# USE CASE STUDY REPORT

**Group No.**: 10

**Student Names**: Srishti Bhandari and Fangsheng Yan

## *Data Mining techniques to predict IMDB scores*


## Executive Summary:

Movies have become a favorite way of entertainment over the centuries. There are thousands of movies coming out each year. IMDB score is a popular and important indicator for people to determine the success of movies. However, it is not certain that movies with high budget or gross earnings will have higher IMDB scores than a fairly lower budget movie with new actors will have a lower IMDB score.

Goal of the study:

The success of a movie isn't purely about entertainment of the audience, the producing firms, directors, actors and the entire crew make huge profits from them. It is essential to keep in mind the kind of profits the movie will make before investing money in the making. Our project aims at providing insights on whether the movie will gain a high IMDB score based on multitude of factors that contribute towards this.

Origin of the data and data processing:

The data is obtained from a verified website: Kaggle. The data consists of 28 variables for 5043 movies, spanning across 100 years in 66 countries. There are about 2399 unique director names and thousands of actors and actresses. The "imdb_score" variable is a column in our data which is the response variable while the other 27 variables are the possible predictors. The data does contain missing records and special characters which gives us an opportunity to apply principals of data cleansing and pre-processing. Various Packages and functions are used for the same.

Data Mining techniques:

The project shows the movie ratings in 3 parameters: Bad, ok, good and excellent. To figure out whether a movie will do good or bad, it is essential to run the models through algorithms of data mining. We have used: Classification tree, K Nearest neighbor and Random forest to process our models. The data for KNN and Random Forest has been divided into 3: training, validation and test.


## I. Background and Introduction

With the Boom of technology, the art of movie making has improved by folds. The story telling has seen changes as well. Movies are a source of entertainment which provides entertainment to everyone irrespective of age and culture. The reason for choosing this topic as our project was to learn algorithms using a dataset that we everyone could relate too and use it with real world examples of things/movies we have watched.

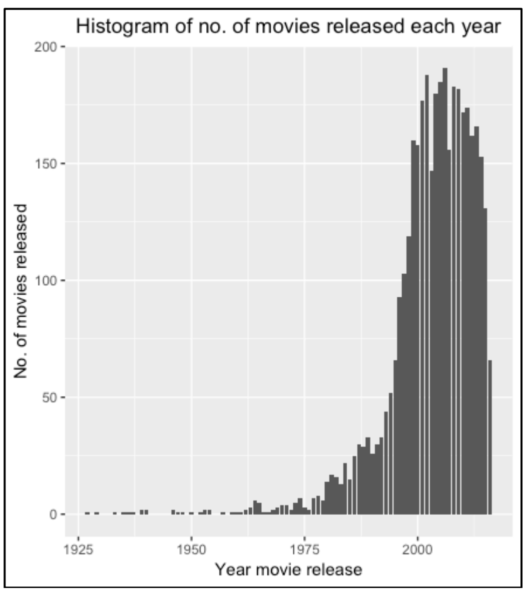
The problem we are trying to resolve is the uncertainty of the movie performing well or not. It is imperative to have some idea of whether the movie is going to make a hit or be a flop or perform somewhere in the middle.

IMDB website has proven to be a very reliable source for providing information to the users based on multiple factors and reviews from audience as well as critics.
The *goal* is to identify a pattern of factors which contribute to the IMDB score the movie receives. The factors are: directors, actors, location, language the movie is in, the country it is produced etc.
We *aim* to firstly clean this data and put it into a format which can be easily understood and used to make modifications on. We then use data visualization techniques to further analyze the data at hand and understand the relation between factors. Moving forward we apply three data mining models to determine the factors contributing to their respective IMDB scores and the accuracy of each of them to find the best suited one.

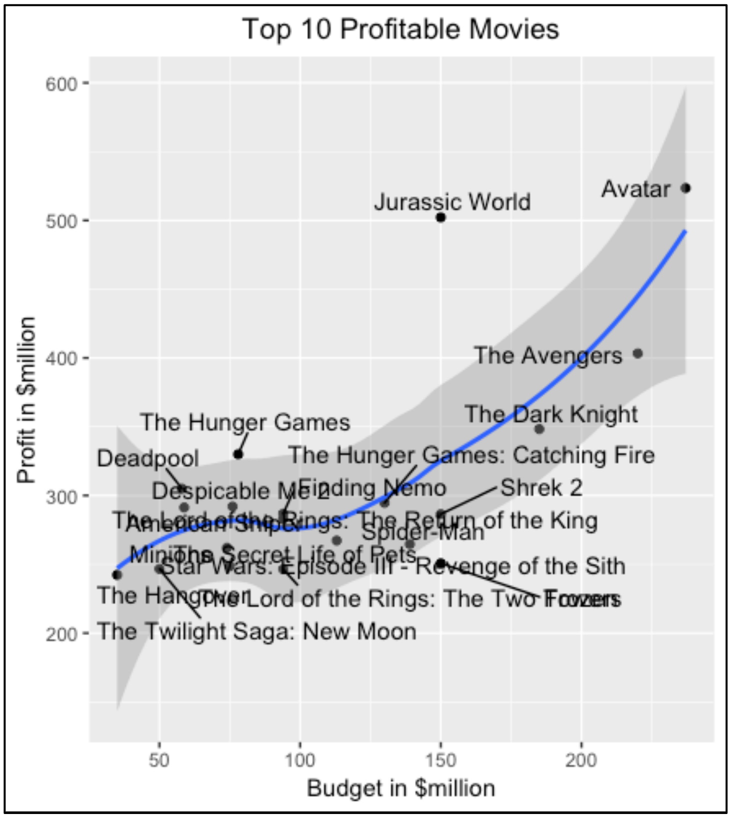
## II. Data Exploration and Visualization

Provide brief description of techniques used to explore the data including: basic charts, distribution plots, correlations, missing values, rescaling, aggregation, hierarchies, zooming, filtering, etc.

1. Plotting histogram of movies released over the years

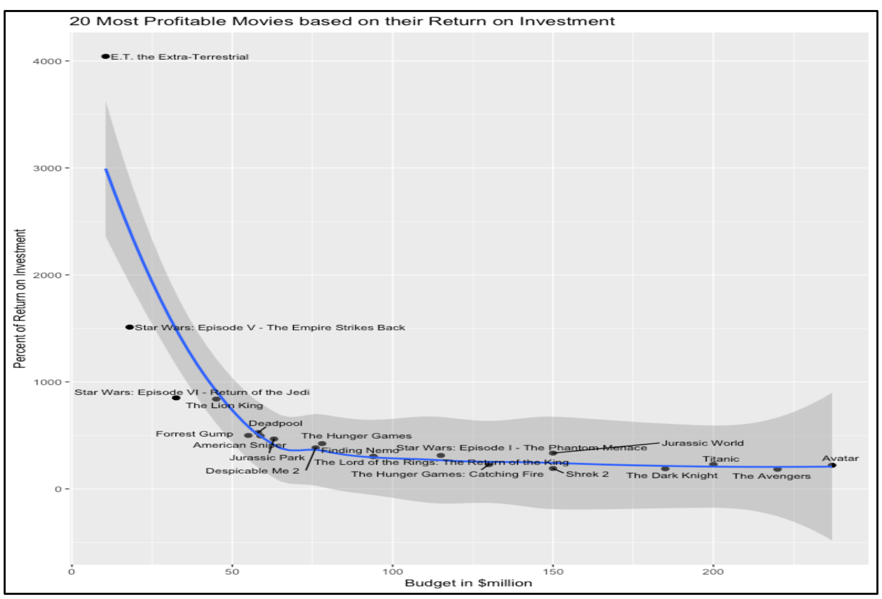



2. It can be seen that the movie boom came after 1980 and thus we represent the data only after 1980.
3. We create 2 new variables: profit and return on investment (%)
   Where profit = gross - budget,
   And return_on_investment_perc = (profit/budget)*100)




Adjacent is the visualization of the top 20 movies based on profits in millions$

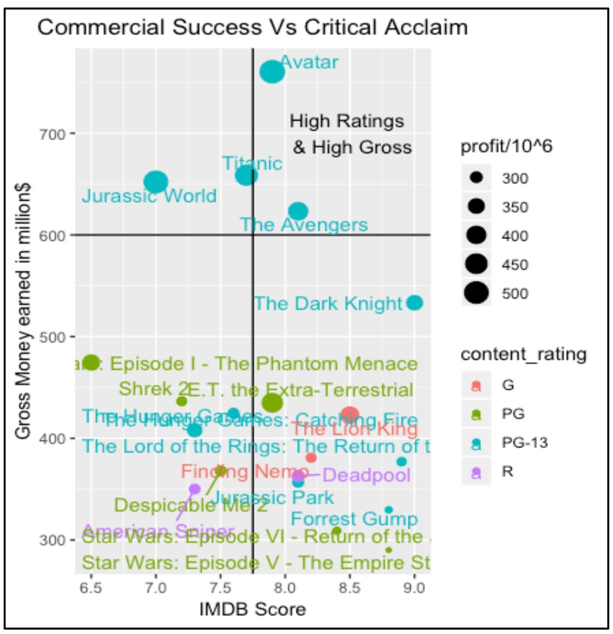
4. Using profits and return on investment variables as criteria to find 20 most profitable movies



5. Visualizing 20 top directors based on the highest IMDB scores

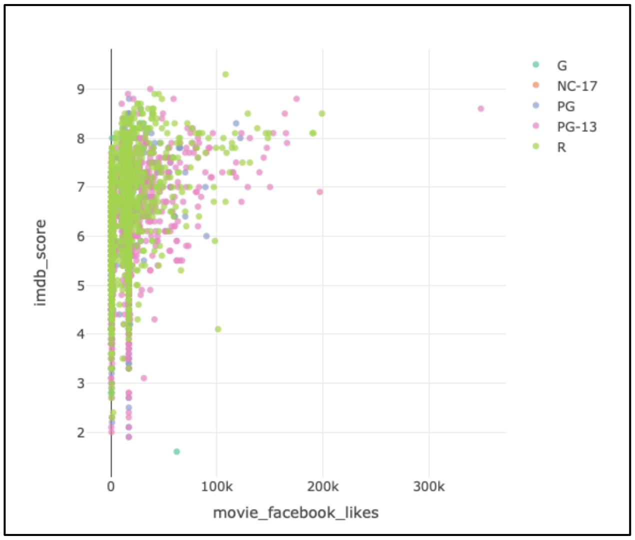
| director_name | avg_imdb |
|---|---|
| Tony Kaye | 8.600000 |
| Damien Chazelle | 8.500000 |
| Majid Majidi | 8.500000 |
| Ron Fricke | 8.500000 |
| Christopher Nolan | 8.425000 |
| Asghar Farhadi | 8.400000 |
| Marius A. Markevicius | 8.400000 |
| Richard Marquand | 8.400000 |
| Sergio Leone | 8.400000 |
| Lee Unkrich | 8.300000 |
| Lenny Abrahamson | 8.300000 |
| Pete Docter | 8.233333 |
| Hayao Miyazaki | 8.225000 |
| Joshua Oppenheimer | 8.200000 |
| Juan José Campanella | 8.200000 |
| Quentin Tarantino | 8.200000 |
| David Sington | 8.100000 |
| Je-kyu Kang | 8.100000 |
| Terry George | 8.100000 |
| Tim Miller | 8.100000 |

6. Plotting a visualization for commercial success vs critical claim




The above observation shows that there is hardly any correlation between critical acclaim and the movie's commercial success

7. Visualizing relation between Facebook likes and IMDB scores




Movies with high Facebook likes can be seen to have higher IMDB score.

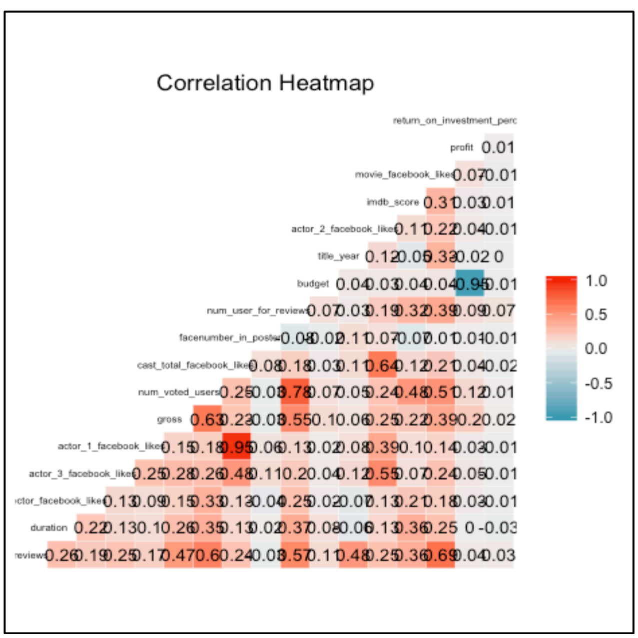
<u>Exploration of Data:</u>

1. Find the total number of directors and actors:

```
> sum(uniqueN(movie_data$director_name))
[1] 1660
> sum(uniqueN(movie_data[, c("actor_1_name", "actor_2_name", "actor_3_name")]))
[1] 3621
```

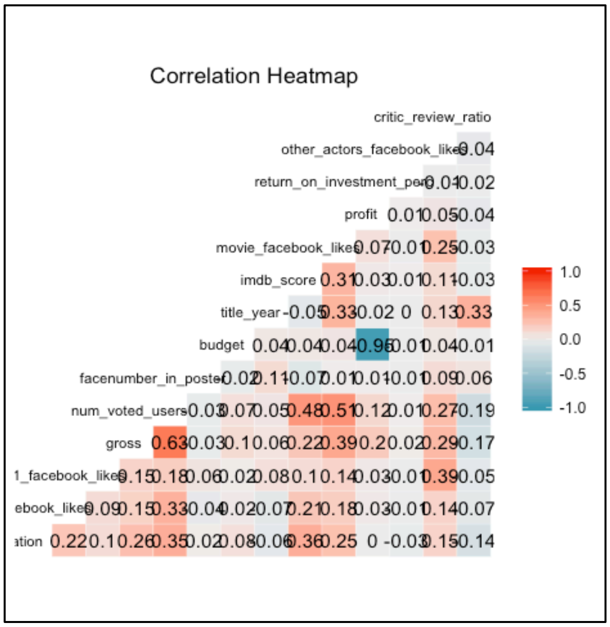
The names of the actors and directors is so distinct that they can't possibly contribute towards predicting.

2. The movie IMDB link is redundant.
3. To avoid multicollinearity, we remove the 2 previously added variables
4. Plotting the map for the remaining values



- Based on the heatmap, we can see some high correlations (>0.7) between predictors. The highest correlation value observed is 0.95 and we can see that actor_1_facebook_likes is highly correlated with the cast_total_facebook_likes and both actor2 and actor3 are also correlated to the total. Thus we modify them into 2 variables: actor_1_facebook_likes and other_actors_facebook_likes.

5. Plotting the heatmap again post deleting the data




No strong correlation of value greater than 0.7 observed.

6. The aim is to build a project wherein the model predicts whether the movie is good or bad. So, bin the scores in four buckets: less than 4(Bad), 4-6(OK), 6-8(Good) and 8-10(Excellent)

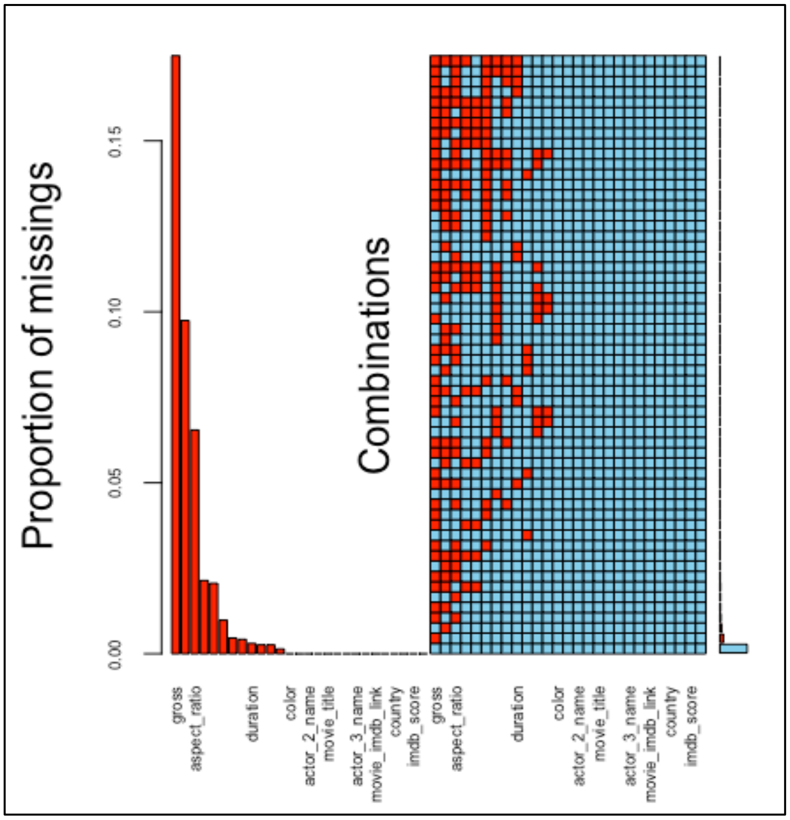
## III. Data Preparation and Preprocessing

We use various methods for data processing and pre-processing, to remove unnecessary data / data with special characters.

We follow the process:

1. Look for duplicates and delete them – remake the data frame by adding only unique values.
2. Movie title across the data has a special character which needs to be corrected- remove the special character and remake the data frame.
3. The genres are widespread and for better understanding we split the genres into multiple data frames. We find the IMDB scores for each of the genres and plot the means in a Bar-plot.
4. We see that the means are all in the range 6-8 and thus it can be concluded that genres is not an impacting factor towards the IMDB score.

5. We then find the aggregate of NAs in the data – plot a heatmap to visualize all the missing values.




```
> colSums(sapply(movie_data, is.na))
                    color             director_name   num_critic_for_reviews                duration
                        0                         0                        1                       1
  director_facebook_likes    actor_3_facebook_likes             actor_2_name  actor_1_facebook_likes
                        0                        10                        0                       3
                    gross              actor_1_name              movie_title          num_voted_users
                        0                         0                        0                       0
cast_total_facebook_likes              actor_3_name     facenumber_in_poster           plot_keywords
                        0                         0                        6                       0
          movie_imdb_link      num_user_for_reviews                 language                 country
                        0                         0                        0                       0
           content_rating                    budget               title_year  actor_2_facebook_likes
                        0                         0                        0                       5
               imdb_score              aspect_ratio      movie_facebook_likes
                        0                        74                        0
```

The number of NAs after removing data.

6. We observe that aspect_ratio still consists of NAs and thus we replace all NAs with 0.
7. Find the aspect_ratio, it is observed the 1.85 and 2.35 are the most common ones and thus we find the mean of values with the same aspect ratio.

```
> mean(movie_data$imdb_score[movie_data$aspect_ratio == 1.85])
[1] 6.373938
>
> mean(movie_data$imdb_score[movie_data$aspect_ratio == 2.35])
[1] 6.508471
```

8. The mean where the aspect_ratio isn't 1.85 and 2.35

```
> mean(movie_data$imdb_score[movie_data$aspect_ratio != 1.85 & movie_data$aspect_ratio != 2.35])
[1] 6.672519
```

9. The mean in either of the cases isn't deviating much and it can be assumed that removing this variable will not affect our analysis.
10. For the variable faceumber_in_poster, we replace the NAs with the column mean and the 0s in the column are replaced with NAs
11. Convert all the 0s in the data to NA
12. Replacing NA in num_critic_for_reviews, duration, director_facebook_likes, actor_3_facebook_likes , actor_1_facebook_, cast_total_facebook_likes , actor_2_facebook_likes , movie_facebook_likes with the average of the column
13. Blanks are to be considered as missing values, removing all of them.
14. Observing the content rating :

```
> table(movie_data$content_rating)

         Approved        G       GP        M    NC-17 Not Rated   Passed       PG    PG-13        R    TV-14
      51       17       91        1        2        6        42        3      573     1314     1723        0
    TV-G    TV-MA    TV-PG     TV-Y    TV-Y7  Unrated         X
       0        0        0        0        0       24        10
```

15. On evaluating content ratings, we observe M = GP = PG, X = NC-17. Replace M and GP with PG and replace X with NC-17
16. Replace "Approved", "Not Rated", "Passed", "Unrated" with the most common rating "R"

```
> table(movie_data$content_rating)

    G NC-17    PG PG-13     R
   91    16   576  1314  1809
```

The new data for content rating.

17. Evaluating the color of the movies:

| | Black and White | Color |
|---|---|---|
| 2 | 124 | 3680 |

It can be observed that the data in color is completely partial towards colored movies and thus it is not an influential factor and we can remove it.

18. Observing the language data

```
> table(movie_data$language)

          Aboriginal     Arabic    Aramaic    Bosnian  Cantonese    Chinese      Czech     Danish       Dari      Dutch
         2         2          1          1          1          7          0          1          3          2          3
  Dzongkha   English   Filipino     French     German      Greek     Hebrew      Hindi  Hungarian  Icelandic Indonesian
         0      3644          1         34         11          0          2          5          1          0          2
   Italian  Japanese    Kannada     Kazakh     Korean   Mandarin       Maya  Mongolian       None  Norwegian    Panjabi
         7        10          0          1          5         14          1          1          1          4          0
   Persian    Polish Portuguese   Romanian    Russian  Slovenian    Spanish    Swahili    Swedish      Tamil     Telugu
         3         0          5          1          1          0         24          0          0          0          0
      Thai      Urdu Vietnamese       Zulu
         3         0          1          1
```

It can be observed that the data in languages is completely partial towards English movies and thus it is not an influential factor and we can remove it.

19. Checking if the country the movie is produced in is an influential factor towards it's score

```
                              Afghanistan            Argentina                Aruba            Australia              Bahamas
                   0                    1                    3                    1                   40                    0
             Belgium               Brazil             Bulgaria             Cambodia             Cameroon               Canada
                   1                    5                    0                    0                    0                   63
               Chile                China             Colombia       Czech Republic              Denmark   Dominican Republic
                   1                   13                    1                    3                    9                    0
               Egypt              Finland               France              Georgia              Germany               Greece
                   0                    1                  103                    1                   79                    1
           Hong Kong              Hungary              Iceland                India            Indonesia                 Iran
                  13                    2                    1                    5                    1                    4
             Ireland               Israel                Italy                Japan                Kenya           Kyrgyzstan
                   7                    2                   11                   15                    0                    0
               Libya               Mexico          Netherlands             New Line          New Zealand              Nigeria
                   0                   10                    3                    1                   11                    0
              Norway        Official site             Pakistan               Panama                 Peru          Philippines
                   4                    1                    0                    0                    1                    1
              Poland              Romania               Russia             Slovakia             Slovenia         South Africa
                   1                    2                    3                    0                    0                    3
         South Korea         Soviet Union                Spain               Sweden          Switzerland               Taiwan
                   8                    0                   22                    0                    0                    2
            Thailand               Turkey                   UK United Arab Emirates                  USA         West Germany
                   4                    0                  316                    0                 3025                    1
```

Approximately 79% movies are form the US, 8% from UK and 13% from other countries. Thus, we collectively represent the movie locations as: US, UK, Others.

```
> table(movie_data$country)

   UK    USA Others
  316   3025    465
>
```

20. To avoid multicollinearity, we remove the 2 previously added variables.

# IV. Data Mining Techniques and Implementation
&
# V. Performance Evaluation

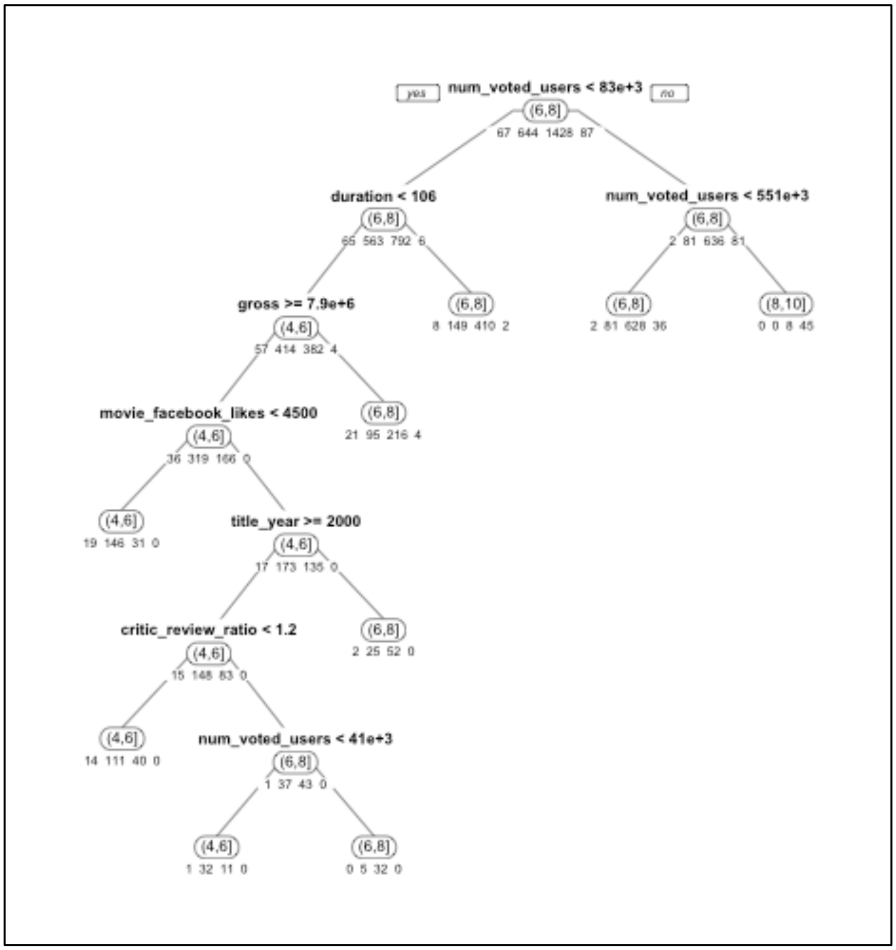
To apply models, splitting the data into training, validation and test sets with the ratio of 6:2:2

CLASSIFICATION TREE

Plotting a full-grown tree:

```
Classification tree:
rpart(formula = binned_score ~ . - imdb_score, data = train,
    method = "class", cp = 1e-05, minsplit = 5, xval = 5)

Variables actually used in tree construction:
 [1] actor_1_facebook_likes      budget                     content_rating             country
 [5] critic_review_ratio         director_facebook_likes    duration                   facenumber_in_poster
 [9] gross                       movie_facebook_likes       num_voted_users            other_actors_facebook_likes
[13] profit                      return_on_investment_perc  title_year

Root node error: 798/2226 = 0.35849
```

```
Root node error: 798/2226 = 0.35849

n= 2226

           CP nsplit rel error  xerror     xstd
1  0.06390977      0   1.00000 1.00000 0.028353
2  0.04636591      3   0.80827 0.86216 0.027322
3  0.01691729      4   0.76190 0.79574 0.026697
4  0.00751880      8   0.69424 0.77694 0.026503
5  0.00626566     10   0.67920 0.77318 0.026464
6  0.00563910     13   0.66040 0.75689 0.026289
7  0.00543024     15   0.64912 0.75689 0.026289
8  0.00501253     20   0.61278 0.75439 0.026262
9  0.00407268     25   0.58772 0.76817 0.026411
10 0.00375940     29   0.57143 0.78195 0.026556
11 0.00325815     45   0.50877 0.79198 0.026659
12 0.00313283     50   0.49248 0.79198 0.026659
13 0.00292398     52   0.48622 0.79699 0.026709
14 0.00250627     55   0.47744 0.79825 0.026722
15 0.00187970    104   0.34461 0.80702 0.026809
16 0.00167084    131   0.27945 0.80326 0.026772
17 0.00156642    143   0.25940 0.80326 0.026772
18 0.00125313    152   0.24436 0.82957 0.027026
19 0.00093985    203   0.17794 0.83208 0.027049
20 0.00083542    207   0.17419 0.82957 0.027026
21 0.00075188    210   0.17168 0.82957 0.027026
22 0.00062657    215   0.16792 0.84586 0.027176
23 0.00027847    221   0.16416 0.84962 0.027210
24 0.00020886    230   0.16165 0.85464 0.027255
25 0.00001000    236   0.16040 0.85589 0.027266
```

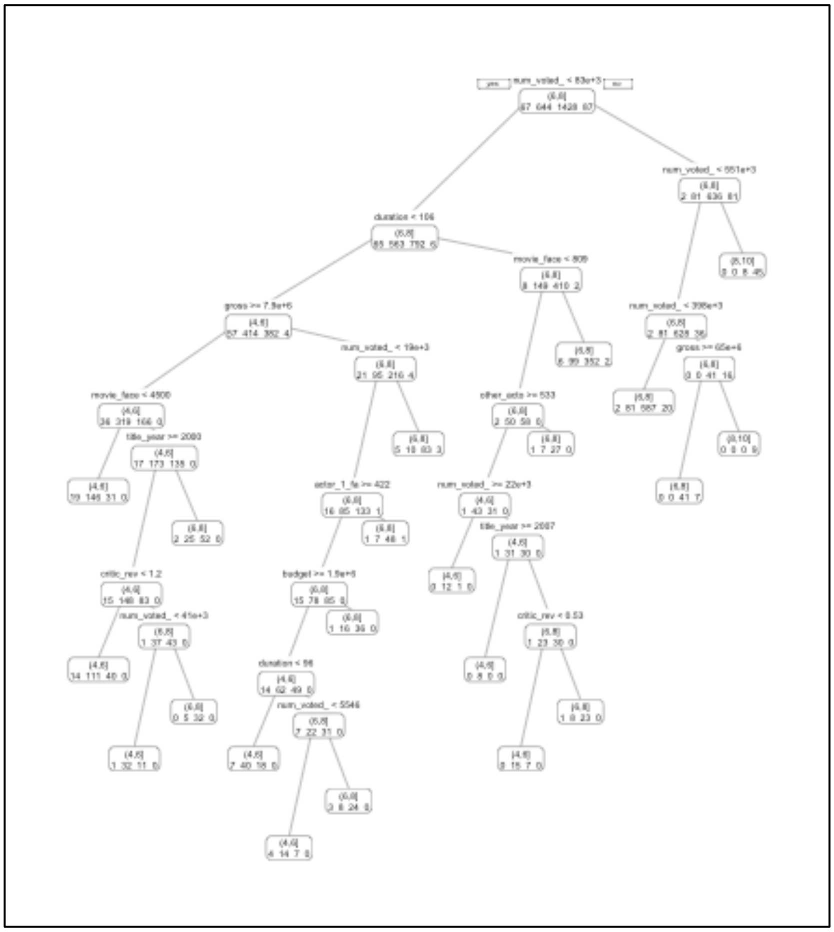



We plot pruned trees as well:

The length of the pruned tree is 21

On Test Data:

```
Confusion Matrix and Statistics

          Reference
Prediction (0,4] (4,6] (6,8] (8,10]
    (0,4]      0     0     0      0
    (4,6]     45   378   115      0
    (6,8]     22   266  1305     33
    (8,10]     0     0     8     54

Overall Statistics

               Accuracy : 0.7803
                 95% CI : (0.7625, 0.7974)
    No Information Rate : 0.6415
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5228

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: (0,4] Class: (4,6] Class: (6,8] Class: (8,10]
Sensitivity                0.0000       0.5870       0.9139       0.62069
Specificity                1.0000       0.8989       0.5977       0.99626
Pos Pred Value                NaN       0.7026       0.8026       0.87097
Neg Pred Value             0.9699       0.8424       0.7950       0.98475
Prevalence                 0.0301       0.2893       0.6415       0.03908
Detection Rate             0.0000       0.1698       0.5863       0.02426
Detection Prevalence       0.0000       0.2417       0.7305       0.02785
Balanced Accuracy          0.5000       0.7429       0.7558       0.80847
```



On Validation set:

```
Confusion Matrix and Statistics

          Reference
Prediction (0,4] (4,6] (6,8] (8,10]
    (0,4]      0     0     0      0
    (4,6]      9    88    62      0
    (6,8]      6   121   424     10
    (8,10]     0     0     5     17

Overall Statistics

               Accuracy : 0.7129
                 95% CI : (0.6789, 0.7453)
    No Information Rate : 0.6617
    P-Value [Acc > NIR] : 0.001616

                  Kappa : 0.345

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: (0,4] Class: (4,6] Class: (6,8] Class: (8,10]
Sensitivity               0.00000       0.4211       0.8635       0.62963
Specificity               1.00000       0.8668       0.4542       0.99301
Pos Pred Value                NaN       0.5535       0.7558       0.77273
Neg Pred Value            0.97978       0.7925       0.6298       0.98611
Prevalence                0.02022       0.2817       0.6617       0.03639
Detection Rate            0.00000       0.1186       0.5714       0.02291
Detection Prevalence      0.00000       0.2143       0.7561       0.02965
Balanced Accuracy         0.50000       0.6439       0.6589       0.81132
```

On Test data:

```
Confusion Matrix and Statistics

          Reference
Prediction (0,4] (4,6] (6,8] (8,10]
    (0,4]      0     0     0      0
    (4,6]      8   107    76      0
    (6,8]      5   105   423     10
    (8,10]     0     0     1      8

Overall Statistics

               Accuracy : 0.7241
                 95% CI : (0.6904, 0.756)
    No Information Rate : 0.6729
    P-Value [Acc > NIR] : 0.001485

                  Kappa : 0.3651

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: (0,4] Class: (4,6] Class: (6,8] Class: (8,10]
Sensitivity                0.0000       0.5047       0.8460       0.44444
Specificity                1.0000       0.8418       0.5062       0.99862
Pos Pred Value                NaN       0.5602       0.7790       0.88889
Neg Pred Value             0.9825       0.8098       0.6150       0.98638
Prevalence                 0.0175       0.2853       0.6729       0.02423
Detection Rate             0.0000       0.1440       0.5693       0.01077
Detection Prevalence       0.0000       0.2571       0.7308       0.01211
Balanced Accuracy          0.5000       0.6733       0.6761       0.72153
```

## <u>K NEAREST NEIGHBOR</u>

Training data accuracy

```
> accuracy.df
    k  accuracy
1   1 0.6671159
2   2 0.6293801
3   3 0.6981132
4   4 0.6940701
5   5 0.6994609
6   6 0.6900270
7   7 0.6886792
8   8 0.6913747
9   9 0.7008086
10 10 0.7048518
11 11 0.6954178
12 12 0.6967655
13 13 0.6954178
14 14 0.6765499
15 15 0.6994609
16 16 0.6927224
17 17 0.6981132
18 18 0.6886792
19 19 0.6927224
20 20 0.6954178
```

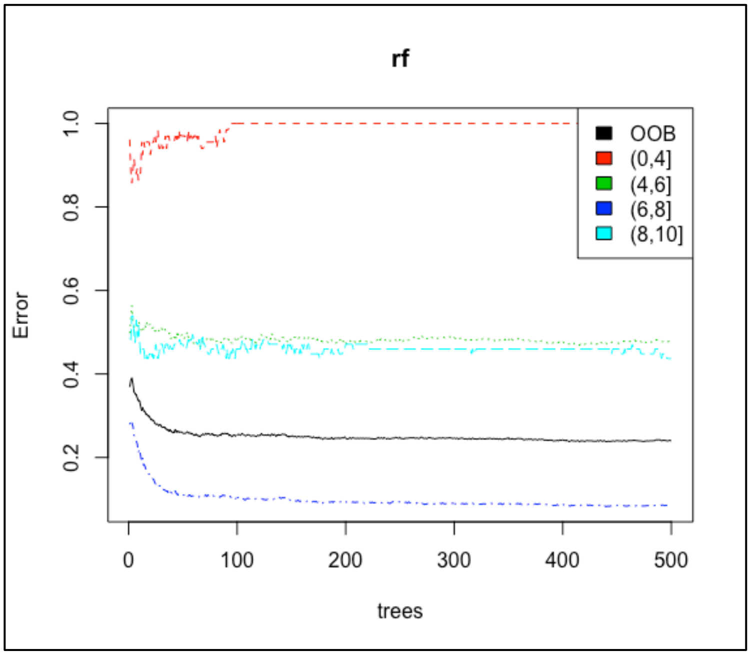
Validation data accuracy

```
> accuracy.df
    k  accura
1   1 0.66711
2   2 0.64555
3   3 0.70080
4   4 0.68867
5   5 0.70080
6   6 0.70350
7   7 0.69002
8   8 0.70485
9   9 0.70754
10 10 0.70485
11 11 0.69676
12 12 0.69002
13 13 0.69407
14 14 0.68328
15 15 0.69676
16 16 0.69272
```

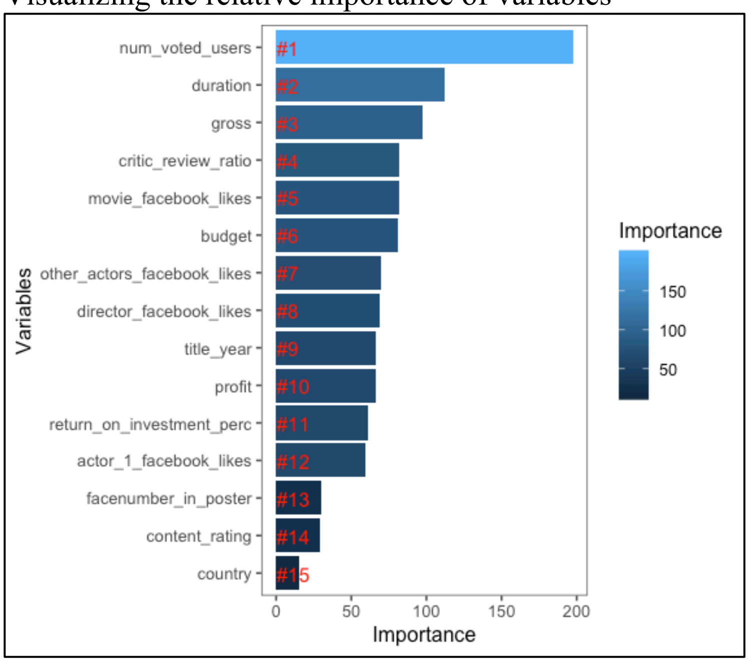
Accuracy of confusion Matrix:
Accuracy
0.7456258

RANDOM FOREST

Visualizing the model error




Visualizing the relative importance of variables



Finding the accuracy of the model

```
Confusion Matrix and Statistics

          Reference
Prediction (0,4] (4,6] (6,8] (8,10]
    (0,4]      0     0     0      0
    (4,6]     10   114    41      0
    (6,8]      5    95   449     12
    (8,10]     0     0     1     15

Overall Statistics

               Accuracy : 0.779
                 95% CI : (0.7474, 0.8083)
    No Information Rate : 0.6617
    P-Value [Acc > NIR] : 1.804e-12

                  Kappa : 0.4934

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: (0,4] Class: (4,6] Class: (6,8] Class: (8,10]
Sensitivity               0.00000       0.5455       0.9145       0.55556
Specificity               1.00000       0.9043       0.5538       0.99860
Pos Pred Value                NaN       0.6909       0.8004       0.93750
Neg Pred Value            0.97978       0.8354       0.7680       0.98347
Prevalence                0.02022       0.2817       0.6617       0.03639
Detection Rate            0.00000       0.1536       0.6051       0.02022
Detection Prevalence      0.00000       0.2224       0.7561       0.02156
Balanced Accuracy         0.50000       0.7249       0.7341       0.77708
```

## VI. Discussion and Recommendation

Observing the accuracy, we can see that:

| Dataset | Decision Tree | KNN | Random Forest |
|---|---|---|---|
| Training | 0.7803 | | |
| Validation | 0.7126 | 0.7143 | 0.7654 |
| Test | 0.7241 | 0.7456 | 0.779 |

We thus decide random forest is the best algorithm to use in this case.

## VII. Summary

The dataset left after pre-processing and removing all the unnecessary data, we find the accuracy of a classification model using classification tree/pruned tree, K Nearest Neighbor and Random Forest. We found Random forest to give the highest accuracy and thus we choose this as the appropriate model.

## Appendix: R Code for use case study

```
#Load Packages
install.packages("ggplot2")
library(ggplot2)
install.packages("ggrepel")
library(ggrepel)
install.packages("ggthemes")
library(ggthemes)
install.packages("scales")
```

```
library(scales)
install.packages("dplyr")
library(dplyr)
install.packages("VIM")
library(VIM)
install.packages("data.table")
library(data.table)
install.packages("formattable")
library(formattable)
install.packages("plotly")
library(plotly)
install.packages("corrplot")
library(corrplot)
install.packages("GGally")
library(GGally)
install.packages("caret")
library(caret)
install.packages("car")
library(car)

#Read Data

movie_data <- read.csv("~/Desktop/movie_metadata.csv", header=TRUE)
str(movie_data)


# DATA EXPLORATION

# Look for duplicates and delete them
sum(duplicated(movie_data))
movie_data <- movie_data[!duplicated(movie_data),]

# Tody the movie title- garbage found before the actual name

library(stringr)
movie_data$movie_title <- gsub("Â", "", as.character(factor(movie_data$movie_title)))
str_trim(movie_data$movie_title, side = "right")

# Check all genres of the movies

head(movie_data$genres)

# Create a dataframe to store the substrings
genres.df <- as.data.frame(movie_data[,c("genres", "imdb_score")])


# Separate different genres
```

```
genres.df$Action <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Action") 1 else 0)
genres.df$Action <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Action") 1 else 0)
genres.df$Adventure <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Adventure") 1 else 0)
genres.df$Animation <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Animation") 1 else 0)
genres.df$Biography <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Biography") 1 else 0)
genres.df$Comedy <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Comedy") 1 else 0)
genres.df$Crime <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Crime") 1 else 0)
genres.df$Documentary <- sapply(1:length(genres.df$genres), function(x) if
 (genres.df[x,1] %like% "Documentary") 1 else 0)
genres.df$Drama <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Drama") 1 else 0)
genres.df$Family <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Family") 1 else 0)
genres.df$Fantasy <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Fantasy") 1 else 0)
genres.df$`Film-Noir` <- sapply(1:length(genres.df$genres), function(x) if
 (genres.df[x,1] %like% "Film-Noir") 1 else 0)
genres.df$History <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "History") 1 else 0)
genres.df$Horror <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Horror") 1 else 0)
genres.df$Musical <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Musical") 1 else 0)
genres.df$Mystery <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Mystery") 1 else 0)
genres.df$News <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "News") 1 else 0)
genres.df$Romance <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Romance") 1 else 0)
genres.df$`Sci-Fi` <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Sci-Fi") 1 else 0)
genres.df$Short <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Short") 1 else 0)
genres.df$Sport <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Sport") 1 else 0)
genres.df$Thriller <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Thriller") 1 else 0)
genres.df$War <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "War") 1 else 0)
genres.df$Western <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1]
 %like% "Western") 1 else 0)
```

```
# Find the mean of imdb score for different genres

means <- rep(0,23)
for (i in 1:23) {
  means[i] <- mean(genres.df$imdb_score[genres.df[i+2]==1])
}

# Plot the Means

barplot(means, main = "Mean od the imdb scores for different genres")

# All means are in the range of 6-8, it can be assumed that not a lot of difference will be
 made to the
#IMDB score if genres were present

movie_data <- subset(movie_data, select = -c(genres))

# Making sure Genres returns a NULL
str(movie_data$genres)

# Find Aggregate of NAs in all columns

colSums(sapply(movie_data, is.na))

# Plotting a heat map to visualize the missing values

missing.values <- aggr(movie_data, sortVars = T, prop = T, sortCombs = T, cex.lab =
 1.5, cex.axis = .6, cex.numbers = 5, combined = F, gap = -.2)

# Gross and Budget have the highest amount of missing values but both of them are
 important factors in determing
# the IMDB score and thus we remove only the rows which have NA in them

movie_data <- movie_data[!is.na(movie_data$gross),]
movie_data <- movie_data[!is.na(movie_data$budget),]

# Checking how much was our daat affected due to removing rows

dim(movie_data)

# 23% of data removed, still consists of 3857 records for analysis
# Rechecking the number of NAs

sum(complete.cases(movie_data))

colSums(sapply(movie_data, is.na))
```

```
# aspect ratio has the highest number of NAs, checking how important aspect ration in
 prediction is

table(movie_data$aspect_ratio)

# Replacing NAs in aspect ration with 0

movie_data$aspect_ratio[is.na (movie_data$aspect_ratio)] <- 0

# Checking mean where aspect ratio is 1.85 and 2.35
mean(movie_data$imdb_score[movie_data$aspect_ratio == 1.85])

mean(movie_data$imdb_score[movie_data$aspect_ratio == 2.35])

# Checking mean where aspect ratio is not 1.85 and 2.35
mean(movie_data$imdb_score[movie_data$aspect_ratio != 1.85 &
 movie_data$aspect_ratio != 2.35])

# Observed: The mean in either of the cases isn't deviating much and it can be assumed
 tht removing this variable
# will not affect our analysis

movie_data <- subset(movie_data, select = -c(aspect_ratio))

# Rechecking if the aspect ratio is still present or is NULL

str(movie_data$aspect_ratio)

# Replacing NAs and 0s in the Data

# Replacing NA in facenumber_in_poster with the average of the column

movie_data$facenumber_in_poster[is.na(movie_data$facenumber_in_poster)] <-
 round(mean(movie_data$facenumber_in_poster, na.rm = TRUE))

# Convert 0s in the data to NAs
movie_data[,c(5,6,8,13,24,26)][movie_data[,c(5,6,8,13,24,26)] == 0] <- NA

# Replacing NA in num_critic_for_reviews with the average of the column

movie_data$num_critic_for_reviews[is.na(movie_data$num_critic_for_reviews)] <-
 round(mean(movie_data$num_critic_for_reviews, na.rm = TRUE))

# Replacing NA in duration with the average of the column

movie_data$duration[is.na(movie_data$duration)] <- round(mean(movie_data$duration,
 na.rm = TRUE))
```

```
# Replacing NA in director_facebook_likes with the average of the column

movie_data$director_facebook_likes[is.na(movie_data$director_facebook_likes)] <-
 round(mean(movie_data$director_facebook_likes, na.rm = TRUE))

# Replacing NA in actor_3_facebook_likes with the average of the column

movie_data$actor_3_facebook_likes[is.na(movie_data$actor_3_facebook_likes)] <-
 round(mean(movie_data$actor_3_facebook_likes, na.rm = TRUE))

# Replacing NA in actor_1_facebook_likes with the average of the column

movie_data$actor_1_facebook_likes[is.na(movie_data$actor_1_facebook_likes)] <-
 round(mean(movie_data$actor_1_facebook_likes, na.rm = TRUE))

# Replacing NA in cast_total_facebook_likes with the average of the column

movie_data$cast_total_facebook_likes[is.na(movie_data$cast_total_facebook_likes)] <-
 round(mean(movie_data$cast_total_facebook_likes, na.rm = TRUE))

# Replacing NA in actor_2_facebook_likes with the average of the column

movie_data$actor_2_facebook_likes[is.na(movie_data$actor_2_facebook_likes)] <-
 round(mean(movie_data$actor_2_facebook_likes, na.rm = TRUE))

# Replacing NA in movie_facebook_likes with the average of the column

movie_data$movie_facebook_likes[is.na(movie_data$movie_facebook_likes)] <-
 round(mean(movie_data$movie_facebook_likes, na.rm = TRUE))

# Finding the missing values in content rating

table(movie_data$content_rating)

# Blanks are to be considered as missing values

movie_data <- movie_data[!(movie_data$content_rating %in% ""),]

# Re-evaluating content ratings
# M = GP = PG, X = NC-17. Replace M and GP with PG and replace X with NC-17

movie_data$content_rating
movie_data$content_rating[movie_data$content_rating == 'M']   <- 'PG'
movie_data$content_rating[movie_data$content_rating == 'GP']  <- 'PG'
movie_data$content_rating[movie_data$content_rating == 'X']   <- 'NC-17'
# Replace “Approved”, “Not Rated”, “Passed”, “Unrated” with the most common rating
 “R”
```

```
movie_data$content_rating[movie_data$content_rating == 'Approved']  <- 'R'
movie_data$content_rating[movie_data$content_rating == 'Not Rated'] <- 'R'
movie_data$content_rating[movie_data$content_rating == 'Passed']    <- 'R'
movie_data$content_rating[movie_data$content_rating == 'Unrated']   <- 'R'
movie_data$content_rating <- factor(movie_data$content_rating)

table(movie_data$content_rating)

# Creating 2 columns profit and percentage of return based on gross and budget
movie_data<- movie_data %>%
mutate(profit = gross - budget,
return_on_investment_perc = (profit/budget)*100)

# Checking if movie color is an influential factor towards it's score

table(movie_data$color)

# It can be observed that the data in color is completely partial towards colored movies
# and thus it is not an influential factor and we can remove it
movie_data <- subset(movie_data, select = -c(color))

# Checking if color is removed from the data and returns a NULL value
movie_data$color

# Checking if movie color is an influential factor towards it's score

table(movie_data$language)

# It can be observed that the data in langauges is completely partial towards english
 movies
# and thus it is not an influential factor and we can remove it

movie_data <- subset(movie_data, select = -c(language))

# Checking if language is removed from the data and returns a NULL value
movie_data$language

# Checking if the country the movie is produced in is an influential factor towards it's
 score

table(movie_data$country)

# Approximately 79% movies are form the US, 8% from UK and 13% from other
 countries
# Thus we collectovely represent the movie locations as: US, UK, Others

levels(movie_data$country) <- c(levels(movie_data$country), "Others")
```

```
movie_data$country[(movie_data$country != 'USA')&(movie_data$country != 'UK')] <-
 'Others'
movie_data$country <- factor(movie_data$country)

# Checking if only 3 locations are available

table(movie_data$country)


# DATA VISUALIZATION

# Histogram of movies released each year
ggplot(movie_data, aes(title_year)) +
  geom_bar() +
  labs(x = "Year movie release", y = "No. of movies released", title = "Histogram of no.
 of movies released each year") +
  theme(plot.title = element_text(hjust = 0.5))

# It can be seen that the movie boom came after 1980 and thus we represent the data only
 after 1980
movie_data <- movie_data[movie_data$title_year >= 1980,]

# Visualizing top 20 movies based on profits in Million$
install.packages("ggrepel")
library(ggrepel)
movie_data %>%
  filter(title_year %in% c(2000:2016)) %>%
  arrange(desc(profit)) %>%
  top_n(20, profit) %>%
  ggplot(aes(x=budget/1000000, y=profit/1000000)) +
  geom_point() +
  geom_smooth() +
  geom_text_repel(aes(label=movie_title)) +
  labs(x = "Budget in $million", y = "Profit in $million", title = "Top 10 Profitable
 Movies") +
  theme(plot.title = element_text(hjust = 0.5))
# Using profits and return on investment variables are criteria to find 20 most profitable
 movies

movie_data %>%
  filter(budget > 100000) %>%
  mutate(profit = gross - budget,
         return_on_investment_perc = (profit/budget)*100) %>%
  arrange(desc(profit)) %>%
  top_n(20, profit) %>%
  ggplot(aes(x=budget/1000000, y = return_on_investment_perc)) +
  geom_point(size = 2) +
  geom_smooth(size = 1) +
```

```
  geom_text_repel(aes(label = movie_title), size = 3) +
  xlab("Budget in $million") +
  ylab("Percent of Return on Investment") +
  ggtitle("20 Most Profitable Movies based on their Return on Investment")


# Visualizing 20 top directors based on the highest IMDB scores
install.packages("formattable")
library(formattable)
movie_data %>%
  group_by(director_name) %>%
  summarise(avg_imdb = mean(imdb_score)) %>%
  arrange(desc(avg_imdb)) %>%
  top_n(20, avg_imdb) %>%
  formattable(list(avg_imdb = color_bar("Red")), align = 'l')

# Plotting commerical success vs critical acclaim
movie_data %>%
  top_n(20, profit) %>%
  ggplot(aes(x = imdb_score, y = gross/10^6, size = profit/10^6, color = content_rating))
 +
  geom_point() +
  geom_hline(aes(yintercept = 600)) +
  geom_vline(aes(xintercept = 7.75)) +
  geom_text_repel(aes(label = movie_title), size = 4) +
  xlab("IMDB Score") +
  ylab("Gross Money earned in million$") +
  ggtitle("Commercial Success Vs Critical Acclaim") +
  annotate("text", x = 8.5, y = 700, label = "High Ratings \n & High Gross") +
  theme(plot.title = element_text(hjust = 0.5))

# The above observation shows that there is hardly any correlation between critical
 acclaim and the movie's commercial success
# Visualizing relation between facebook likes and IMDB scores

library(plotly)
movie_data %>%
  plot_ly(x = ~movie_facebook_likes, y = ~imdb_score, color = ~content_rating , mode =
 "markers",
        text = ~content_rating, alpha = 0.7, type = "scatter")

# Movies with high facebook likes can be seen to have higher IMDB score


# DATA PRE-PROCESSING

install.packages("data.table")
library(data.table)
```

```
# Find number of directors
sum(uniqueN(movie_data$director_name))

# Find number of actors
sum(uniqueN(movie_data[, c("actor_1_name", "actor_2_name", "actor_3_name")]))

# The names of the directors, actors 1 2 3 are so different that it will not contribute in
 predicting the score.
# The plot keyword is too diverse to be used as a predictor
# The movie IMDB link is redundant

movie_data <- subset(movie_data, select = -c(director_name, actor_2_name,
 actor_1_name,
                         movie_title, actor_3_name, plot_keywords,
                         movie_imdb_link))

# To avoid multicollinearity we remove the 2 previously added variables

movie_data <- subset(movie_data, select = -c(profit, return_on_investment_perc))

# Plot heatmap of the entire data as of now

ggcorr(movie_data, label = TRUE, label_round = 2, label_size = 3.5, size = 2, hjust =
 .85) +
  ggtitle("Correlation Heatmap") +
  theme(plot.title = element_text(hjust = 0.5))

# Based on the heatmap, we can see some high correlations (>0.7) between predictors.
# The highest correlation value observed is 0.95 and we can see that
 actor_1_facebook_likes is highly correlated with the cast_total_facebook_likes
# and both actor2 and actor3 are also correlated to the total.
# Thus we modify them into 2 variables: actor_1_facebook_likes and
 other_actors_facebook_likes.

movie_data$other_actors_facebook_likes <- movie_data$actor_2_facebook_likes +
 movie_data$actor_3_facebook_likes

# There is high correlations among num_voted_users, num_user_for_reviews and
 num_critic_for_reviews.
# We want to keep num_voted_users and take the ratio of num_user_for_reviews and
 num_critic_for_reviews.

movie_data$critic_review_ratio <- movie_data$num_critic_for_reviews /
 movie_data$num_user_for_reviews

# Delete Columns
```

```
movie_data <- subset(movie_data, select = -c(cast_total_facebook_likes,
 actor_2_facebook_likes, actor_3_facebook_likes,
                         num_critic_for_reviews, num_user_for_reviews))

# Plotting heatmap to review post changes

ggcorr(movie_data, label = TRUE, label_round = 2, label_size = 4, size = 3, hjust = .85)
 +
  ggtitle("Correlation Heatmap") +
  theme(plot.title = element_text(hjust = 0.5))

# No strong correlation of value greater than 0.7 observed

# The aim is to build a project wherein the model predicts whether the movie is good or
 bad. So bin the scores in four buckets: less than 4(Bad),
# 4-6(OK), 6-8(Good) and 8-10(Excellent)

movie_data$binned_score <- cut(movie_data$imdb_score, breaks = c(0,4,6,8,10))

# Rearranging the data and renaming the column to make it readable

movie_data <- movie_data[,c(9,4,5,14,12,2,3,13,1,6,10,7,8,11,15)]
colnames(movie_data) <- c("budget", "gross", "user_vote", "critic_review_ratio",
                "movie_fb", "director_fb", "actor1_fb", "other_actors_fb",
                "duration", "face_number", "year", "country", "content",
                "imdb_score", "binned_score")

# To apply models, spliting the data into training, validation and test sets with the ratio of
 6:2:2
set.seed(45)
train.index <- sample(row.names(movie_data), dim(movie_data)[1]*0.6)
valid.index <- sample(setdiff(row.names(movie_data), train.index),
 dim(movie_data)[1]*0.2)
test.index <- setdiff(row.names(movie_data), union(train.index, valid.index))
train <- movie_data[train.index, ]
valid <- movie_data[valid.index, ]
test <- movie_data[test.index, ]

# IMPLEMNETATION OF ALGORITHMS

# CLASSIFICATION TREE

# Implementing a full grown tree

library(rpart)
library(rpart.plot)
# Full grown tree
class.tree <- rpart(binned_score ~ . -imdb_score, data = train, method = "class")
```

```
## plot tree
prp(class.tree, type = 1, extra = 1, under = TRUE, split.font = 2, varlen = 0)

# Implementing Best pruned tree

set.seed(51)
cv.ct <- rpart(binned_score ~ . -imdb_score, data = train, method = "class",
            cp = 0.00001, minsplit = 5, xval = 5)
printcp(cv.ct)
pruned.ct <- prune(cv.ct,
               cp = cv.ct$cptable[which.min(cv.ct$cptable[,"xerror"]),"CP"])
length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])
prp(pruned.ct, type = 1, extra = 1, split.font = 1, varlen = -10)

# Apply model on training set
tree.pred.train <- predict(pruned.ct, train, type = "class")
# Generate confusion matrix for training data
confusionMatrix(tree.pred.train, train$binned_score)

# Apply model on validation set
tree.pred.valid <- predict(pruned.ct, valid, type = "class")
# Generate confusion matrix for validation data
confusionMatrix(tree.pred.valid, valid$binned_score)

# Apply model on test set
tree.pred.test <- predict(pruned.ct, test, type = "class")
# Generate confusion matrix for test data
confusionMatrix(tree.pred.test, test$binned_score)


# K NEAREST NEIGHBOUR

library(FNN)
# Using model.matrix() to create dummy variables for country and content
movie_data2 <- movie_data
movie_data2$country <- as.factor(movie_data2$country)
movie_data2$content <- as.factor(movie_data2$content)
movie_data2[,c("country_UK", "country_USA", "country_Others")] <- model.matrix( ~
 country - 1, data = movie_data2)
movie_data2[,c("content_G", "content_NC17", "content_PG", "content_PG13",
 "content_R")] <- model.matrix( ~ content - 1, data = movie_data2)

# Select useful variables for future prediction
movie_data2 <- movie_data2[, c(1,2,3,4,5,6,7,8,9,10,11,16,17,18,19,20,21,22,23,15)]
# Partition the data into training and validation sets
set.seed(52)
train2 <- movie_data2[train.index, ]
valid2 <- movie_data2[valid.index, ]
```

```
test2 <- movie_data2[test.index, ]

# Initializing normalized training, validation, test data, complete data frames to originals
train2.norm <- train2
valid2.norm <- valid2
test2.norm <- test2
movie_data2.norm <- movie_data2

# Using preProcess() from the caret package to normalize predictors
norm.values <- preProcess(train2[, -20], method=c("center", "scale"))
train2.norm[, -20] <- predict(norm.values, train2[, -20])
valid2.norm[, -20] <- predict(norm.values, valid2[, -20])
test2.norm[, -20] <- predict(norm.values, test2[, -20])
movie_data2.norm[, -20] <- predict(norm.values, movie_data2[, -20])

# Finding the best K
# Initialize a data frame with two columns: k, and accuracy

accuracy.df <- data.frame(k = seq(1, 20, 1), accuracy = rep(0, 20))

# Computing knn for different k on validation data

for(i in 1:20) {
  knn.pred <- knn(train2.norm[, -20], valid2.norm[, -20],
                  cl = train2.norm[, 20], k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred, valid2.norm[, 20])$overall[1]
}
accuracy.df

# Applying model on test set

knn.pred.test <- knn(train2.norm[, -20], test2.norm[, -20],
                     cl = train2.norm[, 20], k = 9)

# Generating confusion matrix for test data

accuracy <- confusionMatrix(knn.pred.test, test2.norm[, 20])$overall[1]
accuracy

# RANDOM FOREST

install.packages("randomForest")
library(randomForest)
set.seed(53)
rf <- randomForest(binned_score ~ . -imdb_score, data = train, mtry = 5)
# Show model error
plot(rf)
legend('topright', colnames(rf$err.rate), col=1:5, fill=1:5)
```

```
# Get importance
importance <- importance(rf)
varImportance <- data.frame(Variables = row.names(importance),
                            Importance = round(importance[ ,'MeanDecreaseGini'],2))

# Creating a rank variable based on importance
rankImportance <- varImportance %>%
  mutate(Rank = paste0('#',dense_rank(desc(Importance))))

# Using ggplot2 to visualize the relative importance of variables
ggplot(rankImportance, aes(x = reorder(Variables, Importance),
                           y = Importance, fill = Importance)) +
  geom_bar(stat='identity') +
  geom_text(aes(x = Variables, y = 0.5, label = Rank),
            hjust=0, vjust=0.55, size = 4, colour = 'red') +
  labs(x = 'Variables') +
  coord_flip() +
  theme_few()

install.packages("caret")
library(caret)
set.seed(632)
# apply model on validation set
rf.pred.valid <- predict(rf, valid)
# generate confusion matrix for validation data
confusionMatrix(rf.pred.valid, valid$binned_score)
```