IMDB Data Analysis

Qingsong Wang

May 6th, 2019

1. Introduction

Movies, also known as films, are a type of visual communication which uses moving pictures and sound to tell stories or teach people something. Nowadays, movies are appealing a rapidly-increasing number of audience. There are abundant data resource, recording various kinds of features of thousands of movies. In this project, we aim at gaining comprehensive knowledge about movies, via analyzing related datasets. As listed below, three main goals are set here:

- Have a deep understanding about the dataset.
- Predict the IMDB score of a movie based on several variables.
- Determine whether a movie is profitable, or in other words, worth investing.

We want to achieve our goals by thorough explanatory data analysis as well as building popular machine learning models to predict or classify. Such models include support vector machine, random forest, neural network and etc. As a data analysis report, we try to show to the audience as many vivid results as possible, so the main focus lies on the the EDA part, while we still spend enough time on modelling.

The report is organized as follows: section 1 is a brief introduction; section 2 contains data description and pre-processing; we begin our explanatory analysis in section 3, where the most visualization results are shown; different models for prediction and classification are introduced and trained in section 4 with parameters tuned properly; we close this report with summary and discussion in section 5.

2. Data Description

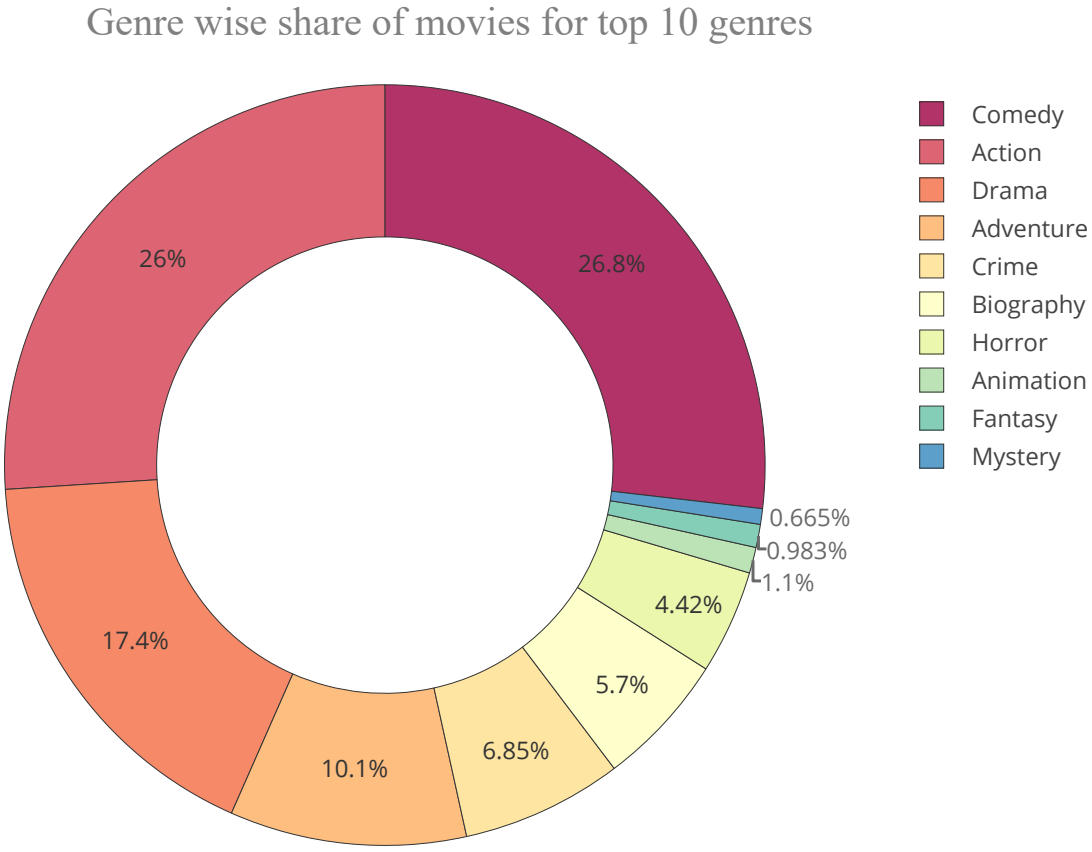
After comparison, we decide to collect two datasets from Kaggle (https://www.kaggle.com/), and combine them. The reason why we choose these two is that they are complete and contain more intereting features. The combined dataset contains nearly 5000 movies as well as 30 features, and after removing NA rows 3636 pieces are left. The meanings of some variable are listed below, while the left are intuitive.

Variable	Description
aspect_ratio	ratio of length to width of screen
budget	the budget of the movie in USD
color	logical, whether is colorful or black-white
facenum_in_posters	number of faces in the poster
genres	movie type e.g. thriller, drama, etc
gross	the revenue of movies in USD
imdb_score	the audience rating
runtime	the duration of movies

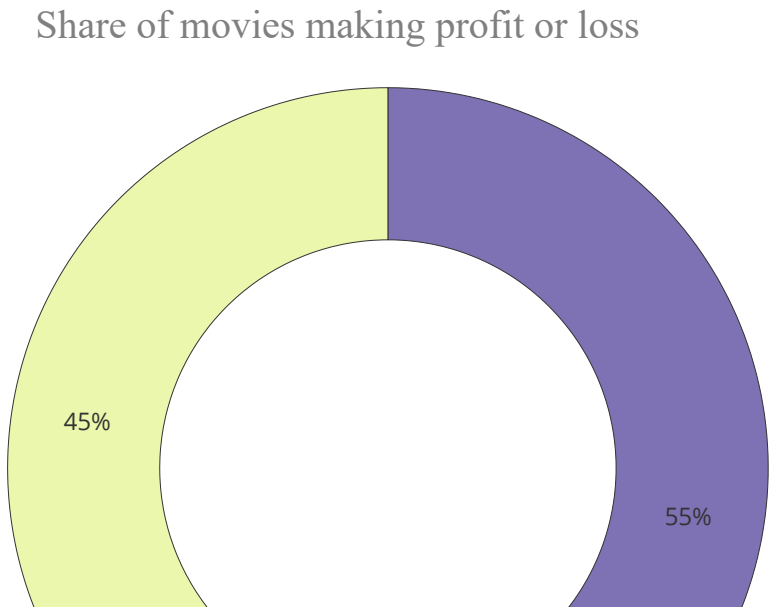
For simplicity, from now on we only analyze 3494 movies in English, to remove the influence of language. And we add two columns, one named **if_profit**, which is a logical value indicating whether a movie is profitable, the other named **return_rate**, i.e. the ratio of net revenue to budget. Notice **return_rate** is more reasonable to measure the revenue, considering inflation. Then we can begin our formal analysis.

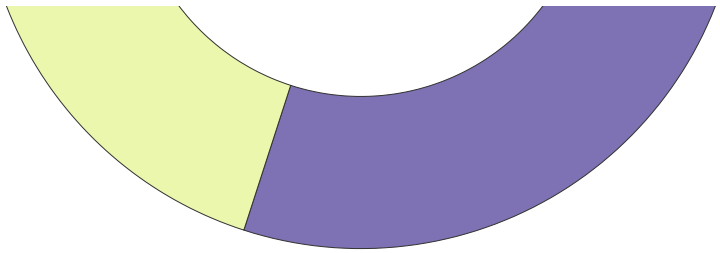
3. Explanatory Data Analysis

First, we want to check which genres most movies belong to. It's easy to find that comedy and action movies rank the first two with share 26.8%, 26% respectively.

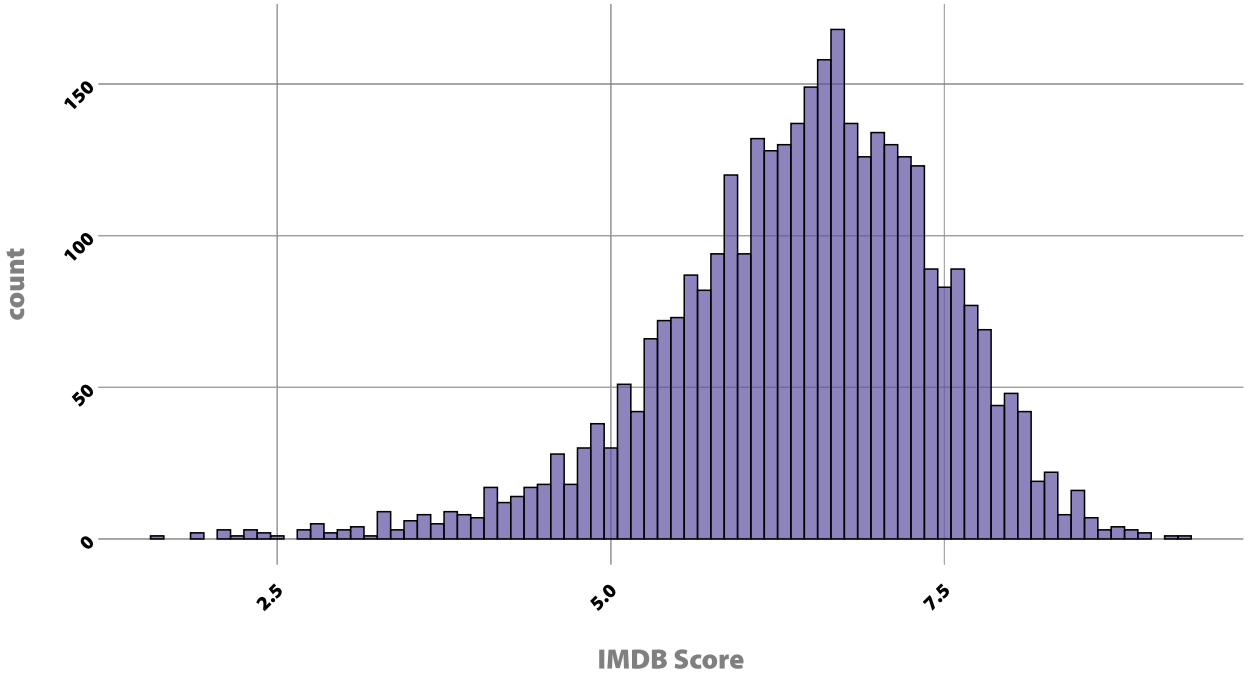


Then we show the proportion of movies making profit or loss. The number of profitable movies is a bit larger, but we can still say that the dataset is balanced. This will not cause difficulty in classification.

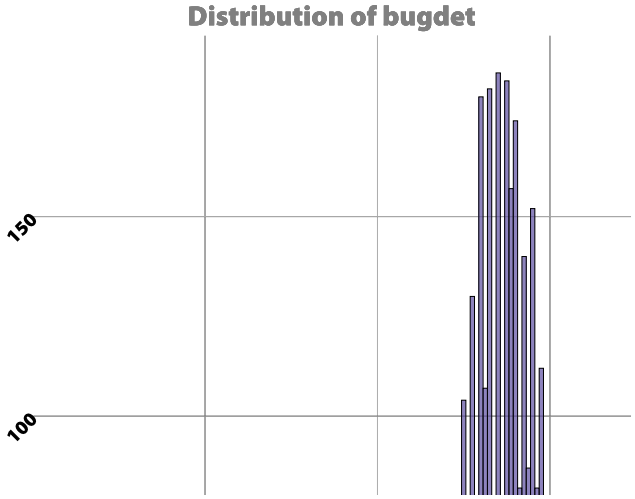


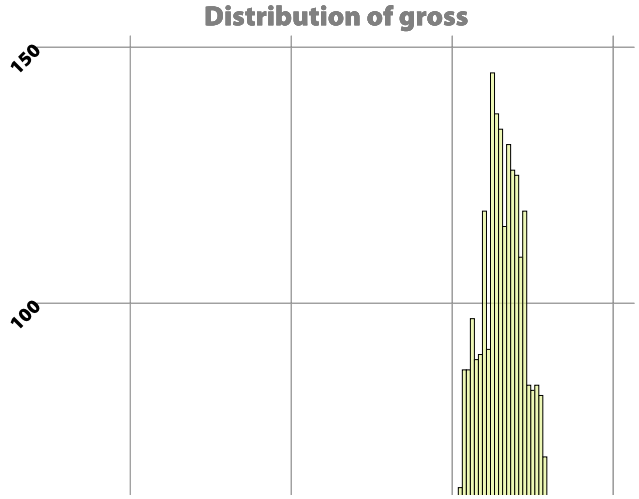


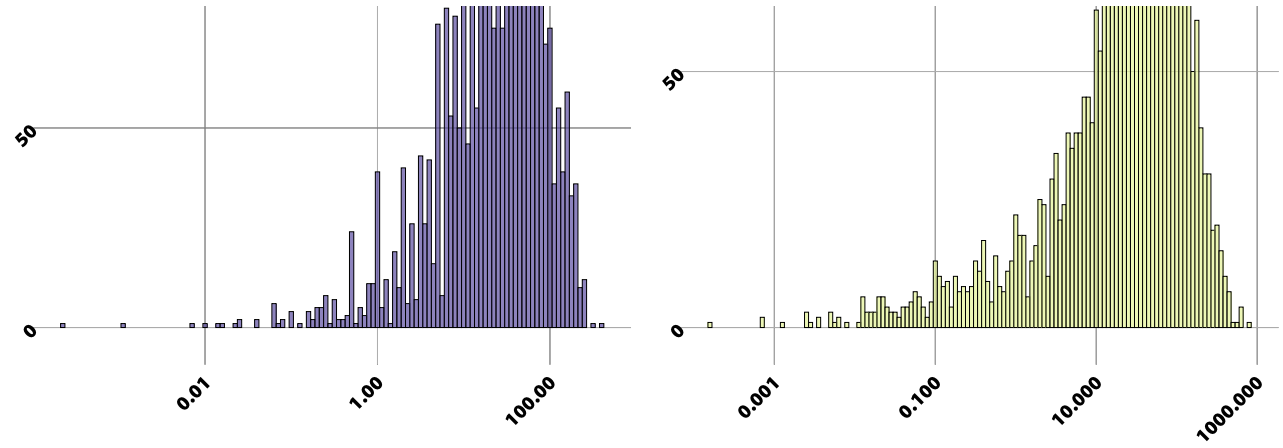
Next we're going to check the distribution of several important continuous variables. The first one is **imdb_score**. The plot shows that there exists right-skewness in the distribution of the IMDB score, most of which lie around 6.7. We might take $\log$ or Box-Cox transformation to handle this problem.



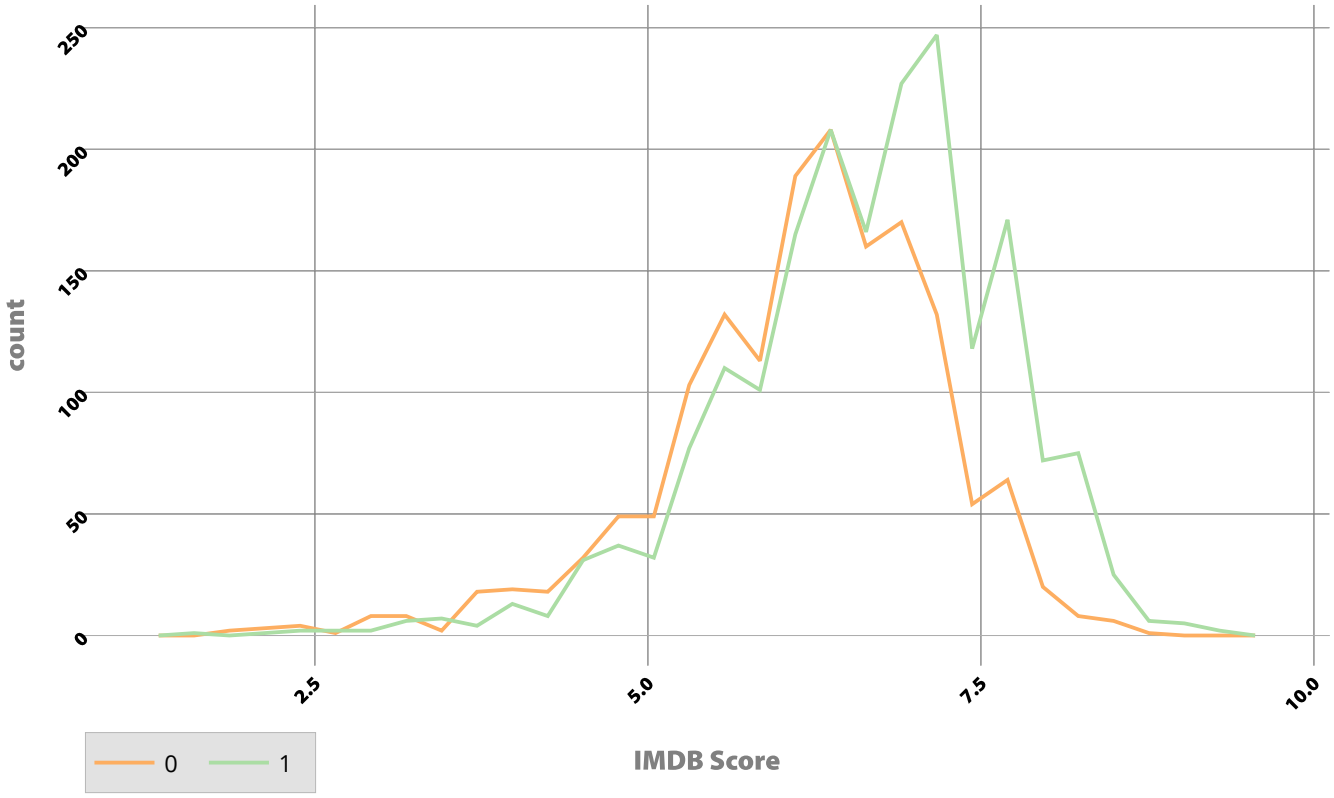
The following is the distribution of **budget** and **gross**, with the values outside their respective inter quartile ranges removed. Similar to **imdb_score**, right-skewness exist and proper transformation may still help.





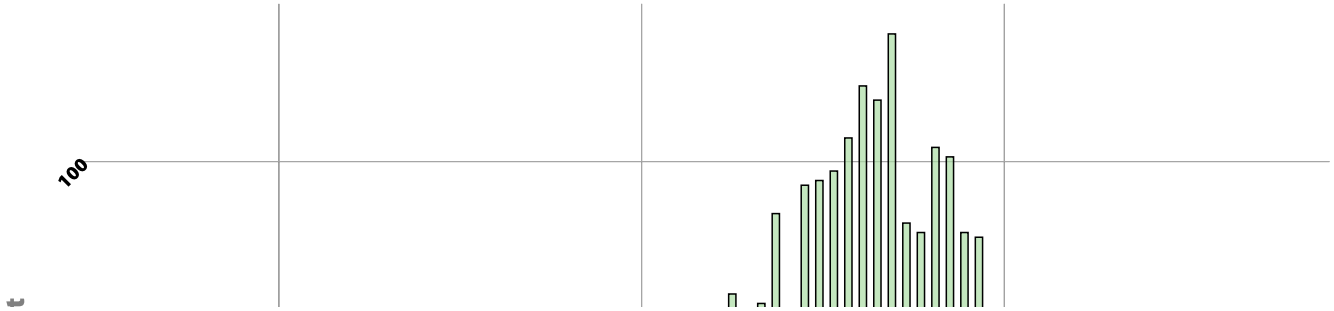


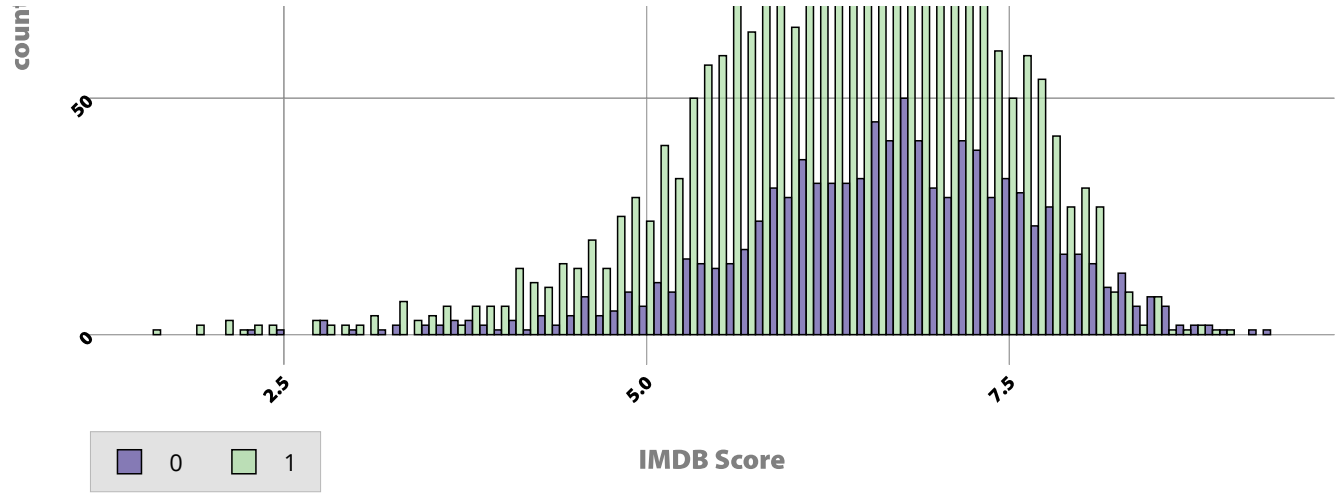
We then check if profitable movies recieve more appreciation by the users. Note that 1 represents profitable movies and 0 the others. Obviously, profitable movies gain higher IMDB score in general. A t-test provides more evidence.



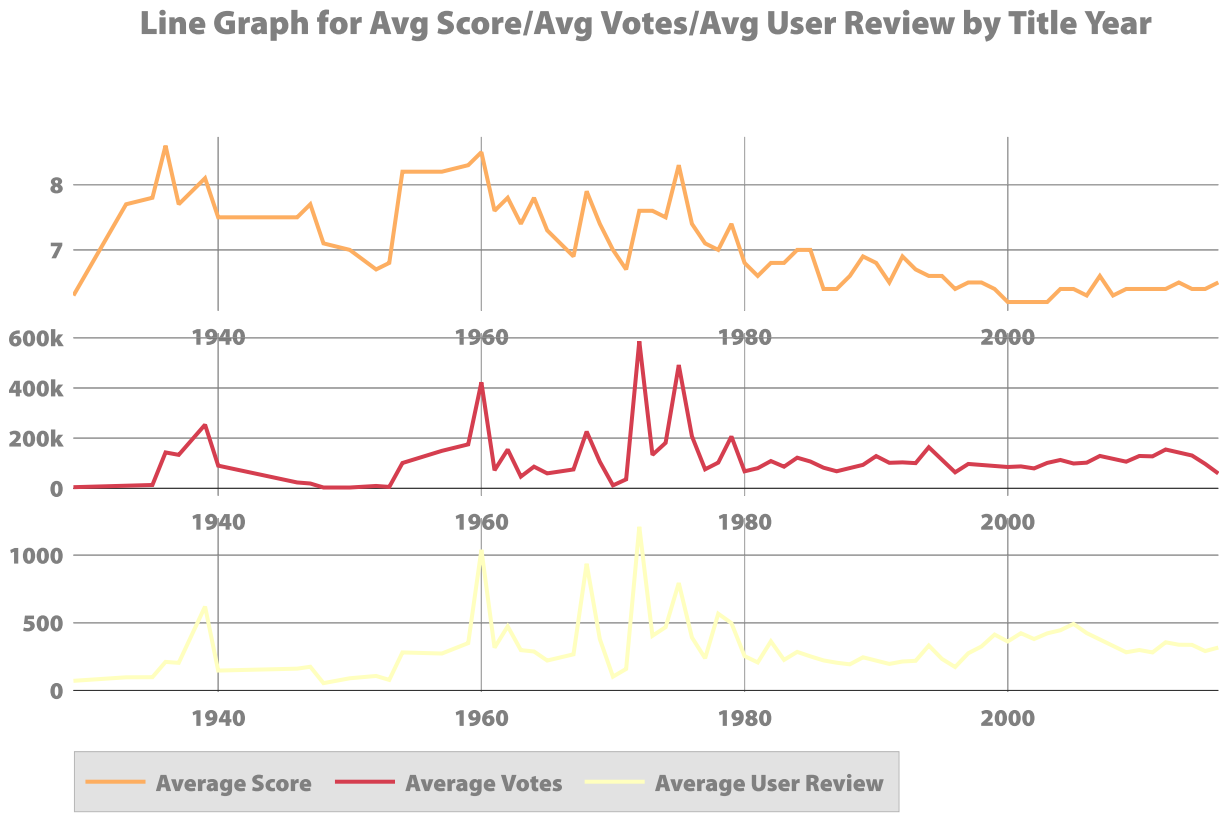
Out of curiosity, we want to check if **imdb_score** is related to **release year**. We set year 2000 as a boundary and find that there's almost no difference in the distribution of **imdb_score** before and after that year.

Distribution of scores before and after year 2000

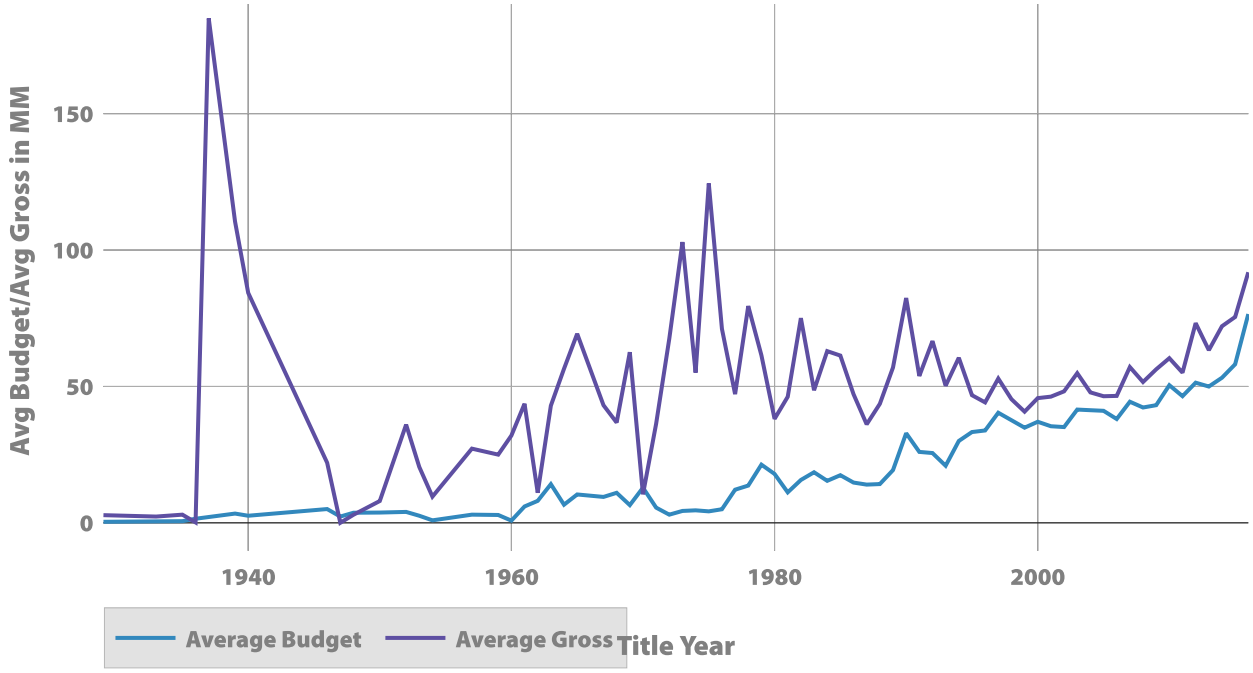




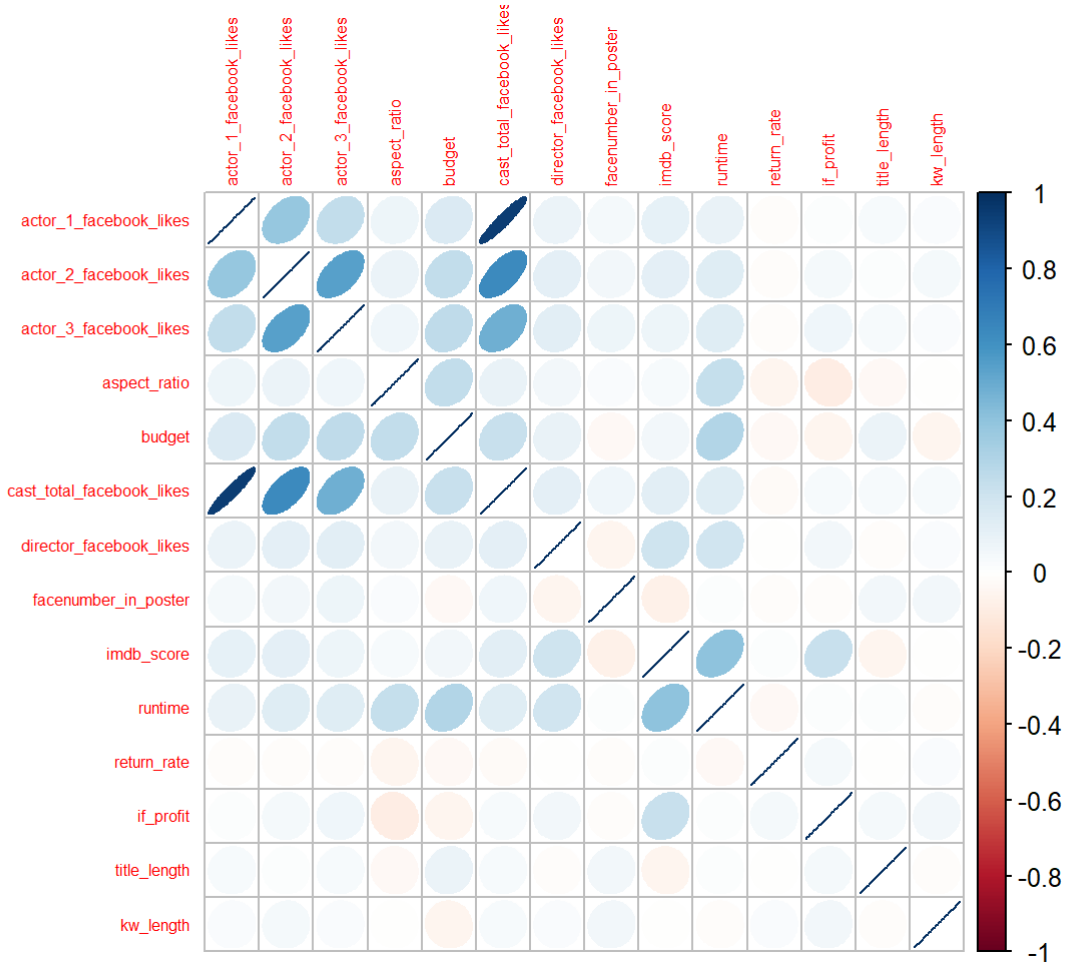
Trend along time is also an interesting point. The following plot shows that audience ratings are decreasing with time, but for user votes or reviews, no such trends occur. However, it's apparent that votes and reviews are peaking where the ratings are higher and vice versa.



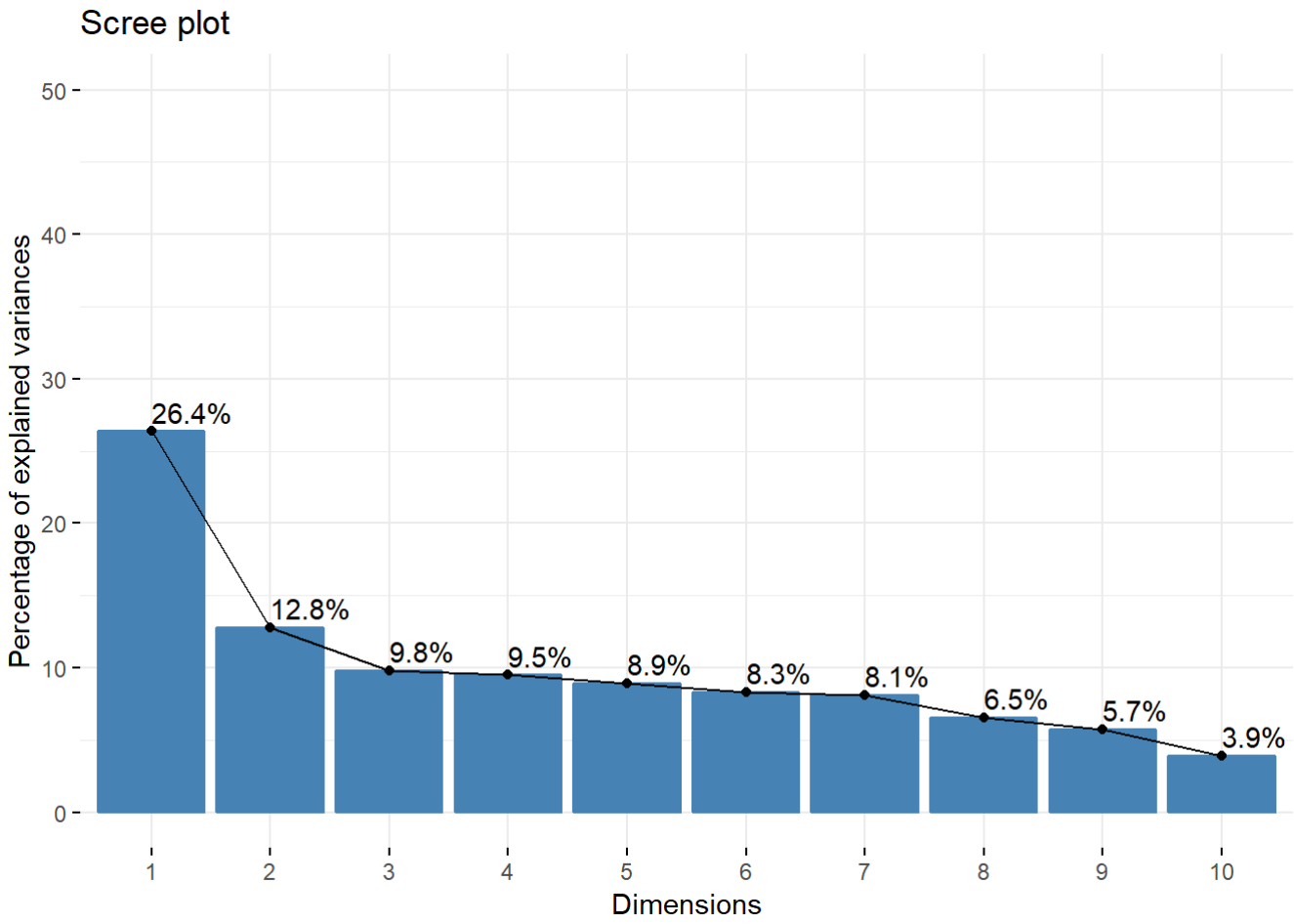
We also check the trend of budget and gross. Evidenlty both budget and gross trends are rising with time. This meant that the costs of producing movies have steadily increased over the years and so have the revenue. The reason why the outlier appears is that there is one movie, of which the gross is not measured in USD.

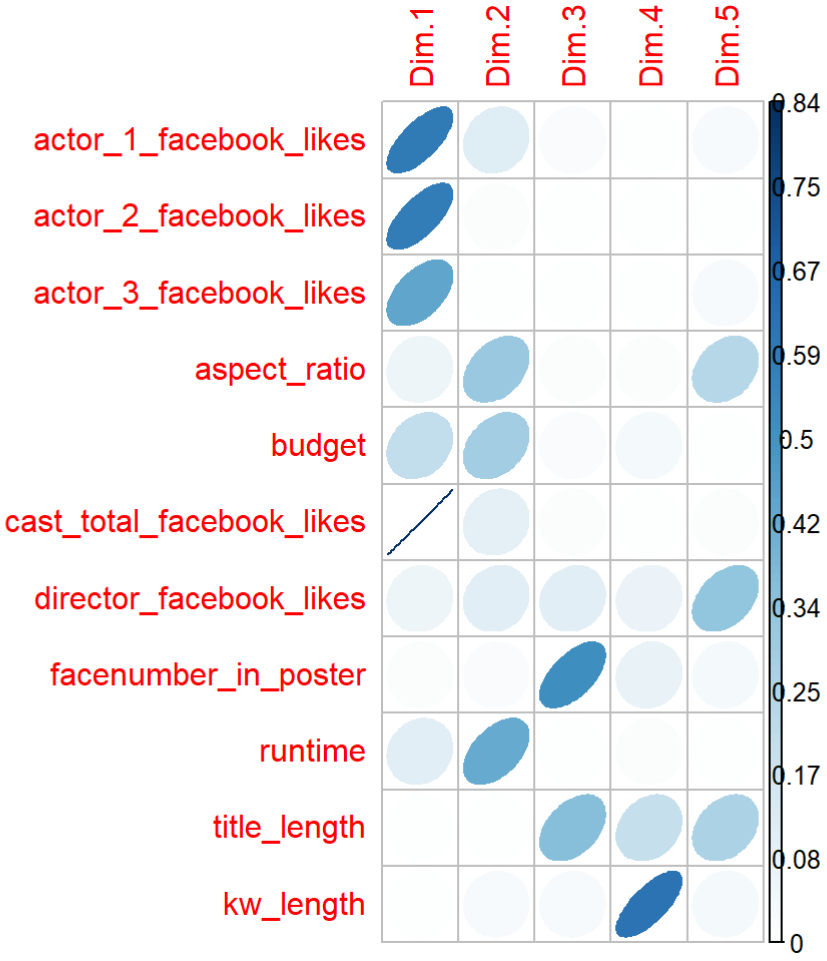


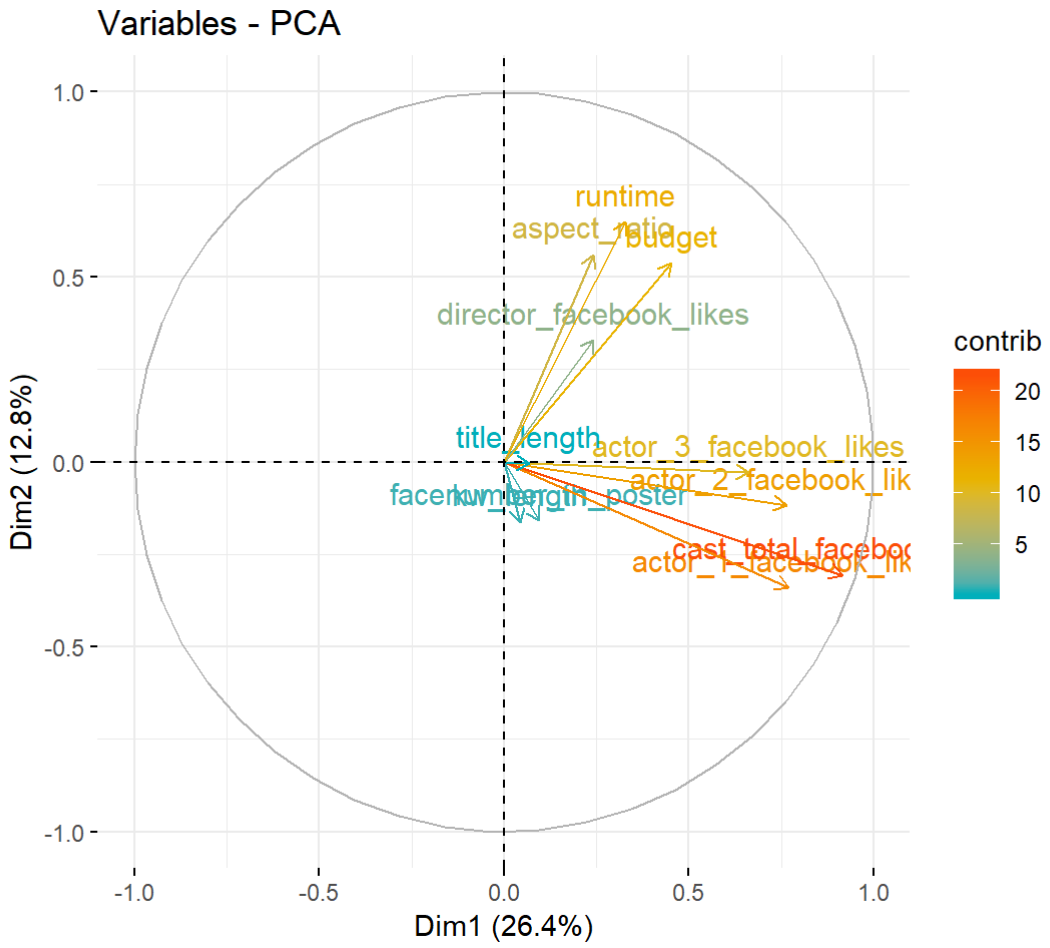
Then, we check the correlation between several variables. This step inspires us to select the most significant variables.



Last but not least, we apply PCA to the continuous variables to see the effect of dimension reduction. But unfortunately, the proportion of variance doesn't concentrate on the first two or three components, so PCA is not very helpful here.


Scree plot
Percentage of explained variances
Dimensions
26.4%
12.8%
9.8%
9.5%
8.9%
8.3%
8.1%
6.5%
5.7%
3.9%


Dim.1
Dim.2
Dim.3
Dim.4
Dim.5
actor_1_facebook_likes
actor_2_facebook_likes
actor_3_facebook_likes
aspect_ratio
budget
cast_total_facebook_likes
director_facebook_likes
facenumber_in_poster
runtime
title_length
kw_length
0.84
0.75
0.67
0.59
0.5
0.42
0.34
0.25
0.17
0.08
0



We have to say EDA requires lots of efforts and energy. What we do in this report is a tip of the iceberg. But due to the limitation of time and space, we only show what we think is the most crucial and interesting part. Now we have a better understanding of the movie dataset, and we will turn to the modelling part.

4. Model

4.1 Review

Since most machine learning models we use later are quite popular and familiar to the audience, we only review one relatively advanced model – Xgboost.

Xgboost, also known as eXtreme Gradient Boosting, is a powerful model in both prediction and classification. Xgboost is actually the ensemble of many CART regression tree. The objective function is

$$\mathcal{L}(\phi) = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k \Omega\left(f_k\right)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2,$$

where T is the number of leaf nodes and w is the weight. By second-order Taylor's expansion,

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2}h_i f_t^2\left(\mathbf{x}_i\right)\right] + \Omega\left(f_t\right),$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$. Now it's easy to solve the problem as follows.

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma.$$

We will apply Xgboost to our classification model.

4.2 Predict IMDB score

As usual, we split the dataset into training and testing set with ratio 0.8. Notice that if we try to decide whether to invest a movie or not, some variables cannot be obtained in advance, such as **num_user_for_reviews**, **num_voted_users**, etc. Some text varaibles should be transformed into factor type, while some hardly have contribution, hence need dropping. Besides, we add two features: **title_length** is the number of words in title and **kw_length** is the number of keywords in poster.

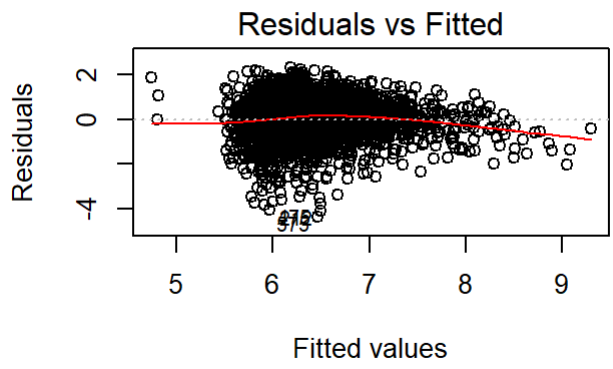
4.2.1 Linear Regression

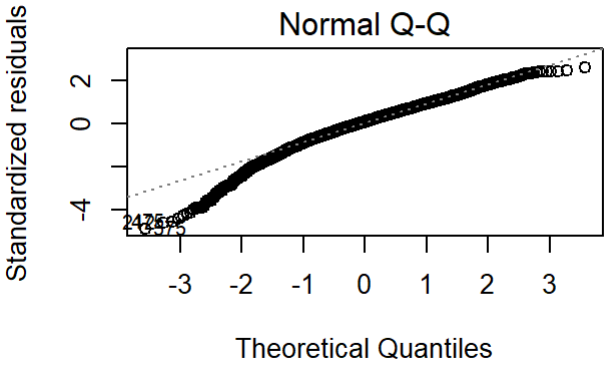
We first fit a simple multiple linear regression to obtain some insights. We find several significant variables including **runtime**, **facenum_in_posters**, etc. But surprisingly and strangely, variables related to facebook likes of actors and **budget** have little with prediction.

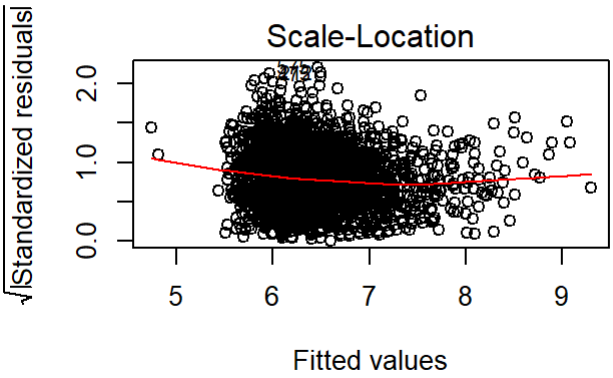
```
## [1] "Significant variables include: (Intercept) colorColor director_facebook_likes facenumber_in_poster genresAdventure genresAnimation genresBiography genresCrime genresDrama runtime"
```

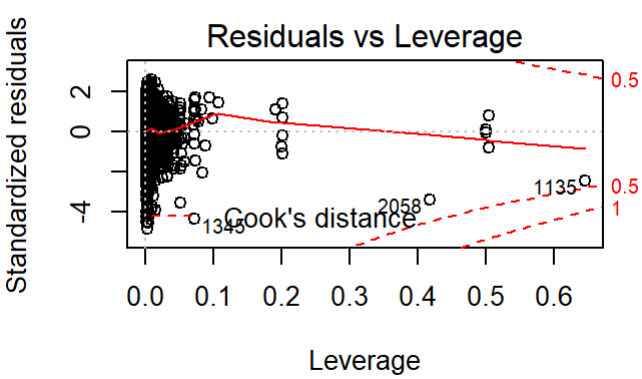
```
## Warning: not plotting observations with leverage one:
##   330, 1080, 2705

## Warning: not plotting observations with leverage one:
##   330, 1080, 2705
```



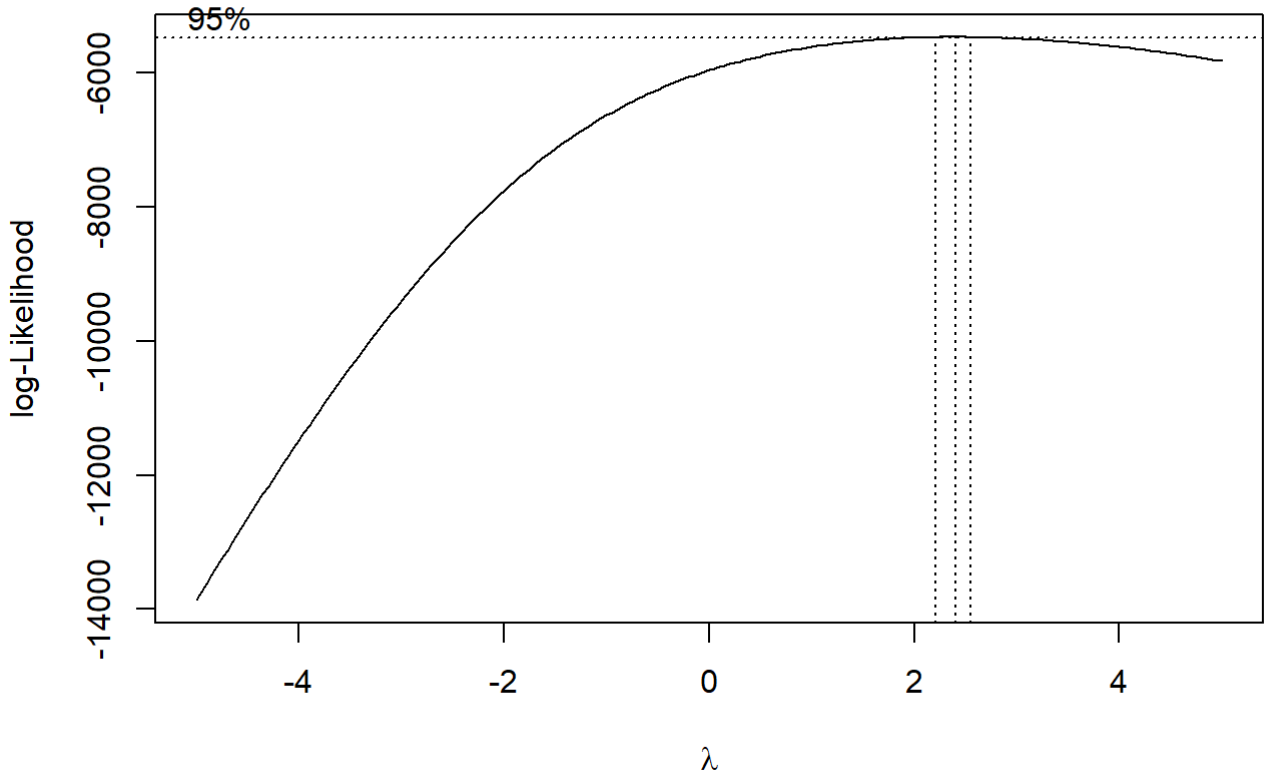







```
## [1] "Linear Regression: RMSE = 0.936738946269746"
```

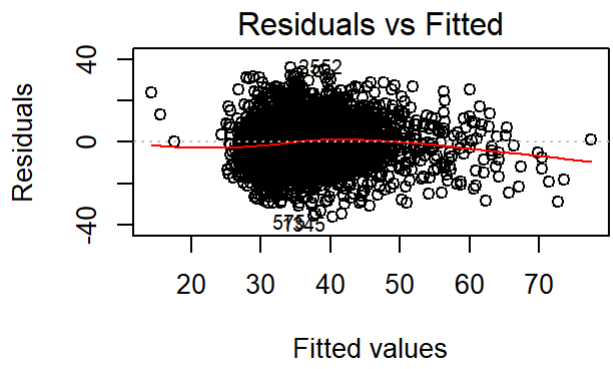
The RMSE is quite satisfying. However, it's easy to find that the QQ-plot is far away from normal case, so next we apply Box-Cox transformation to handle this problem. We can see that such transformation is quite powerful, although accuracy is lost a bit.

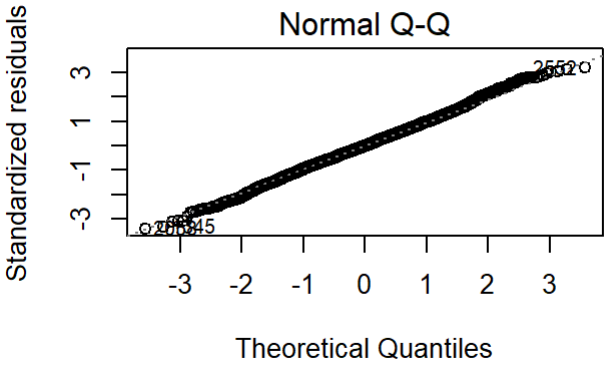


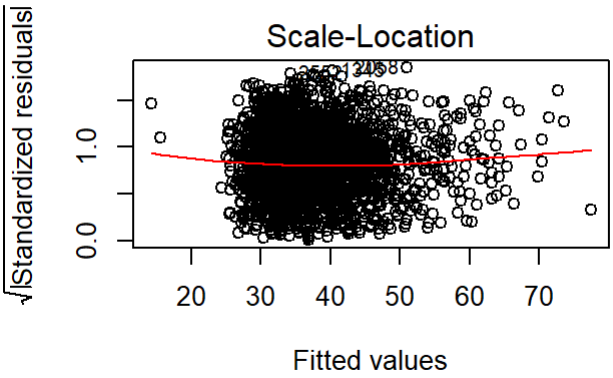
[1] "Significant variables include: (Intercept) colorColor director_facebook_likes facenumber_in_poster genresAdventure genresAnimation genresBiography genresCrime genresDrama runtime"

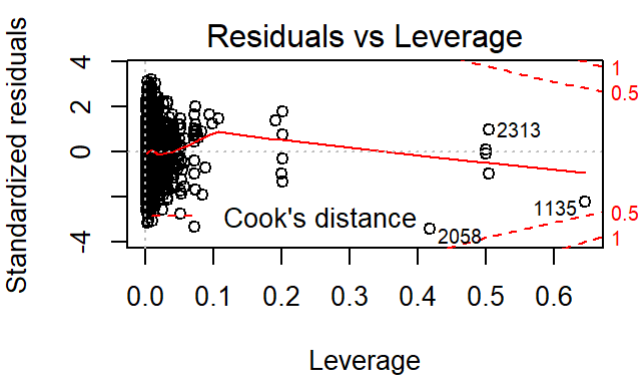
Warning: not plotting observations with leverage one:
330, 1080, 2705

Warning: not plotting observations with leverage one:
330, 1080, 2705



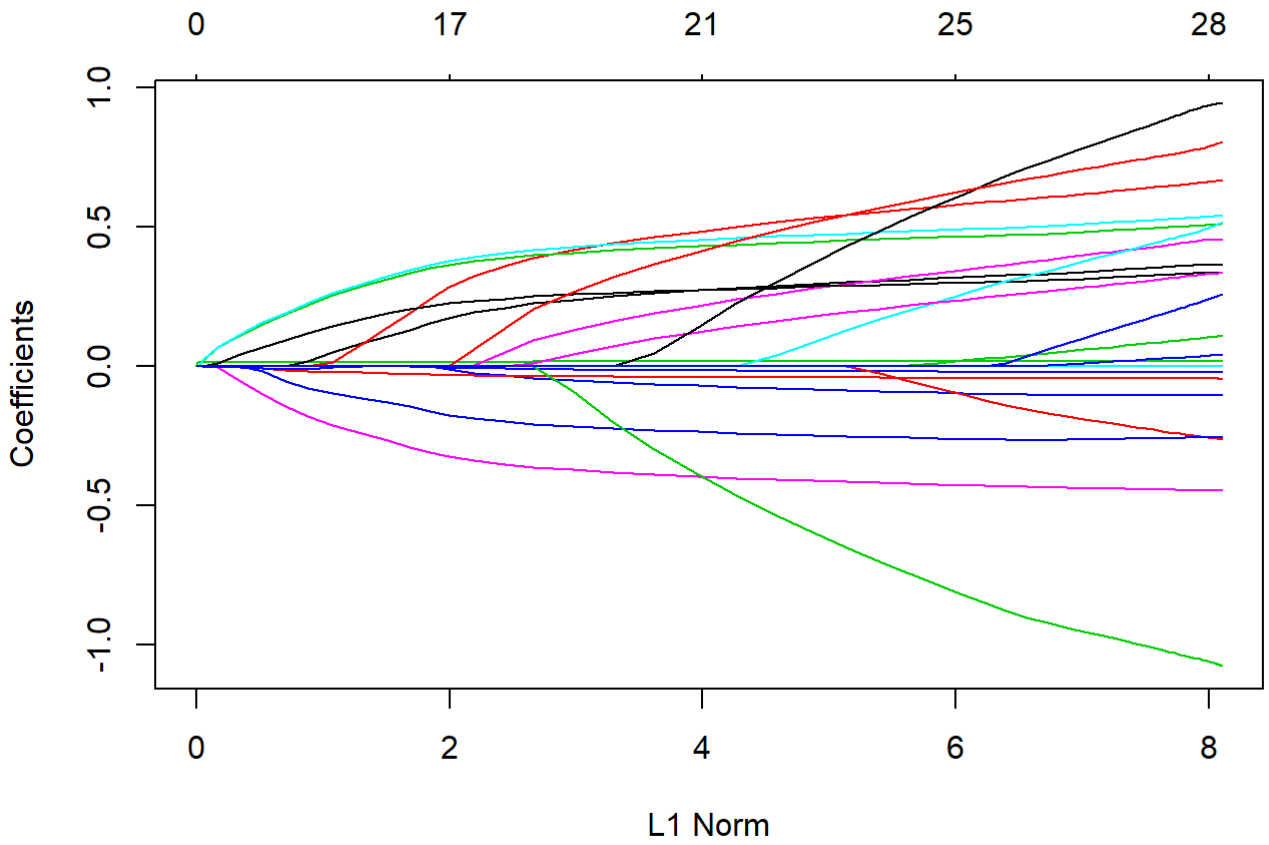


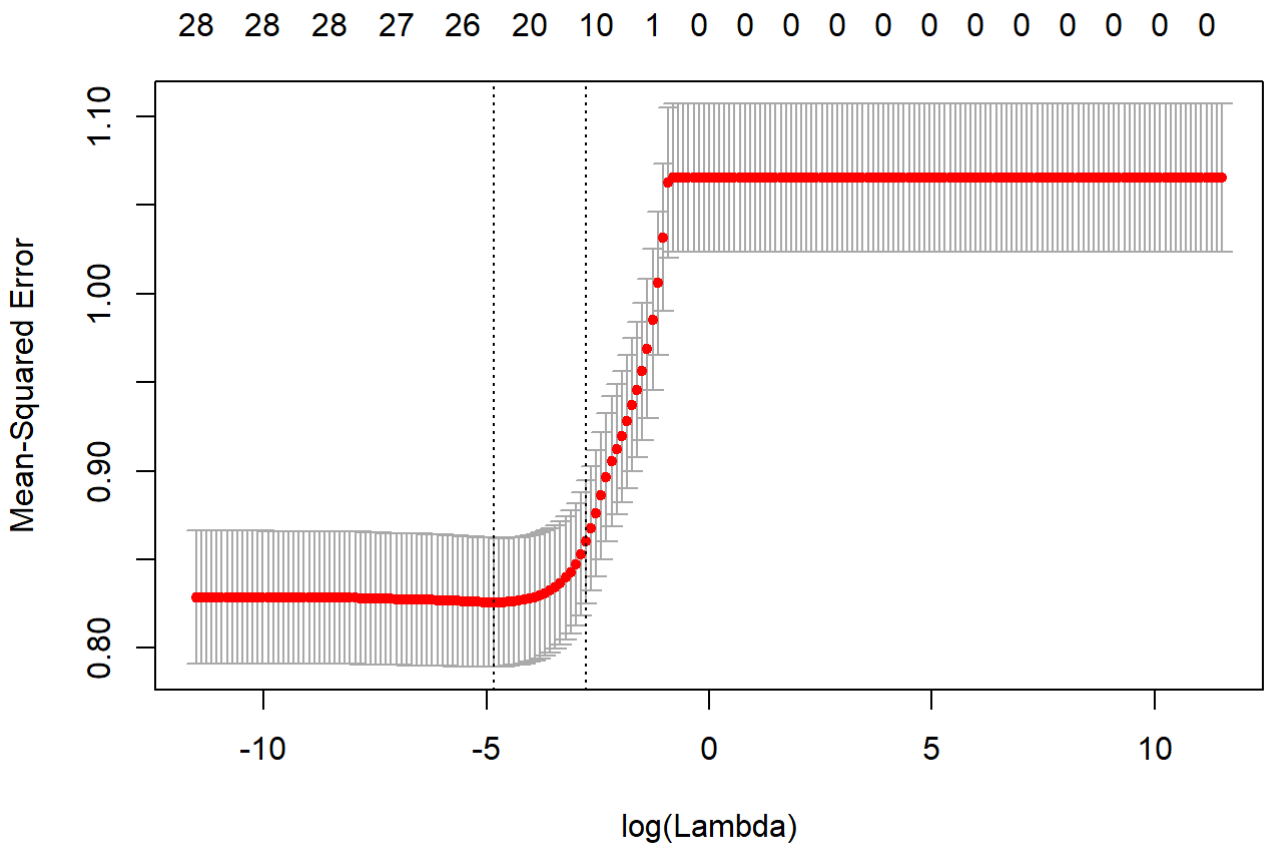





```
## [1] "Linear Regression with Box-Cox: RMSE = 0.945478233414966"
```

We also want to screen really important variables, so variable selection techniques are also applied, the following shows the result of Lasso. Notice we use cross-validation to pick the optimal hyper-parameters. In this case, facebook likes of actors are picked up.


0
17
21
25
28
1.0
0.5
0.0
-0.5
-1.0
Coefficients
0
2
4
6
8
L1 Norm


28 28 28 27 26 20 10 1 0 0 0 0 0 0 0 0 0 0 0 0
1.10
1.00
0.90
0.80
Mean-Squared Error
-10
-5
0
5
10
log(Lambda)

```
## [1] "Significant variables include: (Intercept) actor_1_facebook_likes actor_2_facebook_likes aspect_ratio budget cast_total_facebook_likes director_facebook_likes facenumber_in_poster runtime title_length kw_length colorColor genresAdventure genresAnimation genresBiography genresCrime genresDocumentary genresDrama genresFamily genresHorror genresMusical genresMystery genresRomance genresSci genresThriller"
```

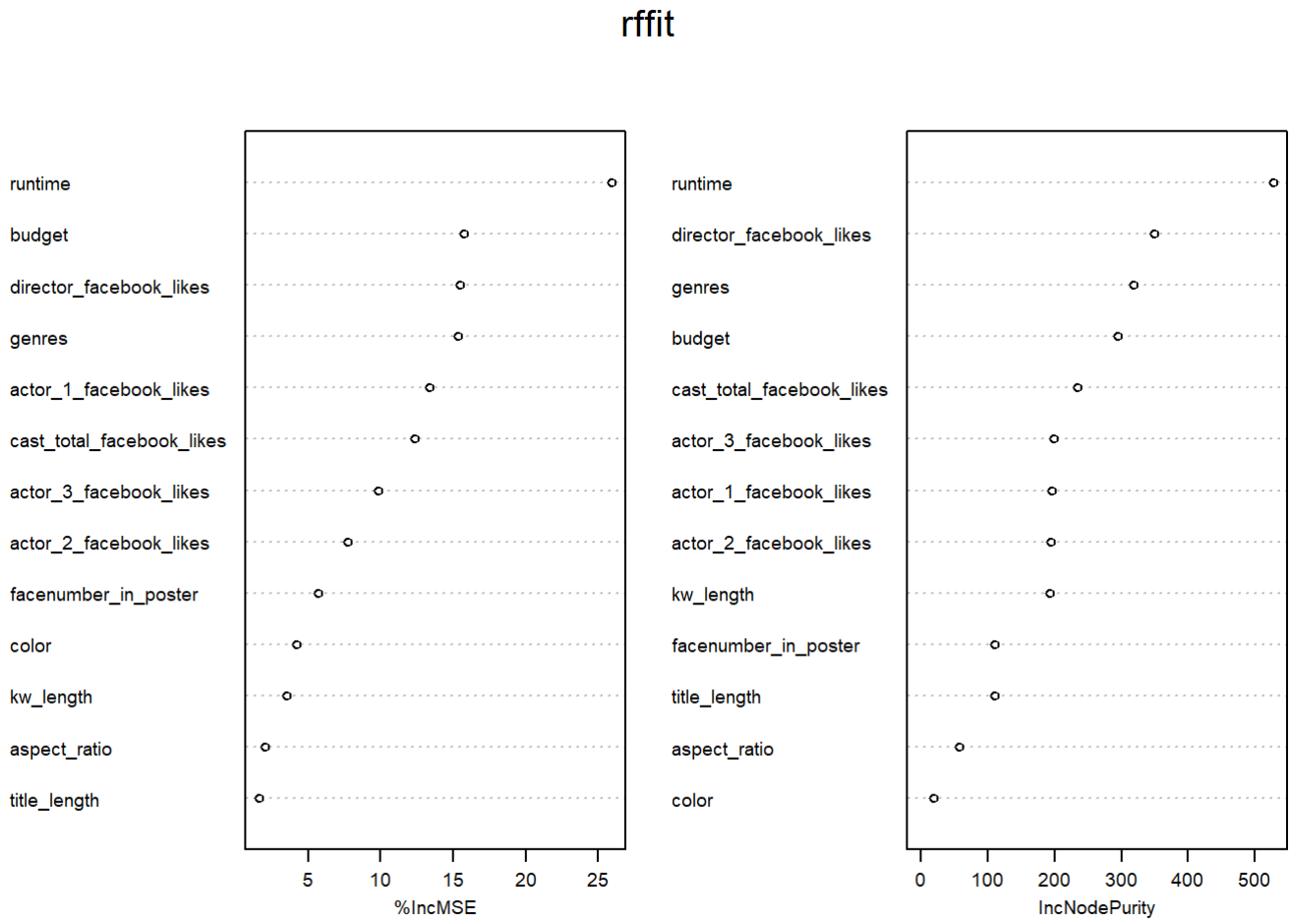
```
## [1] "LASSO: RMSE = 0.936952347074828"
```

4.2.2 SVM & Random Forest

Obviouly, the prediction accuracy on the testing set is improved, especially for random forest. Also, rank of variable importance is shown in the plot below.

```
## [1] "SVM: RMSE = 0.908991617768879"
```

```
## [1] "Random Forest: RMSE = 0.849040564486931"
```

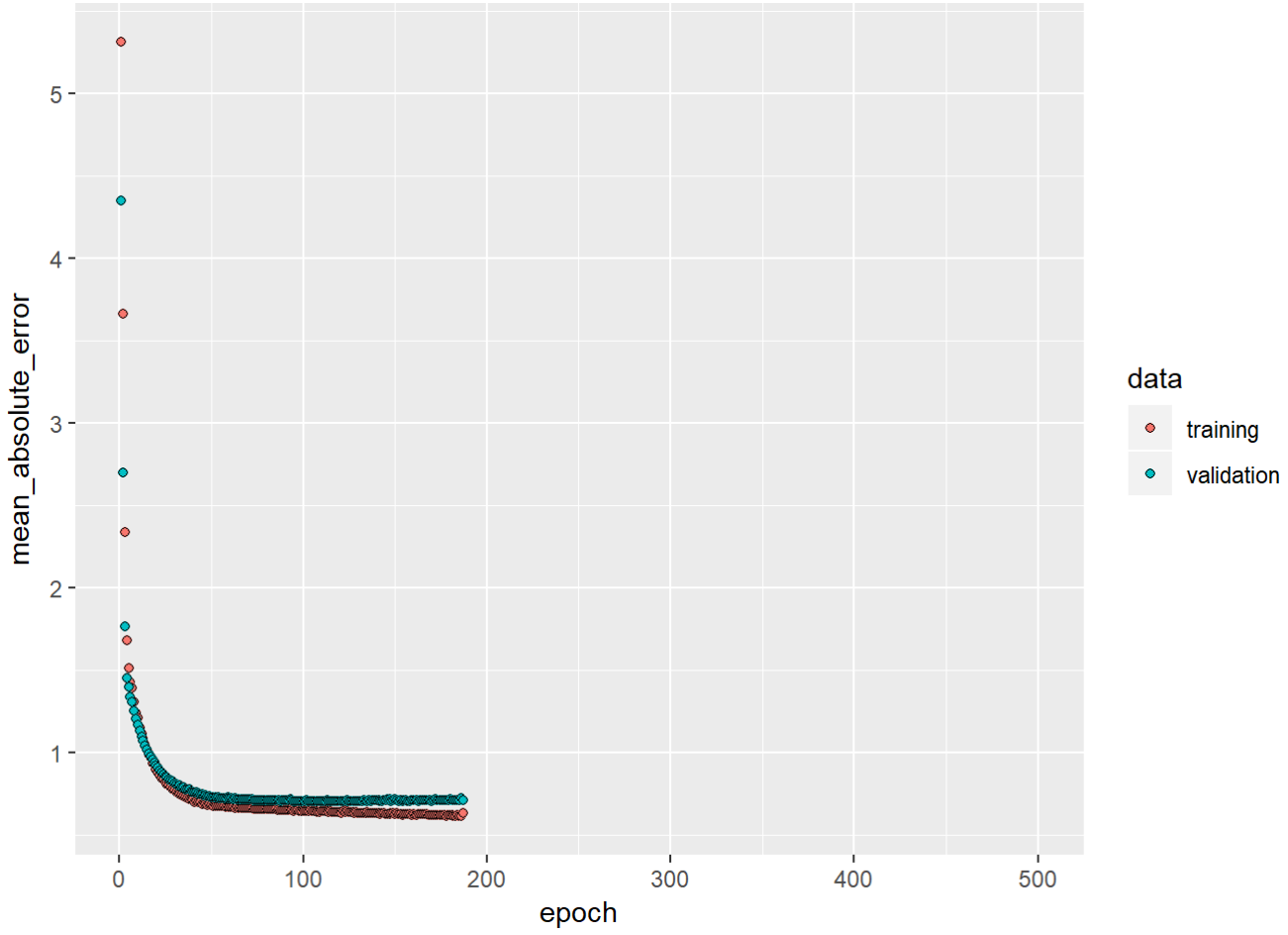


4.2.3 Deep Learning

After building the common machine learning, we now turn to the deep learning techniques. Now keras has already been integrated into R package, thus very convenient to construct a network. Next we train the network and report the results. The common sense is that neural network should be more powerful, but the following results shows the opposite. The accuracy is not improved. We think the reason is due to the overfitting and the lack of variables.

```
## _________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =================================================================
## dense (Dense)                (None, 32)                384
## _________________________________________________________________
## dense_1 (Dense)              (None, 64)                2112
## _________________________________________________________________
## dense_2 (Dense)              (None, 1)                 65
## =================================================================
## Total params: 2,561
## Trainable params: 2,561
## Non-trainable params: 0
## _________________________________________________________________
```

```
##
## ....................................................................................
## ....................................................................................
```



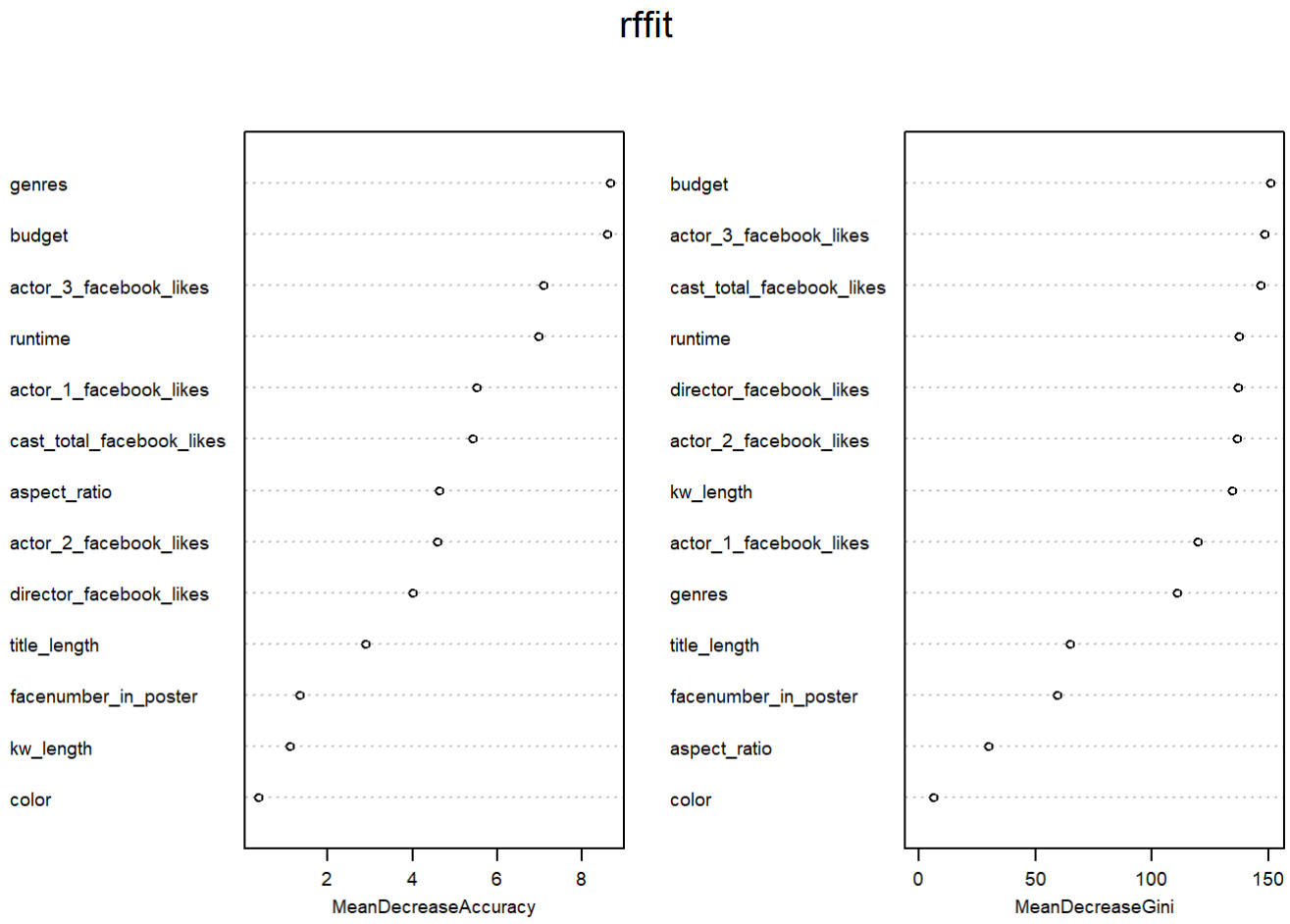

```
## [1] "Deep Learning: RMSE = 0.954686768590835"
```

4.3 Classify Profitable Movie

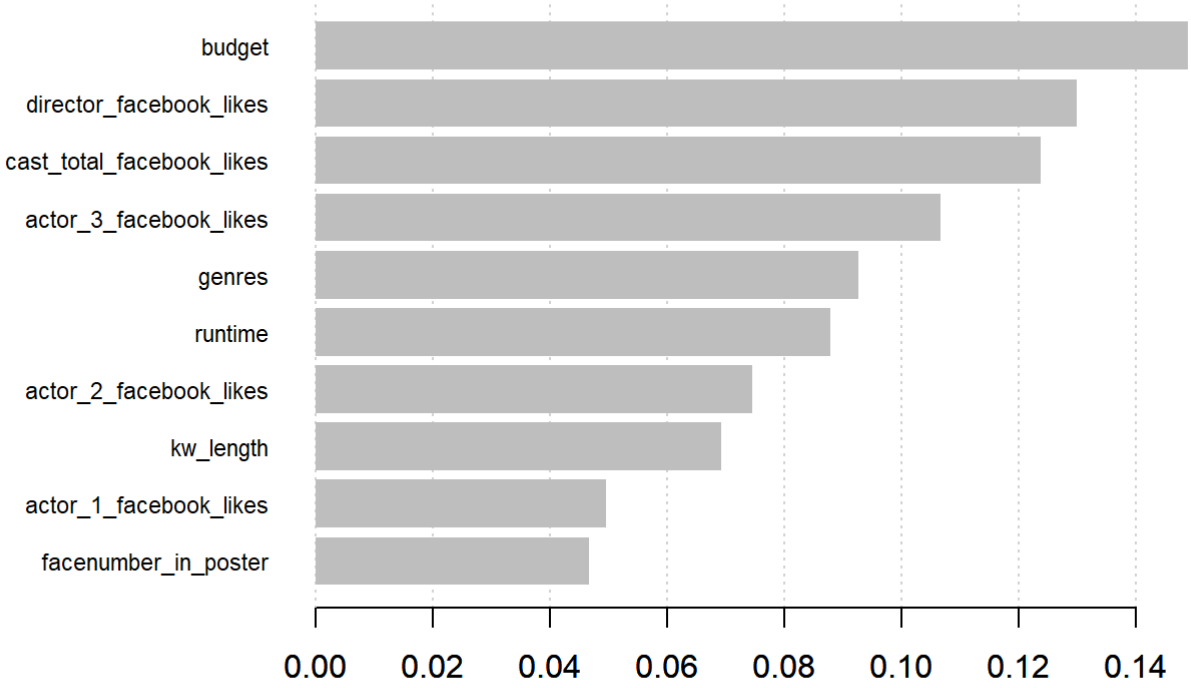
Since this section is imilar to what we do in section 4.2, we just show our results and omit more detailed interpretation.

```
## [1] "Logistic Regression: Rate = 0.591690544412607"
```

```
## [1] "Random Forest: RMSE = 0.60458452722063"
```

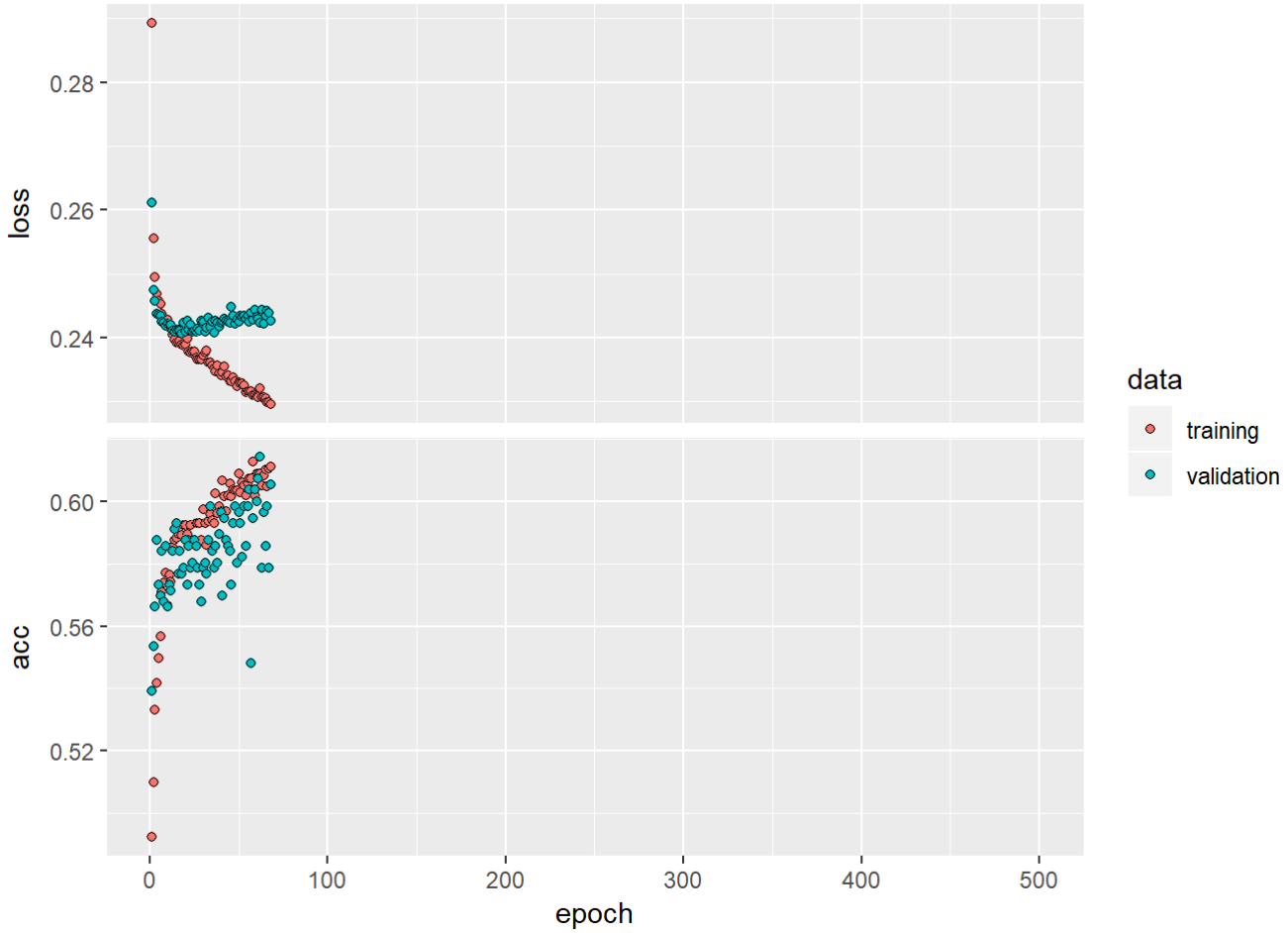



```
## [1] "Xgboost: RMSE = 0.627507163323782"
```

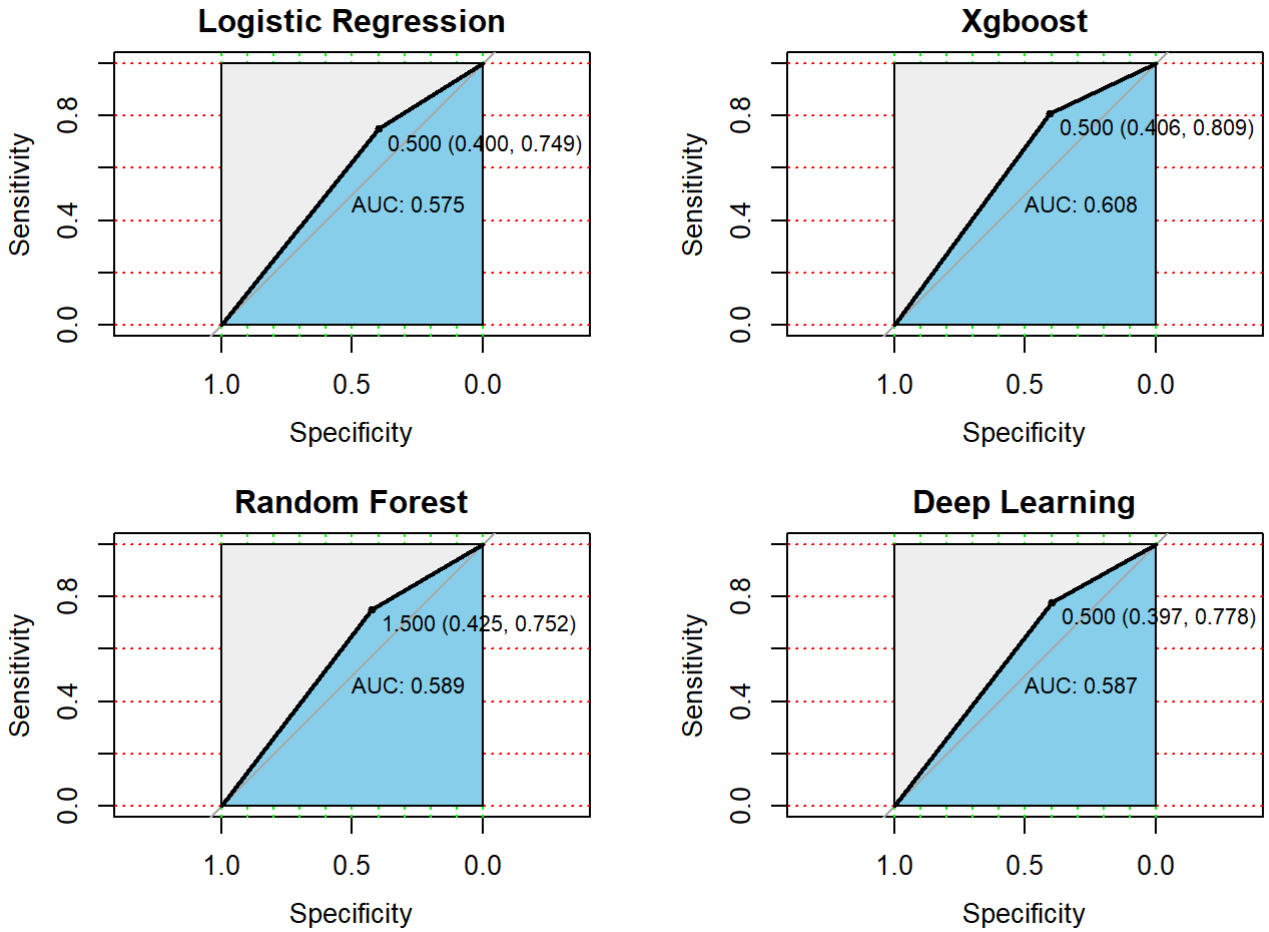



```
## _________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =================================================================
## dense_3 (Dense)              (None, 32)                384
## _________________________________________________________________
## dense_4 (Dense)              (None, 32)                1056
## _________________________________________________________________
## dense_5 (Dense)              (None, 1)                 33
## =================================================================
## Total params: 1,473
## Trainable params: 1,473
## Non-trainable params: 0
## _________________________________________________________________
```

```
##
## ..........
```




```
## [1] "Deep Learning: RMSE = 0.606017191977077"
```



5. Discussion

We have a deeper understanding of the movie dataset. By detailed analysis, we build satisfying prediction systems with RMSE around 0.9, as well as mundane classification systems with accuracy 0.6. The variables we collected are still not enough for better results. After all, it's a hard job to predict score or determine whether profitable before a movie runs to the audience. In fact, if we do not drop the ad-hoc varaibles, both results improve very significantly. But if we still follow the rule of real world, we need to collect more features, such as social media information, network of actors, etc. These can be extensions of this report.