Movie Recommender System with Predictive Modelling

Thesis submitted in partial fulfillment of the
requirements for the

Post Graduate Diploma in Data Science

By

Vijay S
17225760066

Under the guidance of

Dr Subhabaha Pal
Senior Faculty – Data Science and Machine Learning
Manipal Academy of Higher Education
Bangalore

MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL

Movie Recommender System with Predictive Modelling

Thesis submitted in partial fulfillment of the
requirements for

Post Graduate Diploma in Data Science

By

(Signature)
Vijay S
17225760066

Under the guidance of

Dr Subhabaha Pal
Senior Faculty – Data Science and Machine Learning
Manipal Academy of Higher Education
Bangalore

MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL

Movie Recommender System with Predictive Modelling

Thesis submitted in partial fulfillment of the
requirements for

Post Graduate Diploma in Data Science

By

Vijay S
17225760066

Examiner 1	**Examiner 2**
Signature:	Signature:
Name:	Name:

MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL

CERTIFICATE

This is to certify that the project work titled

Movie Recommender System with Predictive Modelling

is a bonafide record of the work done by

Vijay S
17225760066

In partial fulfillment of the requirements for the award of **Post Graduate Diploma in Data Science** under Manipal Academy of Higher Education, Manipal, Manipal and the same has not been submitted elsewhere for any kind of certification/recognition.

(Signature)
Dr Subhabaha Pal
Senior Faculty – Data Science and Machine Learning
Manipal Academy of Higher Education
Bangalore

TABLE OF CONTENTS

ACKNOWLEDGMENTS

First of all I express my sense of gratitude to the Management of Manipal Academy of Higher Education for providing me with excellent facilities in the college.

I have privileged to my mentor **Dr Subhabaha Pal** for his enlightening thoughts and remarkable guidance that helped me in doing my project.

I take this opportunity to express my profound gratitude and deep regards to my guide **Professor Mallikarjuna Doddamane** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I would also like to thank my educators at **Manipal Academy of Higher Education** and Research for sharing their expertise coupled with insightful comments, which played vital role in my thought process of analysing and completing the project.

Last but not the least, I thank my fellow batch-mates in Manipal for the stimulating discussions and sharing their thoughts and encouragement along with all the fun I had in the last one year.

ABSTRACT

Revenue and Result of a movie are the two most important factors that are required for analyzing a movie and it remains a challenging task for many film directors, reviewers and media companies to make accurate prediction on how much revenue a movie will generate and what would be the anticipated result of the movie.

It uses a linear regression model to find the most appropriate variable that helps in determining the revenue of the movie from 45000 rows dataset with good accuracy. It also uses a simple randomforest classification model to find the result of the movie that is able to classify movies as blockbuster, super hit, hit, average hit and flop movies.

It also uses sentiment analysis on movie story line to find if a movie has a positive or negative sentiment by taking the count of positive and negative words and visualize the same with the use of text mining and wordcloud package. Using the sentiment analysis it can also find which genre of movie is mostly created by filmmakers.

To recommend top n movies to the user based on the user's genre preferences it uses a simple function that can filter movies accordingly wherein the filtering is done based on movie popularity.

It also uses two collaborative models such as IBCF and UBCF to filter movies and recommend to user based on their preferences. This recommendation engine was created using variables movieId and userId with value variable as ratings so that recommendation is done using the user ratings. It is also able improve the user experience on the media portals which makes search process very fast with the top list of recommended movies to the user.

2. INTRODUCTION

2.1 MOTIVATION

At present, there are lot of movies being released daily across the world and lot of money is invested in creating those movies. There are lot of human effort behind each movie and large number of people life depends upon the result of these movies. Another interesting fact is that in some countries film and television industries gives good contributions to GDP. So predicting the revenue and result of the movie has become an important task of every film makers and media companies and that has inspired this project in which it performs an extensive EDA on Movie data to narrate the history and the story of Cinema and use this metadata in combination with MovieLens ratings to build various types of Recommender Systems and Predictive Models.

2.2 PROJECT SCOPE

The Movie Recommender System with Predictive Modelling has the following scope such as,

Regression Model

- To predict the revenue of the movies using regression models
- This model will be built using lm() function in R

Classification Model

- To classify the movies as hit or flop based on budget and revenue of the film.
- This model will be built using randomForest classification model in R.

Text Mining

- To do text mining on the movie story line column to do sentiment analysis to find the movie sentiment
- This will be implemented using libraries such as tm, and wordcloud.

Collaborative Filtering

- To build a recommendation engine using IBCF and UBCF model and filtering based on genre.

2.3 PROJECT GOAL

The goal of the project is to provide an efficient recommendation engine to the ecommerce or online media companies with prime focus on increasing their customer count and revenue.

2.4 LITERATURE / MARKET SURVEY

United States Hollywood is biggest film industry in the world as it generated $11.6 billion in the year 2016 making it the profitable film industry in the world. On the other hand India's Bollywood earned around $1.9 billion from films in the rear 2016. In terms of market share, Hollywood enjoys 84% of the world's movie collections, with around 200 movies, while the entire set of Indian movie industries, with around 1000 movies, owns around 4%. Within India, Hindi cinema takes in about 43% of the box office revenues.

Take Vault, an Israel-based artificial intelligence startup that's one of the newest entrant using analytics and algorithms to predict ticket sales. It is neural-network algorithm that relies on 30 years worth of box of office revenues, film budgets, audience demographics, and casting information to help determine box office potential.

Other companies in the race including U.K's Epagogix, founded in 2003, as well as ScriptBook and Boston-based Pilot. Forward-thinking studios like Legendary Pictures also rely on their own in-house analytics teams to devise data-informed marketing strategies.

There are three countries becoming competitive and often surpassing the U.S in terms of volume, quality and appeal and they are South Korea, Nigeria and India. India is by far the largest producer of films in the world. In 2011, india created over 1000 feature

flims and 1500 short films. Indian cinema also covers some of the largest ground, with nine different regions producing their own films in sixteen different languages, making it also one of the most diverse films industries in the world. So this project can be used with Indian movies data to predict the revenue and result of the upcoming movies.

2.5 ORGANISATION OF THE REPORT

The report consists of four major sections:

- **Project Description**, consist of domain understanding, dataset understanding and limitation, and project benefits.
- **Exploratory Data Analysis**, deals with data collection, data cleaning, data transformation.
- **Design**, focuses on feature engineering, understanding trends in data through process like correlation.
- **Modelling**, deals with creating model to predict revenue and result of the movie, recommendation engine for top n movies and followed by conclusion section which talks about finding and future work.

3. PROJECT DESCRIPTION

3.1 BUISNESS UNDERSTANDING

This project can be used by online media websites for

- Predicting the approximate revenue of the movies.
- Can categories movies based on its revenue and budget.
- Can provide effective recommendation engine to the user based on their preferences.
- Can get a quick idea about the story line of the movies instead of reading the entire story.

3.2 PROJECT STAKEHOLDERS

- *Users* who are using the services of online media portals.
- *Ecommerce Companies* who are holding product and user data and wants to retain their customers.
- *Media Companies* who wants to give good interface to their customer in order to increase their revenue.

3.2 DATASET UNDERSTANDING

- The dataset contains metadata for all 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include keywords, budget, revenue, release dates, languages, TMDB vote counts and vote averages.
- This dataset also has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1 - 5 and was obtained from the official GroupLens website.
- There are 17 columns in the dataset that are as follows,

S.No	Column Name		Description
1	Adult	-	Contain True or False where True means the films has adult content and vice versa
2	Budget	-	Budget invested in creating the film
3	Genres	-	Genre of the movie in json text format
4	Id	-	Id of the movie
5	Original Language	-	Language in which films was created
6	Original title	-	Actual title of the movie in its regional language
7	Overview	-	Story of the movie
8	Popularity	-	Percentage of movie popularity
9	Release Date	-	Actual date of movie release
10	Revenue	-	Total revenue generated by the movie
11	Runtime	-	Actual running time of the movie in minutes
12	Status	-	Contains details of the movie whether it has been released, rumoured, cancelled, in production and post production
13	Tagline	-	Short explanation of the movie story plot
14	Title	-	Title of the movie in English
15	Video	-	Available in video format or not
16	Vote Average	-	Average votes received by the movie
17	Vote Count	-	Total number of votes given to the movie in percentage

3.3 DATA LIMITATIONS

The main technical challenge it poses to classifying the result of the movie as there is no output column like result to train the model. The profit percentage generated by the movie to be calculated and needs to be classified as blockbuster, super hit, hit, average hit and flop categories.

The Genres column in the dataset in in character format but actual format of the column is json text format that is in key value pairs and needs to be treated accordingly to extract the genre out of it in dataframe format.

The overview and tagline columns that describes the story of the movie need text mining that requires lot of data cleaning process before doing topic modeling or sentiment analysis.

The ratings dataset size is too large and it is difficult to handle in R with less system configuration. Therefore, the IBCF and UBCF model was trained with only the first one lakhs rows of the ratings data.

3.4 BENEFITS OF THE PROJECT

With lot of money invested in making films, the revenue prediction will not only help the investors and producers but also the people who worked day and night to create the films with increased wages on every film’s success. Predicting the result of the movie at box office will ensure continuous flow of money in film industry and thus it will contribute to countries GDP. In today’s world, we all are in digital era so most of the people watch movies online in media portals and this recommendation engine will help the digital users with good interface and search option to watch their favorite movies.

4. EXPLORATORY DATA ANALYSIS

4.1 DATA COLLECTION

This dataset is an ensemble of data collected from TMDB and GroupLens. The Movie Details have been collected from the TMDB Open API. This product uses the TMDB API but is not endorsed or certified by TMDB. Their API also provides access to data on many additional movies, actors and actresses, crew members, and TV shows. The Movie Links and Ratings have been obtained from the Official GroupLens website.

4.2 DATA EXPLORATION

First the dataset is completed checked for any missing values and if it needs any imputation,

```
#### Check for NA values
```{r}
colSums(is.na(movies))
```
```

adult	budget	genres	id	original_language	original_title
0	0	0	0	0	0
overview	**popularity**	**release_date**	**revenue**	**runtime**	**status**
0	3	0	3	260	0
tagline	**title**	**video**	**vote_average**	**vote_count**	
0	0	3	3	3	

The runtime columns has most of the NA values in it and since the entire 260 rows has either NA or 0's in it, It doesn't need any imputation and thus all the NA values can be omitted.

The NA values are omitted using na.omit() finction in R, which removes entire rows consisting NA values. Before the removal of NA values the movies data had 45463 rows but after removing NA values the dataset now has 45203 rows.

dim(movies) # dimension of the dataset before removing the NA values

```
[1] 45463    17
```

dim(movies_df) # dimensions of the dataset after removing the NA values

```
[1] 45203    17
```

Removing the NA values

```{r}
movies_df = na.omit(movies)
```

Final check for NA values

```{r}
colSums(is.na(movies_df))
```

adult	budget	genres	id	original_language	original_title
0	0	0	0	0	0
overview	**popularity**	**release_date**	**revenue**	**runtime**	**status**
0	0	0	0	0	0
tagline	**title**	**video**	**vote_average**	**vote_count**	
0	0	0	0	0	

The output column for revenue prediction is already given in the dataset but for developing classification model the output column result is not given, it need to be derived from the existing columns.

First a column named “Difference” is derived from taking the difference between budget and revenue column.

Another column named “Profit Percent” is derived from “Difference” column and finally the output column named “Result” is derived with levels such as,

- Blockbuster, *if profit_percent >= 80*
- Super Hit, *if profit_percent >= 50 and < 80*
- Hit, *if profit_percent >= 30 and < 50*
- Average Hit, *if profit_percent > 0 and < 30*
- Neutral, *if profit_percent == 0 &*
- Flop, *if profit_percent < 0*

There were some column which are in character type that need to be converted to factor columns for avoiding runtime error in model creation. Columns like, Original

language, Original title, Result and Status.The dataset further divided into numeric and categorical datasets to get understanding of the data.

4.3 DATA COMPLEXITY

The dataset has a complex data like genres column, which is in JSON text format, and it is in key value pairs. So proper conversion of JSON text format to dataframe is required. Another challenging task is to clean the genre column like removing single quotes and replacing it with double quotes as JSON format doesn't support single quotes. If we try to convert without replacing the single quote R will through error.

```
#### Creating new dataframe for genre column in movies dataset which is in JSON format
```{r}
movies_df$genres = gsub("'","\"", movies_df$genres)

genre_df = movies_df %>% filter(nchar(genres) > 2) %>%
                      mutate(js = lapply(genres, fromJSON)) %>%
                      unnest(js) %>% select(id, title, genre = name)

genre_bud_rev = movies_df %>% select(id, budget, revenue)

genre = merge(genre_df, genre_bud_rev, by.x = "id", by.y = "id", all.x = T)

movie_genre = genre %>% filter(budget != 0 & revenue != 0)
```
```

The original rating dataset has 26 million ratings of the user and is of huge size in which it requires higher system configuration to handle it. So in order to avoid this complexity and make the model run on low commodity hardware, the rating dataset is limited to 1 lakhs rows to build IBCF and UBCF recommendation engine.

4.4 DATA CLEANING

The tagline column in the dataset needs some data cleaning for doing sentiment analysis on it. To achieve this, the column is cleaned by removing the stopwords, punctuations, whitespaces and numbers. A corpus is created which then been converted to get a document term matrix for doing sentiment analysis.

Exploring the tagline column to gather information on most used tagwords

```{r}

# taking the tagline column and separating each words
tagline = gsub("[^A-Za-z///]", " ", movies_df$tagline)

# Creating the corpus for text mining
corpus = Corpus(VectorSource(list(movies_df$tagline)))

# Removing punctuation from the corpus
corpus = tm_map(corpus,removePunctuation)

# Converting uppercase letters to lowecase
corpus = tm_map(corpus,content_transformer(tolower))

# Removing the unwanted whitespaces
corpus = tm_map(corpus,stripWhitespace)

# Removing the stopwords from the corpus
corpus = tm_map(corpus,removeWords,stopwords("english"))

# Finally creating the document term matrix for doing sentiment analysis
dtm_tag = DocumentTermMatrix(VCorpus(VectorSource(corpus[1]$content)))
```

4.5 DATA TRANSFORMATION

Using the genre column a dataframe has been created but for doing collaborative filtering and it needs some transformations like genre of the movies must be transformed to columns and movieId must be transformed to rows.

For each movieId the genre will be either 0 or 1, where 0 means NO and 1 means YES for that genre. To filter the top movies it needs popularity column so it has to be merged with movies data to get the data ready to make collaborative filtering on genre.

Collabarative Filtering using Movie Genre

```{r}
m = cast(movie_genre, formula = title~genre)
colfil = merge(m, movies_df, by.x = "title", by.y = "title", all.x = T)
colfil_df = colfil[,c(1:21,23,29,31,36,37)]
```

(Before transformation)

id <int>	title <chr>	genre <chr>
5	Four Rooms	Crime
5	Four Rooms	Comedy
11	Star Wars	Adventure
11	Star Wars	Action
11	Star Wars	Science Fiction
12	Finding Nemo	Family
12	Finding Nemo	Animation
13	Forrest Gump	Comedy
13	Forrest Gump	Drama
13	Forrest Gump	Romance

(After transformation)

title <chr>	Action <int>	Adventure <int>	Animation <int>	Comedy <int>
(500) Days of Summer	0	0	0	1
[REC]	0	0	0	0
[REC]Â²	0	0	0	0
10 Cloverfield Lane	0	0	0	0
10 Things I Hate About You	0	0	0	1
10 to Midnight	0	0	0	0
10,000 BC	1	1	0	0
101 Dalmatians	0	0	0	1
102 Dalmatians	0	0	0	1
10th & Wolf	1	0	0	0

5. DESIGN

5.1 ANALYTICAL METHODS AND TECHNOLOGY USED

For analysing the dataset, it was separated into numeric columns dataset and categorical columns dataset and methods like,

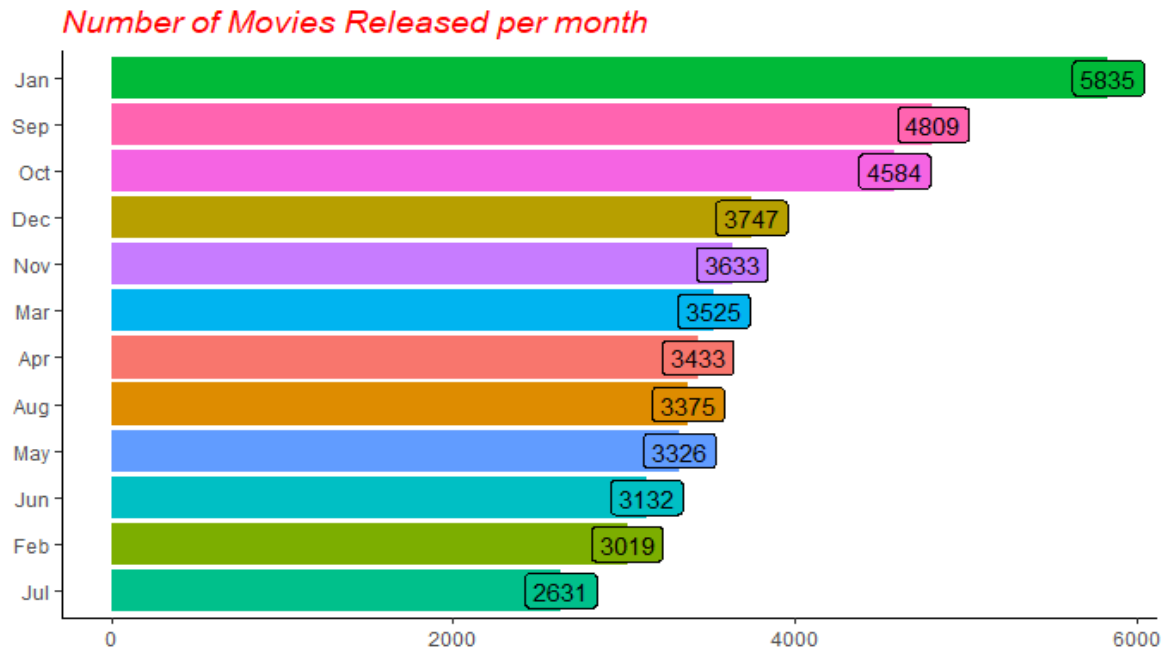
- Text Mining
- Predictive Modelling
- Collaborative Filtering
- Recommendation System, were used to get the overall understanding of the dataset.

Technologies used,

- R Programming, for model creation &
- ggplot2, for visualization of the data.

5.2 DATA VISUALIZATION

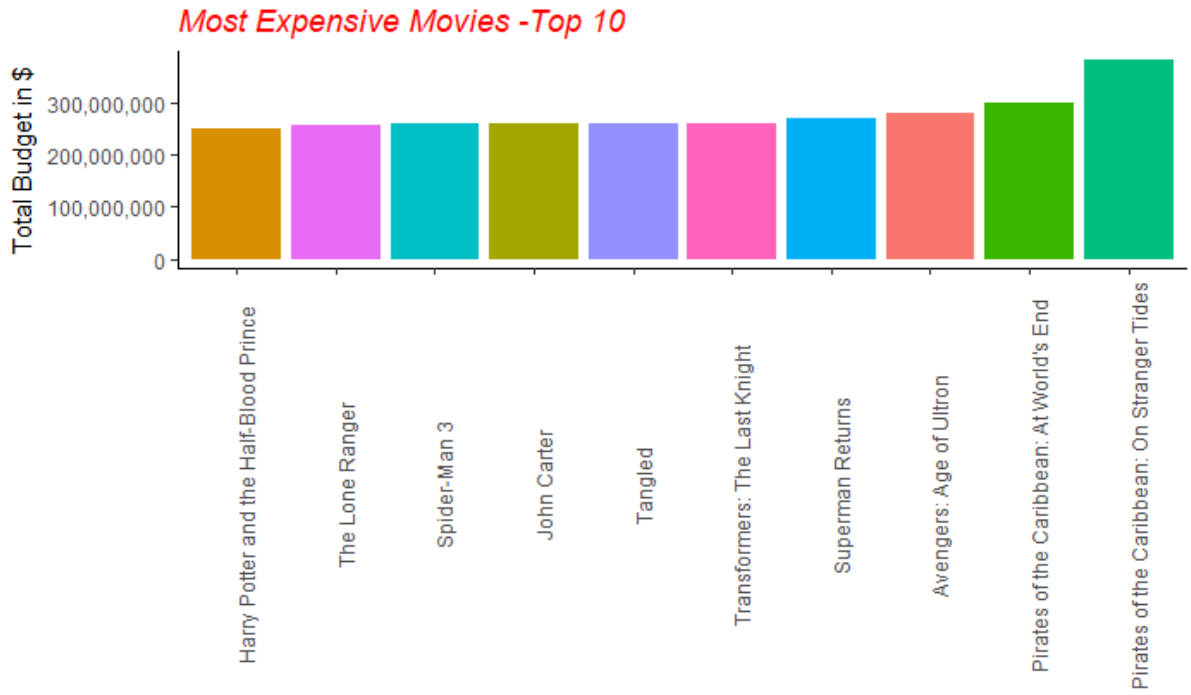
5.2.1 Which year has seen maximum release of movies?



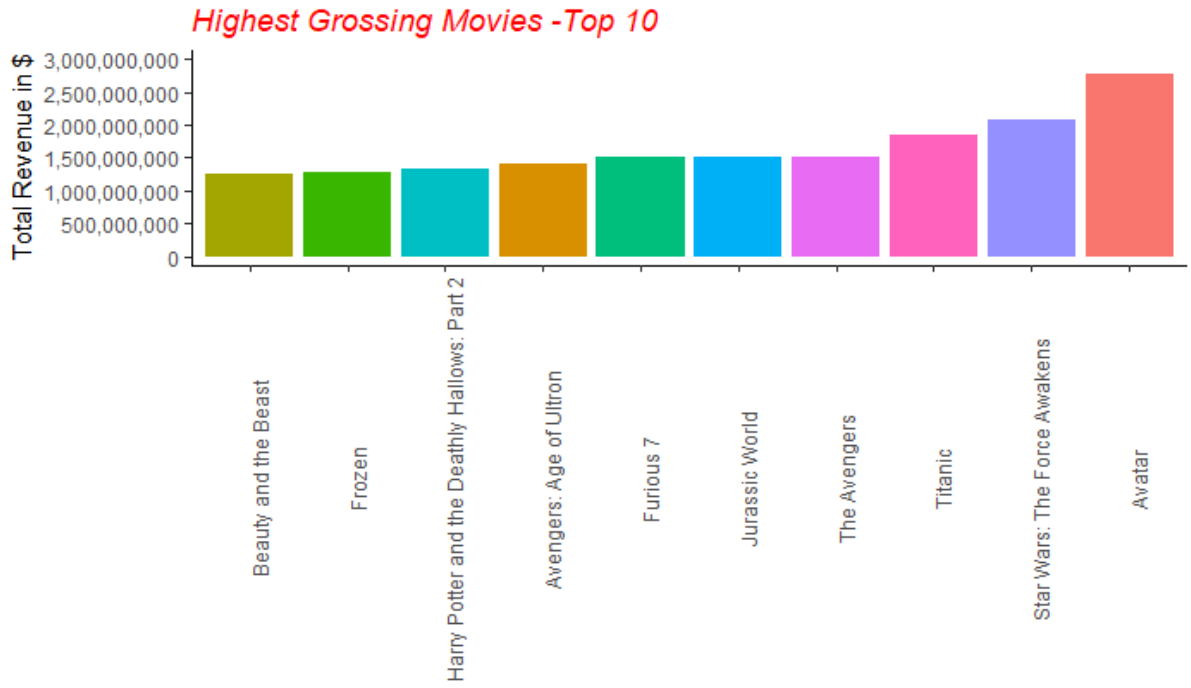
It appears that **January** is the most popular month when it comes to movie releases. In Hollywood circles, this is also known as the *the dump month* when sub par movies are released by the dozen.

In the month of September and October of every year more number of movies were released as most of the holidays comes during these months so that more people tend to watch their movie in theatres.

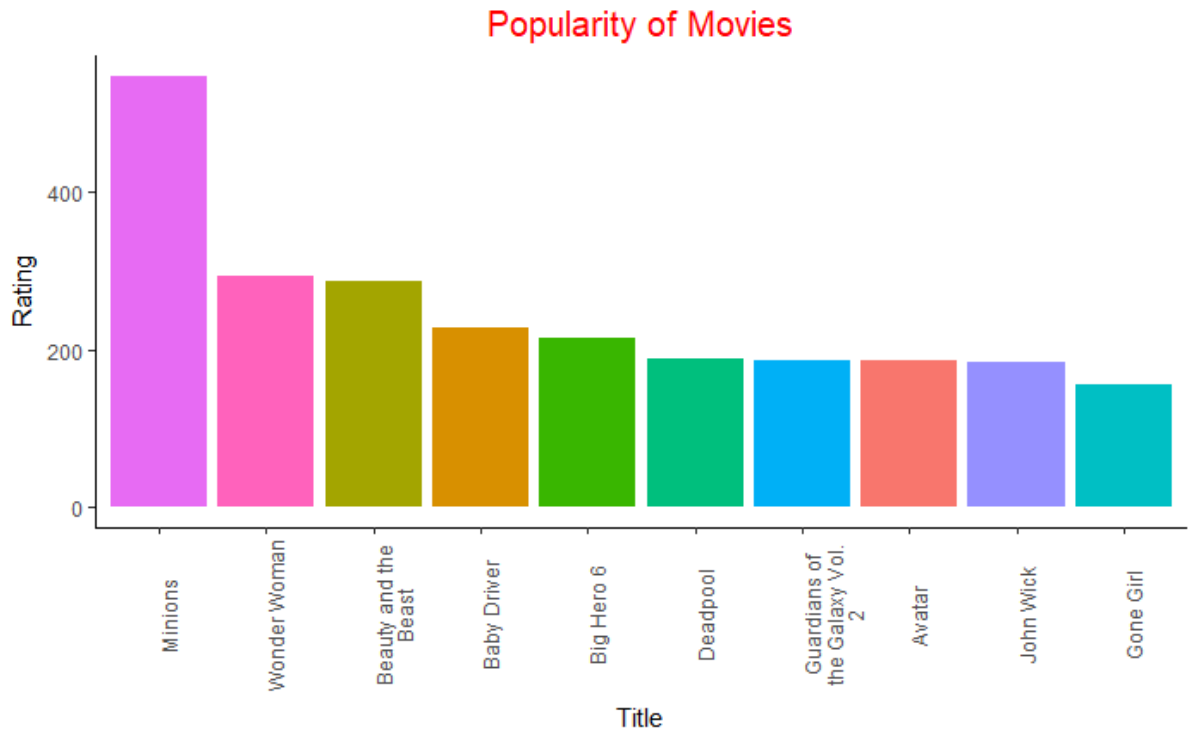
5.2.2 Most Expensive Movies



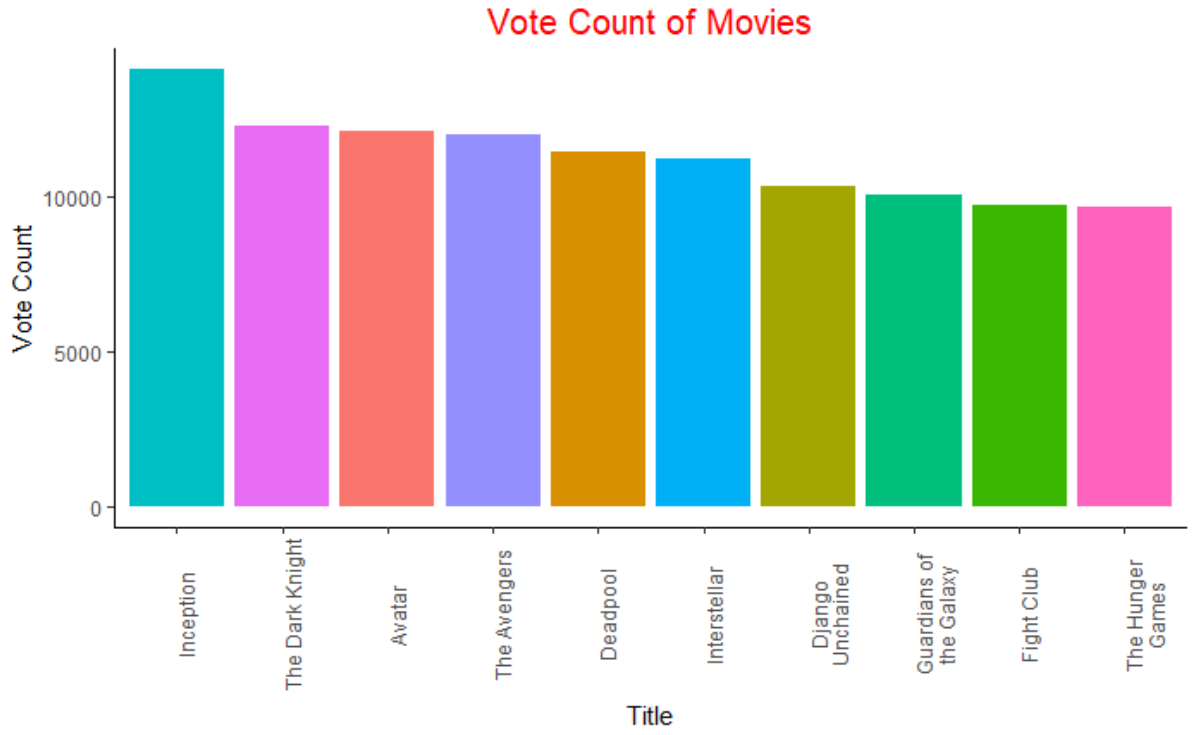
5.2.3 Highest Grossing Movies



5.2.4 Most Popular Movies



5.2.5 Movies with Highest Vote Count



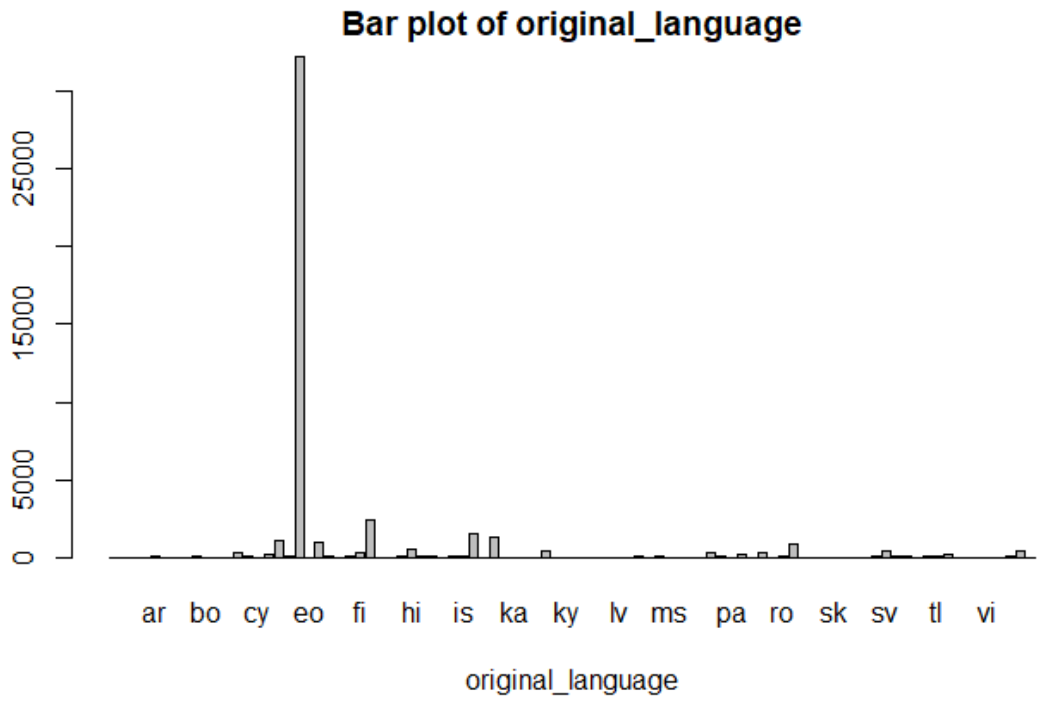
5.2.6 Wordcloud for Positive Words in Movie Tagline

5.2.7 Wordcloud for Negative Words in Movie Tagline

Positive Sentiment : 18071

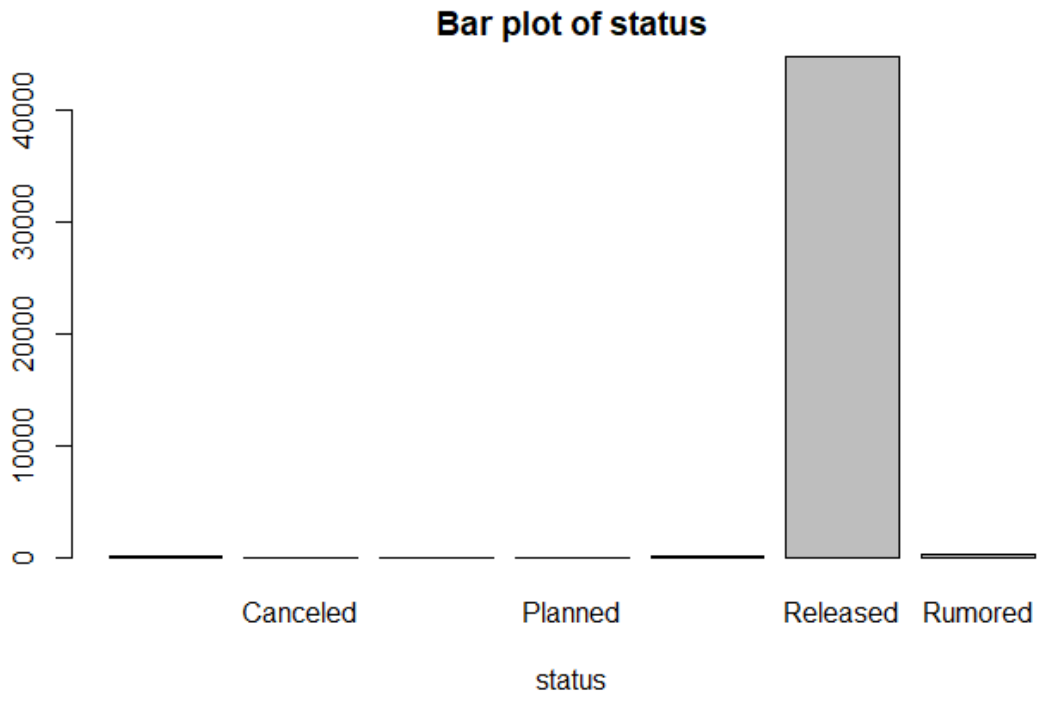
Negative Sentiment : 15578

5.2.8 Language in which most number of movies released.

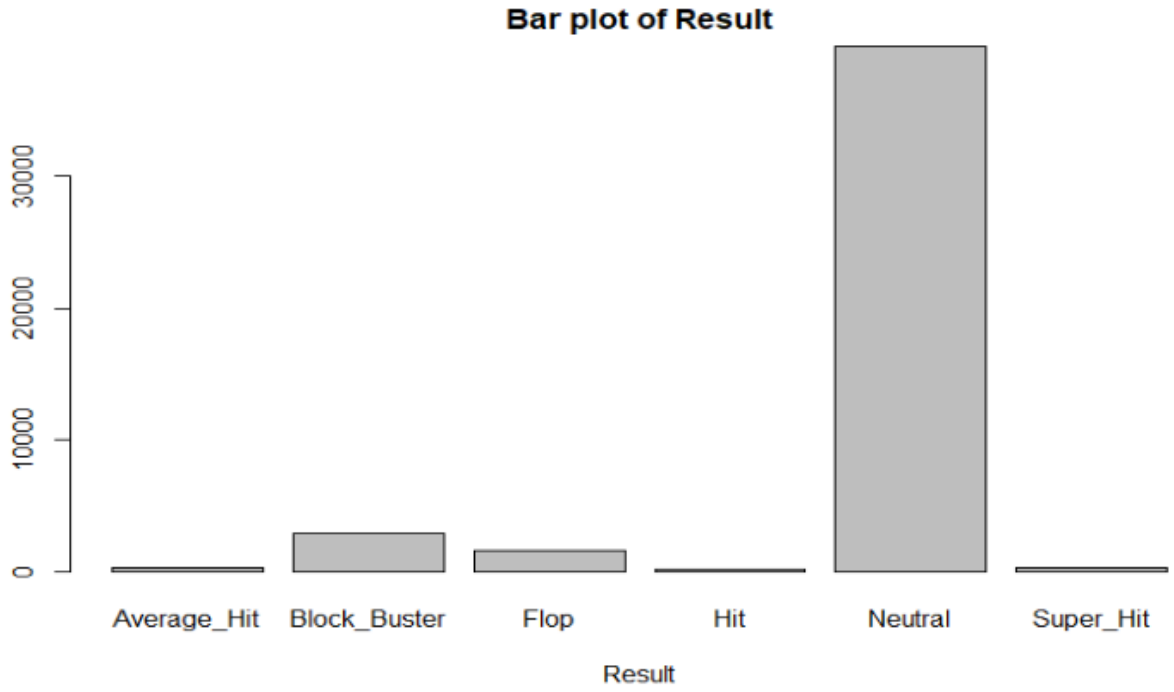


The above plot says that most of the movies released in Spanish Esperanto Language. Esperanto is spoken by some 2 million people as a second language in 115 countries, most of them in Central and Eastern Europe, East Asia, and South America.

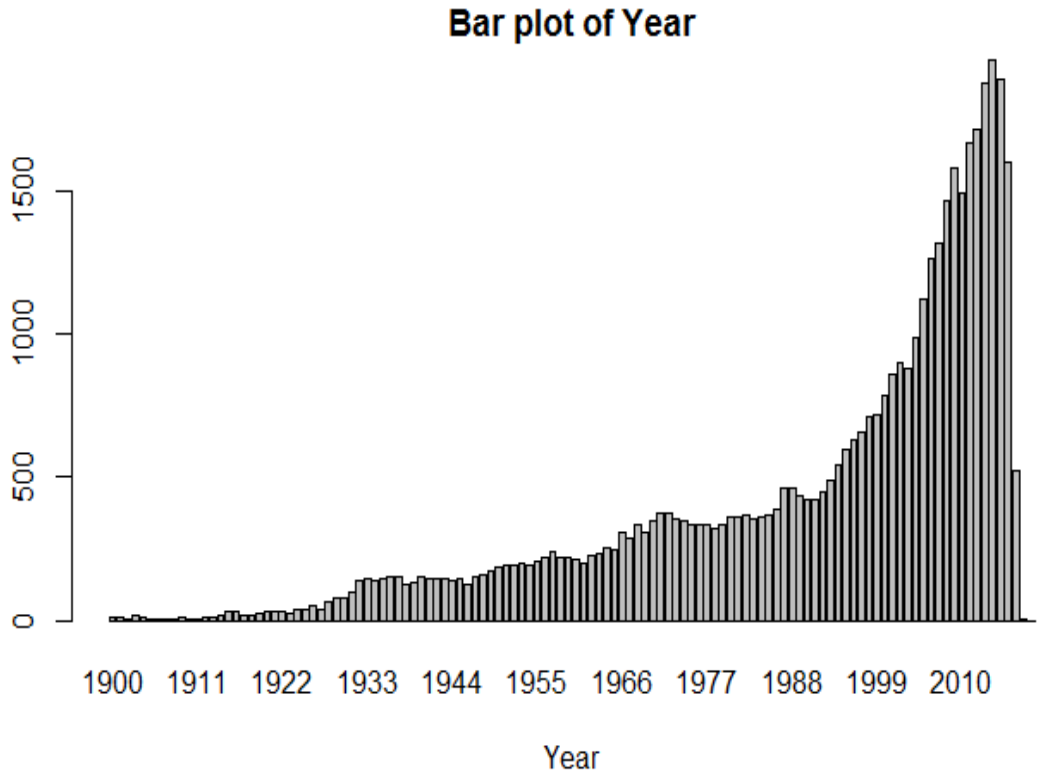
5.2.9 Status of the movies



5.2.10 Result of the movies

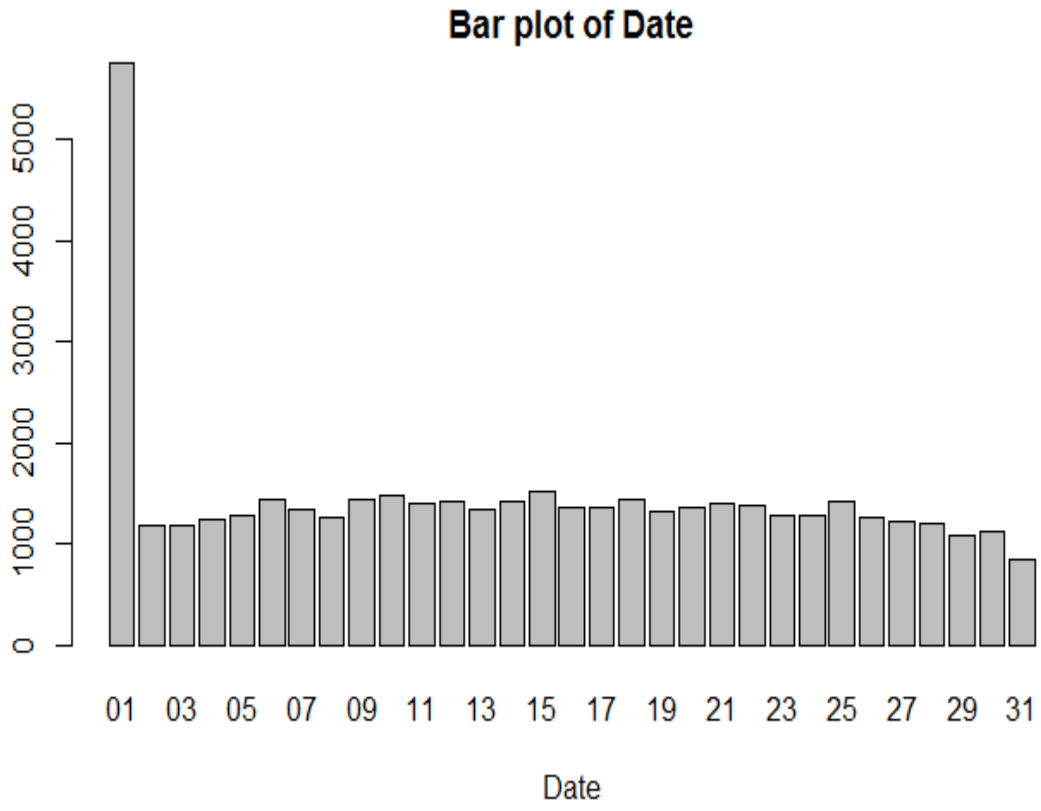


5.2.11 Year wise movie release



More Number of Movies released during 20th Century – The Golden era of Flims

5.2.12 Day wise movie release



Most of the movies were released during the 1st day of every month.

5.1 FEATURE ENGINEERING

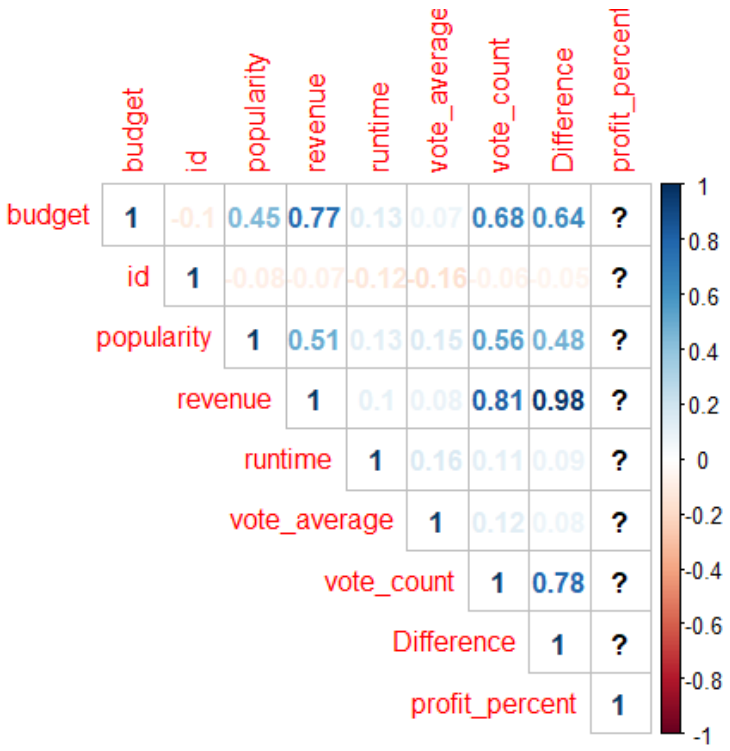
To train the classification model it needs the corresponding output variable. Since there is no output classifier in the dataset I went on to derive 2 new variables,

1. Difference
2. Profit_Percent

While evaluating the best model for predicting revenue the derived column “Difference” turned out to be important in obtaining the best performance from the regression model.

To find the best predictor for regression model a simple correlation analysis was done and found that only columns such as “Difference”, ”vote_count” and “budget” showed good correlation with the output column “revenue”

```r
```{r}
#### Correlation Analysis
corr <- cor(movies_df[sapply(movies_df, is.numeric)])
corrplot(corr,type = "upper",method = "number") # Plot correlation matrix
corr[lower.tri(corr)] = 0
corr <- corr-diag(nrow(corr))
index <- which(abs(corr) > 0.7,arr.ind = T)
df <- data.frame(rowname=rownames(index),colname=colnames(corr)[index[,2]],cor_value=corr[index])
df
```
```



rowname <fctr>	colname <fctr>	cor_value <dbl>
budget	revenue	0.7687322
revenue	vote_count	0.8119861
revenue	Difference	0.9760239
vote_count	Difference	0.7813089

To improve the accuracy of the classification model I removed some the variable such as, original_language, original_title, tagline, title, overview & genres.

5.2 SHORT DATA SNAPSHOTS

Movies Dataset:

adult	budget	genres	id	original_language	original_title
FALSE	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, ...	862	en	Toy Story
FALSE	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, 'name': 'Fantasy'}, ...	8844	en	Jumanji
FALSE	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, 'name': 'Comedy'}]	15602	en	Grumpier Old Men
FALSE	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}, {'id'...	31357	en	Waiting to Exhale
FALSE	0	[{'id': 35, 'name': 'Comedy'}]	11862	en	Father of the Bride Part II
FALSE	60000000	[{'id': 28, 'name': 'Action'}, {'id': 80, 'name': 'Crime'}, {'id': 1...	949	en	Heat
FALSE	58000000	[{'id': 35, 'name': 'Comedy'}, {'id': 10749, 'name': 'Romance'}]	11860	en	Sabrina
FALSE	0	[{'id': 28, 'name': 'Action'}, {'id': 12, 'name': 'Adventure'}, {'i...	45325	en	Tom and Huck
FALSE	35000000	[{'id': 28, 'name': 'Action'}, {'id': 12, 'name': 'Adventure'}, {'i...	9091	en	Sudden Death
FALSE	58000000	[{'id': 12, 'name': 'Adventure'}, {'id': 28, 'name': 'Action'}, {'i...	710	en	GoldenEye
FALSE	62000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}, {'id'...	9087	en	The American President
FALSE	0	[{'id': 35, 'name': 'Comedy'}, {'id': 27, 'name': 'Horror'}]	12110	en	Dracula: Dead and Loving It
FALSE	0	[{'id': 10751, 'name': 'Family'}, {'id': 16, 'name': 'Animation'...	21032	en	Balto
FALSE	44000000	[{'id': 36, 'name': 'History'}, {'id': 18, 'name': 'Drama'}]	10858	en	Nixon
FALSE	98000000	[{'id': 28, 'name': 'Action'}, {'id': 12, 'name': 'Adventure'}]	1408	en	Cutthroat Island
FALSE	52000000	[{'id': 18, 'name': 'Drama'}, {'id': 80, 'name': 'Crime'}]	524	en	Casino
FALSE	16500000	[{'id': 18, 'name': 'Drama'}, {'id': 10749, 'name': 'Romance'}]	4584	en	Sense and Sensibility
FALSE	4000000	[{'id': 80, 'name': 'Crime'}, {'id': 35, 'name': 'Comedy'}]	5	en	Four Rooms
FALSE	30000000	[{'id': 80, 'name': 'Crime'}, {'id': 35, 'name': 'Comedy'}, {'id':...	9273	en	Ace Ventura: When Nature Calls
FALSE	60000000	[{'id': 28, 'name': 'Action'}, {'id': 35, 'name': 'Comedy'}, {'id'...	11517	en	Money Train
FALSE	30250000	[{'id': 35, 'name': 'Comedy'}, {'id': 53, 'name': 'Thriller'}, {'id...	8012	en	Get Shorty

popularity	release_date	revenue	runtime	status	tagline	title	video
21.946943	30-10-1995	373554033	81	Released		Toy Story	FALSE
17.015539	15-12-1995	262797249	104	Released	Roll the dice and unleash the excitement!	Jumanji	FALSE
11.712900	22-12-1995	0	101	Released	Still Yelling. Still Fighting. Still Ready for Love.	Grumpier Old Men	FALSE
3.859495	22-12-1995	81452156	127	Released	Friends are the people who let you be yourself... and ne...	Waiting to Exhale	FALSE
8.387519	10-02-1995	76578911	106	Released	Just When His World Is Back To Normal... He's In For The ...	Father of the Bride Part II	FALSE
17.924927	15-12-1995	187436818	170	Released	A Los Angeles Crime Saga	Heat	FALSE
6.677277	15-12-1995	0	127	Released	You are cordially invited to the most surprising merger o...	Sabrina	FALSE
2.561161	22-12-1995	0	97	Released	The Original Bad Boys.	Tom and Huck	FALSE
5.231580	22-12-1995	64350171	106	Released	Terror goes into overtime.	Sudden Death	FALSE
14.686036	16-11-1995	352194034	130	Released	No limits. No fears. No substitutes.	GoldenEye	FALSE
6.318445	17-11-1995	107879496	106	Released	Why can't the most powerful man in the world have the ...	The American President	FALSE
5.430331	22-12-1995	0	88	Released		Dracula: Dead and Loving It	FALSE
12.140733	22-12-1995	11348324	78	Released	Part Dog. Part Wolf. All Hero.	Balto	FALSE
5.092000	22-12-1995	13681765	192	Released	Triumphant in Victory, Bitter in Defeat. He Changed the ...	Nixon	FALSE
7.284477	22-12-1995	10017322	119	Released	The Course Has Been Set. There Is No Turning Back. Prep...	Cutthroat Island	FALSE
10.137389	22-11-1995	116112375	178	Released	No one stays at the top forever.	Casino	FALSE
10.673167	13-12-1995	135000000	136	Released	Lose your heart and come to your senses.	Sense and Sensibility	FALSE
9.026586	09-12-1995	4300000	98	Released	Twelve outrageous guests. Four scandalous requests. A...	Four Rooms	FALSE
8.205448	10-11-1995	212385533	90	Released	New animals. New adventures. Same hair.	Ace Ventura: When Nature Calls	FALSE
7.337906	21-11-1995	35431113	103	Released	Get on, or GET OUT THE WAY!	Money Train	FALSE
12.669608	20-10-1995	115101622	105	Released	The mob is tough, but itâ€™s nothing like show business.	Get Shorty	FALSE

Rating Dataset:

	userId	movieId	rating	timestamp
1	1	31	2.5	1260759144
2	1	1029	3.0	1260759179
3	1	1061	3.0	1260759182
4	1	1129	2.0	1260759185
5	1	1172	4.0	1260759205
6	1	1263	2.0	1260759151
7	1	1287	2.0	1260759187
8	1	1293	2.0	1260759148
9	1	1339	3.5	1260759125
10	1	1343	2.0	1260759131
11	1	1371	2.5	1260759135
12	1	1405	1.0	1260759203
13	1	1953	4.0	1260759191
14	1	2105	4.0	1260759139
15	1	2150	3.0	1260759194
16	1	2193	2.0	1260759198
17	1	2294	2.0	1260759108
18	1	2455	2.5	1260759113
19	1	2968	1.0	1260759200
20	1	3671	3.0	1260759117
21	2	10	4.0	835355493
22	2	17	5.0	835355681

5.5 SHORT CODE SNIPPETS

```
# Deriving Difference column
movies_df$Difference = ifelse(movies_df$budget == 0 | movies_df$revenue == 0,
                              0, movies_df$revenue - movies_df$budget)

# Deriving Profit percent column
movies_df$profit_percent = ((movies_df$Difference*100)/movies_df$budget)

movies_df$Result = ifelse(movies_df$profit_percent >= 80,"Block_Buster",
                          ifelse(movies_df$profit_percent >= 50 & movies_df$profit_percent < 80, "Super_Hit",
                                 ifelse(movies_df$profit_percent >= 30 & movies_df$profit_percent < 50, "Hit",
                                        ifelse(movies_df$profit_percent > 0 & movies_df$profit_percent < 30, "Average_Hit",
                                               ifelse(movies_df$profit_percent == 0, "Neutral","Flop")))))
```

```
# Converting character column to catagorical column
movies_df$original_language = as.factor(movies_df$original_language)
movies_df$original_title = as.factor(movies_df$original_title)
movies_df$status = as.factor(movies_df$status)
movies_df$Result = as.factor(movies_df$Result)

# Sparating numeric and categorical column
numeric_col <- movies_df[sapply(movies_df, is.numeric)]

categorical_col <- movies_df[sapply(movies_df,is.factor)]
```

Highest Budget& Revenue Movies

```
```{r}
movies_df %>% select(original_title,budget) %>%
              drop_na(original_title) %>%
              arrange(desc(budget)) %>%
              head(10) %>%
              ggplot(aes(reorder(original_title,budget), budget,fill=original_title)) +
              geom_bar(stat="identity") + theme_classic() +
              theme(axis.text.x = element_text(angle=90),
                    plot.title=element_text(color="Red",face="italic"),
                    legend.position="none") + scale_y_continuous(labels=scales::comma) +
              labs(x="",y="Total Budget in $",title="Most Expensive Movies -Top 10")
```
```

Highest Grossing Movies

```
```{r}
movies_df %>% select(original_title,revenue) %>% drop_na(original_title) %>%
             arrange(desc(revenue)) %>%
             head(10) %>%
             ggplot(aes(reorder(original_title,revenue),revenue,fill=original_title)) +
             geom_bar(stat="identity") + theme_classic() +
             theme(axis.text.x=element_text(angle=90),plot.title=element_text(color="Red",face="italic"),
                   legend.position="none") + scale_y_continuous(limits=c(0,3000000000),
                                                                breaks=seq(0,3000000000,500000000),
                                                                labels=scales::comma) +
             labs(x="",y="Total Revenue in $",title="Highest Grossing Movies -Top 10")
```
```

Movies with Highest Popularity

```
```{r}
movies_df %>% group_by(original_title) %>%
              arrange(desc(popularity)) %>%
              head(10) %>%
              ggplot(aes(factor(original_title,levels=original_title),
                         popularity,fill=original_title)) +
              geom_bar(stat="identity") +
              theme_classic() +
              theme(axis.text.x=element_text(angle=90,hjust=0.5),
                    plot.title=element_text(hjust=0.5,size=15,color="red"),
                    legend.position="none") +
              labs(x="Title",y="Rating",title="Popularity of Movies") +
              scale_x_discrete(labels=function(x) str_wrap(x,width=15))
```
```

6. MODELLING

6.1 SELECTION OF MODEL / TECHNIQUE

LINEAR REGRESSION

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

- If the goal is prediction, or forecasting, or error reduction, linear regression can be used to fit a predictive model to an observed dataset of values of the response and explanatory variables. After developing such a model, if additional values of the explanatory variables are collected without an accompanying response value, the fitted model can be used to make a prediction of the response.
- If the goal is to explain variation in the response variable that can be attributed to variation in the explanatory variables, linear regression analysis can be applied to quantify the strength of the relationship between the response and the explanatory variables, and in particular to determine whether some explanatory variables may have no linear relationship with the response at all, or to identify which subsets of explanatory variables may contain redundant information about the response.

Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the "lack of fit" in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares cost function as in ridge regression (L^2-norm penalty) and lasso (L^1-norm penalty). Conversely, the least squares approach can be used to fit

models that are not linear models. Thus, although the terms "least squares" and "linear model" are closely linked, they are not synonymous.

RANDOM FOREST

A main drawback of decision trees is that they tend to overfit the training data. Random forests are one way to address this problem. A random forest is essentially a collection of decision trees, where each tree is slightly different from the others. The idea behind random forests is that each tree might do a relatively good job of predicting, but will likely overfit on part of the data. If we build many trees, all of which work well and overfit in different ways, we can reduce the amount of overfitting by averaging their results. This reduction in overfitting, while retaining the predictive power of the trees, can be shown using rigorous mathematics.

To implement this strategy, we need to build many decision trees. Each tree should do an acceptable job of predicting the target, and should also be different from the other trees. Random forests get their name from injecting randomness into the tree building to ensure each tree is different. There are two ways in which the trees in a random forest are randomized: by selecting the data points used to build a tree and by selecting the features in each split test.

USER BASED COLLABORATIVE FILTERING

A user-based recommendation engine recommends movies based on what other users with similar profiles have watched and liked in the past. As an example of a user-based recommender, imagine there is a movie lover who watches movies regularly, every Friday evening. He’s an unmarried man who’s also a working professional. A user-based recommender could go and look up movie recommendations based on what other unmarried, professional men who watch movies regularly have liked.

Imagine that we want to recommend a movie to our friend Stanley. We could assume that similar people will have similar taste. Suppose that me and Stanley have seen the same movies, and we rated them all almost identically. But Stanley hasn’t seen ‘The

Godfather: Part II' and I did. If I love that movie, it sounds logical to think that he will too. With that, we have created an artificial rating based on our similarity.

Well, UB-CF uses that logic and recommends items by finding similar users to the *active user* (to whom we are trying to recommend a movie).

In other words, we are creating a User-Item Matrix, predicting the ratings on items the active user has not see, based on the other similar users. This technique is memory-based.

ITEM BASED COLLABORATIVE FILTERING

An item-based recommender would make recommendations based on similarities between movies; in other words, it would recommend movies that are similar to ones that a user already likes. As an example of this, imagine you watch the movie 'Kung Fu Panda' and you liked it so you gave it five stars.

An item-based collaborative filtering system would then look into similar movies from the same genre (perhaps animated, fighting, comedy or based on similar storyline) and then recommend to you similar movies based on the preference you indicated when you gave 'Kung Fu Panda' five stars. In fact, and item-based collaborative filtering system can even make recommendations based on any variety of common elements, such as movies about pandas, movies from the same producers, directors, etc. For this example, it's most likely that the primary suggestions will include 'Kung Fu Panda 2' and 'Kung Fu Panda 3' followed by other cases.

It is a model-based algorithm for making recommendations. In the algorithm, the similarities between different items in the dataset are calculated by using one of a number of similarity measures, and then these similarity values are used to predict ratings for user-item pairs not present in the dataset.

The similarity values between items are measured by observing **all the users who have rated both the items**. The similarity between two items is dependent upon the ratings given to the items by users who have rated both of them.

The main idea behind this model is that, Suppose we have two different users: *A* and *B*. Also, we have *item I* and *item J*. *User A* rated *item I* with 1 star and the *item J* with 1.5. If the *User B* rated *Item I* with a 2. We can make the assumption that the difference between both items will be the same as *User A*. With this in mind, *User B* would rate *Item J* as: 2+ (1,5–1) = 2,5.

TEXT MINING

Words express various kinds of sentiments that may be positive, negative, strong, or weak. To perform sentiment analysis, it is important to understand the polarity of words and classify sentiments into categories such as positive, negative, or neutral. This task can be accomplished through the use of sentiment lexicons.

There are different types of sentiment lexicons available that have words classified as having positive or negative sentiments. The online lexicons may not be sufficient because they may not have enough words or do not pertain to the domain or topic of discussion. In such a case, a new sentiment lexicon can be built.

This can be done using techniques such as a semi-supervised lexicon, which leverages a small amount of information (e.g., a few labeled examples or a few hand-built patterns) to bootstrap a complete lexicon through the learning of lexicons. In a bootstrapping approach, a high-precision classifier is first used to identify some subjective and objective sentences. A set of patterns is learned from these identified subjective and objective sentences. The learned patterns are then used to extract more subjective and objective sentences. This process can then be repeated until the desired lexicon is built.

6.2 CHALLENGES FACED

First the ratings dataset is too large and needs high system configuration for analysing and model building. To avoid this the ratings dataset is limited to 100000 rows and the collaborative model is trained on it. Secondly the dataset has many values as 0 which made it difficult whether to treat 0 as missing values and impute them or just to omit them.

6.3 EVALUATION AND MODEL INTERPRETATION

LINEAR REGRESSION

```r
# Preparing train and test data for model building
mov_train = movies_df[sample(1:nrow(movies_df),0.8*nrow(movies_df)),]
mov_test = movies_df[sample(1:nrow(movies_df),0.2*nrow(movies_df)),]


revenue_model_1 = lm(revenue~budget, data = mov_train) # Model 1 with only budget

revenue_model_2 = lm(revenue~budget+vote_count, data = mov_train) # Model 2 with budget and vote_count

revenue_model_3 = lm(revenue~budget+Difference, data = mov_train) # Model 3 with budget and difference

revenue_model_4 = lm(revenue~budget+vote_count+Difference, data = mov_train) # Model 4 with budget, vote_count and difference


# Summaray of all the models to find which model performs well
summary(revenue_model_1)
summary(revenue_model_2)
summary(revenue_model_3)
summary(revenue_model_4)
```

Checking the Adjusted R-Square value for find the best model

For revenue_model_1 - Adjusted R-Square value = 0.58

For revenue_model_2 - Adjusted R-Square value = 0.75

For revenue_model_3 - Adjusted R-Square value = 0.9896

For revenue_model_4 - Adjusted R-Square value = 0.9897

The models 3 and 4 show good performance with 98% each but the revenue_model_4 has multicollinearity problem as the columns “difference” and “vote_count” has high correlation between them so the revenue_model_3 with only budget and difference columns it is able to give 98.96% accuracy and it does not violate the basic assumptions of linear regression and it the best performing model. To proof it all the four models has been evaluated with making predictions and checking the RMSE value of each model as follows,

Revenue Predictions

```r
mov_test$predicted_Result_1 = predict(revenue_model_1, mov_test %>% select(-revenue))
mov_test$predicted_Result_2 = predict(revenue_model_2, mov_test %>% select(-revenue))
mov_test$predicted_Result_3 = predict(revenue_model_3, mov_test %>% select(-revenue))
mov_test$predicted_Result_4 = predict(revenue_model_4, mov_test %>% select(-revenue))
```

```
#### Checking the RMSE value for better performing model
```{r}
rmse_1 = sqrt(mean(mov_test$revenue - mov_test$predicted_Result_1)^2)

rmse_2 = sqrt(mean(mov_test$revenue - mov_test$predicted_Result_2)^2)

rmse_3 = sqrt(mean(mov_test$revenue - mov_test$predicted_Result_3)^2)

rmse_4 = sqrt(mean(mov_test$revenue - mov_test$predicted_Result_4)^2)

rmse = c(rmse_1,rmse_2,rmse_3,rmse_4)
lm_model = c("revenue_model_1","revenue_model_2","revenue_model_3","revenue_model_4")

rmse_df = data.frame(rmse, lm_model)

rmse_df %>% arrange(rmse_df$rmse) %>% head(1) # Best performing model = revenue_model_3

```
```

	rmse <dbl>	lm_model <fctr>
1	25737.28	revenue_model_3

The RMSE value of revenue_model_3 is very less as compared to other models and it turn out to be the best performing model.

RANDOM FOREST

```
#### Random Forest Classification Model
```{r}
# Test and train data
rf_train = data[sample(1:nrow(data),0.7*nrow(data)),]
rf_test = data[sample(1:nrow(data),0.3*nrow(data)),]

mtry = round(sqrt(length(colnames(rf_train))-1)) # Sampling the columns of train data

model_rf = randomForest(Result~., data = rf_train %>% select(-original_language,
                                                             -original_title,
                                                             -tagline,
                                                             -title,
                                                             -overview,
                                                             -genres),
                        ntree = 100, mtry = mtry) # Model building

rf_test$Predicted_Result_1 = predict(model_rf, rf_test) # Predicting the result

cm = confusionMatrix(rf_test$Predicted_Result_1, rf_test$Result) # Checking the model performance
cm
cm$overall['Accuracy']*100

```
```

```
Confusion Matrix and Statistics

             Reference
Prediction     Average_Hit Block_Buster  Flop   Hit Neutral Super_Hit
  Average_Hit          115           0     0     0       0         0
  Block_Buster           0         882     0     0       0         0
  Flop                   0           0   481     0       0         0
  Hit                    0           0     0    65       0         1
  Neutral                0           0     1     0   11936         0
  Super_Hit              0           0     0     1       0        78

Overall Statistics

               Accuracy : 0.9998
                 95% CI : (0.9994, 1)
    No Information Rate : 0.8802
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.999
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Average_Hit Class: Block_Buster Class: Flop Class: Hit Class: Neutral Class: Super_Hit
Sensitivity                    1.000000            1.00000     0.99793   0.984848         1.0000         0.987342
Specificity                    1.000000            1.00000     1.00000   0.999926         0.9994         0.999926
Pos Pred Value                 1.000000            1.00000     1.00000   0.984848         0.9999         0.987342
Neg Pred Value                 1.000000            1.00000     0.99992   0.999926         1.0000         0.999926
Prevalence                     0.008481            0.06504     0.03555   0.004867         0.8802         0.005826
Detection Rate                 0.008481            0.06504     0.03547   0.004794         0.8802         0.005752
Detection Prevalence           0.008481            0.06504     0.03547   0.004867         0.8803         0.005826
Balanced Accuracy              1.000000            1.00000     0.99896   0.992387         0.9997         0.993634
Accuracy
99.97788
```

The Random Forest model gives almost 99% accuracy in predicting the result of the movies.

COLLABORATIVE FILTERING

It provides a unique filtering option that can be included in the media portals to recommend movies as per the user preferences on movie genre. It works by filtering the genre based on the popularity of the movie.

The user may select his genre preferences and recommendation works based on the popularity that gives top 10 movies to the user.

```
#### Getting input from the user to filter movies accordingly
```{r}
n = as.integer(readline(prompt = "Enter the number of genre you want to filter not greater than 5: "))
Genre = c()
for (i in 1:n) {
  Genre[i] = readline(prompt = "Enter Movie genre: ")
  print(Genre)
}
MoviesUmayLIKE(Genre) # Function to get the output
```
```

```
#### Collabrative filtering for Movie Genre
```{r}
MoviesUmayLIKE = function(x){
if(length(Genre) == 1 & any(Genre %in% colnames(colfil_df))){
  Recommended_Movies = colfil_df %>% filter(colfil_df[Genre] == 1) %>% arrange(-popularity) %>% head(10) %>%
                               select("title","popularity","budget","revenue")
  }else if(length(Genre) == 2 & any(Genre %in% colnames(colfil_df))){
    Recommended_Movies = colfil_df %>% filter(colfil_df[Genre][1] == 1 & colfil_df[Genre][2] == 1) %>%
      arrange(-popularity) %>% head(10) %>% select("title","popularity","budget","revenue")
    }else if(length(Genre) == 3 & any(Genre %in% colnames(colfil_df))){
      Recommended_Movies = colfil_df %>% filter(colfil_df[Genre][1] == 1 & colfil_df[Genre][2] == 1 &
                                                  colfil_df[Genre][3] == 1) %>% arrange(-popularity) %>%
        head(10) %>% select("title","popularity","budget","revenue")
      }else if(length(Genre) == 4 & any(Genre %in% colnames(colfil_df))){
        Recommended_Movies = colfil_df %>% filter(colfil_df[Genre][1] == 1 & colfil_df[Genre][2] == 1 &
                                                    colfil_df[Genre][3] == 1 & colfil_df[Genre][4] == 1) %>%
          arrange(-popularity) %>% head(10) %>% select("title","popularity","budget","revenue")
        }else if(length(Genre) == 5 & any(Genre %in% colnames(colfil_df))){
          Recommended_Movies = colfil_df %>% filter(colfil_df[Genre][1] == 1 & colfil_df[Genre][2] == 1 &
                                                      colfil_df[Genre][3] == 1 & colfil_df[Genre][4] == 1 &
                                                      colfil_df[Genre][5] == 1) %>% arrange(-popularity) %>%
            head(10) %>% select("title","popularity","budget","revenue")
          }else{
    print("Genre not found in our database")
          }
 return(Recommended_Movies)
}
```
```

OUTPUT

```
Enter the number of genre you want to filter not greater than 5: 2
> Genre = c()
> for (i in 1:n) {
+   Genre[i] = readline(prompt = "Enter Movie genre: ")
+   print(Genre)
+ }
Enter Movie genre: Action
[1] "Action"
Enter Movie genre: Comedy
[1] "Action" "Comedy"
> MoviesUmayLIKE(Genre) # Function to get the output
```

	title <chr>	popularity <dbl>	budget <int>	revenue <dbl>
1	Big Hero 6	213.84991	165000000	652105443
2	Deadpool	187.86049	58000000	783112979
3	Guardians of the Galaxy Vol. 2	185.33099	200000000	863416141
4	Pirates of the Caribbean: Dead Men Tell No Tales	133.82782	230000000	794191988
5	Captain Underpants: The First Epic Movie	88.56124	38000000	110824373
6	Logan	54.58200	97000000	616801808
7	Now You See Me 2	39.54065	90000000	334901337
8	Despicable Me 3	36.63152	80000000	1020063384
9	Baywatch	35.63769	69000000	177856751
10	Kingsman: The Secret Service	28.22421	81000000	414351546

USER BASED COLLABORATIVE FILTERING

```
#### User Based Collabrative Filtering
```{r}
# Creating the rating matrix with rows as uers and columns as movies
ratings_matrix = as.matrix(dcast(data = rating, userId~movieId, value.var = "rating"))

# We can now remove the user ids and convert ratings_matrix to real rating matrix
ranking_matrix = as(ratings_matrix[,-1], 'realRatingMatrix')

# Create Recommender Model.The parameters are UBCF and Cosine similarity. We take 10 nearest neighbours
ubcf = Recommender(ranking_matrix, method = 'UBCF', param = list(method = 'Cosine', nn = 10))

# Predicting the recommendation of 4th user
result = predict(ubcf, ranking_matrix[4, ])

# Top 5 movies recommended to 4th user
movies_sugg = as((bestN(result, n=5)),"list")[[1]]

# Output with movieid and title
movies %>% filter(id %in% movies_sugg) %>% select(id,title)

```
```

OUTPUT

id <int>	title <chr>
527	Once Were Warriors
2762	Young and Innocent
778	Monsieur Hulot's Holiday
2959	License to Wed
5995	Miffo

ITEM BASED COLLABORATIVE FILTERING

```
#### Item Based Collabrative Filtering
```{r}
# Creating the rating matrix with rows as movies and columns as users
ratings_matrix_1 = as.matrix(dcast(data = rating, movieId~userId, value.var = 'rating'))

# We can now remove the movie ids and convert ratings_matrix to real rating matrix
ranking_matrix_1 = as(ratings_matrix[,-1], 'realRatingMatrix')

# Create Recommender Model.The parameters are IBCF and Cosine similarity. We take 10 nearest neighbours
ibcf = Recommender(ranking_matrix_1, method = "IBCF", param = list(method = "Cosine", k = 10))

# Predicting the recommendation of 4th user
result_1 = predict(ibcf, ranking_matrix_1[4, ])

# Top 5 movies recommended to 4th user
movies_sugg_1 = as((bestN(result_1, n=5)), "list")[[1]]

# Output with movieid and title
movies %>% filter(movieId %in% movies_sugg_1) %>% select(movieId,title)

```
```

OUTPUT

movieId <int>	title <fctr>
40	Cry, the Beloved Country (1995)
52	Mighty Aphrodite (1995)
71	Fair Game (1995)
81	Things to Do in Denver When You're Dead (1995)
102	Mr. Wrong (1996)

EVALUATION OF UBCF AND IBCF MODELS

```
#### Evaluation of IBCF and UBCF model to find the best performing model
```{r}
# create evaluation scheme splitting taking 90% of the date for training and leaving 10% for validation or test
u1 <- evaluationScheme(ranking_matrix, method="split", train=0.9, given=-1)
u2 <- evaluationScheme(ranking_matrix, method="split", train=0.8, given=-1)
u3 <- evaluationScheme(ranking_matrix, method="split", train=0.7, given=-1)

i1 <- evaluationScheme(ranking_matrix_1, method="split", train=0.9, given=-1)
i2 <- evaluationScheme(ranking_matrix_1, method="split", train=0.8, given=-1)
i3 <- evaluationScheme(ranking_matrix_1, method="split", train=0.7, given=-1)

# creation of recommender model based on ubcf
Rec.ubcf_1 <- Recommender(getData(u1, "train"), "UBCF")
Rec.ubcf_2 <- Recommender(getData(u2, "train"), "UBCF")
Rec.ubcf_3 <- Recommender(getData(u3, "train"), "UBCF")

# creation of recommender model based on ibcf for comparison
Rec.ibcf_1 <- Recommender(getData(i1, "train"), "IBCF")
Rec.ibcf_2 <- Recommender(getData(i2, "train"), "IBCF")
Rec.ibcf_3 <- Recommender(getData(i3, "train"), "IBCF")

# making predictions on the test data set
p.ubcf_1 <- predict(Rec.ubcf_1, getData(u1, "known"), type="ratings")
p.ubcf_2 <- predict(Rec.ubcf_2, getData(u2, "known"), type="ratings")
p.ubcf_3 <- predict(Rec.ubcf_3, getData(u3, "known"), type="ratings")
```

```
# making predictions on the test data set
p.ibcf_1 <- predict(Rec.ibcf_1, getData(i1, "known"), type="ratings")
p.ibcf_2 <- predict(Rec.ibcf_2, getData(i2, "known"), type="ratings")
p.ibcf_3 <- predict(Rec.ibcf_3, getData(i3, "known"), type="ratings")

# obtaining the error metrics for both approaches and comparing them
error.ubcf_1<-calcPredictionAccuracy(p.ubcf_1, getData(u1, "unknown"))
error.ubcf_2<-calcPredictionAccuracy(p.ubcf_2, getData(u2, "unknown"))
error.ubcf_3<-calcPredictionAccuracy(p.ubcf_3, getData(u3, "unknown"))

error.ibcf_1<-calcPredictionAccuracy(p.ibcf_1, getData(i1, "unknown"))
error.ibcf_2<-calcPredictionAccuracy(p.ibcf_2, getData(i2, "unknown"))
error.ibcf_3<-calcPredictionAccuracy(p.ibcf_3, getData(i3, "unknown"))
error <- rbind(error.ubcf_1,error.ubcf_2,error.ubcf_3,error.ibcf_1,error.ibcf_2,error.ibcf_3)
rownames(error) <- c("model_1_UBCF","model_2_UBCF","model_3_UBCF","model_1_IBCF","model_2_IBCF","model_3_IBCF")
error
```
```

OUTPUT

	RMSE	MSE	MAE
model_1_UBCF	1.089115	1.186172	0.8241684
model_2_UBCF	1.080654	1.167812	0.8318207
model_3_UBCF	1.079447	1.165206	0.8488714
model_1_IBCF	1.725179	2.976241	1.4564592
model_2_IBCF	1.394449	1.944489	1.0666345
model_3_IBCF	1.648262	2.716768	1.2444659

From the above we can conclude that for UBCF model_2_UBCF is the best performing model and,

For IBCF model_2_IBCF is the best performing model. So these models can be used to make accurate recommendation to the users.

7. CONCLUSION

7.1 SUMMARY OF THE PROJECT

In this project I have created 2 predictive models and 3 recommendation engines based on different ideas and algorithims. They are as follows,

PREDICTIVE MODELS

1. **Revenue Prediction Model:** This model uses budget and the difference amount between budget and revenue as main variable to make accurate prediction of revenue of the movies. The model is built using a simple linear regression algorithim with 98% accuracy.
2. **Result Classification Model:** This model uses subset of column except the character columns and number of samples of data to classify the result of the movies as blockbuster, super hit, hit, average hit, neutral and flop respectively. This model is built using random forest algorithim with 99% accuracy.

RECOMMENDATION ENGINES

1. **Genre Based Recommendation System:** This system is uses a simple collaborative filtering technique by making recommendations based on the popularity of the movies. It gives an option of user to enter their preferred movie genre and the system recommends top 10 similar kind of movies to the user. The system is built using a user defined function MoviesUMayLike() in R.
2. **User Based Collaborative Filtering:** This engine is built mainly on movieId and userId in which users are listed in rows and movies are listed in columns and recommendation is made. However this engine has few disadvantages like cold start, scalability and sparsity but it has advantages as it is easy to build, context independent and is more accurate.
3. **Item Based Collaborative Filtering:** This engine is built mainly on movieId and userId in which movies are listed in rows and users are listed in columns and recommendation is made. However this engine is easy to implement and maintain, it works with little user feedback and is considerably more accurate.

7.2 FUTURE WORK

Even though the recommendation engines gives, good accuracy it has some disadvantages to overcome and that will be overcomed using a hybrid recommendation engine, which gives better accuracy in making recommendation. These models have been trained using small dataset and will be trained with large dataset to give a better accurate recommendation engines.

REFERENCES

1. M. Balabanović and Y. Shoham (1997). Fab: content-based, collaborative recommendation. Communications of the ACM, 40(3):66-72.

2. Prem Melville and Vikas Sindhwani (2010), Recommender Systems, Encyclopedia of Machine Learning.

3. Francesco Ricci and Lior Rokach and Bracha Shapira (2011), Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, pp. 1-35

4. R. J. Mooney & L. Roy (1999). Content-based book recommendation using learning for text categorization. In Workshop Recom. Sys: Algo. and Evaluation.

5. Rubens, Neil; Elahi, Mehdi; Sugiyama, Masashi; Kaplan, Dain (2016). "Active Learning in Recommender Systems". In Ricci, Francesco; Rokach, Lior; Shapira, Bracha. Recommender Systems Handbook (2 ed.). Springer US.

6. Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; Riedl, J. T. (January 2004). "Evaluating collaborative filtering recommender systems". ACM Trans. Inf. Syst. 22 (1): 5–53.

7. Lakiotaki, K.; Matsatsinis; Tsoukias, A. "Multicriteria User Modeling in Recommender Systems". IEEE Intelligent Systems. 26 (2): 64–76.

APPENDIX

S.NO	NAME	ABBREVIATION
1	UBCF	USER BASED COLLABORATIVE FILTERING
2	IBCF	ITEM BASED COLLABORATIVE FILTERING
3	LM	LINEAR MODEL
4	TM	TEXT MINING
4	RMSE	ROOT MEAN SQUARED ERROR
5	MSE	MEAN SQUARED ERROR
6	MAE	MEAN ABSOLUTE ERROR
7	CM	CONFUSION MATRIX