Netflix Consumption Analysis And Recommendation Model

Submitted by:

Venkata Sai Akshay Kishore Khanderao (A20458999)

Soumya Elizabeth Thomas (A20456962)

FNU Deepanshu (A20449479)

Table of Contents

Abstract

Over-the-top content platforms such as Netflix, Amazon prime video, and others have experienced exponential growth in the last few years. An exciting aspect of this exponential growth is the difference in age and the way people watch TV-shows. Besides the time management advantages that online TV brings to people, people often choose online and on-demand TV in the absence of commercial breaks, the ability to watch where they want and whichever device they want. Netflix platform is one of the parties that jumped into the world of online streaming services. To better serve the user's needs, every online platform needs a good system that helps decrease the time to discover new content.

The project's main aim is to create a recommendation system based on the user. A recommendation system provides suggestions to the user based on user preferences and watches history. The user information is taken as an input to provide the recommendation based on their preferences. In this, we are using the Item-based Collaborative Filter Recommendation. An item-based Collaborative Filter Recommendation provides recommendations with respect to other users who have similar preferences.

Introduction

Netflix has evolved to become one of the most popular over-the-top content platform provider and production companies in the world from a humble DVD-by-mail service. It is one of the first companies to see the potential for an over-the-top content platform and started the transition in early 2007. Since the transition, the annual revenue has grown exponentially from 1.36 billion to around 15.8 billion in just ten years. It has also considerably increased its subscriber base from 22 million in 2011 to 150 million in 2019.

The dataset consists of all the tv shows and movies available on Netflix till 2019. The dataset is collected from Flixable, which is a third-party Netflix search engine. They released a report which shows the number of TV shows on Netflix has tripled, and the movies have decreased by 2000 in number since 2010. This provides to further incentivize us to explore all other valuable insights into the data.

Data Summary

Description:

The dataset was found on Kaggle; it contains the data related to Netflix over-the-top content platform data related to both Tv-Shows and Movies.

This, coupled with ImdbRatings and ImdbMovies datasets, are used to get interesting observations, which are used to answer the following questions.

Specific Questions:

Content added to Netflix over the years across various countries.

Top movie directors, top genres

Which countries are producing most TV shows and movies?

The top TV show directors and genres

The Average movie duration in each country

Most frequent show categories

The TV shows having the largest number of seasons

Most prolific directors and actors who are associated with most movies on Netflix

Data Preparation

The initial step consists of transforming the raw data into a more understandable format to feed it as an input to the model. Data preparation is a wrangling method where data transformation occurs, i.e., inconsistent, incomplete data with errors is converted to an understandable format.

Import Datasets

In this step, we import all the datasets collected and required for the generation of a model. In this, we use multiple data files that are which are netflix_titles.csv, IMDb movies, IMDb ratings. The most important job in this step is finding the specific features required for the model building. During the data preparation, we understood that only weighted_average_vote in IMDb ratings. and 'title','year','genre','reviews_from_users' are required for the model building. We take the features mentioned above in the IMDb movies, IMDb ratings to make a new data frame by joining them on the imdb_title_id column. As both datasets are generated from the same source, no conditions were implemented during the join process.

Dropping Duplicates

In this step, we drop the duplicates from the newly created Imdb dataset based on the title, country, type, release_year variables, which corroborates that the data is duplicated. Such data doesn't add any valuable insights; hence we drop the duplicates.

Handling Missing Values

In this step, we check for missing values it is done so that it does not cause any issues during model training. If there is a non-numerical value missing, we remove the entire row which has a missing value. There is a probability that we miss some important data, but this is compensated by a large amount of data available to us. But if there is a numerical value is missing, we replace it mode.

Checking for categorical Data

ML models perform complex mathematical operations for it to be done, data should appear in a numerical format, and non-numerical values cannot be given as an input. To handle this situation, Data should be in a numerical form to perform computation on it. To perform this "as.numeric()" and "as.factors()" methods are used.

There are also some features in which data has been recorded in multiple formats; one such is the data_added feature; to this, we convert the object data type into DateTime value and then convert it into the required date-time format, which helps during the model development.

Split data into training, validation and evaluation sets

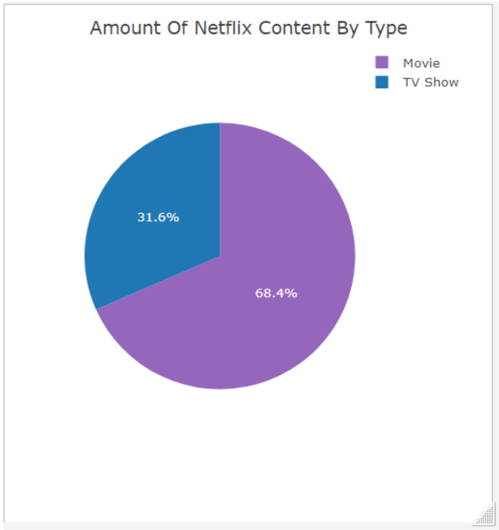
To check how the model works, we split the data into 3 different sets: first is the training set to train the model, second is the validation set to validate the accuracy of our model, and finally, the last is the test set to test the performance of our model to see how the model functions on the new data. This is used to predict the ratings of movies and TV shows.

Exploratory Data Analytics

EDA is the phase in which we understand the data using different analytical methods and Visualizations to get valuable insights into the data.

Ratio of TV Show vs Movies

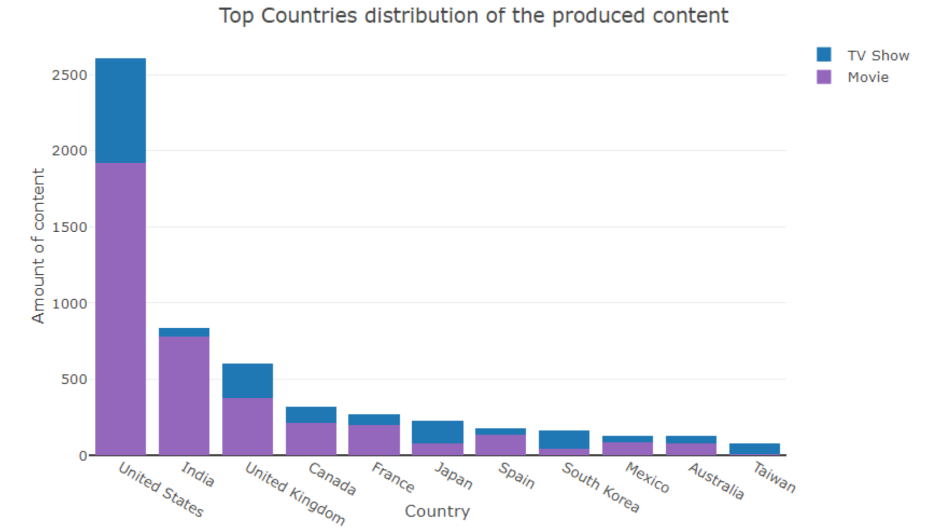
First, we see the ratio of TV Show and movies available on the Netflix OTT. Below Pie Chart shows the ratio. As we see from the PI chart below, there are twice the movies available than the TV-Shows.



Movies and TV shows are made by several countries

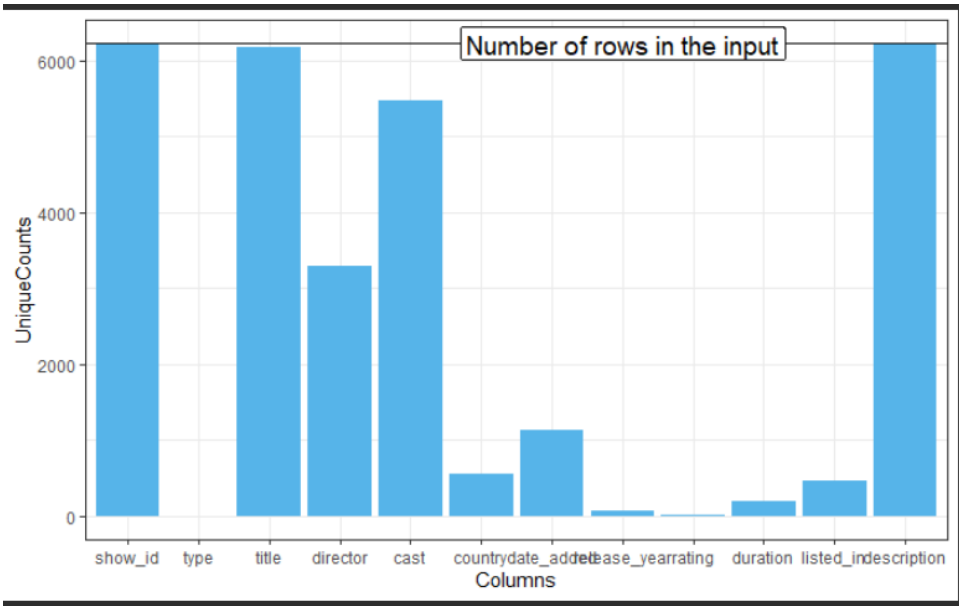
As the movies and tv shows are produced by different countries, to correctly count the total amount of content created by each country, we need to split strings in-country variable and count each country's entire occurrence on its own.

From the below bar graph, we can see the United States has produced the highest number of TV shows and movies, followed by India and other countries.



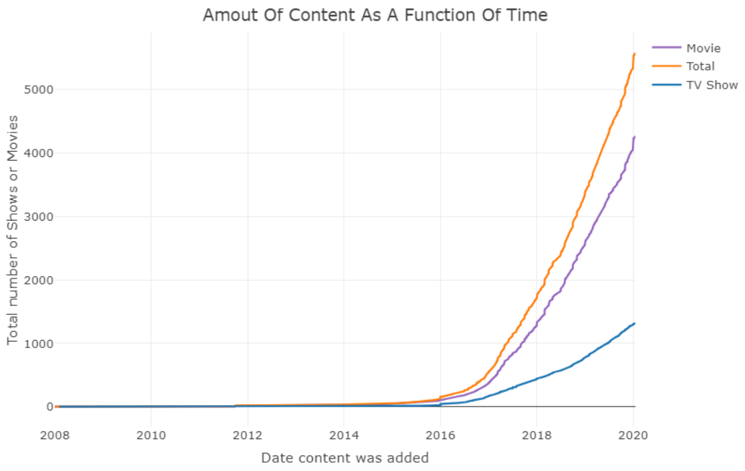
Unique values in each column

From the below figure, we can see that each entry in the show_id has a unique value, representing that a unique id represents each show id. Show titles and descriptions are also almost unique for each show indicating that we don't have to worry too much about duplicate entries in the input data.



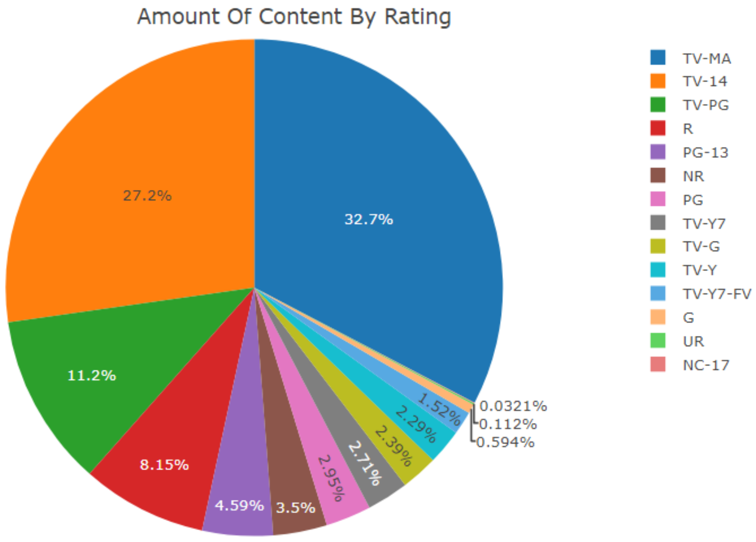
Total number of TV Shows /Movies available on Netflix over the years

From the below graph, we can see the rapid increase in the number of Tv Shows and movies produced since the year 2016. Netflix has about 4200 or more movies as in 2019 and 1400 TV shows. The number of movies it is streaming has increased exponentially over the years 2017 to 2019.



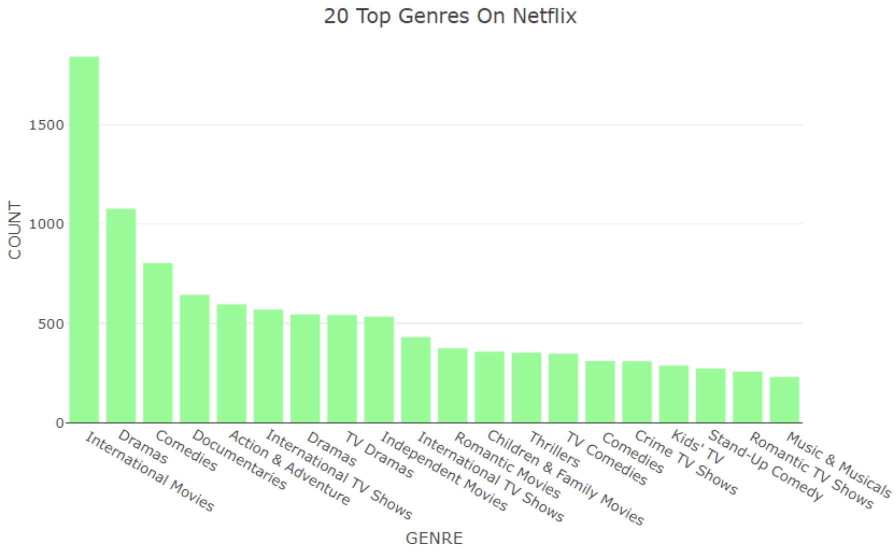
Distribution of content by rating classes

The TV parental guidelines are a television content rating system according to which each show and movie must be rated. To see which content has the highest amount of shows, we use the below figure.



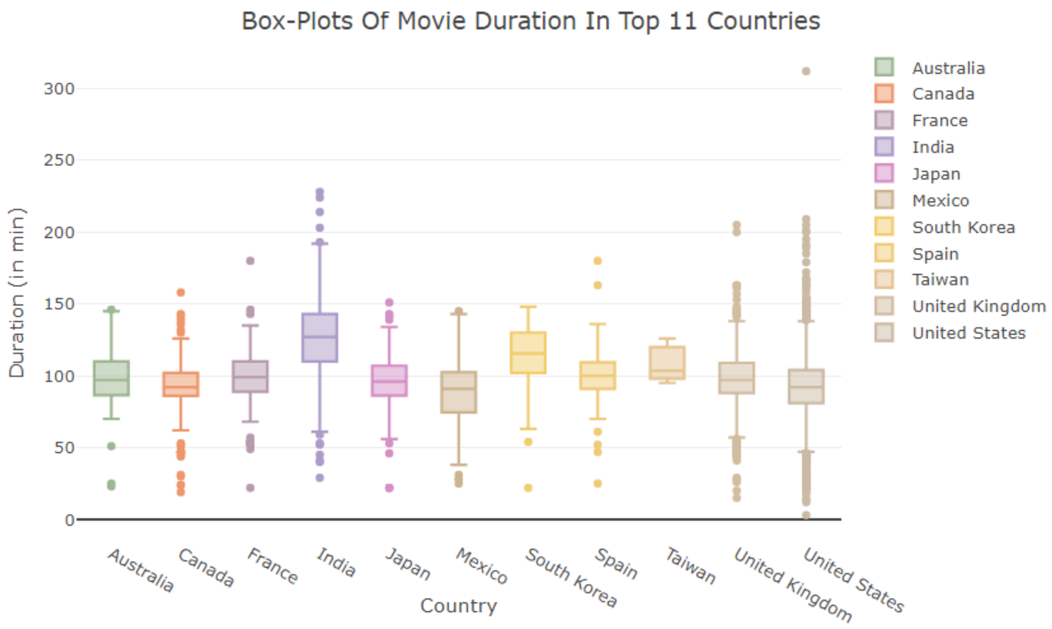
Top genres in Netflix

The below bar graph shows the total number of movies and TV-shows that are produced for each genre. From the figure, we can see international movies are highest produced, followed by Drama.



Movie Duration by Countries

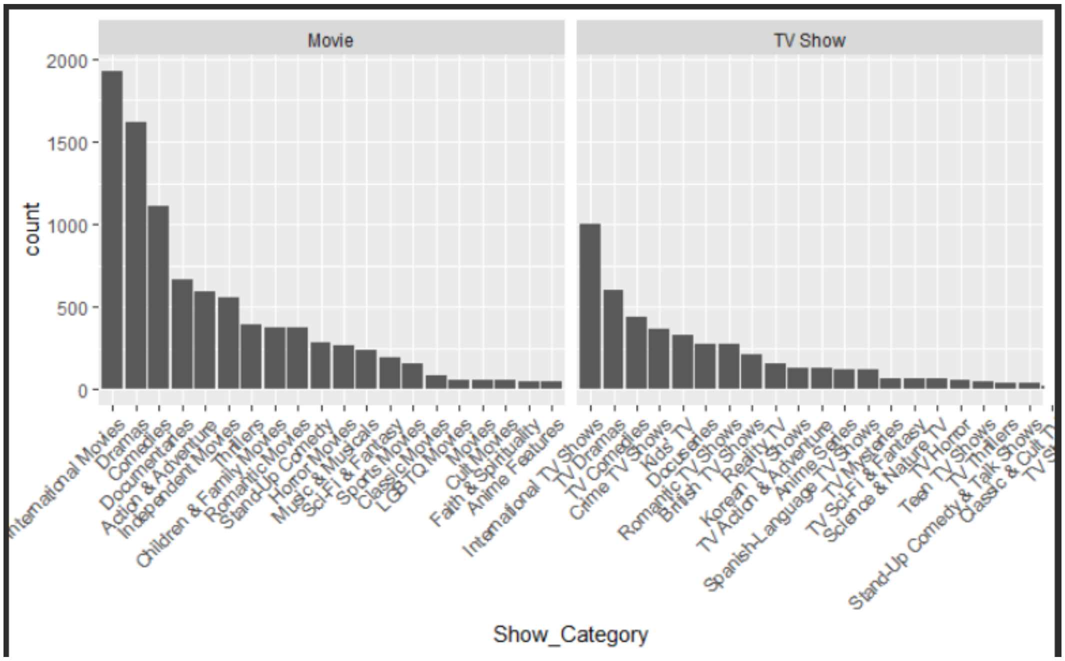
The below box plot shows that India has the highest duration movie, followed by the United States and the United Kingdom.



Most frequent movie or Tv Show categories

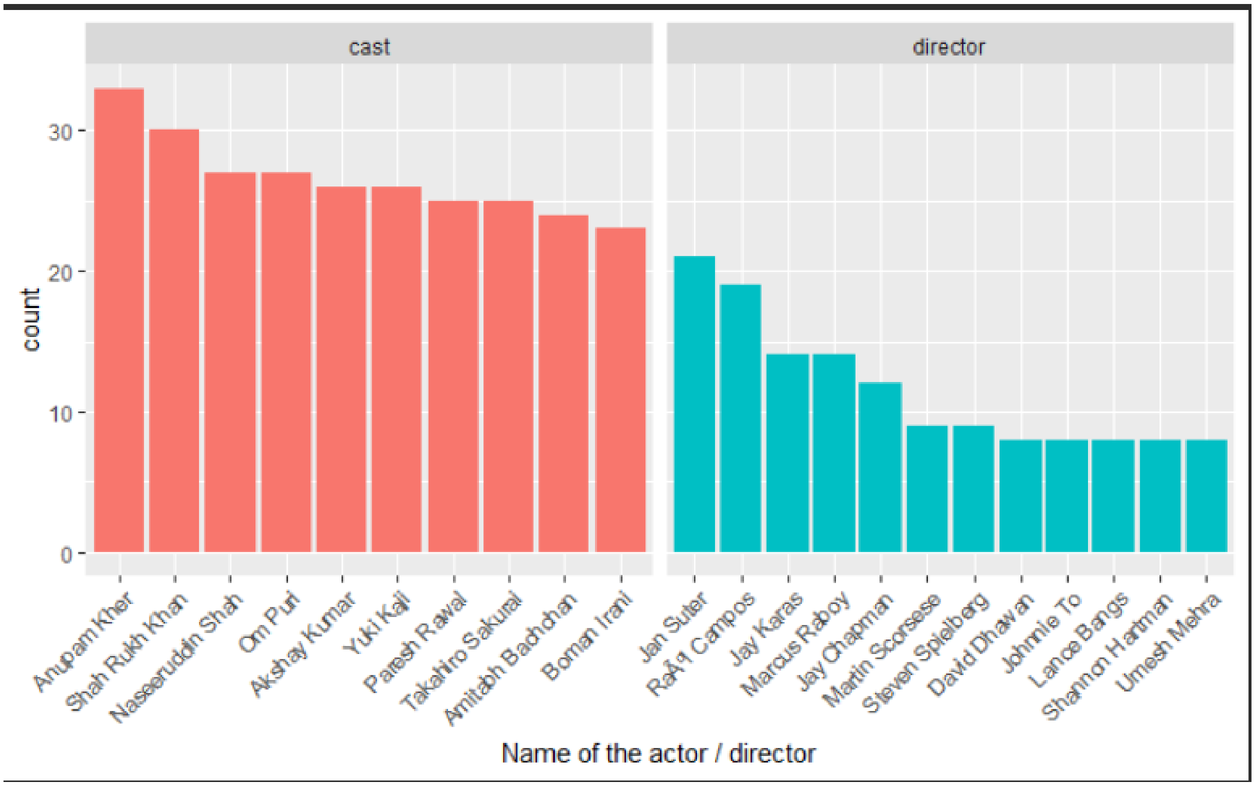
The below bar graph displays the number of TV shows according to each category. From the graph, we can see that International Movies / TV Shows are showing up as the dominant category in both Movies and TV shows, followed by Drama and Comedy.

Action movies are more popular (number-wise) than action TV shows.



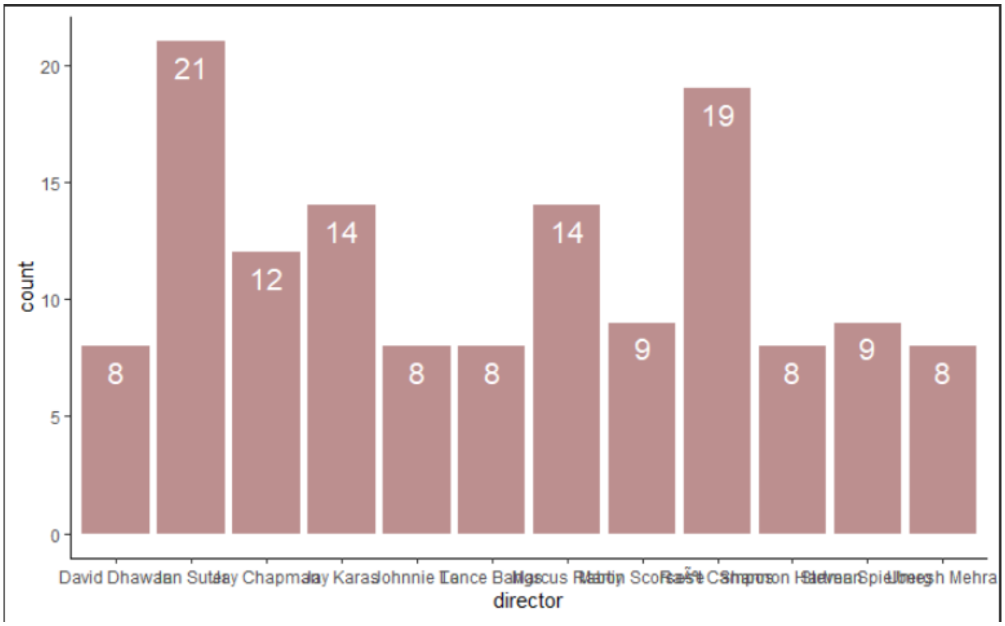
List of Prolific directors, actors who are associated with most movies

From the below bar graph, we can see that among Actors, "Anupam Kher" followed by Shah Rukh Khan has been associated with most Movies. Whereas among directors Jan Suter has been associated with most Movies.



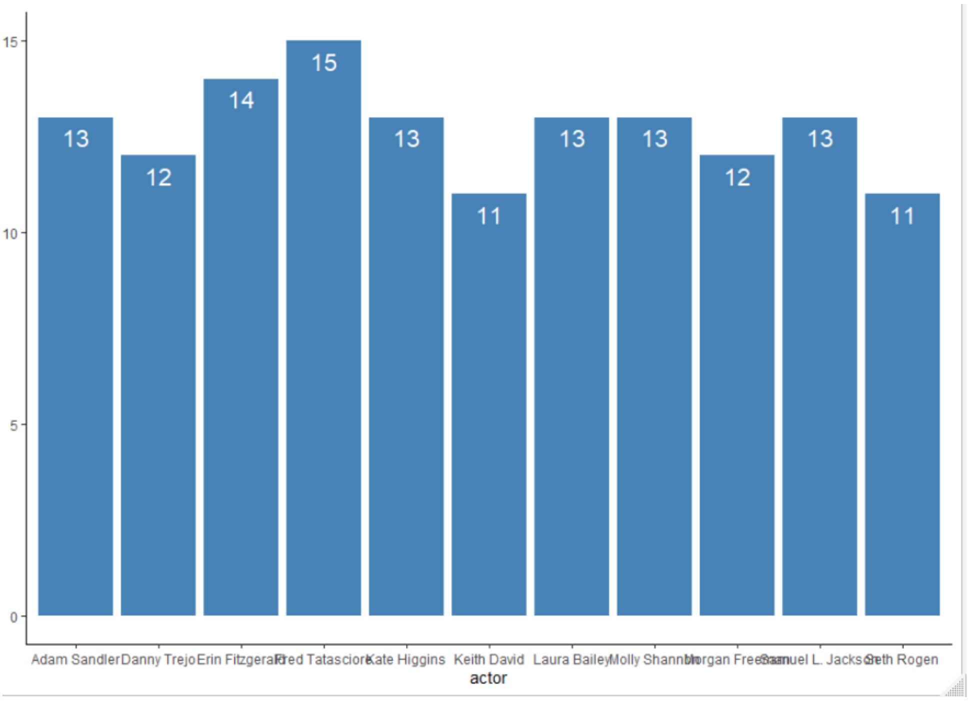
Top 10 directors by the amount of content on Netflix

The below list shows the top 10 directors according to the amount of content on Netflix.



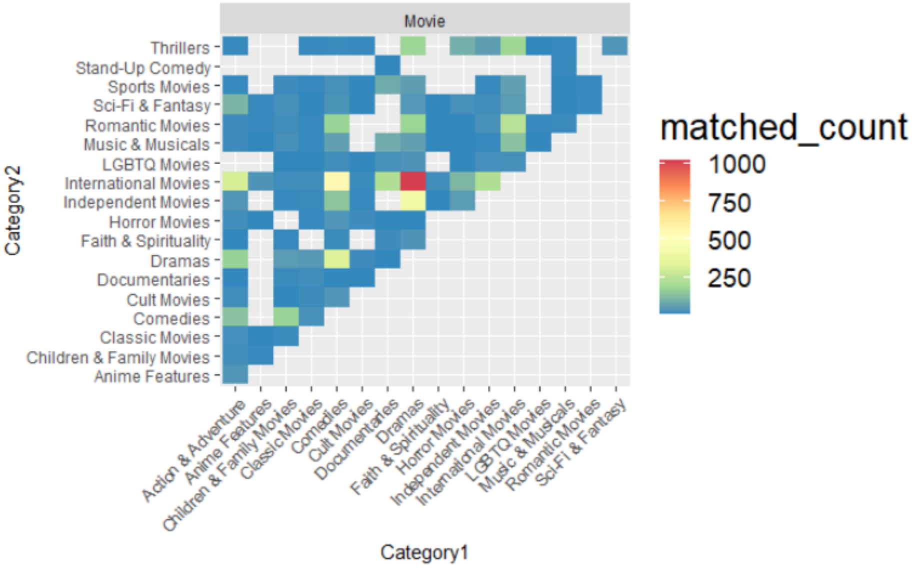
Top 10 actors by the amount of content on Netflix in US

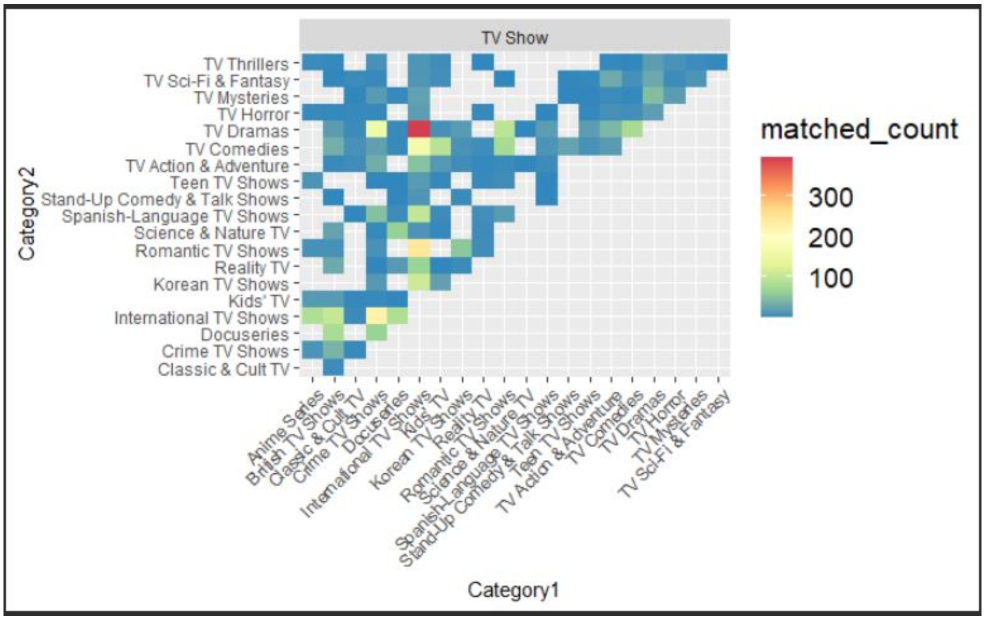
The below list shows the top 10 directors according to the amount of content on the Netflix.



Correlation of categories with TV shows/movies

Below Heatmap figures show the correlation of features. From this, we can see many international movies are listed "Dramas" that contribute to the higher correlation between the two features. There is a significant overlap between "Drama" and "Comedy" as expected.





Modeling

Recommendation Model

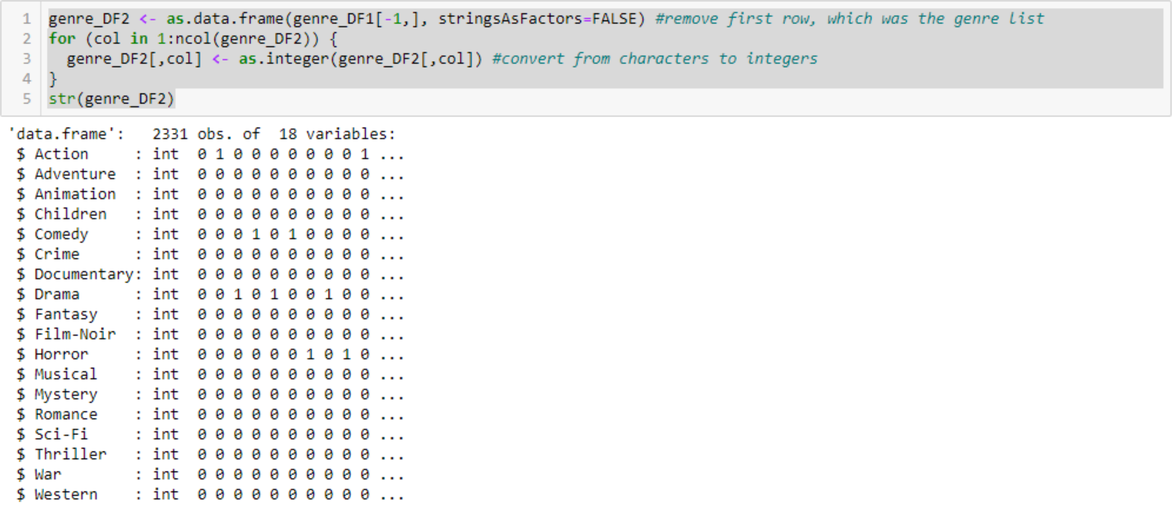
We must convert our matrix into a sparse matrix one for our movie recommendation system to make sense of our ratings through recommenderlabs. We implement Item Based Collaborative Filtering. Collaborative Filtering includes suggesting movies to the users that are based on collecting preferences from many other users. For example, if user-1 prefers to watch action movies, and so does user-2, then the movies that the user-2 will watch in the future will be suggested to user-1 and vice-versa. Therefore, recommending movies is dependent on forming a relationship of similarity in the behavior of the two users. With the help of recommenderlab, we can find similarities using various operators like Cosine, Pearson as well as Jaccard.

The algorithm first makes a similar-items table of the customers who have purchased them into a combination of similar items. This is then supplied into the recommendation system. The similarity between a single product and related products can be determined with the following algorithm:

1. Check each Item i1 present in the product catalog, purchased by customer C.
2. And, also for each item, i2 also purchased by customer C.
3. Create a record that the customer purchased items i1 and i2.
4. Calculate the similarity between i1 and i2.
5. We build this filtering system by splitting the dataset into 80% training set and 20% test set.

Modeling Steps:

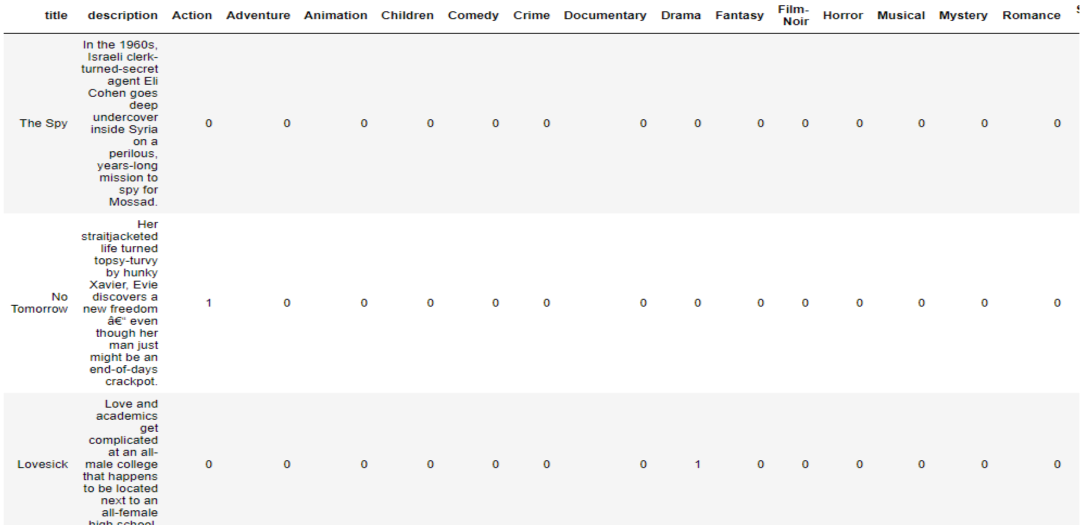
Step1: We need to convert the genres present in the "movie_data" data frame into a more usable format by the users. To do so, we will first make a one-hot encoding to generate a matrix that comprises corresponding genres for each of the films.



```
genre_DF2 <- as.data.frame(genre_DF1[-1,], stringsAsFactors=FALSE) #remove first row, which was the genre list
for (col in 1:ncol(genre_DF2)) {
  genre_DF2[,col] <- as.integer(genre_DF2[,col]) #convert from characters to integers
}
str(genre_DF2)
```

```
'data.frame':	2331 obs. of  18 variables:
 $ Action     : int  0 1 0 0 0 0 0 0 0 1 ...
 $ Adventure  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Animation  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Children   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Comedy     : int  0 0 0 1 0 1 0 0 0 0 ...
 $ Crime      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Documentary: int  0 0 0 0 0 0 0 0 0 0 ...
 $ Drama      : int  0 0 1 0 1 0 0 1 0 0 ...
 $ Fantasy    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Film-Noir  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Horror     : int  0 0 0 0 0 0 1 0 1 0 ...
 $ Musical    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Mystery    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Romance    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Sci-Fi     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Thriller   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ War        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Western    : int  0 0 0 0 0 0 0 0 0 0 ...
```

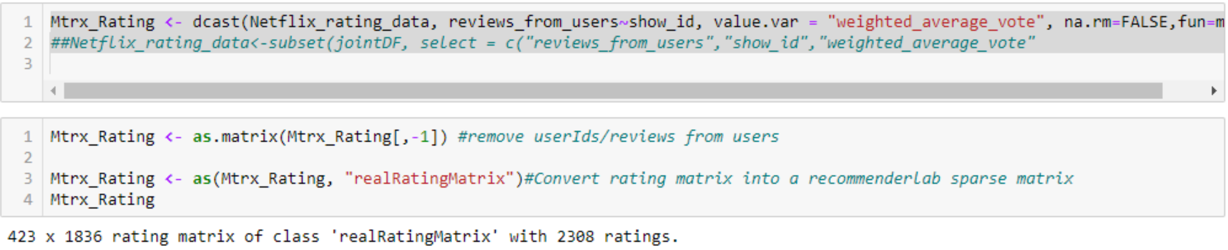
Step2: We will generate a 'search matrix' that will allow us to perform an easy search of the movies by specifying the genre present in our list.



title	description	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance
The Spy	In the 1960s, Israeli clerk-turned-secret agent Eli Cohen goes deep undercover inside Syria on a perilous, years-long mission to spy for Mossad.	0	0	0	0	0	0	0	0	0	0	0	0	0	0
No Tomorrow	Her straitjacketed life turned topsy-turvy by hunky Xavier, Evie discovers a new freedom â€" even though her man just might be an end-of-days crackpot.	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Lovesick	Love and academics get complicated at an all-male college that happens to be located next to an all-female high school	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Step3: Some movies have several genres; for example, Toy Story, an animated film, also falls under Children, Fantasy and Comedy. This implements in the majority of the movies.

We have to transform our matrix into a sparse matrix one for our movie recommendation model to make sense of the ratings through recommenderlabs. This new transformed matrix is of the class 'realRatingMatrix'. This is performed as follows:

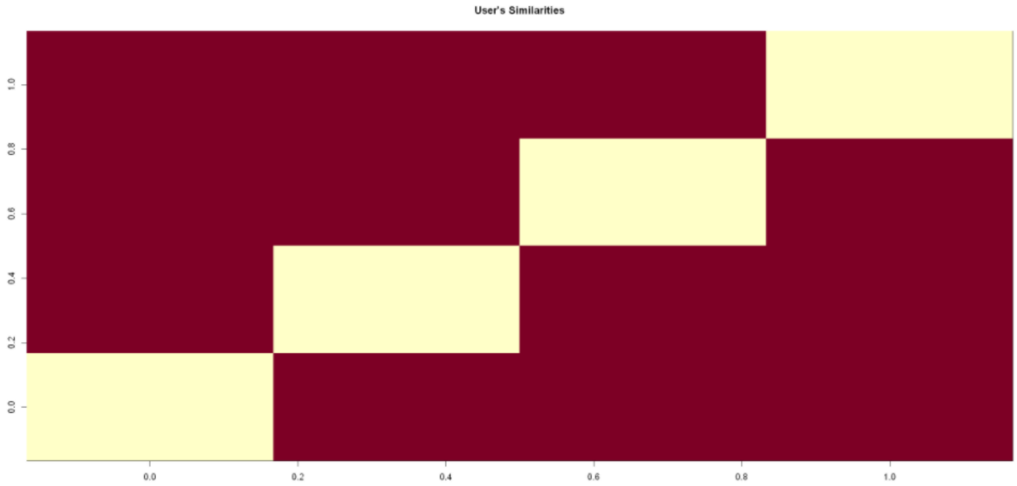


```
Mtrx_Rating <- dcast(Netflix_rating_data, reviews_from_users~show_id, value.var = "weighted_average_vote", na.rm=FALSE,fun=m
##Netflix_rating_data<-subset(jointDF, select = c("reviews_from_users","show_id","weighted_average_vote"

```

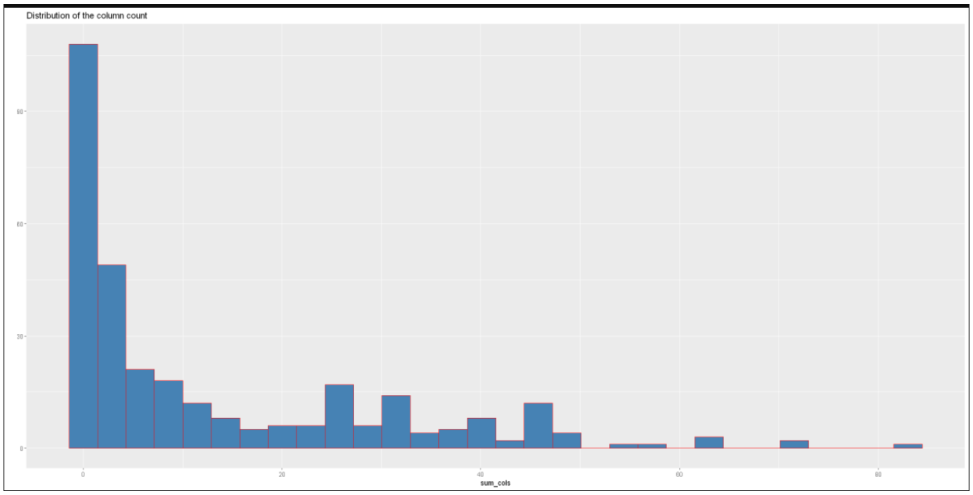
```
Mtrx_Rating <- as.matrix(Mtrx_Rating[,-1]) #remove userIds/reviews from users

Mtrx_Rating <- as(Mtrx_Rating, "realRatingMatrix")#Convert rating matrix into a recommenderlab sparse matrix
Mtrx_Rating
```

```
423 x 1836 rating matrix of class 'realRatingMatrix' with 2308 ratings.
```

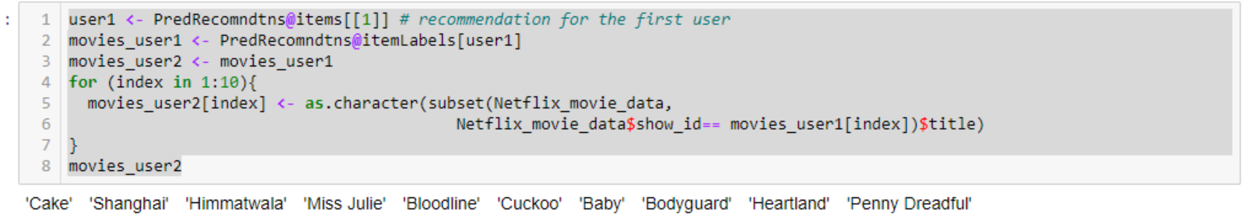
Step 4: We compute similarities between two users using recommenderlab. We do this by using HeatMap.



Step 5: The sum of rows and columns with the similarity of the objects above is 0. We will visualize the sum of columns through distribution as follows:

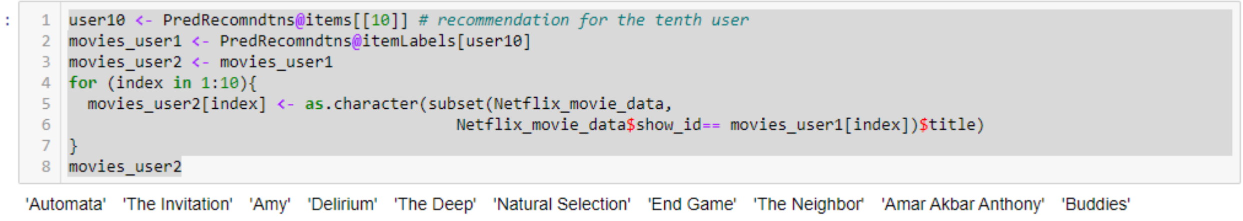


Step 6: Create a top_recommendations variable, which will be initialized to 10, specifying the number of films to each user. We will then use the predict() function to identify similar items and rank them appropriately. Here, each rating is used as a weight. Each weight is multiplied with related similarities. Finally, everything is added at the end.



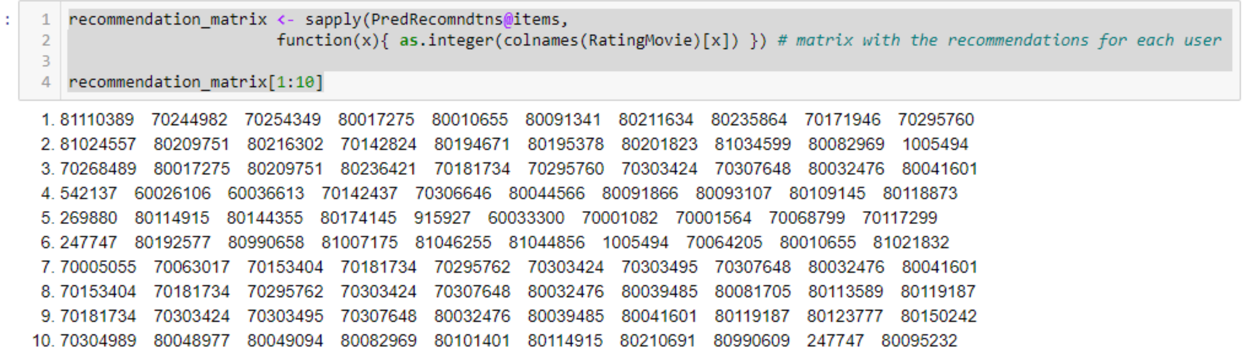
```
user1 <- PredRecomndtns@items[[1]] # recommendation for the first user
movies_user1 <- PredRecomndtns@itemLabels[user1]
movies_user2 <- movies_user1
for (index in 1:10){
  movies_user2[index] <- as.character(subset(Netflix_movie_data,
                                       Netflix_movie_data$show_id== movies_user1[index])$title)
}
movies_user2
```

```
'Cake'  'Shanghai'  'Himmatwala'  'Miss Julie'  'Bloodline'  'Cuckoo'  'Baby'  'Bodyguard'  'Heartland'  'Penny Dreadful'
```



```
user10 <- PredRecomndtns@items[[10]] # recommendation for the tenth user
movies_user1 <- PredRecomndtns@itemLabels[user10]
movies_user2 <- movies_user1
for (index in 1:10){
  movies_user2[index] <- as.character(subset(Netflix_movie_data,
                                       Netflix_movie_data$show_id== movies_user1[index])$title)
}
movies_user2
```

```
'Automata'  'The Invitation'  'Amy'  'Delirium'  'The Deep'  'Natural Selection'  'End Game'  'The Neighbor'  'Amar Akbar Anthony'  'Buddies'
```



```
recommendation_matrix <- sapply(PredRecomndtns@items,
                      function(x){ as.integer(colnames(RatingMovie)[x]) }) # matrix with the recommendations for each user

recommendation_matrix[1:10]
```

```
 1. 81110389   70244982   70254349   80017275   80010655   80091341   80211634   80235864   70171946   70295760
 2. 81024557   80209751   80216302   70142824   80194671   80195378   80201823   81034599   80082969   1005494
 3. 70268489   80017275   80209751   80236421   70181734   70295760   70303424   70307648   80032476   80041601
 4. 542137   60026106   60036613   70142437   70306646   80044566   80091866   80093107   80109145   80118873
 5. 269880   80114915   80144355   80174145   915927   60033300   70001082   70001564   70068799   70117299
 6. 247747   80192577   80990658   81007175   81046255   81044856   1005494   70064205   80010655   81021832
 7. 70005055   70063017   70153404   70181734   70295762   70303424   70303495   70307648   80032476   80041601
 8. 70153404   70181734   70295762   70303424   70307648   80032476   80039485   80081705   80113589   80119187
 9. 70181734   70303424   70303495   70307648   80032476   80039485   80041601   80119187   80123777   80150242
10. 70304989   80048977   80049094   80082969   80101401   80114915   80210691   80990609   247747   80095232
```

As can be seen above for User 1, the movies “cake”, “shangai”, “Himmatwala and so on have been (10 recommendations) recommended based on users like that of user 1.

Another example for user 10 is also shown with movies “Automata”, “The invitation, “Amy” have been recommended.

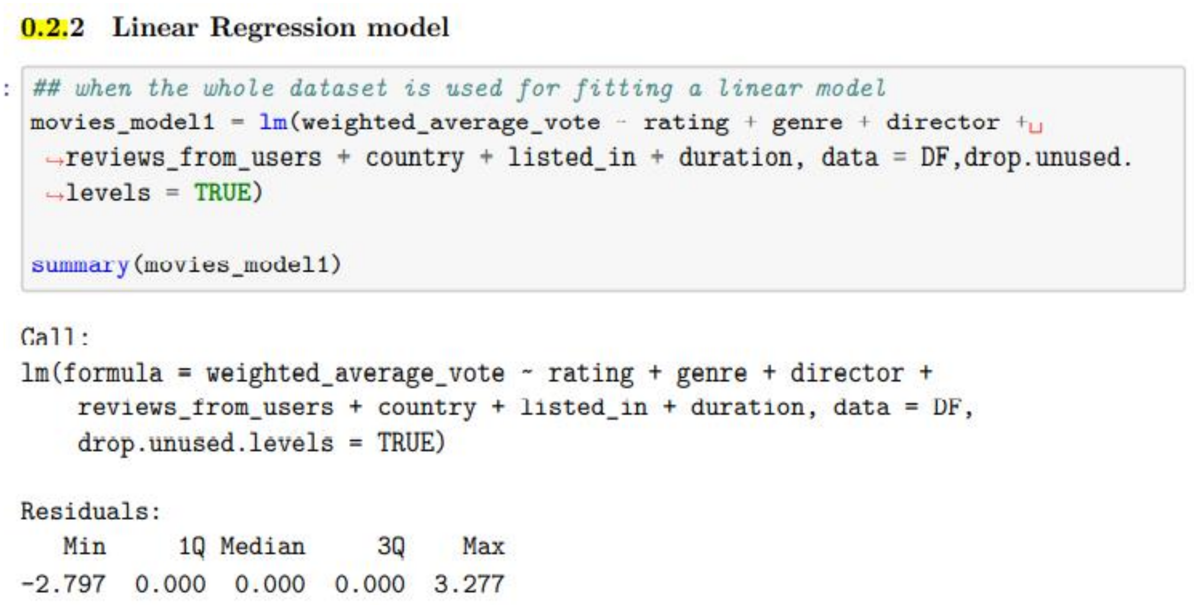
Prediction Model:

In the prediction model, we try to predict the weighted vote averages, representing the Movie/TV Show rating. We are using a Linear regression Model to predict movie ratings.

Step-1: In this, we omit the missing values (NA)

Step-2: Convert the feature to Factor levels

Step-3: then the linear regression is applied to data as the weight average vote as dependent vector and other as independent vector

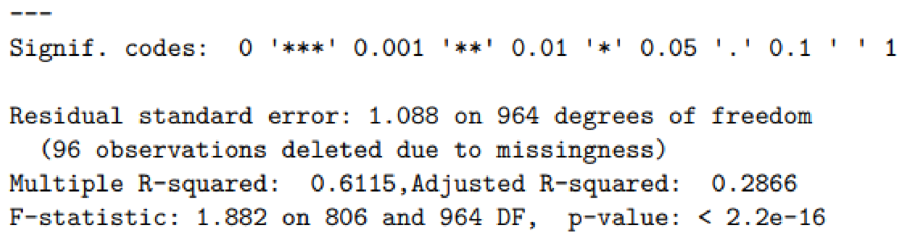


0.2.2 Linear Regression model

```
: ## when the whole dataset is used for fitting a linear model
  movies_model1 = lm(weighted_average_vote ~ rating + genre + director +␣
   ↪reviews_from_users + country + listed_in + duration, data = DF,drop.unused.
   ↪levels = TRUE)

  summary(movies_model1)
```

```
Call:
lm(formula = weighted_average_vote ~ rating + genre + director + 
    reviews_from_users + country + listed_in + duration, data = DF, 
    drop.unused.levels = TRUE)

Residuals:
   Min     1Q Median     3Q    Max 
-2.797  0.000  0.000  0.000  3.277 
```

The performance of the regression model is measured using the following measures.



```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.088 on 964 degrees of freedom
  (96 observations deleted due to missingness)
Multiple R-squared:  0.6115,Adjusted R-squared:  0.2866 
F-statistic: 1.882 on 806 and 964 DF,  p-value: < 2.2e-16
```

R-squared increases every time you include a variable to model. The R-squared never decreases. A regression model that contains more independent variables can look like it provides a better fit merely because it contains more variables.

The adjusted R-squared is used to compare the goodness-of-fit for regression models containing differing numbers of independent variables. The adjusted R-squared adjusts for the number of terms in the model, i.e., its value increases only when the new term improves the model fit more than expected by chance alone. The adjusted R-squared value decreases when the term doesn't improve the model fit by enough.

As there is a difference between R square and adjusted R square in our model, we should be looking at which all the variables are to be included in our regression model. Both the coefficient estimates and predicted values have larger margins of error around them. That's why you don't want to include too many terms in the regression model.

Conclusion & Future Work:

Recommendation systems are becoming more popular with the advent of big data. The base of recommendation systems is to give different services to different types of users. The collaborative filter recommendation techniques utilize the information of an active user's like-minded neighborhood to make recommendations. Personalized recommendation systems have been widely implemented to deal with information overload queries and create personalized recommendations for user's information in streaming platforms. Recommender systems have their origins in various research areas, including information retrieval and information filtering. Currently, famous recommendation algorithms are mainly divided into content-based recommendations, collaborative filtering (CF) recommendations, hybrid recommendations, and other algorithms.

From the EDA, we can see the content of the OTT platforms is increasing exponentially, so usage of a collaborative filtering algorithm based on user preference clustering reduces the data overload problem and makes it convenient for the user to navigate content. In this project, we have developed an item-based collaborative filtering recommendation. Collaborative filtering recommendation aims to calculate a list of interesting features to target users based on other like-minded users' preferences. We use discrete features such as genres, directors, and actors in movies to generate a recommendation.

To further this, we are implementing a rating prediction model that predicts the rating of the movies; for that, we are planning to use TFIDF for text processing and do the predict modeling.

The linear regression model generated is relatively a poor model when we compare the R square and adjusted R square. The current model has both the coefficient estimates having large margins of error. That’s why we don’t want to include too many terms in the regression model.

In the future, we are looking ahead to build regression models like GBM, Adaboost, Ridge, Lasso Regression, and Elasticnet Regression to see which models predict the ratings better.

Data Sources

- Data Repository

  https://www.kaggle.com/shivamb/netflix-shows?select=netflix_titles.csv

  https://www.kaggle.com/netflix-inc/netflix-prize-data

  https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset

Bibliography

- Reference resources or Related Work:

  https://www.econstor.eu/bitstream/10419/215745/1/169430860X.pdf

  https://journals.sagepub.com/doi/abs/10.1177/1749602019834554

  https://journals.uic.edu/ojs/index.php/fm/article/view/6138/4999

  https://www.researchgate.net/publication/270665559_The_Netflix_Effect_Teens_Binge_Watching_and_On-Demand_Digital_Media_Trends

  https://journals.sagepub.com/doi/full/10.1177/1354856519890856

  https://www.cs.vu.nl/~sbhulai/papers/paper-fernandez.pdf

  https://beta.vu.nl/nl/Images/werkstuk-postmus_tcm235-877824.pdf

- Supplemental resources:

  https://data.world/chasewillden/netflix-shows

  https://public.tableau.com/profile/dilyan.penev#!/vizhome/MyNetflixStats/MyNetflixStats

  https://public.tableau.com/profile/andreask28#!/vizhome/NetflixUsageV2/NetflixUsage