

Data Mining Project Description

COVID -19 STAY AT HOME: FORECAST MOVIE RATING(SCORE) ON
IMDB

Greeshma Vijayakumar | MIS 545 | 08/30/2020

Contents

Background	2
PROBLEM STATEMENT.....	2
Project Data description	3
Data Exploration	4
Step 1: Loading the dataset.....	4
Step 2: Duplicate Removal	5
Step 3: Cleaning up Movie_Title Field.....	5
Step 4: Genre Categorization	6
Data Cleansing	7
Identify Missing Values.....	7
Add Purposeful Columns	10
Delete Irrelevant Columns.....	10
Data Visualization.....	11
Movies Released: Year	11
Data Pre-Processing	16
Remove Variables	16
Group dependent variable	18
Organize Dataset.....	19
Split Data	19
Data Mining Algorithms- Implementation.....	20
Decision / Classification Tree – Predictive Data Analysis Technique.....	20
Full grown Tree	20
Applying the model – Evaluation Criteria.....	24
Applying the model on the training set	24
Applying the model on validation set	26
Apply model on test data	27
K-Nearest Neighbors – Descriptive Data Mining Technique.....	27
Data Preparation	28
Data Normalization.....	28

Find the Best K using models – Evaluation Criteria	28
Apply model on test-set	29
Random forest – Multitude of decision trees.....	30
Build Model	30
Apply the model – Validation Data Set	32
Apply the model – Test Data set.....	33
Project Data Analysis conclusion	34
References.....	34

BACKGROUND

During the COVID-19 stay at home situation, there has been multiple studies on human behavioral aspects including change in human habits and hobbies on a daily basis with most of the population across the globe either working from home or taking online classes. With respect to the recent data released by Comcast for USA it was noticed that people are actually watching so much more video/movie content on websites and TV that weekday and weekend video content viewing habits are now blurring together. This is from the initial research analysis of what the TV watching trends during COVID-19 are showing, including that households are viewing a full workday's worth of content more each week. Data from Comcast also shows that there is a slow decrease in DVR usage, while at the same time there's a **50% increase in video on demand usage**. There is also comcast data asserting the customer asks on "what to watch" and "surprise me." This suggests that people have watched more of the shows on their lists and are looking for something new. While they are digging in for something new, the data mining project here will help them make this an easier task, as one of the primary deciding factors for watching or reading content on the internet or TV now is review / rating / scoring.

PROBLEM STATEMENT

This individual project covers a data mining report on predicting how we can help the people watch valuable content on TV or the internet by providing them a rating or review score for each movie they wish to watch. Taking into account, the value of people's time during this pandemic I would like to build a credible movie rating or scoring prediction model using the IMDB data set which is one of the top 10 movie scoring websites trusted by the audience. This will help the audience choose their movie watching content wisely without having spent their valuable time on something which would not satisfy their interests or entertainment needs. While we focus on the video content watchers as our primary consumers of this data, we also have to keep in mind the commercial success of a movie generates a tremendous amount of profit for the video makers as well. So, our secondary consumer of this data model would be the film companies / video content creators.

This is an important problem statement especially during these times because based on the massive movie information available on the internet and TV , it would be super useful to develop an understanding of what are the important factors that make a movie more watchable or interesting than others available in the pool. This saves people's time taken to decide on what to watch, help people watch valuable content based on their interest, help makers create audient favorite data etc.

With that said , I would like to focus my data mining algorithms to what kind of movies are more in demand, in other words, what kind of movies can a get higher IMDB score which will in turn help the audient and movie maker decision making easier.

PROJECT DATA DESCRIPTION

To organize the data and the code I have created a githib repository of the project. You can view the complete data below. The primary dataset is collected Data World and IMDB websites. I have also collected and inserted datasets from TMDb for additional reference and data analysis purposes.

<https://github.com/greeshvMIS545Project/PredictRatingMoviesTVShows/commits?author=greeshvMIS545Project>

From the initial look of the data, following are my observations: there is metadata for

- ~5000 movies and it contains 28 different variables.
- 19 records do not contain the value for whether the movie is color or black & white. Further data digging will help me understand if we need to delete these records from the dataset as part of data cleaning.
- Spanning across 66 countries in a time range of 100 years.
- 2399 unique director name records and currently uncounted number of actors / actresses.

The identified dependent variable here would be class named ***"imdb_score"*** while the other 27 variables are possible independent variables which are predicting factors for IMDB score or rating.

Variable Name	Description
movie_title	Title of the Movie
duration	Duration in minutes
director_name	Name of the Director of the Movie
director_facebook_likes	Number of likes of the Director on his Facebook Page
actor_1_name	Primary actor starring in the movie
actor_1_facebook_likes	Number of likes of the Actor_1 on his/her Facebook Page
actor_2_name	Another actor starring in the movie
actor_2_facebook_likes	Number of likes of the Actor_2 on his/her Facebook Page
actor_3_name	Another actor starring in the movie
actor_3_facebook_likes	Number of likes of the Actor_3 on his/her Facebook Page
num_user_for_reviews	Number of users who gave a review
num_critic_for_reviews	Number of critical reviews on imdb
num_voted_users	Number of people who voted for the movie

cast_total_facebook_likes	Total number of FB likes of the entire cast of the movie
movie_facebook_likes	Number of Facebook likes in the movie page
plot_keywords	Keywords describing the movie plot
facenumber_in_poster	Number of the actor who featured in the movie poster
color	Film colorization. 'Black and White' or 'Color'
genres	Film categorization like 'Animation', 'Comedy', 'Romance', 'Horror', 'Sci-Fi', 'Action', 'Family'
title_year	The year in which the movie is released (1916:2016)
language	English, Arabic, Chinese, French, German, Danish, Italian, Japanese etc
country	Country where the movie is produced
content_rating	Content rating of the movie
aspect_ratio	Aspect ratio the movie was made in
movie_imdb_link	IMDB link of the movie
gross	Gross earnings of the movie in Dollars
budget	Budget of the movie in Dollars
imdb_score	IMDB Score of the movie on IMDB

DATA EXPLORATION

In this section we will perform data exploration which is the initial step in data analysis. Here we are going to explore the *IMDB_movie_metadata* data set to discover initial patterns, characteristics and our specific points of interests to come up with an accurate prediction for our problem statement.

Step 1: Loading the dataset

Firstly, let's load all the necessary libraries and packages before reading in double click at the dataset. **We have 5043 observations of 28 variables.** The response variable "imdb_score" is numerical, and the predictors are mixed with numerical and categorical variable.

```

> IMDB <- read.csv("IMDB_movie_metadata.csv")
> str(IMDB)
'data.frame': 5043 obs. of 28 variables:
 $ color                : chr  "Color" "Color" "Color" "Color" ...
 $ director_name        : chr  "James Cameron" "Gore Verbinski" "Sam Mend$
 $ num_critic_for_reviews : int  723 302 602 813 NA 462 392 324 635 375 ...
 $ duration             : int  178 169 148 164 NA 132 156 100 141 153 ...
 $ director_facebook_likes : int  0 563 0 22000 131 475 0 15 0 282 ...
 $ actor_3_facebook_likes : int  855 1000 161 23000 NA 530 4000 284 19000 1$
 $ actor_2_name         : chr  "Joel David Moore" "Orlando Bloom" "Rory K$
 $ actor_1_facebook_likes : int  1000 40000 11000 27000 131 640 24000 799 2$
 $ gross                : int  760505847 309404152 200074175 448130642 NA$
 $ genres               : chr  "Action|Adventure|Fantasy|Sci-Fi" "Action|$
 $ actor_1_name         : chr  "CCH Pounder" "Johnny Depp" "Christoph Wal$
 $ movie_title          : chr  "Avatar" "Pirates of the Caribbean: At W$
 $ num_voted_users      : int  886204 471220 275868 1144337 8 212204 3830$
 $ cast_total_facebook_likes: int  4834 48350 11700 106759 143 1873 46055 203$
 $ actor_3_name         : chr  "Wes Studi" "Jack Davenport" "Stephanie Si$
 $ facenumber_in_poster : int  0 0 1 0 0 1 0 1 4 3 ...
 $ plot_keywords        : chr  "avatar|future|marine|native|paraplegic" "$
 $ movie_imdb_link      : chr  "http://www.imdb.com/title/tt0499549/?ref_$
 $ num_user_for_reviews : int  3054 1238 994 2701 NA 738 1902 387 1117 97$
 $ language            : chr  "English" "English" "English" "English" ...
 $ country             : chr  "USA" "USA" "UK" "USA" ...
 $ content_rating       : chr  "PG-13" "PG-13" "PG-13" "PG-13" ...
 $ budget              : num  2.37e+08 3.00e+08 2.45e+08 2.50e+08 NA ...
 $ title_year          : int  2009 2007 2015 2012 NA 2012 2007 2010 2015$
 $ actor_2_facebook_likes : int  936 5000 393 23000 12 632 11000 553 21000 $
 $ imdb_score          : num  7.9 7.1 6.8 8.5 7.1 6.6 6.2 7.8 7.5 7.5 ...
 $ aspect_ratio        : num  1.78 2.35 2.35 2.35 NA 2.35 2.35 1.85 2.35$
 $ movie_facebook_likes : int  33000 0 85000 164000 0 24000 0 29000 11800$

```

Step 2: Duplicate Removal

In the IMDB movie dataset, it has come to the notice that there are multiple duplicate rows. As part of data exploration, make sure to predict the IMDB rating for the consumers based on accurate input data. *We want to remove the 45 duplicated rows and keep the unique ones. We get 4998 observations left.*

```

> sum(duplicated(IMDB))
[1] 45
> IMDB <- IMDB[!duplicated(IMDB), ]
> |

```

Step 3: Cleaning up Movie_Title Field.

In the dataset, it was found that the movie titles have a special character towards the end, and some have additional whitespaces. As part of data exploration and preparing the dataset, make sure the irrelevant special characters are removed. Now the output looks like the following screenshot:

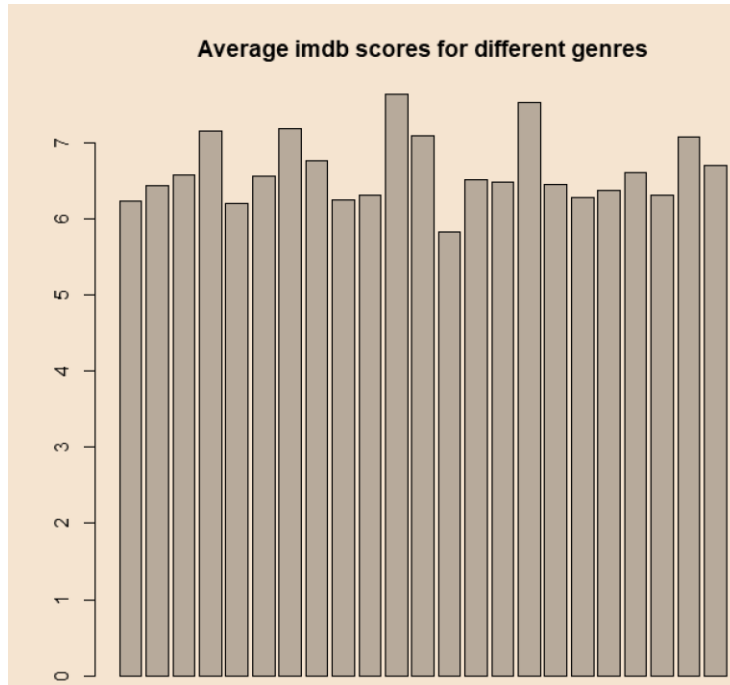
```
[4951] "This Is Martin Bonner"  
[4952] "A True Story"  
[4953] "George Washington"  
[4954] "Smiling Fish & Goat on Fire"  
[4955] "Dawn of the Crescent Moon"  
[4956] "Raymond Did It"  
[4957] "The Last Waltz"  
[4958] "Run, Hide, Die"  
[4959] "The Exploding Girl"  
[4960] "The Legend of God's Gun"  
[4961] "Mutual Appreciation"  
[4962] "Her Cry: La Llorona Investigation"
```

Step 4: Genre Categorization

The genre categorization in the dataset seems to be clumsy and this will further cause trouble in our data analysis. Before we start categorizing the data based on genres, we need to understand if genre is related to IMDB rating / scores. For this, let's divide the string into several substrings by the separator '|' and save each substring along with its corresponding IMDB score in the other data frame named genres.

- Created a new data frame.
- Separated different genres into different columns.
- get the mean of IMDB score for different genres.
- # plot the means.

Then let's plot a histogram to get a visual view for the score and genres to see if they are relative or not. There isn't much difference in the averages of IMDB score related to different genres, almost all the averages are in the same range of 6~8. So, the predictor "genres" can be removed because it's was found to be unrelated to the score.

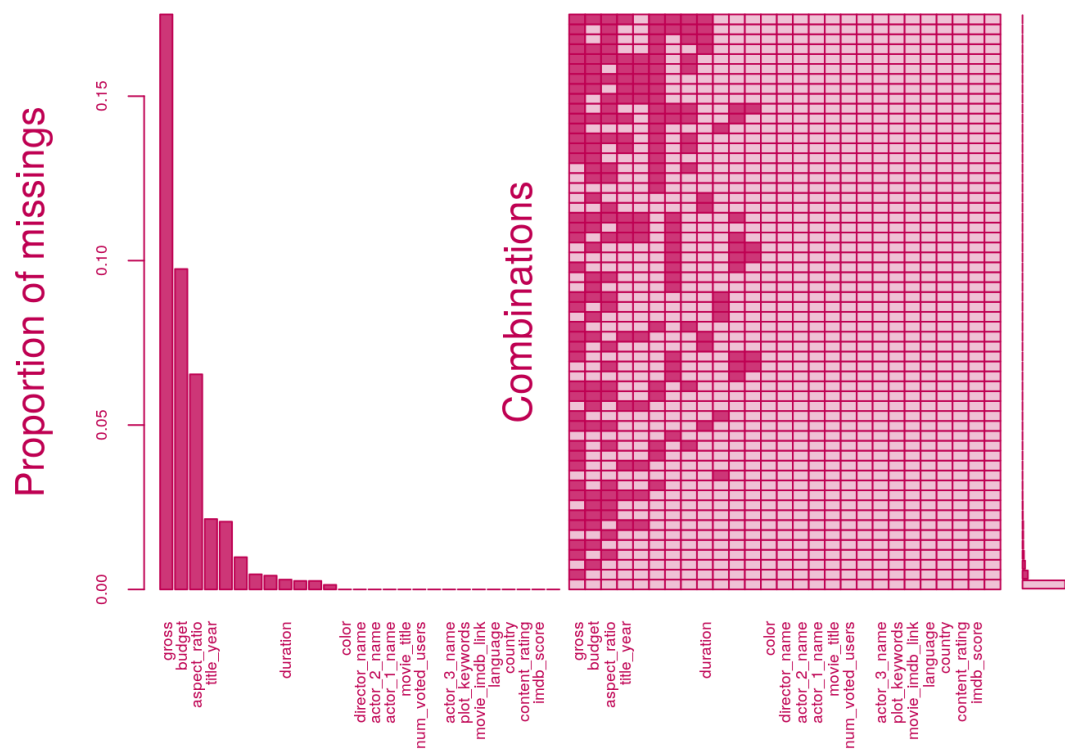


DATA CLEANSING

In this section we will explore the data cleaning technique applied for this project data set. Attempted to detect and correct corrupt or inaccurate records from the record set to identifying incomplete, incorrect, inaccurate, or irrelevant parts of the data.

Identify Missing Values

To find missing values in each column utilized the `colSums()` function to aggregate NA in each column and also used a heatmap to visualize missing values.



Step 1: Deleting Null Values

It was noticed that gross and budget have too many missing values, and we would need to keep these two variables for the project analysis, hence we can only delete rows with null values for gross and budget.

```
> IMDB <- IMDB[!is.na(IMDB$gross), ]
> IMDB <- IMDB[!is.na(IMDB$budget), ]
> dim(IMDB)
[1] 3857 27
> sum(complete.cases(IMDB))
[1] 3768
```

After deleting the null values, now the dataset has 3857 observations. So, there are still $3857 - 3768 = 89$ rows with NAs.

Step 2: Aspect Ratio Analysis

Now that we have removed missing values from gross and budget, we need to look at the remaining columns with missing values.

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

Now aspect_ratio has 74 missing values. Here wanted to check how important is this variable before cleaning up the data.

```
> table(IMDB$aspect_ratio)
1.18 1.33 1.37 1.5 1.66 1.75 1.77 1.78 1.85 2 2.2 2.24 2.35 2.39 2.4
1 19 50 1 40 2 1 41 1600 3 10 1 1995 11 3
2.55 2.76 16
1 3 1
```

Notice that the most common aspect ratios are 1.85 and 2.35. For analyzing purpose, we group other ratios together. To compute the mean of IMDB score for various **aspect_ratio**, let's switch NA with zero first. From the means of IMDB score for various aspect ratios, note that there's no significant difference, all the means fall within the range of 6.3~6.8. So, removing this variable won't affect our following analysis.

```
> IMDB$aspect_ratio[is.na(IMDB$aspect_ratio)] <- 0
> mean(IMDB$imdb_score[IMDB$aspect_ratio == 1.85])
[1] 6.373938
> mean(IMDB$imdb_score[IMDB$aspect_ratio == 2.35])
[1] 6.508471
> mean(IMDB$imdb_score[IMDB$aspect_ratio != 1.85 & IMDB$aspect_ratio != 2.35])
[1] 6.672519
> IMDB <- subset(IMDB, select = -c(aspect_ratio))
```

Step 3: Deal with zeros

It has come to the notice that there are zero values that should also be regarded as missing value except for predictor **facenumber_in_poster**. First we need to replace NA with column average for **facenumber_in_poster**, then replace zeros in other predictors with NA, and finally replace all NAs with their respective column mean.

Note that now imputing the numeric missing values is completed, let's look at the categorical fields. Now we finished imputing the numeric missing values. There are still some categorical missing values, let's look.

Step 4: Missing Values in content_rating

Note that there missing values in content_rating, which are marked as “”. Since these missing values cannot be replaced with reasonable data, we delete these rows.

```
table(IMDB$content_rating)
```

	Approved	G	GP	M	NC-17	Not Rated	Passed
51	17	91	1	2	6	42	3
PG	PG-13	R	Unrated	X			
573	1314	1723	24	10			

Based on the research and the history of naming different content ratings, we find M = GP = PG, X = NC-17. We want to replace M and GP with PG, replace X with NC-17, because these are the new codes used recently. We want to replace “Approved”, “Not Rated”, “Passed”, “Unrated” with the most common rating “R”. Now we only have 5 different content ratings.

```
> IMDB <- IMDB[!(IMDB$content_rating %in% ""),]
> IMDB$content_rating[IMDB$content_rating == 'M'] <- 'PG'
> IMDB$content_rating[IMDB$content_rating == 'GP'] <- 'PG'
> IMDB$content_rating[IMDB$content_rating == 'X'] <- 'NC-17'
> IMDB$content_rating[IMDB$content_rating == 'Approved'] <- 'R'
> IMDB$content_rating[IMDB$content_rating == 'Not Rated'] <- 'R'
> IMDB$content_rating[IMDB$content_rating == 'Passed'] <- 'R'
> IMDB$content_rating[IMDB$content_rating == 'Unrated'] <- 'R'
> IMDB$content_rating <- factor(IMDB$content_rating)
> table(IMDB$content_rating)
```

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

Add Purposeful Columns

Based on the problem statement and research data, it was inferred that we would need two additional columns to make an accurate prediction through this data analysis.

- Profit
- percentage return on investment

Delete Irrelevant Columns

Here we are going to take a stab at the data which might not be relevant for the data analysis.

- Predictor “color” is an irrelevant predictor as this is nearly constant and more than 95% movies are colored.
- Predictor “language” is an irrelevant predictor as over 95% movies are in English, which means this variable is nearly constant.

- Predictor “**country**” is an *irrelevant predictor* as around 79% movies are from USA, 8% from UK, 13% from other countries. Let’s group other countries together to make this categorical variable with less levels: USA, UK, Others.

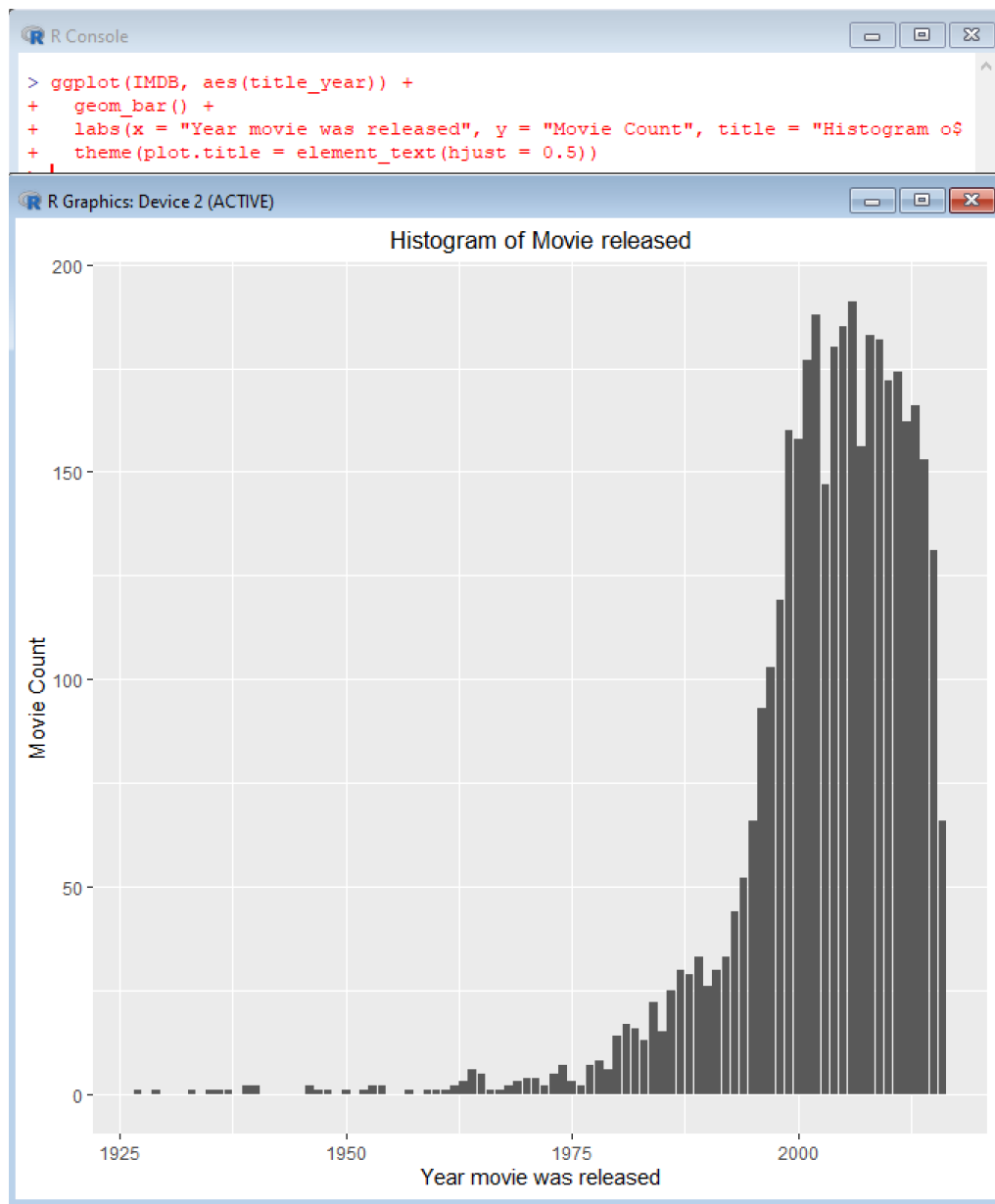
```
> table(IMDB$color)
      Black and White      Color
      2             124      3680
>
> IMDB <- subset(IMDB, select = -c(color))
> table(IMDB$language)
      Aboriginal      Arabic      Aramaic      Bosnian      Cantonese      Czech
      2             2             1             1             7             1
      Danish      Dari      Dutch      English      Filipino      French      German
      3             2             3             3644             1             34             11
      Hebrew      Hindi      Hungarian      Indonesian      Italian      Japanese      Kazakh
      2             5             1             2             7             10             1
      Korean      Mandarin      Maya      Mongolian      None      Norwegian      Persian
      5             14             1             1             1             4             3
      Portuguese      Romanian      Russian      Spanish      Thai      Vietnamese      Zulu
      5             1             1             24             3             1             1
> IMDB <- subset(IMDB, select = -c(language))
> table(IMDB$country)
      Afghanistan      Argentina      Aruba      Australia      Belgium
      1             3             1             40             1
      Brazil      Canada      Chile      China      Colombia
      5             63             1             13             1
      Czech Republic      Denmark      Finland      France      Georgia
      3             9             1             103             1
      Germany      Greece      Hong Kong      Hungary      Iceland
      79             1             13             2             1
      India      Indonesia      Iran      Ireland      Israel
      5             1             4             7             2
      Italy      Japan      Mexico      Netherlands      New Line
      11             15             10             3             1
      New Zealand      Norway      Official site      Peru      Philippines
      11             4             1             1             1
      Poland      Romania      Russia      South Africa      South Korea
      1             2             3             3             8
      Spain      Taiwan      Thailand      UK      USA
      22             2             4             316             3025
      West Germany
      1
> levels(IMDB$country) <- c(levels(IMDB$country), "Others")
> IMDB$country[(IMDB$country != 'USA') & (IMDB$country != 'UK')] <- 'Others'
> IMDB$country <- factor(IMDB$country)
> table(IMDB$country)
      Others      UK      USA
      465      316      3025
```

DATA VISUALIZATION

In this section we will showcase the visualization of dataset and a graphical representation of information we collected and processed in the previous steps.

Movies Released: Year

It does look like the movie production industry just spiked after year 1990. Based on the research articles it shows that this was due to the advancement in technology and the increased utilization of the internet.



Note: From the histogram it looks like it is safe to remove records before 1980 because they might not be representative.

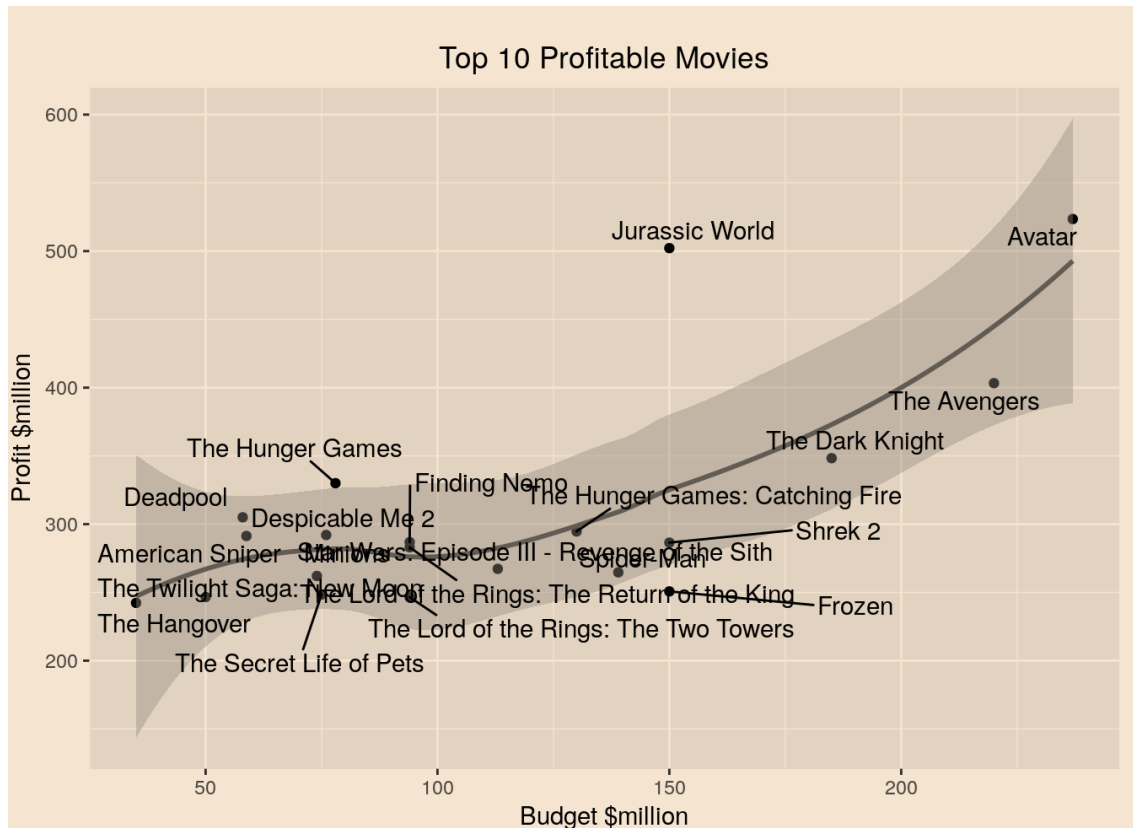
Profit: Highest 20 movies.

These are the highest 20 movies supported the Profit earned (Gross - Budget). It may be inferred from this plot that top budget movies tend to earn more profit. The trend is sort of linear, with profit increasing with the rise in budget.

```

```{r}
IMDB %>%
 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 $million", y = "Profit $million", title = "Top 10 Profitable Movies")
 theme(plot.title = element_text(hjust = 0.5))
```

```



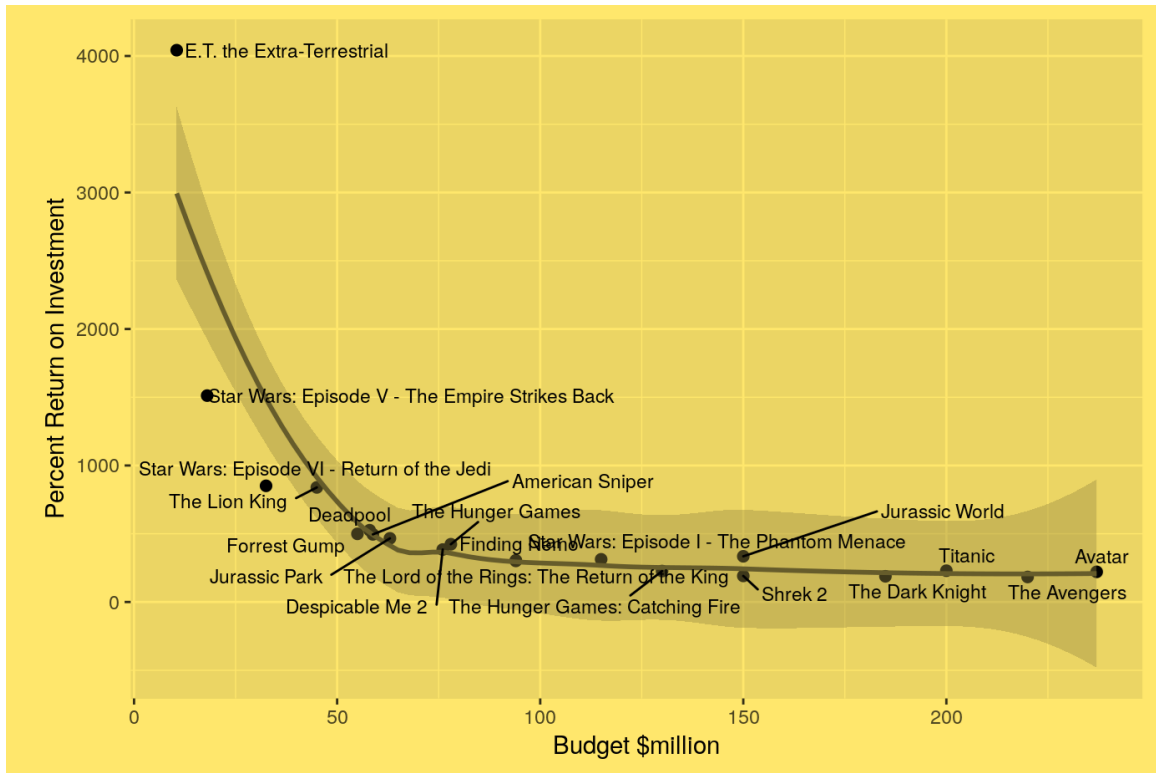
Return on Investment: Highest 20 movies.

These are the highest 20 movies supported its Percentage Return on Investment $((\text{profit}/\text{budget}) * 100)$. Since profit earned by a movie doesn't provides a clear picture about its monetary success over the years, this analysis, over absolutely the value of the Return on Investment(ROI) across its Budget, would supply better results. As hypothesized, the ROI is high for Low Budget Films and reduces because the budget of the movie increases.

```

IMDB %>%
  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 $million") +
  ylab("Percent Return on Investment") +
  ggtitle("20 Most Profitable Movies based on its Return on Investment")

```



Highest Avg IMDB Score: Directors

This is the result for the top 20 directors with the highest average IMDB scores from the dataset.

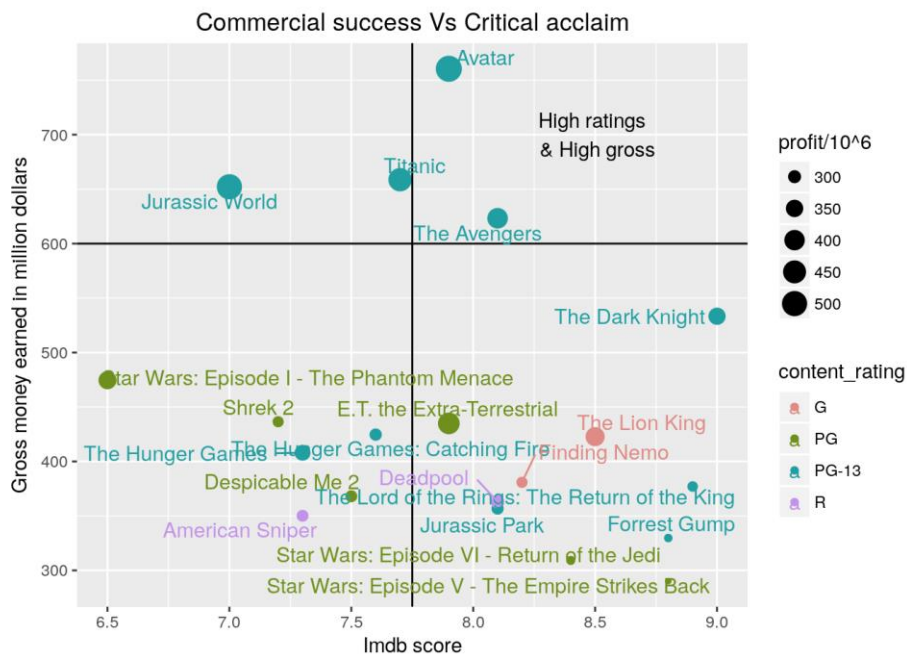
| director_name | avg_imdb |
|-----------------------|----------|
| Tony Kaye | 8.6 |
| Damien Chazelle | 8.5 |
| Majid Majidi | 8.5 |
| Ron Fricke | 8.5 |
| Christopher Nolan | 8.425 |
| Asghar Farhadi | 8.4 |
| Marius A. Markevicius | 8.4 |

| | |
|----------------------|----------|
| Richard Marquand | 8.4 |
| Sergio Leone | 8.4 |
| Lee Unkrich | 8.3 |
| Lenny Abrahamson | 8.3 |
| Pete Docter | 8.233333 |
| Hayao Miyazaki | 8.225 |
| Joshua Oppenheimer | 8.2 |
| Juan José Campanella | 8.2 |
| Quentin Tarantino | 8.2 |
| David Singleton | 8.1 |
| Je-kyu Kang | 8.1 |
| Terry George | 8.1 |
| Tim Miller | 8.1 |

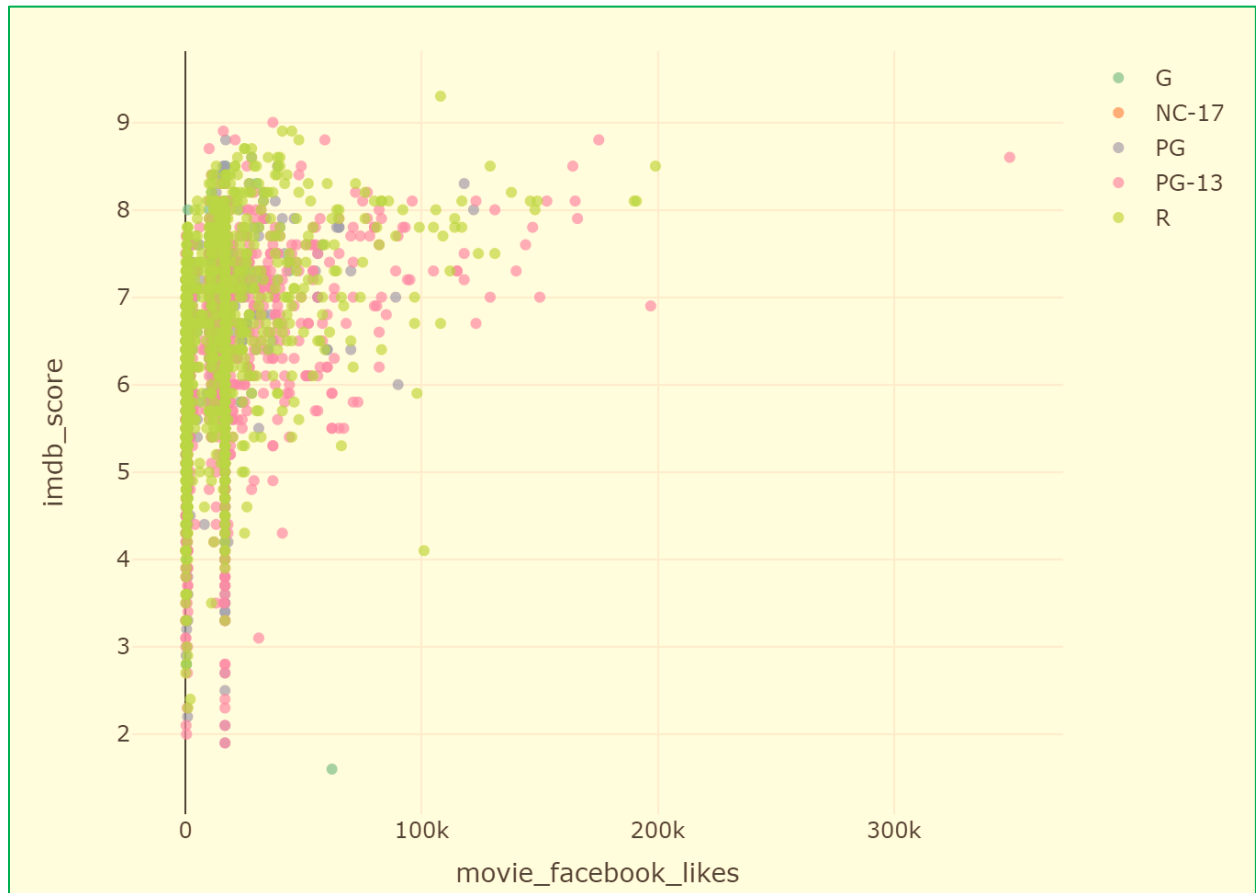
Scope down Data to Production/Commercial Success Vs Critical Acclaim

We are assuming the commercial success attained by the movie depends on the gross and profit gained, here we are performing an analysis for production success movies versus critically acclaimed movies. Here we will compare the IMDB scores for these 2 categories. This plot showcases the expected results which is , most of the critically renowned movies do not collect as much profit.

As expected, there is not much correlation since most critically acclaimed movies do not do much well commercially.



Based on history and research it is noted that we need to split this plot based on content rating. We are associating the social media (Facebook) likes to the IMDB score.



DATA PRE-PROCESSING

Remove Variables

Names

Let's validate the number of directors and actors in this dataset.

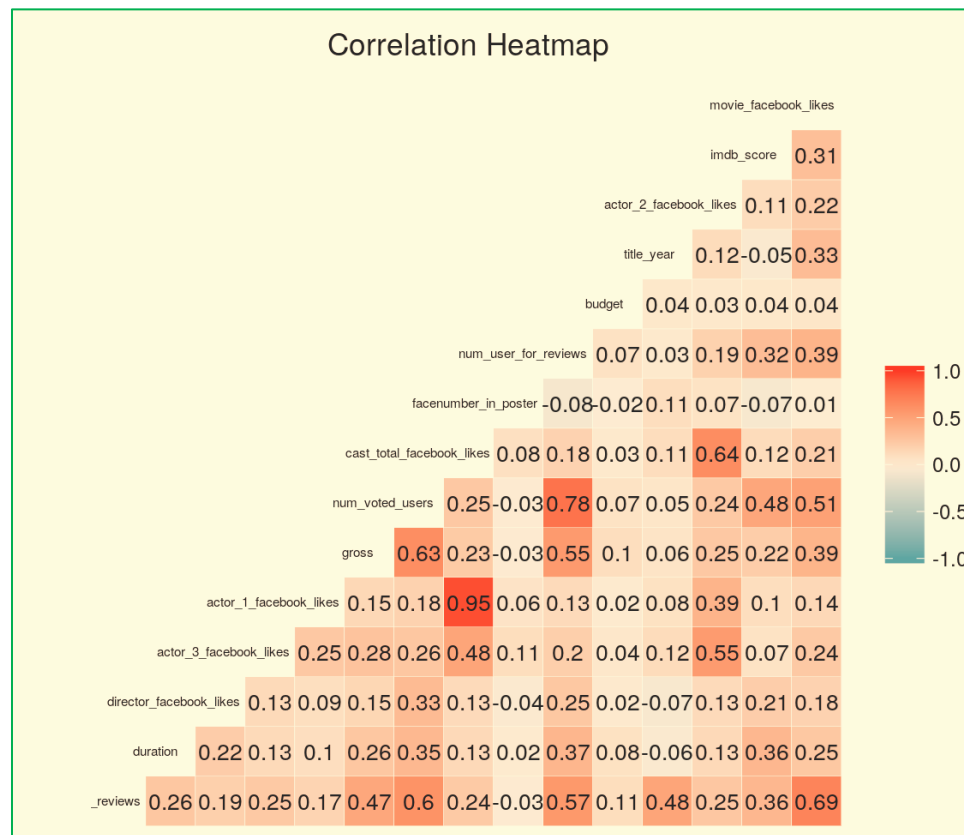
- 1660 directors
- 3621 actors

As the director and actor names are different for each entry the ability of these variables to predict the score would be irrelevant. We shall remove the movie link as that is a redundant variable too.

Highly correlated Variables

Let's create a heat map for defining the co-relation for the dataset.

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



Based on the heatmap, here are the observations:

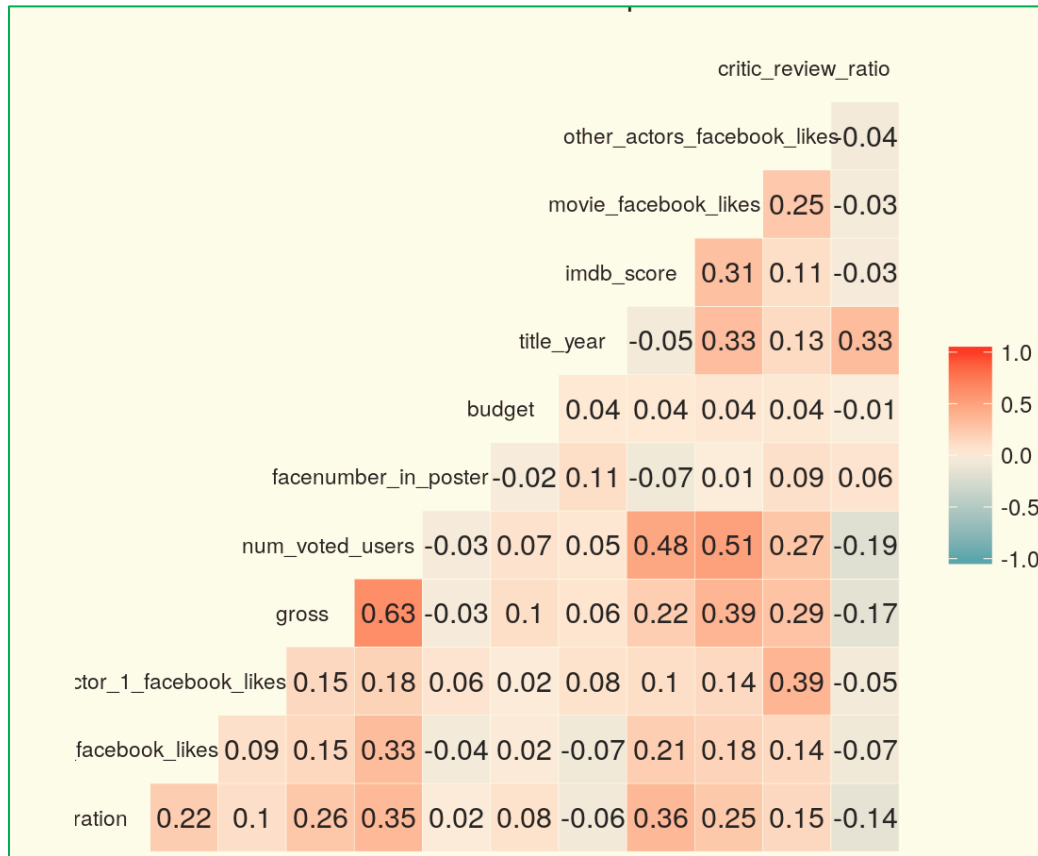
- There are high correlations (greater than 0.7) between predictors.
- Based on the highest correlation value, which is 0.95, we infer that actor_1_facebook_likes is highly correlated with the cast_total_facebook_likes
- We can also conclude that both actor2 and actor3 are correlated.
- high correlations among num_voted_users, num_user_for_reviews and num_critic_for_reviews
- keep num_voted_users and take the ratio of num_user_for_reviews and num_critic_for_reviews.

So, based on our above observations, let's categorize them into 2 variables:

- actor_1_facebook_likes
- other_actors_facebook_likes

While we have made the above conclusions, the new correlation heat map would look like the below:

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



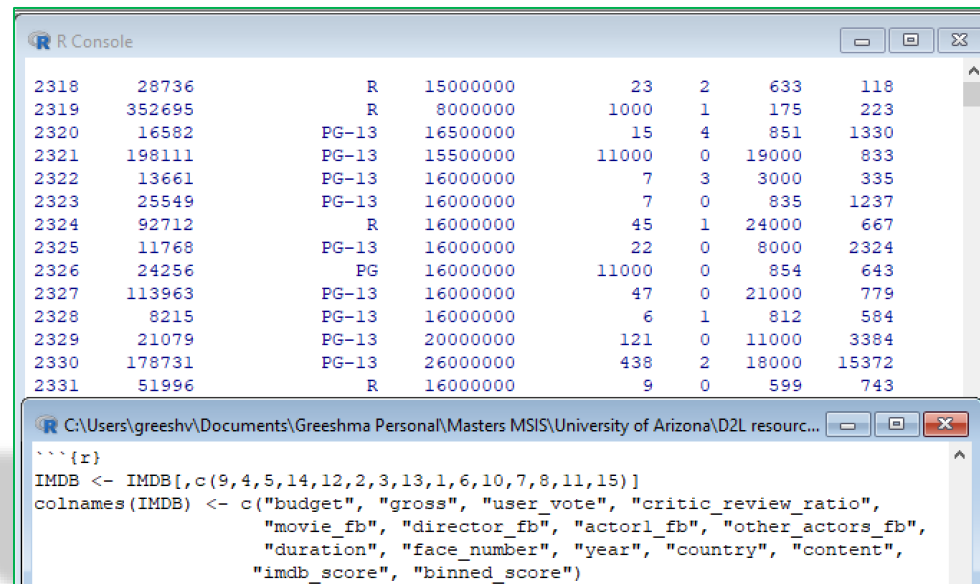
Group dependent variable

Let's bin the IMDB scores into 4 buckets. Part of the output is pasted as a screenshot:

- less than 4 - BAD
- 4~6 - OK
- 6~8 - GOOD
- 8~10 - EXCELLENT

```
> IMDB$binned_score <- cut(IMDB$imdb_score, breaks = c(0,4,6,8,10))
> IMDB$binned_score
[1] (6,8] (6,8] (6,8] (8,10] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8]
[11] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (8,10] (6,8] (6,8] (6,8]
[21] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (8,10] (4,6] (6,8] (6,8]
[31] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (4,6] (4,6] (6,8] (6,8]
[41] (6,8] (4,6] (8,10] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8]
[51] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (8,10] (6,8] (4,6] (6,8]
[61] (4,6] (6,8] (6,8] (6,8] (8,10] (8,10] (6,8] (6,8] (6,8] (4,6]
[71] (4,6] (6,8] (4,6] (6,8] (6,8] (4,6] (8,10] (6,8] (6,8] (6,8]
[81] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (4,6]
[91] (8,10] (6,8] (8,10] (8,10] (8,10] (6,8] (6,8] (6,8] (6,8] (6,8]
[101] (6,8] (4,6] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (4,6]
[111] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (8,10] (6,8] (4,6] (6,8]
[121] (6,8] (6,8] (6,8] (6,8] (8,10] (6,8] (6,8] (6,8] (4,6] (4,6]
[131] (6,8] (6,8] (4,6] (6,8] (6,8] (4,6] (6,8] (6,8] (4,6] (6,8]
[141] (4,6] (6,8] (6,8] (6,8] (6,8] (4,6] (6,8] (6,8] (6,8] (6,8]
[151] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (6,8] (4,6] (4,6]
[161] (6,8] (6,8] (4,6] (4,6] (4,6] (6,8] (6,8] (6,8] (6,8] (4,6]
```

Organize Dataset



The screenshot shows an R console window with a data table and R code. The data table has 8 columns: index, budget, gross, user_vote, critic_review_ratio, movie_fb, director_fb, actor1_fb, other_actors_fb, duration, face_number, year, country, content, imdb_score, and binned_score. The R code below the table loads the IMDB dataset and sets column names.

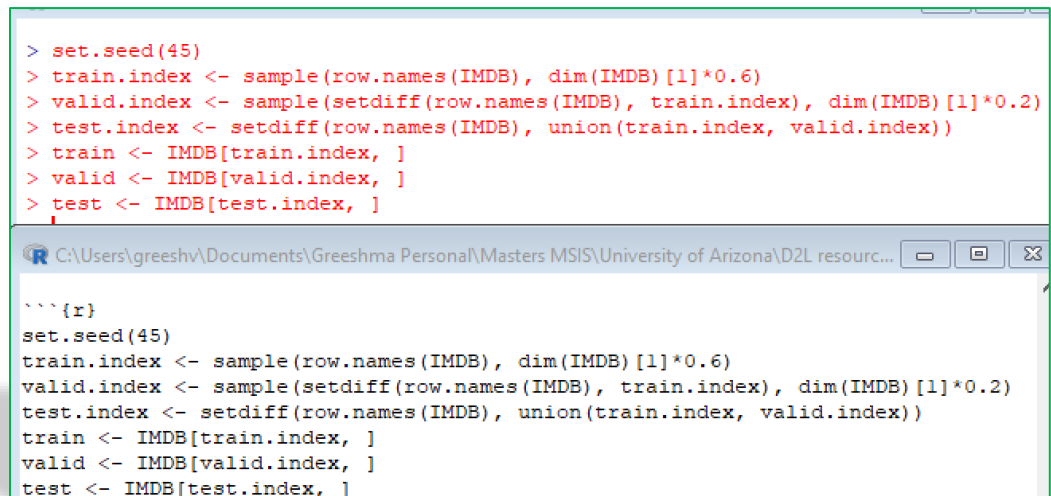
| | | | | | | | | | | | | | | | |
|------|--------|-------|----------|-------|---|-------|-------|--|--|--|--|--|--|--|--|
| 2318 | 28736 | R | 15000000 | 23 | 2 | 633 | 118 | | | | | | | | |
| 2319 | 352695 | R | 8000000 | 1000 | 1 | 175 | 223 | | | | | | | | |
| 2320 | 16582 | PG-13 | 16500000 | 15 | 4 | 851 | 1330 | | | | | | | | |
| 2321 | 198111 | PG-13 | 15500000 | 11000 | 0 | 19000 | 833 | | | | | | | | |
| 2322 | 13661 | PG-13 | 16000000 | 7 | 3 | 3000 | 335 | | | | | | | | |
| 2323 | 25549 | PG-13 | 16000000 | 7 | 0 | 835 | 1237 | | | | | | | | |
| 2324 | 92712 | R | 16000000 | 45 | 1 | 24000 | 667 | | | | | | | | |
| 2325 | 11768 | PG-13 | 16000000 | 22 | 0 | 8000 | 2324 | | | | | | | | |
| 2326 | 24256 | PG | 16000000 | 11000 | 0 | 854 | 643 | | | | | | | | |
| 2327 | 113963 | PG-13 | 16000000 | 47 | 0 | 21000 | 779 | | | | | | | | |
| 2328 | 8215 | PG-13 | 16000000 | 6 | 1 | 812 | 584 | | | | | | | | |
| 2329 | 21079 | PG-13 | 20000000 | 121 | 0 | 11000 | 3384 | | | | | | | | |
| 2330 | 178731 | PG-13 | 26000000 | 438 | 2 | 18000 | 15372 | | | | | | | | |
| 2331 | 51996 | R | 16000000 | 9 | 0 | 599 | 743 | | | | | | | | |

```
```\r}\nIMDB <- IMDB[,c(9,4,5,14,12,2,3,13,1,6,10,7,8,11,15)]\ncolnames(IMDB) <- c("budget", "gross", "user_vote", "critic_review_ratio",\n                    "movie_fb", "director_fb", "actor1_fb", "other_actors_fb",\n                    "duration", "face_number", "year", "country", "content",\n                    "imdb_score", "binned_score")
```

## Split Data

Data is being split into the following:

- Training
- Validation
- test sets



The screenshot shows an R console window with R code for splitting the IMDB dataset into training, validation, and test sets. The code uses the sample and setdiff functions to create the indices.

```
> set.seed(45)\n> train.index <- sample(row.names(IMDB), dim(IMDB)[1]*0.6)\n> valid.index <- sample(setdiff(row.names(IMDB), train.index), dim(IMDB)[1]*0.2)\n> test.index <- setdiff(row.names(IMDB), union(train.index, valid.index))\n> train <- IMDB[train.index,]\n> valid <- IMDB[valid.index,]\n> test <- IMDB[test.index,]\n\n```\r}\nset.seed(45)\ntrain.index <- sample(row.names(IMDB), dim(IMDB)[1]*0.6)\nvalid.index <- sample(setdiff(row.names(IMDB), train.index), dim(IMDB)[1]*0.2)\ntest.index <- setdiff(row.names(IMDB), union(train.index, valid.index))\ntrain <- IMDB[train.index,]\nvalid <- IMDB[valid.index,]\ntest <- IMDB[test.index,]
```

### Full grown Tree

Based on the tree above, let's formulate some rules and classify them into the bin we prepared earlier:

<b>(user_vote &gt;= 551000)</b>	<b>(8,10]</b>
(83000 <= user_vote < 551000)	(6,8]
(user_vote < 83000) && (duration >= 106)	(6,8]
(user_vote < 83000) and (duration < 106) and (gross < 7900000)	(6,8]
(user_vote < 83000) and (duration < 106) and (gross >= 7900000) and (movie_fb < 4500)	(4,6]
(user_vote < 83000) and (duration < 106) and (gross >= 7900000) and (movie_fb >= 4500) and (year < 2000)	(6,8]
(user_vote < 83000) and (duration < 106) and (gross >= 7900000) and (movie_fb >= 4500) and (year >= 2000) and (critic_review_ratio < 1.2)	(4,6]
(user_vote < 41000) and (duration < 106) and (gross >= 7900000) and (movie_fb >= 4500) and (year >= 2000) and (critic_review_ratio >= 1.2)	(4,6]
(41000 <= user_vote < 83000) and (duration < 106) and (gross >= 7900000) and (movie_fb >= 4500) and (year >= 2000) and (critic_review_ratio >= 1.2)	(6,8]

Conclusions we assume from the above classification table are:

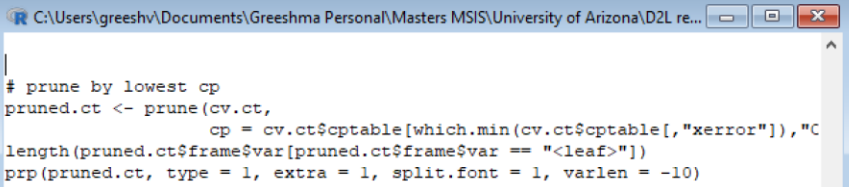
- lot of votes in imdb website tend to own a higher score.
- If a movie has lower number of votes even if it's a good movie only if its length is longer.

### To find the 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)
```

```
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:
```



```
prune by lowest cp
pruned.ct <- prune(cv.ct,
 cp = cv.ct$cptable[which.min(cv.ct$cptable[, "xerror"]), "C
length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])
prp(pruned.ct, type = 1, extra = 1, split.font = 1, varlen = -10)
...
```

Variables actually used in tree construction:

```
[1] actor1_fb budget content
[4] country critic_review_ratio director_fb
[7] duration face_number gross
[10] movie_fb other_actors_fb user_vote
[13] year
```

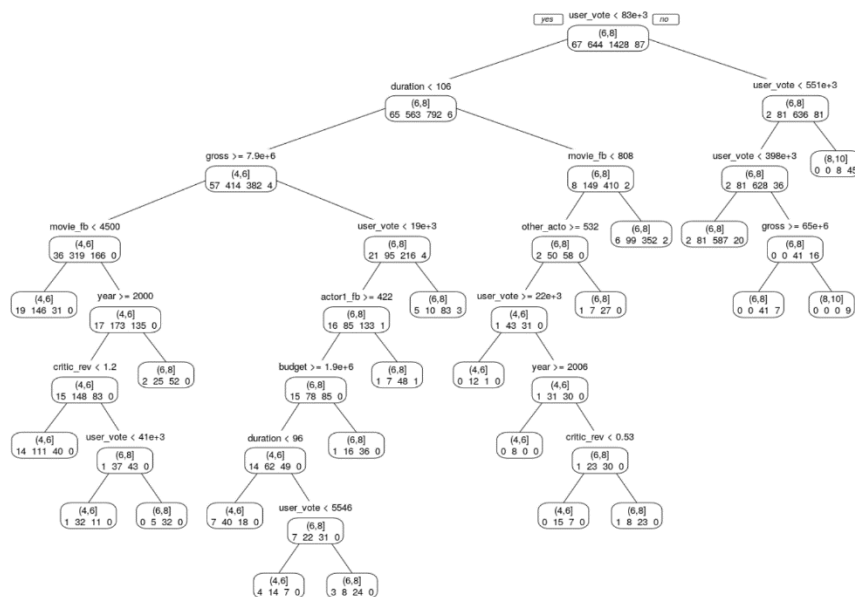
Root node error: 798/2226 = 0.35849

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	0.76316	0.026357
6	0.00563910	13	0.66040	0.75815	0.026303
7	0.00543024	15	0.64912	0.75815	0.026303
8	0.00501253	20	0.61278	0.75564	0.026276
9	0.00407268	25	0.58772	0.76817	0.026411
10	0.00375940	29	0.57143	0.77820	0.026517
11	0.00325815	45	0.50877	0.78070	0.026543
12	0.00322234	50	0.49248	0.78446	0.026582
13	0.00313283	57	0.46992	0.78446	0.026582
14	0.00292398	63	0.45113	0.79574	0.026697
15	0.00250627	66	0.44236	0.79574	0.026697
16	0.00219298	112	0.31830	0.80827	0.026821
17	0.00187970	120	0.29950	0.80451	0.026784
18	0.00167084	133	0.27444	0.80576	0.026797
19	0.00156642	142	0.25940	0.80576	0.026797
20	0.00125313	151	0.24436	0.83208	0.027049
21	0.00093985	200	0.18045	0.84712	0.027188
22	0.00083542	204	0.17669	0.84336	0.027154
23	0.00062657	207	0.17419	0.85714	0.027278
24	0.00050125	213	0.17043	0.85714	0.027278
25	0.00027847	218	0.16792	0.86842	0.027376
26	0.00025063	227	0.16541	0.86842	0.027376
27	0.00001000	237	0.16291	0.86842	0.027376

lowest cross-validation  
error: 0.75564





```

C:\Users\greeshv\Documents\Greeshma Personal\Masters MSIS\University of Arizona\D2L re...
argument cp sets the smallest value for the complexity parameter.
...
The 8th tree has the lowest cross-validation error (xerror): 0.75564.

prune by lowest cp
pruned.ct <- prune(cv.ct,
 cp = cv.ct$cpstable[which.min(cv.ct$cpstable[, "xerror"])], "%
length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])
prp(pruned.ct, type = 1, extra = 1, split.font = 1, varlen = -10)
...

```

## APPLYING THE MODEL – EVALUATION CRITERIA

Applying the model on the training set

```

R Console
> # 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$binmed_score)
Confusion Matrix and Statistics

C:\Users\greeshv\Documents\Greeshma Personal\Masters MSIS\University of Arizona\D2L re...
{r}
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$binmed_score)
...

```

## 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

## Applying the model on validation set

```
> tree.pred.valid <- predict(pruned.ct, valid, type = "class")
> # generate confusion matrix for validation data
> confusionMatrix(tree.pred.valid, valid$binmed_score)
Confusion Matrix and Statistics
```

```
R C:\Users\greeshv\Documents\Greeshma Personal\Masters MSIS\University of Arizona\D2L re..
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$binmed_score)|
...

```

### 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

## Apply model on test data

```
> tree.pred.test <- predict(pruned.ct, test, type = "class")
> # generate confusion matrix for test data
> confusionMatrix(tree.pred.test, test$binmed_score)
```

```
C:\Users\greeshv\Documents\Greeshma Personal\Masters MSIS\University of Arizona\D2L re...
tree.pred.test <- predict(pruned.ct, test, type = "class")
generate confusion matrix for test data
confusionMatrix(tree.pred.test, test$binmed_score)|
```

### 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

## K-NEAREST NEIGHBORS – DESCRIPTIVE DATA MINING TECHNIQUE

K nearest neighbors is a simple predictive data mining algorithm that stores all available cases and classifies new cases based on a similarity measure mostly distance functions.

## Data Preparation

- Create dummy variables for country variable and content variable.
- Select potential variables for future prediction.
- Partition the data into training, validation, and test data sets.

```
> IMDB2 <- IMDB
> IMDB2$country <- as.factor(IMDB2$country)
> IMDB2$content <- as.factor(IMDB2$content)
> IMDB2[,c("country_UK", "country_USA", "country_Others")] <- model.matrix(~ c$
> IMDB2[,c("content_G", "content_NC17", "content_PG", "content_PG13", "content_$

R C:\Users\greeshv\Documents\Greeshma Personal\Masters MSIS\University of Arizona\D2L re...
IMDB2 <- IMDB
IMDB2$country <- as.factor(IMDB2$country)
IMDB2$content <- as.factor(IMDB2$content)
IMDB2[,c("country_UK", "country_USA", "country_Others")] <- model.matrix(~
IMDB2[,c("content_G", "content_NC17", "content_PG", "content_PG13", "content
#
IMDB2 <- IMDB2[, c(1,2,3,4,5,6,7,8,9,10,11,16,17,18,19,20,21,22,23,15)]
#
set.seed(52)
train2 <- IMDB2[train.index,]
valid2 <- IMDB2[valid.index,]
test2 <- IMDB2[test.index,]
```

## Data Normalization

```
> train2.norm[, -20] <- predict(norm.values, train2[, -20])
> valid2.norm[, -20] <- predict(norm.values, valid2[, -20])
> test2.norm[, -20] <- predict(norm.values, test2[, -20])
> IMDB2.norm[, -20] <- predict(norm.values, IMDB2[, -20])

R C:\Users\greeshv\Documents\Greeshma Personal\Masters MSIS\University of Arizona\D2L re...

train2.norm <- train2
valid2.norm <- valid2
test2.norm <- test2
IMDB2.norm <- IMDB2
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])
IMDB2.norm[, -20] <- predict(norm.values, IMDB2[, -20])|
```

## Find the Best K using models – Evaluation Criteria

Let' set the value of k from 1:20. We will build 20 different models and calculate the accuracy to find the best k based on highest accuracy.

Highest Accuracy

	k	accuracy
1	1	0.6671159
2	2	0.5916442
3	3	0.6940701
4	4	0.6792453
5	5	0.6954178
6	6	0.6913747
7	7	0.6886792
8	8	0.6873315
9	9	0.7142857
10	10	0.7061995
11	11	0.7021563
12	12	0.6873315
13	13	0.6940701
14	14	0.6846361
15	15	0.7008086
16	16	0.6886792
17	17	0.6913747
18	18	0.6886792
19	19	0.6873315
20	20	0.6927224

```
R C:\Users\greeshv\Documents\Greeshma Personal\Masters MSIS\University of Arizona\D2L re...
initialize a data frame with two columns: k, and accuracy.
accuracy.df <- data.frame(k = seq(1, 20, 1), accuracy = rep(0, 20))
compute 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
```

Apply model on test-set

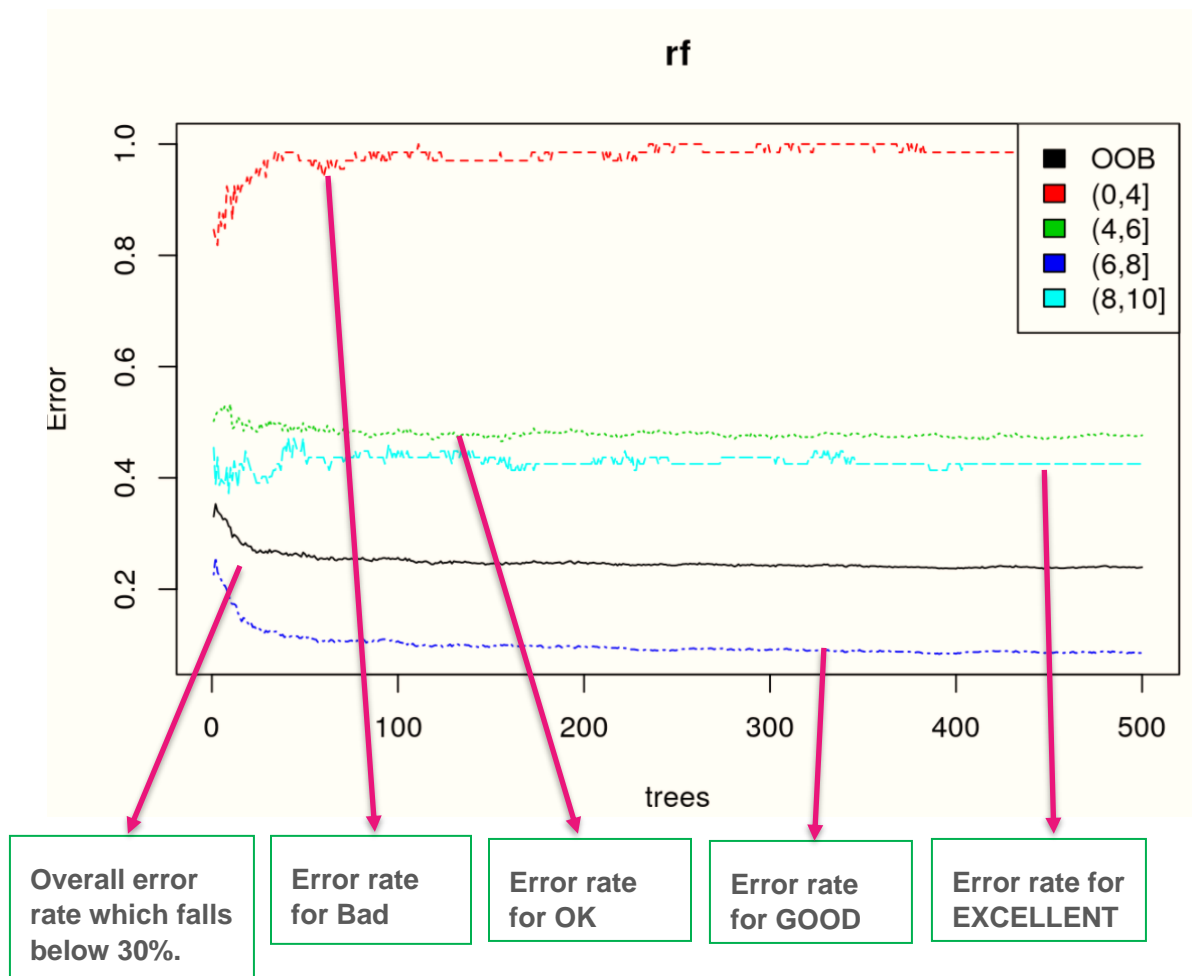
```
apply model on test set
knn.pred.test <- knn(train2.norm[, -20], test2.norm[, -20],
 cl = train2.norm[, 20], k = 9)
generate confusion matrix for test data
accuracy <- confusionMatrix(knn.pred.test, test2.norm[, 20])$overall[1]
accuracy
```

Accuracy  
0.7456258

## RANDOM FOREST – MULTITUDE OF DECISION TREES

### Build Model

```
C:\Users\greeshv\Documents\Greeshma Personal\Masters MSIS\University of Arizona\D2L re...
install.packages("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)
```

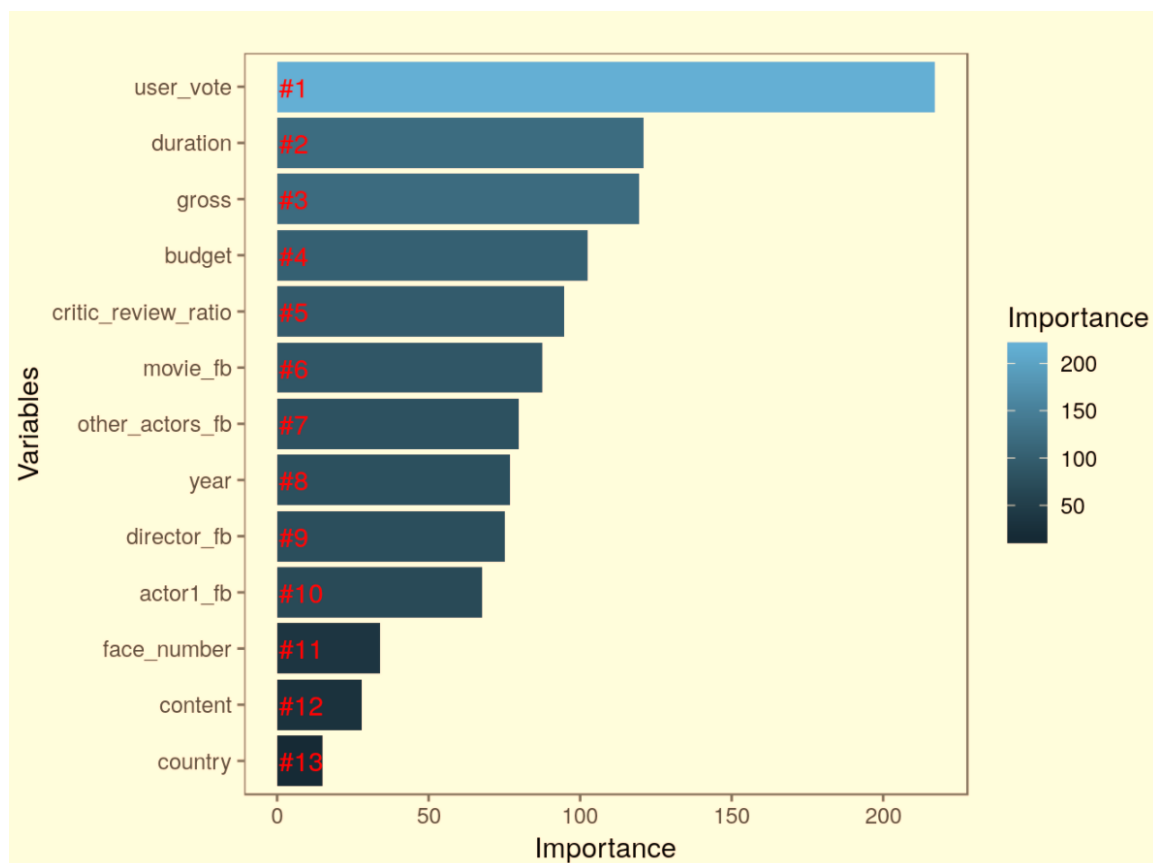


Calculating the variable importance:

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

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

Use 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()
```



It is now proved that the **User\_vote** is a crucial and relevant variable for the decision making, while **face\_number**, **content** and **country** are irrelevant to this project analysis.



## Apply the model – Validation Data Set

```
> set.seed(632)
> # apply model on validation set
> rf.pred.valid <- predict(rf, valid)
```

R C:\Users\greeshv\Documents\Greeshma Personal\Masters MSIS\University of Arizona\D2L resour...

### ### 6.3.2 Apply Model

```
```{r}
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)
```
```

#### Confusion Matrix and Statistics

|            | Reference |       |       |        |
|------------|-----------|-------|-------|--------|
| Prediction | (0,4]     | (4,6] | (6,8] | (8,10] |
| (0,4]      | 0         | 0     | 0     | 0      |
| (4,6]      | 7         | 106   | 43    | 0      |
| (6,8]      | 8         | 103   | 446   | 12     |
| (8,10]     | 0         | 0     | 2     | 15     |

#### Overall Statistics

Accuracy : 0.7642

95% CI : (0.7319, 0.7943)

No Information Rate : 0.6617

P-Value [Acc > NIR] : 8.023e-10

Kappa : 0.4547

Mcnemar's Test P-Value : NA

#### Statistics by Class:

|                      | Class: (0,4] | Class: (4,6] | Class: (6,8] | Class: (8,10] |
|----------------------|--------------|--------------|--------------|---------------|
| Sensitivity          | 0.00000      | 0.5072       | 0.9084       | 0.55556       |
| Specificity          | 1.00000      | 0.9062       | 0.5100       | 0.99720       |
| Pos Pred Value       | NaN          | 0.6795       | 0.7838       | 0.88235       |
| Neg Pred Value       | 0.97978      | 0.8242       | 0.7399       | 0.98345       |
| Prevalence           | 0.02022      | 0.2817       | 0.6617       | 0.03639       |
| Detection Rate       | 0.00000      | 0.1429       | 0.6011       | 0.02022       |
| Detection Prevalence | 0.00000      | 0.2102       | 0.7668       | 0.02291       |
| Balanced Accuracy    | 0.50000      | 0.7067       | 0.7092       | 0.77638       |

## Apply the model – Test Data set

```
R Console

> set.seed(633)
> # apply model on test set
> rf.pred.test <- predict(rf, test)

C:\Users\greeshv\Documents\Greeshma Personal\Masters MSIS\University of Arizona\D2L resour..
set.seed(633)
apply model on test set
rf.pred.test <- predict(rf, test)
generate confusion matrix for test data
confusionMatrix(rf.pred.test, test$binned_score)|
```

Confusion Matrix and Statistics

|                                     | Reference |     |     |    |
|-------------------------------------|-----------|-----|-----|----|
| Prediction (0,4] (4,6] (6,8] (8,10] |           |     |     |    |
| (0,4]                               | 0         | 1   | 0   | 0  |
| (4,6]                               | 8         | 114 | 51  | 0  |
| (6,8]                               | 5         | 97  | 447 | 10 |
| (8,10]                              | 0         | 0   | 2   | 8  |

Overall Statistics

|                          |                  |
|--------------------------|------------------|
| Accuracy :               | 0.7658           |
| 95% CI :                 | (0.7337, 0.7958) |
| No Information Rate :    | 0.6729           |
| P-Value [Acc > NIR] :    | 1.796e-08        |
| Kappa :                  | 0.4515           |
| Mcnemar's Test P-Value : | NA               |

Statistics by Class:

|                      | Class: (0,4] | Class: (4,6] | Class: (6,8] | Class: (8,10] |
|----------------------|--------------|--------------|--------------|---------------|
| Sensitivity          | 0.000000     | 0.5377       | 0.8940       | 0.44444       |
| Specificity          | 0.998630     | 0.8889       | 0.5391       | 0.99724       |
| Pos Pred Value       | 0.000000     | 0.6590       | 0.7996       | 0.80000       |
| Neg Pred Value       | 0.982480     | 0.8281       | 0.7120       | 0.98636       |
| Prevalence           | 0.017497     | 0.2853       | 0.6729       | 0.02423       |
| Detection Rate       | 0.000000     | 0.1534       | 0.6016       | 0.01077       |
| Detection Prevalence | 0.001346     | 0.2328       | 0.7524       | 0.01346       |
| Balanced Accuracy    | 0.499315     | 0.7133       | 0.7165       | 0.72084       |

## PROJECT DATA ANALYSIS CONCLUSION

While we have conducted the data analysis using three different mining algorithms or machine learning algorithms, in this section I would like to tabulate the results and present the overall inference from the dataset useful for predicting the IMDB scores for customers during this COVID-19 pandemic.

Accuracy table for different models:

| Dataset    | Decision Tree                     | K-NN   | Random Forest |
|------------|-----------------------------------|--------|---------------|
| Training   | 0.7803                            | n/a    | n/a           |
| Validation | 0.7129                            | 0.7143 | 0.7642        |
| Test       | 0.7241                            | 0.7456 | 0.7658        |
| Accuracy   | Higher accuracy for training data | n/a    | n/a           |

For Decision tree data mining model, it is noticed that there is a higher accuracy for training data which is expected as the tree was built based on the training data.

Based on the overall analysis, based on my learnings and understanding we find the best model is random forest algorithm in this case and it is seen that it gives a high accuracy around 0.76

## REFERENCES

- I also referred to the dataset provided in the suggested Project Dataset list.  
<https://www.imdb.com/interfaces/>
- <https://www.raindance.org/top-10-film-review-websites/>
- <https://www.mlive.com/coronavirus/2020/05/covid-19-tv-habits-show-a-staggering-change-in-viewing-hours-and-what-were-watching.html>
- <https://www.comscore.com/Insights/Blog/US-TV-Viewing-Is-Increasing-During-Coronavirus-Pandemic>
- <https://builtin.com/data-science/random-forest-algorithm>
- <https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm>
- <https://www.knowledgehut.com/blog/data-science/knn-for-machine-learning>