){actor60$ranklag2[i]=actor60$ranklag1[i]}
}

actor70<-actor60[,-c(1,2,5,6,7)]
actor80<-scale(actor70)

set.seed(1)
# K-Means Cluster Analysis
fit0 <- kmeans(actor80, 5) # 5 cluster solution
# get cluster means
aggregate(actor70,by=list(fit0$cluster),FUN=mean)
#library(fpc)
plotcluster(actor80,fit0$cluster, main = "Cluster plot with count    ")
# append cluster assignment
actor90 <- data.frame(actor60, fit0$cluster)

save(actor9, file="actor_cluster.rds")
load(file="actor_cluster.rds")
plotcluster(actor70,fit0$cluster, main = "Cluster plot with count    ")
```

- Code for the comparison of Annual Average Revenue and GDP

```
data<-read.csv("Data Merged( Kaggle TMDB Studio, no NAs).csv")
library(vcd)
library(grid)
library(plyr)
library(ggplot2)
head(data)
attach(data)
PoB<-(data$Revenue-data$Budget)/data$Budget
data[which(PoB==9919),]
data.year.mean<-aggregate(data[,c(33,35)],list(data$Year),mean)
class(data.year.mean)
data.year.PoB<-(data.year.mean[,3]-data.year.mean[,2])/data.year.mean[,2]

data.bigtest<-read.csv("TMDB_split.csv")

data2 <- data[, c("Year", "Budget", "Revenue")]
data2["Year"] <- sapply(data2["Year"], as.character)
data2["Year"] <- sapply(data2["Year"], as.numeric)
data3 <- data2[complete.cases(data2),]
```

```
data3
data4<-data3[which(data3[,2]>1000 & data3[,3]>1000),]

data4.year.mean<-aggregate(data4[,c(2,3)],list(data4$Year),mean)
data4.year.PoB<-(data4.year.mean[,3]-data4.year.mean[,2])/data4.year.mean[,2]
data4.combined<-cbind(data4.year.PoB,count.year[,1])
data4.combined

plot(data4.combined[50:75]~data4.combined[c(50:75),2],type="l",col="Blue",xlab = "Year",ylab = "Profit/Budget")
title(main = "Relationship Between Profit/Budget and Year")
usgdp<-c(2.3, 6.1, 4.4, 5.8, 6.4, 6.5, 2.5, 4.8, 3.1, 3.206807257, 3.295476728, 5.263262776, 5.643124848, -
0.517154562, -0.197678537, 5.386090051, 4.608597407, 5.561684929, 3.17569075, -0.244596225,
     2.594470388, -1.910891068, 4.632457181, 7.259086959, 4.238737521, 3.511614499, 3.461747692,
     4.203971979, 3.680524033, 1.919370297, -0.074084531, 3.555396148, 2.745856719, 4.037643425,
     2.718975789, 3.795881229, 4.487026493, 4.449910963, 4.685199608, 4.092176449, 0.975981834,
     1.786127687, 2.806775956, 3.78574285, 3.345216063, 2.666625826, 1.77857024, -0.291621459, -
2.775529574, 2.531920616, 1.601454672, 2.224030854, 1.489524949, 2.427795636, 2.425970526)

par(new = T)
with(d.df,plot(usgdp[30:55],type = "l",pch=16, axes=F, col="Red",xlab=NA, ylab=NA, cex=1.2))
axis(side = 4)
legend("topright",legend=c("Profit/Budget","US GDP Grouth rate %"),lty=c(1,1),col=c("blue","red"),cex=0.6)
```