Movie Recommendation System

Navya Sai Reddy Gorre, Chetana alekhya reddy Gorre, Maheswar Reddy Peram

11/27/2022

Introduction

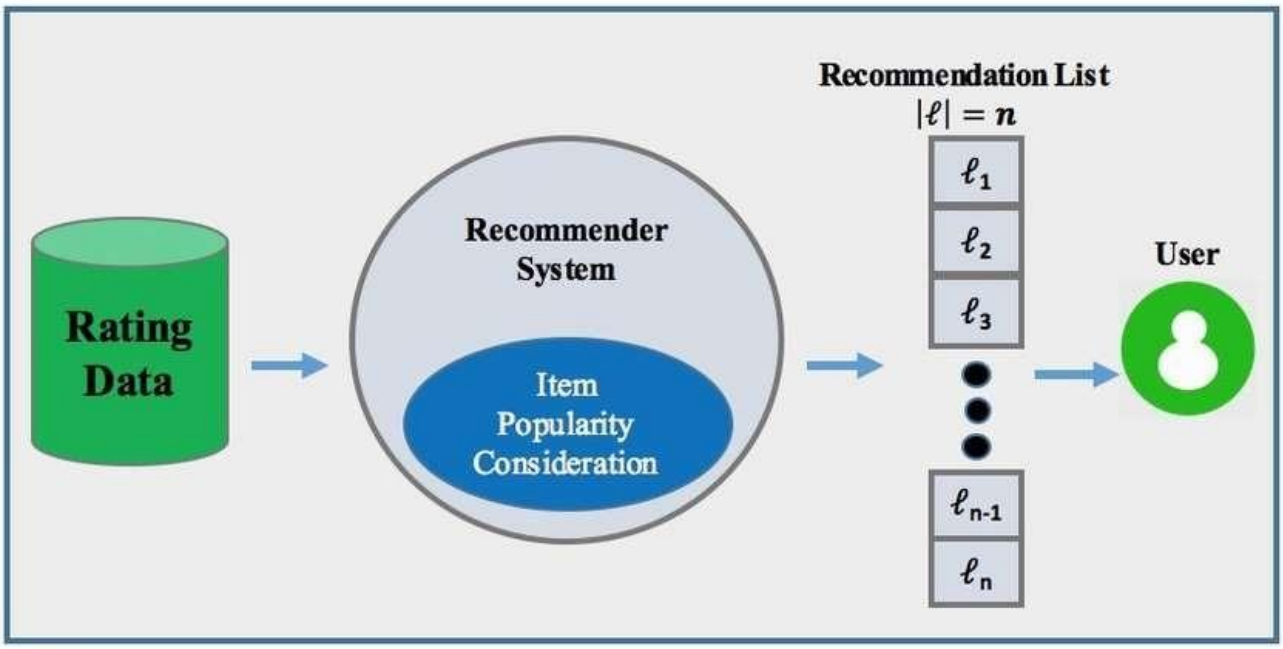
Earlier, when there are no movie recommendation systems, people use to rely on the suggestions and recommendations done by other people based on their personal interests. In order to overcome this, now a days, many OTT platforms such as Netflix, Amazon Prime Video, Hulu and so on are implementing a special feature, recommendation system, in their application on considering the one's watch history and most liked movies. It is possible for business organizations to exploit the usage of these Recommendation Systems by suggesting items that are lovable to the users. Ever wondered how these applications suggest or recommend the movies which are appealing to us using such recommendation systems? This system is used to learn the watching patterns of the user and recommends the movies accordingly. We all come across these systems once in a while. These systems are not only used in entertainment applications but they are also used in many E-Commerce websites such as Amazon, Ebay and Best Buy.

We have come up with the similar idea of designing our own Movie recommendation system which helps the user by recommending the top movies based on his/her preferences and ratings history.

There are three algorithms which helps us to build Recommender Systems 1. Popularity based Filtering 2. Content based Filtering 3. User based Collaborative Filtering

Popularity based Filtering:

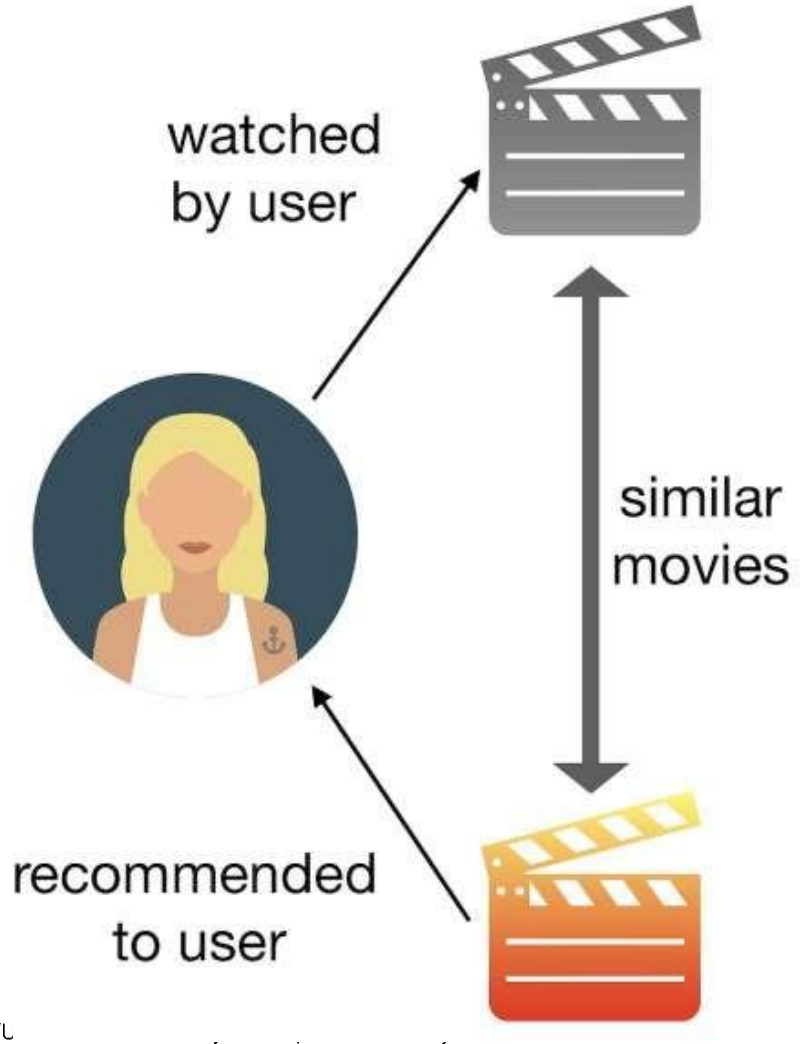
This is a simple Recommender System which recommends the movies to all users in a generalized way based on popularity. The basic idea behind this is that the user likes the movies which are highly popular. So, we cannot expect user preference based recommendations in this model.



popularity based filtering

Content based Filtering:

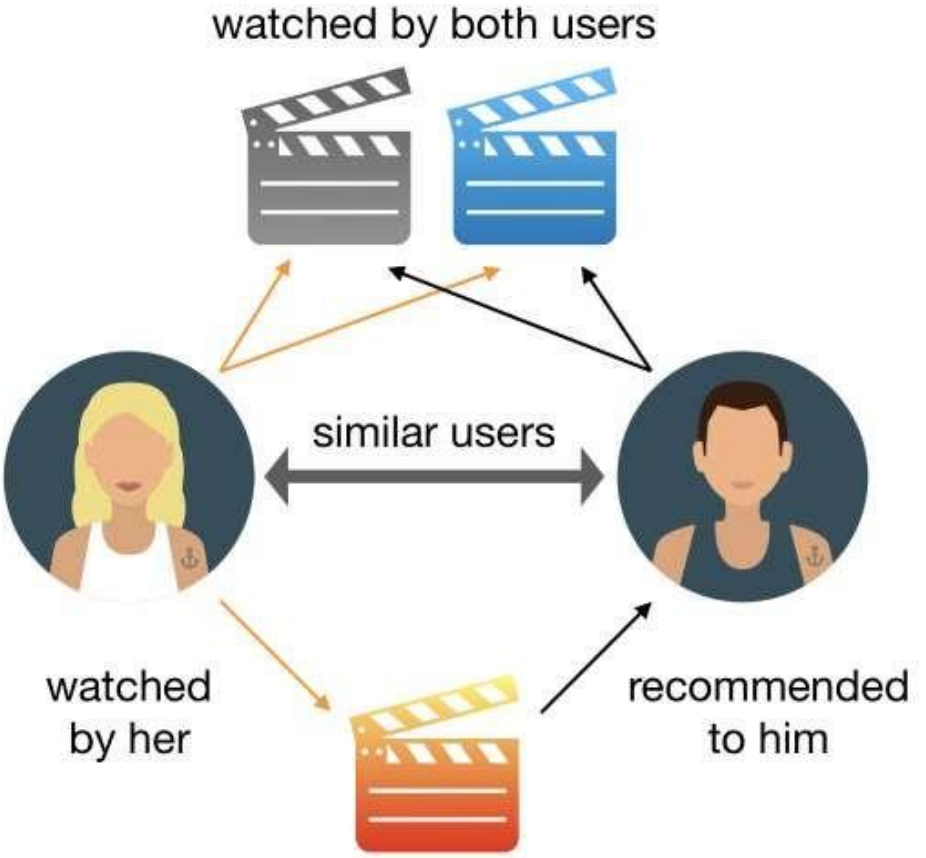
This is a Recommender System which recommends the movies by his/her preference and their data. This system does not take any data from other users. The basic idea behind this model is if the user likes one movie he may like the movies which are similar to it. Attributes such as genre, director, description and actors will be used by this model.



Content based filtering

Collaborative based Filtering

This is a Recommender System which recommends the movies by his/her preference and it also considers data of other similar users. The watch history of every user will be used in this algorithm. The basic idea behind this model is that movies watched by one user are recommended to similar users. On using Collaborative based filtering, the outcome of the recommendation is obtained from multiple users' data and it doesn't rely on single user data.



collaborative filtering

The main motivation behind doing this project is the Netflix application. It suggests movies according to my interests. So, it raised a bean of curiosity to learn more about these recommendation systems and how it works.

We built a recommender system using the three algorithms mentioned above and also observed how best these algorithms worked.

Related Work

We went through various academic papers to complete this work. We got to know how to build a data mining pipeline.

There is an exponential growth in data nowadays. So, it is so important to extract usasble data. Now, data mining pipeline comes into picture[1].

We learnt business purposes of recommendation systems.

We learnt how the movie recommendation systems are being implemented. We learnt the details of evaluation of model from this paper[5].

These references helped us to finish our work efficiently without any hardships. There is possibility of building Hybrid Recommendation models[2]. We came across these Hybrid Recommendation Systems in our research. We are going to save this Hybrid model as the future progress of our work.

Methods

For this Recommender Systems to work we need a dataset of movies with ratings and popularity. And we also need metadata of the movie to build a Content based Filtering model.

There's a dataset called "The Movies Dataset" on the Kaggle website which consists of data which is suitable for our project. The metadata of 45000 movies are available in this data. It is taken from MovieLens Dataset. These files contain movies released before 2017. Attributes such as cast, crew, keywords, overview, budget, revenue, release dates, languages, countries, vote counts and vote averages are contained in this dataset. 26 million ratings of all the movies are also present in this file. These ratings are obtained from a website called GroupLens.

```
# loading required packages

#install.packages("tidyverse")

library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.3.6      ✓ purrr   0.3.4
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.4.1
## ✓ readr   2.1.2      ✓ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
##
## The following object is masked from 'package:purrr':
##
##     flatten
```

```
library(dplyr)
library(purrr)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
# importing meta data of movies
movies_md = read.csv("B:/my-work/kdd/final/Movie_Recommender/movies_metadata.csv")

summary(movies_md)
```

```
##     adult           belongs_to_collection    budget            genres
##  Length:45466       Length:45466          Length:45466       Length:45466
##  Class :character   Class :character      Class :character   Class :character
##  Mode  :character   Mode  :character      Mode  :character   Mode  :character
##
##
##
##
##    homepage              id              imdb_id          original_language
##  Length:45466       Length:45466       Length:45466       Length:45466
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  original_title       overview          popularity        poster_path
##  Length:45466       Length:45466       Length:45466       Length:45466
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  production_companies production_countries release_date
##  Length:45466         Length:45466         Length:45466
##  Class :character     Class :character     Class :character
##  Mode  :character     Mode  :character     Mode  :character
##
##
##
##
##     revenue             runtime        spoken_languages      status
##  Min.   :0.000e+00   Min.   :   0.00   Length:45466       Length:45466
##  1st Qu.:0.000e+00   1st Qu.:  85.00   Class :character   Class :character
##  Median :0.000e+00   Median :  95.00   Mode  :character   Mode  :character
##  Mean   :1.121e+07   Mean   :  94.13
##  3rd Qu.:0.000e+00   3rd Qu.: 107.00
##  Max.   :2.788e+09   Max.   :1256.00
##  NA's   :6           NA's   :263
##    tagline             title              video            vote_average
##  Length:45466       Length:45466       Length:45466       Min.   : 0.000
##  Class :character   Class :character   Class :character   1st Qu.: 5.000
##  Mode  :character   Mode  :character   Mode  :character   Median : 6.000
##                                                           Mean   : 5.618
##                                                           3rd Qu.: 6.800
##                                                           Max.   :10.000
##                                                           NA's   :6
##    vote_count

##  Min.   :    0.0
##  1st Qu.:    3.0
##  Median :   10.0
##  Mean   :  109.9
##  3rd Qu.:   34.0
##  Max.   :14075.0
##  NA's   :6
```

Data Preparation

We collected the data from the Kaggle website and MovieLens website to build Recommender Systems using three different approaches.

Data Cleaning:

We removed the rows which contain Null values. We removed all the duplicate values present in the data. We changed data-types of particular columns to do the computation according to our requirement.

Data Filtering:

We removed the unwanted columns in our dataset. This will also reduce the burden to the system. We added a few extra columns from existing columns which helped us to do Analysis.

Outliers Removal:

We removed only the one outlier present in our data.

```
# removing rows that contains
movies_md[movies_md=="[]"] = NA
movies_md = na.omit(movies_md)

dim(movies_md)
```

```
## [1] 32024    24
```

```
# removing duplicate rows

names(movies_md)[6] =  "m_id"
movies_md = distinct(movies_md)

dim(movies_md)
```

```
## [1] 32013    24
```

```
summary(movies_md)
```

```
##     adult           belongs_to_collection    budget             genres
##  Length:32013       Length:32013          Length:32013       Length:32013
##  Class :character   Class :character      Class :character   Class :character
##  Mode  :character   Mode  :character      Mode  :character   Mode  :character
##
##
##
##    homepage             m_id             imdb_id          original_language
##  Length:32013       Length:32013       Length:32013       Length:32013
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  original_title       overview          popularity        poster_path
##  Length:32013       Length:32013       Length:32013       Length:32013
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  production_companies production_countries release_date
##  Length:32013         Length:32013         Length:32013
##  Class :character     Class :character     Class :character
##  Mode  :character     Mode  :character     Mode  :character
##
##
##
##     revenue             runtime       spoken_languages      status
##  Min.   :0.000e+00   Min.   :  0.00   Length:32013       Length:32013
##  1st Qu.:0.000e+00   1st Qu.: 88.00   Class :character   Class :character
##  Median :0.000e+00   Median : 96.00   Mode  :character   Mode  :character
##  Mean   :1.585e+07   Mean   : 98.06
##  3rd Qu.:0.000e+00   3rd Qu.:109.00
##  Max.   :2.788e+09   Max.   :931.00
##    tagline             title              video            vote_average
##  Length:32013       Length:32013       Length:32013       Min.   : 0.000
##  Class :character   Class :character   Class :character   1st Qu.: 5.300
##  Mode  :character   Mode  :character   Mode  :character   Median : 6.100
##                                                           Mean   : 5.846
##                                                           3rd Qu.: 6.800
##                                                           Max.   :10.000
##    vote_count
##  Min.   :    0.0
##  1st Qu.:    5.0
##  Median :   16.0
##  Mean   :  152.6
##  3rd Qu.:   60.0
##  Max.   :14075.0
```

```
# changing data types for particular columns

movies_md$release_date = as.Date(movies_md$release_date)
movies_md$revenue = as.numeric(movies_md$revenue)
movies_md$budget = as.numeric(movies_md$budget)

summary(movies_md)
```

```
##    adult           belongs_to_collection     budget
##  Length:32013       Length:32013          Min.   :        0
##  Class :character   Class :character      1st Qu.:        0
##  Mode  :character   Mode  :character      Median :        0
##                                          Mean   :  5914718
##                                          3rd Qu.:       12
##                                          Max.   :380000000
##
##    genres            homepage             m_id             imdb_id
##  Length:32013       Length:32013       Length:32013       Length:32013
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  original_language  original_title       overview          popularity
##  Length:32013       Length:32013       Length:32013       Length:32013
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  poster_path        production_companies production_countries
##  Length:32013       Length:32013         Length:32013
##  Class :character   Class :character     Class :character
##  Mode  :character   Mode  :character     Mode  :character
##
##
##
##
##   release_date           revenue             runtime       spoken_languages
##  Min.   :1878-06-14   Min.   :0.000e+00   Min.   :  0.00   Length:32013
##  1st Qu.:1974-03-26   1st Qu.:0.000e+00   1st Qu.: 88.00   Class :character
##  Median :1999-09-11   Median :0.000e+00   Median : 96.00   Mode  :character
##  Mean   :1990-09-14   Mean   :1.585e+07   Mean   : 98.06
##  3rd Qu.:2010-11-05   3rd Qu.:0.000e+00   3rd Qu.:109.00
##  Max.   :2020-12-16   Max.   :2.788e+09   Max.   :931.00
##  NA's   :4
##    status            tagline             title              video
##  Length:32013       Length:32013       Length:32013       Length:32013
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   vote_average      vote_count
##  Min.   : 0.000   Min.   :    0.0
##  1st Qu.: 5.300   1st Qu.:    5.0
##  Median : 6.100   Median :   16.0
##  Mean   : 5.846   Mean   :  152.6
##  3rd Qu.: 6.800   3rd Qu.:   60.0
##  Max.   :10.000   Max.   :14075.0
##
```

```
#adding a new column year for detailed analysis
movies_md$year = format(movies_md$release_date,format="%Y")
movies_md$year = as.integer(movies_md$year)

glimpse(movies_md)
```

```
## Rows: 32,013
## Columns: 25
## $ adult                 <chr> "False", "False", "False", "False", "False", "Fa…
## $ belongs_to_collection <chr> "{'id': 10194, 'name': 'Toy Story Collection', '…
## $ budget                <dbl> 30000000, 65000000, 0, 16000000, 0, 60000000, 58…
## $ genres                <chr> "[{'id': 16, 'name': 'Animation'}, {'id': 35, 'n…
## $ homepage              <chr> "http://toystory.disney.com/toy-story", "", "", …
## $ m_id                  <chr> "862", "8844", "15602", "31357", "11862", "949",…
## $ imdb_id               <chr> "tt0114709", "tt0113497", "tt0113228", "tt011488…
## $ original_language     <chr> "en", "en", "en", "en", "en", "en", "en", "en", …
## $ original_title        <chr> "Toy Story", "Jumanji", "Grumpier Old Men", "Wai…
## $ overview              <chr> "Led by Woody, Andy's toys live happily in his r…
## $ popularity            <chr> "21.946943", "17.015539", "11.7129", "3.859495",…
## $ poster_path           <chr> "/rhIRbceoE9lR4veEXuwCC2wARtG.jpg", "/vzmL6fP7aP…
## $ production_companies  <chr> "[{'name': 'Pixar Animation Studios', 'id': 3}]"…
## $ production_countries  <chr> "[{'iso_3166_1': 'US', 'name': 'United States of…
## $ release_date          <date> 1995-10-30, 1995-12-15, 1995-12-22, 1995-12-22,…
## $ revenue               <dbl> 373554033, 262797249, 0, 81452156, 76578911, 187…
## $ runtime               <dbl> 81, 104, 101, 127, 106, 170, 127, 97, 106, 130, …
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'English'}]", "[{'…
## $ status                <chr> "Released", "Released", "Released", "Released", …
## $ tagline               <chr> "", "Roll the dice and unleash the excitement!",…
## $ title                 <chr> "Toy Story", "Jumanji", "Grumpier Old Men", "Wai…
## $ video                 <chr> "False", "False", "False", "False", "False", "Fa…
## $ vote_average          <dbl> 7.7, 6.9, 6.5, 6.1, 5.7, 7.7, 6.2, 5.4, 5.5, 6.6…
## $ vote_count            <int> 5415, 2413, 92, 34, 173, 1886, 141, 45, 174, 119…
## $ year                  <int> 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995, …
```

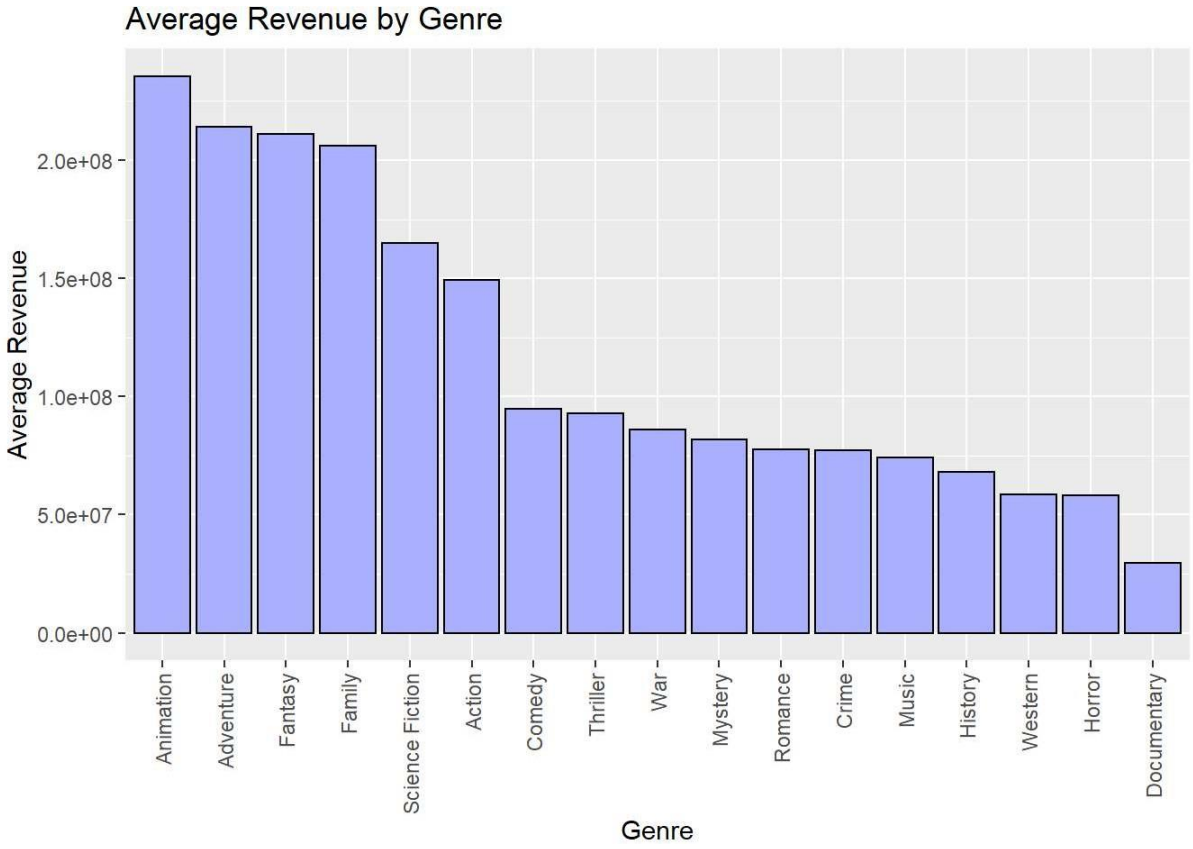
```
# Filtering data by removing unwanted columns
movies_filtered = filter(movies_md, revenue > 1000000, budget > 1000, runtime > 0) %>%
  # Removing unwanted columns
  select(-"adult", -"belongs_to_collection", -"homepage", -"overview", -"poster_path", -"s
tatus", -"video", -"tagline", -"production_companies", -"production_countries", -"spoken_l
anguages") %>%
  # Adding columns for net return and Return on investment (ROI) to see which movies colle
cted most at box-office
  mutate(net_return = revenue - budget, ROI = net_return / budget * 100)

movies_filtered = movies_filtered[ !(movies_filtered$m_id %in% c(363093)), ]

glimpse(movies_filtered)
```

```
## Rows: 4,714
## Columns: 16
## $ budget            <dbl> 30000000, 65000000, 16000000, 60000000, 35000000, 58…
## $ genres            <chr> "[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name'…
## $ m_id              <chr> "862", "8844", "31357", "949", "9091", "710", "9087"…
## $ imdb_id           <chr> "tt0114709", "tt0113497", "tt0114885", "tt0113277", …
## $ original_language <chr> "en", "en", "en", "en", "en", "en", "en", "en", "en"…
## $ original_title    <chr> "Toy Story", "Jumanji", "Waiting to Exhale", "Heat",…
## $ popularity        <chr> "21.946943", "17.015539", "3.859495", "17.924927", "…
## $ release_date      <date> 1995-10-30, 1995-12-15, 1995-12-22, 1995-12-15, 199…
## $ revenue           <dbl> 373554033, 262797249, 81452156, 187436818, 64350171,…
## $ runtime           <dbl> 81, 104, 127, 170, 106, 130, 106, 192, 119, 178, 136…
## $ title             <chr> "Toy Story", "Jumanji", "Waiting to Exhale", "Heat",…
## $ vote_average      <dbl> 7.7, 6.9, 6.1, 7.7, 5.5, 6.6, 6.5, 7.1, 5.7, 7.8, 7.…
## $ vote_count        <int> 5415, 2413, 34, 1886, 174, 1194, 199, 72, 137, 1343,…
## $ year              <int> 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995…
## $ net_return        <dbl> 343554033, 197797249, 65452156, 127436818, 29350171,…
## $ ROI               <dbl> 1145.18011, 304.30346, 409.07598, 212.39470, 83.8576…
```

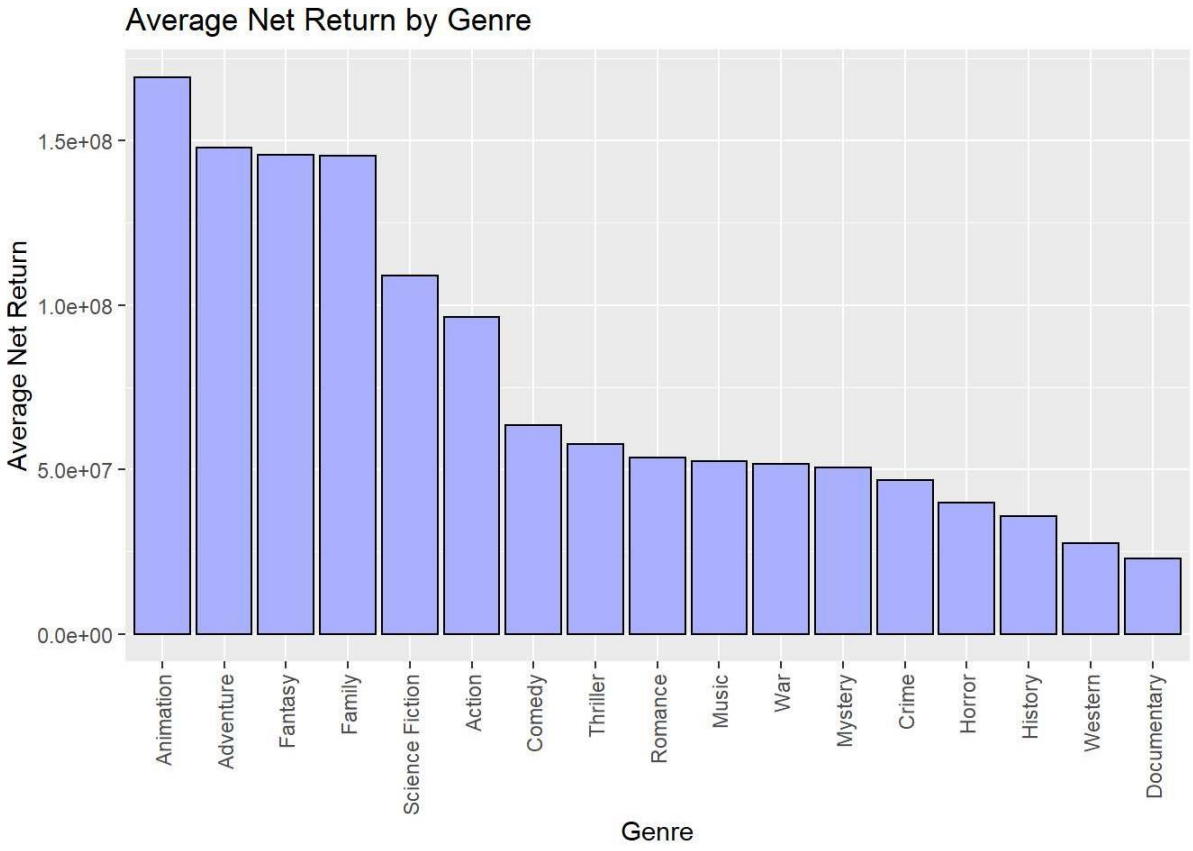
Analysis of the Data

```
# Genre-wise distribution of the data

horror          = filter(movies_filtered, grepl('Horror'         , genres))
mystery         = filter(movies_filtered, grepl('Mystery'        , genres))
action          = filter(movies_filtered, grepl('Action'         , genres))
adventure       = filter(movies_filtered, grepl('Adventure'      , genres))
fantasy         = filter(movies_filtered, grepl('Fantasy'        , genres))
comedy          = filter(movies_filtered, grepl('Comedy'         , genres))
thriller        = filter(movies_filtered, grepl('Thriller'       , genres))
documentary     = filter(movies_filtered, grepl('Documentary'    , genres))
animation       = filter(movies_filtered, grepl('Animation'      , genres))
romance         = filter(movies_filtered, grepl('Romance'        , genres))
family          = filter(movies_filtered, grepl('Family'         , genres))
western         = filter(movies_filtered, grepl('Western'        , genres))
music           = filter(movies_filtered, grepl('Music'          , genres))
science_fiction = filter(movies_filtered, grepl('Science Fiction', genres))
crime           = filter(movies_filtered, grepl('Crime'          , genres))
history         = filter(movies_filtered, grepl('History'        , genres))
war             = filter(movies_filtered, grepl('War'            , genres))

# collections-wise distribution of every genre

genre_collections = data.frame(name = c("Horror", "Mystery", "Action", "Adventure", "Fanta
sy", "Comedy", "Thriller", "Documentary", "Animation",
"Romance", "Family", "Western", "Music", "Science Fiction", "Crime", "History", "War"),
                               average_revenue    = c(mean(horror$revenue), mean(mystery$r
evenue), mean(action$revenue),
                                                      mean(adventure$revenue), mean(fantas
y$revenue), mean(comedy$revenue),
                                                      mean(thriller$revenue), mean(documen
tary$revenue), mean(animation$revenue),
                                                      mean(romance$revenue), mean(family$r
evenue), mean(western$revenue),
                                                      mean(music$revenue), mean(science_fi
ction$revenue), mean(crime$revenue),
                                                      mean(history$revenue), mean(war$reve
nue)),
                               average_net_return = c(mean(horror$net_return), mean(myster
y$net_return), mean(action$net_return),
                                                      mean(adventure$net_return), mean(fan
tasy$net_return), mean(comedy$net_return),
                                                      mean(thriller$net_return), mean(docu
mentary$net_return), mean(animation$net_return),
                                                      mean(romance$net_return), mean(famil
y$net_return), mean(western$net_return),
                                                      mean(music$net_return), mean(science
_fiction$net_return), mean(crime$net_return),
                                                      mean(history$net_return), mean(war$n
et_return)),
                               average_ROI        = c(mean(horror$ROI), mean(mystery$ROI),
mean(action$ROI),
                                                      mean(adventure$ROI), mean(fantasy$RO
I), mean(comedy$ROI),
                                                      mean(thriller$ROI), mean(documentary
$ROI), mean(animation$ROI),
                                                      mean(romance$ROI), mean(family$ROI),
mean(western$ROI),
                                                      mean(music$ROI), mean(science_fictio
```

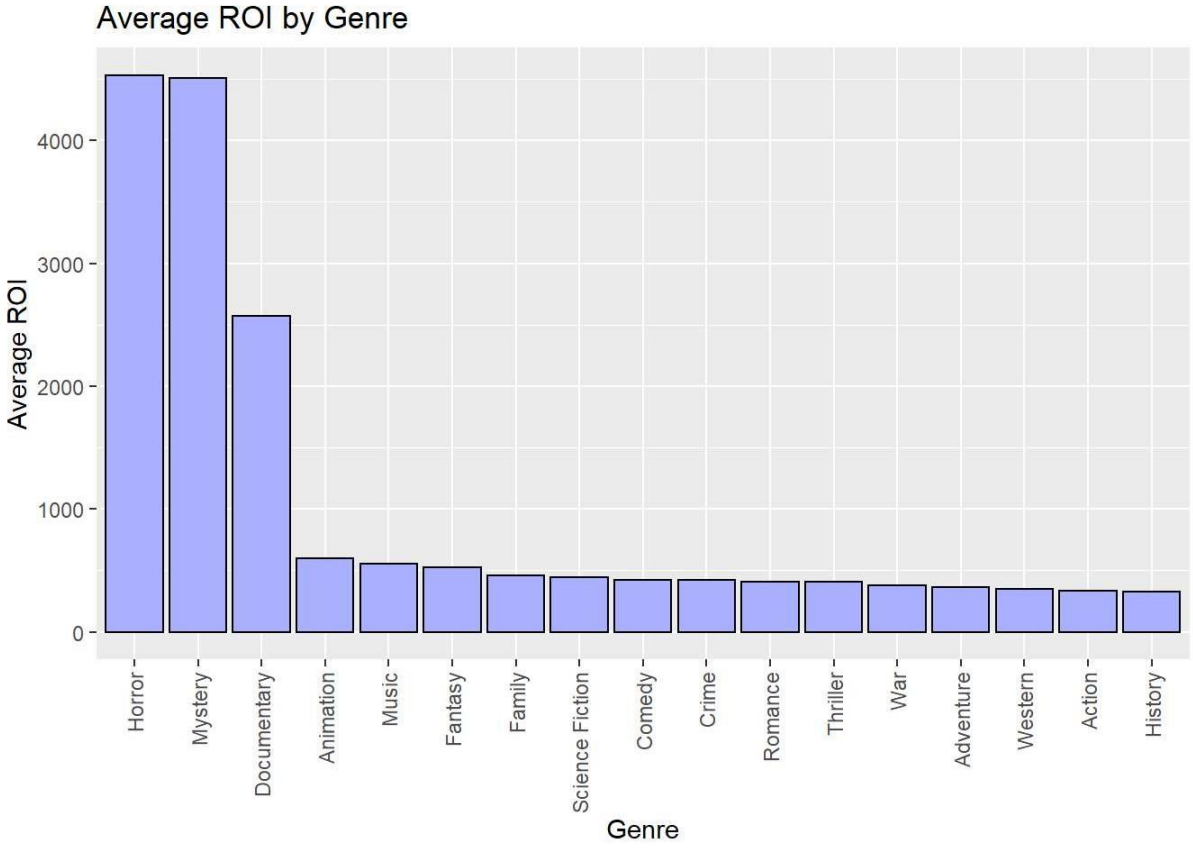
```
n$ROI), mean(crime$ROI),
                                                    mean(history$ROI), mean(war$ROI)),
                                 average_budget     = c(mean(horror$budget), mean(mystery$bu
dget), mean(action$budget),
                                                    mean(adventure$budget), mean(fantasy
$budget), mean(comedy$budget),
                                                    mean(thriller$budget), mean(document
ary$budget), mean(animation$budget),
                                                    mean(romance$budget), mean(family$bu
dget), mean(western$budget),
                                                    mean(music$budget), mean(science_fic
tion$budget), mean(crime$budget),
                                                    mean(history$budget), mean(war$budge
t)))

# Average Revenue by Genre
ggplot(data = genre_collections, mapping = aes(x = reorder(name, -average_revenue), y = av
erage_revenue)) + geom_bar(color = "black", fill = "#AAAFFF", stat='identity') + labs(titl
e = "Average Revenue by Genre", y = "Average Revenue", x = "Genre") + theme(axis.text.x =
 element_text(angle = 90, vjust = 0.5, hjust=1))
```




```
# Average Net Return by Genre
ggplot(data = genre_collections, mapping = aes (x = reorder(name, -average_net_return), y
 = average_net_return)) + geom_bar(color = "black", fill = "#AAAFFF", stat='identity') + l
abs(title = "Average Net Return by Genre", y = "Average Net Return", x = "Genre") + theme
(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

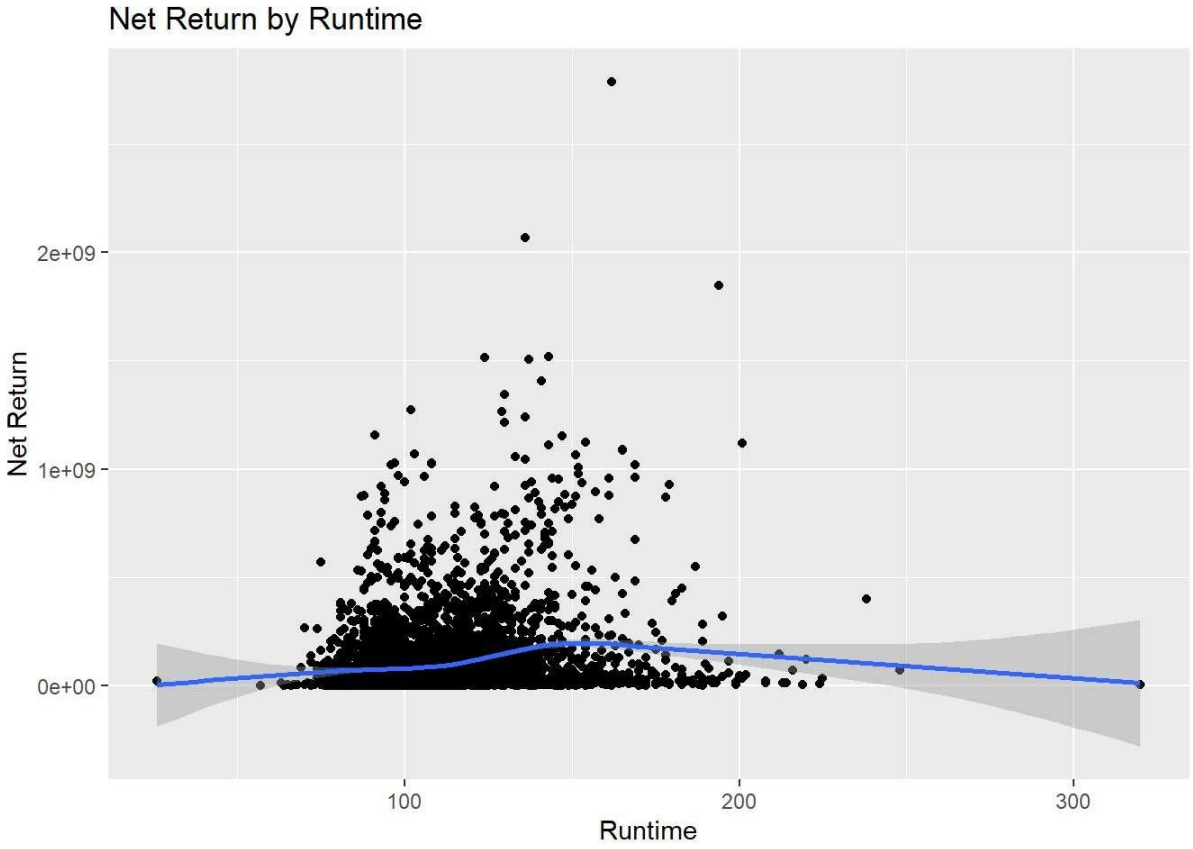



```
# Average ROI by Genre
ggplot(data = genre_collections, mapping = aes (x = reorder(name, -average_ROI), y = avera
ge_ROI)) + geom_bar(color = "black", fill = "#AAAFFF", stat='identity') + labs(title = "Av
erage ROI by Genre", y = "Average ROI", x = "Genre") + theme(axis.text.x = element_text(an
gle = 90, vjust = 0.5, hjust=1))
```



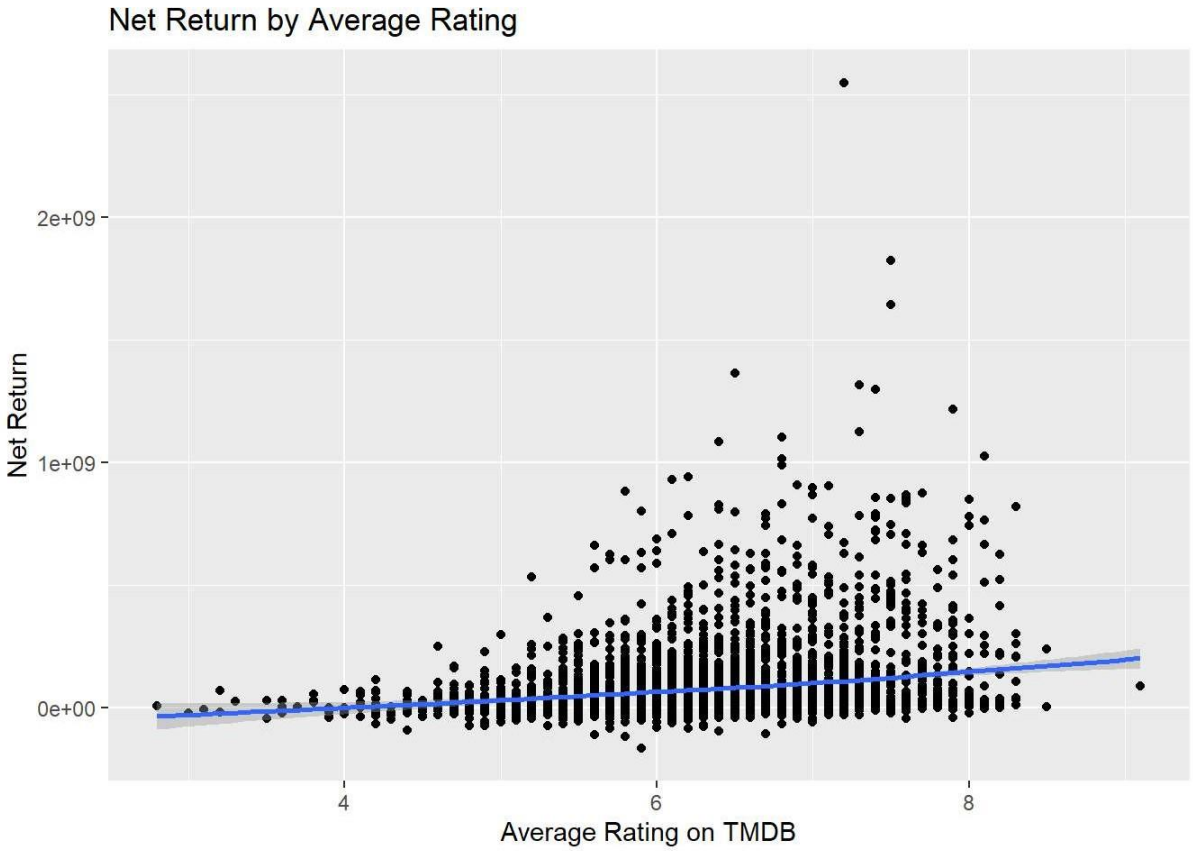
```
# Runtime vs net-return

ggplot(data = movies_filtered, mapping = aes (x = runtime, y = revenue)) + geom_point( sta
t='identity') + geom_smooth() + labs(title = "Net Return by Runtime", y = "Net Return", x
 = "Runtime")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
# tmdb ratings vs net-return
ggplot(data = filter(movies_filtered, vote_count > 50), mapping = aes(x = vote_average, y
 = net_return)) + geom_point(stat='identity') + geom_smooth() + labs(title = "Net Return b
y Average Rating", y = "Net Return", x = "Average Rating on TMDB")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Popularity based Model

This is a simple model we created using basic idea that is, users are more likely to watch the movies which collected more revenue. These are the results that we obtained. We can see those movies are highly appraised movies. But there is lack of user preference in this model. But we can use this model to attract large mass of population rather than a single user.

```
top_10 = movies_filtered %>% arrange(desc(revenue)) %>% slice(1:10)
top_10_titles = data.frame(top_10$original_title)
print(top_10_titles)
```

```
##                          top_10.original_title
## 1                                       Avatar
## 2                 Star Wars: The Force Awakens
## 3                                      Titanic
## 4                                The The Avengers
## 5                              Jurassic World
## 6                                   Furious 7
## 7                       Avengers: Age of Ultron
## 8  Harry Potter and the Deathly Hallows: Part 2
## 9                                       Frozen
## 10                        Beauty and the Beast
```

Data Preparation for Content-based approach

```
movies_df = read.csv("B:/my-work/kdd/final/Movie_Recommender/movies.csv")
head(movies_df)
```

```
##   movieId                               title
## 1       1                   Toy Story (1995)
## 2       2                     Jumanji (1995)
## 3       3            Grumpier Old Men (1995)
## 4       4           Waiting to Exhale (1995)
## 5       5 Father of the Bride Part II (1995)
## 6       6                         Heat (1995)
##                                       genres
## 1 Adventure|Animation|Children|Comedy|Fantasy
## 2                  Adventure|Children|Fantasy
## 3                              Comedy|Romance
## 4                        Comedy|Drama|Romance
## 5                                      Comedy
## 6                       Action|Crime|Thriller
```

```
dim(movies_df)
```

```
## [1] 62423     3
```

```
ratings_df = read.csv("B:/my-work/kdd/final/Movie_Recommender/ratings_small.csv")
ratings_df = ratings_df[1:3]
head(ratings_df)
```

```
##   userId movieId rating
## 1      1      31    2.5
## 2      1    1029    3.0
## 3      1    1061    3.0
## 4      1    1129    2.0
## 5      1    1172    4.0
## 6      1    1263    2.0
```

```
dim(ratings_df)
```

```
## [1] 100004      3
```

```
# Creating Binary class data frame from ratings to see whether user likes movie or not
# rating > 3 is equal to 1 (1 means like)
# rating <= 3 is equal to -1 (-1 means dislike)
binaryratings = ratings_df
for (i in 1:nrow(binaryratings)){
  if (binaryratings[i,3] > 3){
    binaryratings[i,3] = 1
  }
  else{
    binaryratings[i,3] = -1
  }
}

binaryratings2 = dcast(binaryratings, movieId~userId, value.var = "rating", na.rm=FALSE)
```

```
## Warning in dcast(binaryratings, movieId ~ userId, value.var = "rating", : The
## dcast generic in data.table has been passed a data.frame and will attempt to
## redirect to the reshape2::dcast; please note that reshape2 is deprecated, and
## this redirection is now deprecated as well. Please do this redirection yourself
## like reshape2::dcast(binaryratings). In the next version, this warning will
## become an error.
```

```
for (i in 1:ncol(binaryratings2)){
  binaryratings2[which(is.na(binaryratings2[,i]) == TRUE),i] = 0
}
binaryratings2 = binaryratings2[,-1]
```

```
movieIds = (unique(movies_df$movieId)) #62483
ratingmovieIds = (unique(ratings_df$movieId)) #9066
movies2_df = movies_df[-which((movieIds %in% ratingmovieIds) == FALSE),]
rownames(movies2_df) = NULL

genre_matrix3 = genre_matrix2[-which((movieIds %in% ratingmovieIds) == FALSE),]
rownames(genre_matrix3) = NULL

result = matrix(0,18,671)
for (c in 1:ncol(binaryratings2)){
  for (i in 1:ncol(genre_matrix3)){
    result[i,c] = sum((genre_matrix3[,i]) * (binaryratings2[,c]))
  }
}

for(j in 1:ncol(result)){
for (i in 1:nrow(result)){
  if (result[i,j] < 0){
    result[i,j] = 0
  }
  else {
    result[i,j] = 1
  }
}
}
```

Content Based Filtering approach

```
# recommendations for user1

result2 = result[,1]#First user's profile
result2
```

```
##  [1] 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0
```

```
sim_mat = rbind.data.frame(result2, genre_matrix3)
sim_mat = data.frame(lapply(sim_mat,function(x){as.integer(x)}))
sim_mat[1,] #user liked genres
```

```
##   Action Adventure Animation Children Comedy Crime Documentary Drama Fantasy
## 1      0         0         1        0      0     1           0     0       0
##   Film.Noir Horror Musical Mystery Romance Sci.Fi Thriller War Western
## 1         1      0       0       1       0      0        1   0       0
```

```
library(proxy)
```

```
## Warning: package 'proxy' was built under R version 4.2.2
```

```
##
## Attaching package: 'proxy'
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
sim_results = dist(sim_mat, method = "Cosine")
sim_results = as.data.frame(as.matrix(sim_results[1:8863]))
rows = which(sim_results == min(sim_results))
#Recommended movies
movies_df[rows,]
```

```
##      movieId                       title                              genres
## 144      145           Bad Boys (1995) Action|Comedy|Crime|Drama|Thriller
## 1004    1026 So Dear to My Heart (1949)                     Children|Drama
## 1286    1319    Kids of Survival (1996)                        Documentary
## 2910    3001      Suburbans, The (1999)                              Drama
## 3231    3323      Chain of Fools (2000)                       Comedy|Crime
```

```
# recommendations for user2

result2 = result[,2]#Second user's profile
result2
```

```
##  [1] 0 1 0 0 0 1 1 1 0 0 1 0 1 1 0 0 1 0
```

```
sim_mat = rbind.data.frame(result2, genre_matrix3)
sim_mat = data.frame(lapply(sim_mat,function(x){as.integer(x)}))
sim_mat[1,] #user liked genres
```

```
##   Action Adventure Animation Children Comedy Crime Documentary Drama Fantasy
## 1      0         1         0        0      0     1           1     1       0
##   Film.Noir Horror Musical Mystery Romance Sci.Fi Thriller War Western
## 1         0      1       0       1       1      0        0   1       0
```

```
library(proxy)
sim_results = dist(sim_mat, method = "Cosine")
sim_results = as.data.frame(as.matrix(sim_results[1:8863]))
rows = which(sim_results == min(sim_results))
#Recommended movies

movies_df[rows,]
```

```
##      movieId                              title        genres
## 6918    7042 Betty Blue (37°2 le matin) (1986) Drama|Romance
```

Data Preparation for User-Based Collaborative filtering approach

We used recommenderlab package to build this model.

```
# Data preparation
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
ratingmat = dcast(ratings_df, userId~movieId, value.var = "rating", na.rm=FALSE)
ratingmat = as.matrix(ratingmat[,-1])

#install.packages("recommenderlab", repos = "https://mhahsler.r-universe.dev")
#install.packages("matrixStats", type = "binary")
#install.packages('BBmisc')

library(recommenderlab)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loading required package: arules
```

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
## Loading required package: registry
```

```
## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy
```

```
library(BBmisc)
```

```
##
## Attaching package: 'BBmisc'
```

```
## The following object is masked from 'package:recommenderlab':
##
##     normalize
```

```
## The following objects are masked from 'package:dplyr':
##
##     coalesce, collapse
```

```
## The following object is masked from 'package:base':
##
##     isFALSE
```

```
ratingmat2 = as(ratingmat, "realRatingMatrix")

#binarize the data
ratingmat_bin = binarize(ratingmat2,minRating=3)
```

User-Based Collaborative Filtering

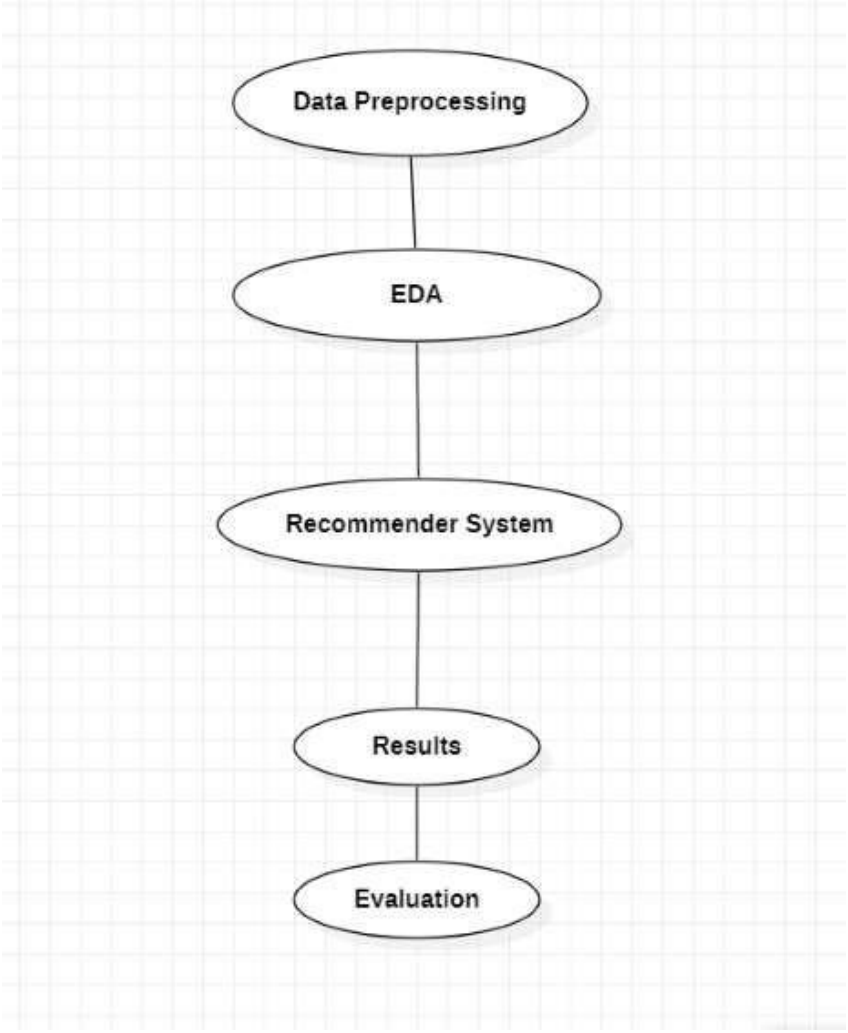
```
#Creating Recommender Model. "UBCF" stands for User-Based Collaborative Filtering

recommender_model = Recommender(ratingmat_bin, method = "UBCF", param=list(method="Cosine"
,nn=30))
recom = predict(recommender_model, ratingmat2[1], n=10) #Obtain top 10 recommendations for
1st user in dataset
recom_list = as(recom, "list")

#Obtain recommendations
recom_result = matrix(0,10)
for (i in c(1:10)){
  recom_result[i] = movies_df[as.integer(recom_list[[1]][i]),2]
}

print(recom_result)
```

```
##       [,1]
##  [1,] "Heidi Fleiss: Hollywood Madam (1995)"
##  [2,] "Love & Human Remains (1993)"
##  [3,] "My Crazy Life (Mi vida loca) (1993)"
##  [4,] "Diabolique (1996)"
##  [5,] "Truth About Cats & Dogs, The (1996)"
##  [6,] "Visitors, The (Visiteurs, Les) (1993)"
##  [7,] "Cimarron (1931)"
##  [8,] "Marty (1955)"
##  [9,] "Rushmore (1998)"
## [10,] "Last Days, The (1998)"
```

```
recommender_model2 = Recommender(ratingmat_bin, method = "UBCF", param=list(method="Cosin
e",nn=30))
recom = predict(recommender_model2, ratingmat2[2], n=10) #Obtain top 10 recommendations fo
r 2nd user in dataset
recom_list = as(recom, "list")

#Obtain recommendations
recom_result = matrix(0,10)
for (i in c(1:10)){
  recom_result[i] = movies_df[as.integer(recom_list[[1]][i]),2]
}

print(recom_result)
```

```
##       [,1]
##  [1,] "Balto (1995)"
##  [2,] "When Night Is Falling (1995)"
##  [3,] "Nico Icon (1995)"
##  [4,] "City Hall (1996)"
##  [5,] "Happy Gilmore (1996)"
##  [6,] "Headless Body in Topless Bar (1995)"
##  [7,] "Apollo 13 (1995)"
##  [8,] "Batman Forever (1995)"
##  [9,] "Casper (1995)"
## [10,] "Feast of July (1995)"
```

The Basic methodology we used is below:



methodology

Results and Discussion

Popularity based model

We used movies metadata file to build this model. The ideology behind this model is that user likes movies that were highly successful in profits. So, we just used data preparation techniques to build this model. So, it'll display the top 10 movies that gained higher profits at box-office.

Advantages

The model can suggest movies without any information about the user.

Disadvantages

Every user will have the same recommendation list of movies.

Content based model

We used ratings.csv file to build this model. Here, we predicted the ratings based on the features of the movies. We construct a binary matrix with rows consisting of movies and columns consisting of features based on whether a feature is available in that movie or not. We calculated the cosine similarities between pairs of movies. The ratings of the movies can be predicted using the weighted averaging technique for recommendation purpose.

Advantages

Learns user's preferences and Highly personalized for the user.

Disadvantages

Doesn't take into account what others think of the item, so low quality item recommendations might happen.

Collaborative based model

All users' ratings for films were extracted using the ratings.csv file, which we got from the GroupLens website. We used user-based CF model in recommenderlab package. Using centred cosine similarity between two users, you may determine which two people have similar tastes in movies. When two users are similar, their ratings' similarity value is close to 1 (or positive compared to other users), which indicates that these two individuals are probably to watch similar movies. Therefore, the system suggests a movie to the first user who hasn't seen it if the other person has seen it and gave it a high rating.

Advantages

Takes other user's ratings into consideration and Adapts to the user's interests which might change over time.

Disadvantages

Privacy issues when trying to learn the user's preferences and There might be a low of amount of users to recommend.

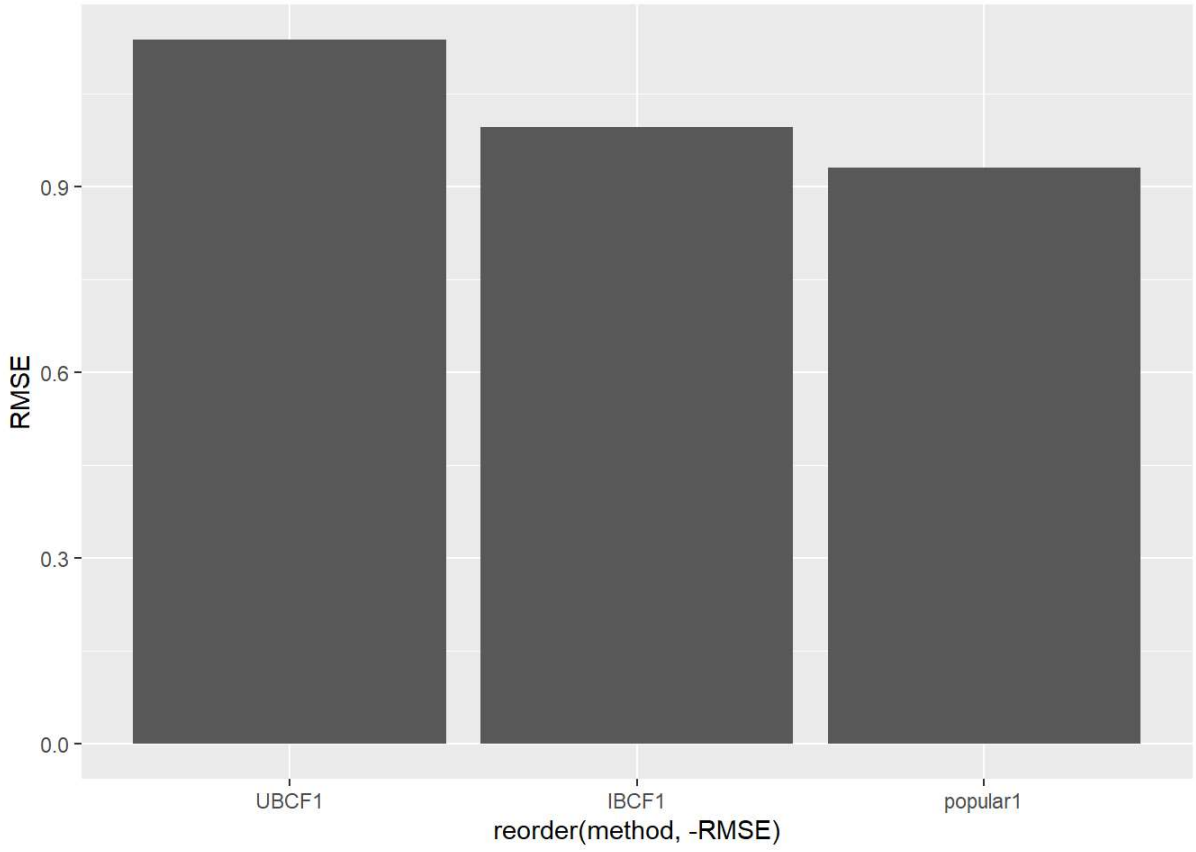
We evaluated our models by using recommenderlab package present in Rlanguage. The results are as below.

Evaluation

```
# Evaluation of models


scheme = evaluationScheme(ratingmat2, method = 'cross-validation',k = 5, given = -1,goodRa
ting = 5)

algs = list('popular' = list(name = 'POPULAR',param = NULL),
            'IBCF' = list(name = 'IBCF',parameter = list(method = "Cosine")),
            'UBCF' = list(name = 'UBCF',param = list(nn = 30)))

set.seed(1)
results_list = evaluate(scheme,method = algs,type = 'ratings')
```

```
## POPULAR run fold/sample [model time/prediction time]
##   1  [0sec/0.19sec]
##   2  [0.02sec/0.14sec]
##   3  [0.02sec/0.16sec]
##   4  [0.02sec/0.16sec]
##   5  [0.01sec/0.16sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [94.31sec/0.2sec]
##   2  [97.44sec/0.09sec]
##   3  [71.79sec/0.1sec]
##   4  [74.59sec/0.08sec]
##   5  [123.81sec/0.19sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.02sec/6.84sec]
##   2  [0sec/6.04sec]
##   3  [0.02sec/6.82sec]
##   4  [0sec/4.19sec]
##   5  [0.02sec/3.67sec]
```

```
library(dplyr)
labels = c('RMSE','MSE','MAE')
results = avg(results_list) %>% unlist() %>% as.data.frame()
results$label  = labels
results$method = row.names(results)
colnames(results)[1]  = 'RMSE'
library(ggplot2)
results %>% filter(label == 'RMSE') %>% ggplot(aes(reorder(method,-RMSE),RMSE)) + geom_bar
(stat = 'identity')
```



We evaluated ou model with the parameter **RMSE**. RMSE for every model is low.

Every model has its own advantages and disadvantages.

Conclusion

We built three types of recommendation systems. But there are other models out there. We have a data of only 671 users and their reviews which might not be the sufficient data for proper Recommendation System. Our work is better as far as possible.

There are advantages and disadvantages of every model. So, There are Hybrid Recommendation System which overcomes the disadvantages of these models.

Disadvantages

Popularity based model - There is no user preference.

Content based model - The proper data distribution is mandatory for the model to work efficiently

User based Collaborative model - This model takes more memory as it should calculate similarity among every user. If number of users are in millions. It will take more time for computation.

So, the future scope of our project will be building Hybrid Recommendation System.

Data and Software Availability

You can find our work here :

https://github.com/maheswarreddy01/movie_recommendation_system (https://github.com/maheswarreddy01/movie_recommendation_system)

References

[1]. Marija Juodyte, "Overview: Data Mining Pipeline", URL: https://www5.in.tum.de/lehre/seminare/datamining/ss17/paper_pres/01_pipeline/Data_Mining_Pipeline.pdf (https://www5.in.tum.de/lehre/seminare/datamining/ss17/paper_pres/01_pipeline/Data_Mining_Pipeline.pdf)

[2]. Rajeev Kumar |Guru Basava | Felicita Furtado "An Efficient Content, Collaborative – Based and Hybrid Approach for Movie Recommendation Engine" Published in International Journal of Trend in Scientific Research and Development(ijtsrd), ISSN: 2456-6470, Volume-4 |Issue-3,April 2020, pp.894-904, URL: <www.ijtsrd.com/papers/ijtsrd30737.pdf>

[3]. MaríaN.Moreno, SaddysSegreraVivian, F.López,María Dolores Muñoz, Ángel LuisSánchez, "Web mining based framework for solving usual problems in recommender systems. A case study for movies' recommendation", Neurocomputing, volume 176, February 2016. https://doi.org/10.1016/j.neucom.2014.10.097 (https://doi.org/10.1016/j.neucom.2014.10.097)

[4]. SRS Reddy, Sravani Nalluri, Subramanyam Kunisetti, S. Ashok & B. Venkatesh, Content-Based Movie Recommendation System Using Genre Correlation, URL: https://link.springer.com/chapter/10.1007/978-981-13-1927-3_42#Sec3 (https://link.springer.com/chapter/10.1007/978-981-13-1927-3_42#Sec3)

[5]. Mojdeh Saadati, Syed Shihab, Mohammed Shaiqur Rahman, Movie Recommender Systems: Implementation and Performace Evaluation URL: https://arxiv.org/ftp/arxiv/papers/1909/1909.12749.pdf (https://arxiv.org/ftp/arxiv/papers/1909/1909.12749.pdf)