Movie recommendation system

Project Final Report

Name of the student	CWID Number
Mohana uma sushmanth Penumarthi	A20525576
Vinnapala Sai Ramya	A20526253
Harini Vaidya	A20525926

Data Preparation & Analysis – (CSP 571)

DATE: 21st April 2024

Table of contents

Abstract

In the realm of online streaming platforms, recommendation systems play a pivotal role in enhancing user engagement and satisfaction by delivering relevant content. This project aims to develop an efficient recommendation system for movies based on ratings analysis using the IMDB movie dataset. Through data collection, exploration, and preparation, insights were gleaned regarding movie genres, ratings, and audience preferences.Three machine learning models—linear regression, KNN regression, and decision tree—were constructed and evaluated to predict movie ratings. The KNN regression model emerged as the most effective, offering superior accuracy and precision for rating prediction, thereby facilitating informed decision-making for streaming platform companies and enhancing user experience.

Problem statement

In today's digital world, online streaming platforms are majorly focused on captivating users. Delivering relevant content is key to capturing their attention. Recommendation systems play a crucial role in sectors like e-commerce and online streaming services, including platforms like Netflix, YouTube, and Amazon to attract users by recommending the relevant content what they like to watch. Precise recommendations for the next product, song, or movie elevate user satisfaction, extend their interaction, and drive sales and profit expansion.

Introduction

The main objective of this project is to develop a recommendation system that assists users in discovering the best movie content based on ratings. For this project, we take IMDB movie dataset from the official IMDB website and would like to analyze what kind of movies are more successful or got a higher reach to audiences. we build the effecient machine learning model to predict the ratings based on the certain features like movie type, runtime of the movie, number of votes given etc. The results from this project can also help the streaming platform companies to understand the factors of successful movie and make a decision regarding future movie acquisitions.

Methodology

The proposed methodology includes the following steps:

Fig 1: Flowchart of the project

In data Collection, we gathered the data from official IMDB website. We have collected two datasets. (Movies dataset and ratings dataset). In the next phase, we process the data by cleaning and removing null values and transforming it into a suitable format for subsequent analysis. We perform data analysis in data exploration stage. we will use exploratory data analysis (EDA) techniques to identify trends, patterns in the data. When the data is ready, we go to model development stage. This stage involves the development of various machine learning and statistical models, such as linear regression, KNN regression, and decision tree also employing model selection techniques to determine the most suitable model for our dataset. In model evaluation, we will evaluate the performance of our models. We will use a variety of evaluation metrics, such as accuracy, Mean Absolute Error, Mean squared error (loss function), Root Mean squared error, and R-squared to assess the performance of our models

Data collection:

we have collected the datasets from official IMDB website. In this project we are considering two datasets - movie dataset and rating dataset. Movie dataset (title.basics.tsv.gz) consist of 10,48,575 rows and 9 features. Rating dataset (title.ratings.tsv.gz) consist of 10,48,575 rows and 3 features.

Dataset link: https://developer.imdb.com/non-commercial-datasets/

Movie Dataset:

	tconst <chr>	titleType <chr>	primaryTitle <chr>	originalTitle <chr>	isAdult <int>	startYear <int>	endYear <int>	runtimeMinutes <int>	genres <chr>
1	tt0000001	short	Carmencita	Carmencita	0	1894	NA	1	Documentary,Short
2	tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	NA	5	Animation,Short
3	tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	NA	4	Animation,Comedy,Romance
4	tt0000004	short	Un bon bock	Un bon bock	0	1892	NA	12	Animation,Short
5	tt0000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	NA	1	Comedy,Short
6	tt0000006	short	Chinese Opium Den	Chinese Opium Den	0	1894	NA	1	Short

Fig 2: Movie dataset

Ratings Dataset:

	tconst <chr>	averageRating <dbl>	numVotes <int>
1	tt0000001	5.7	2024
2	tt0000002	5.7	272
3	tt0000003	6.5	1962
4	tt0000004	5.4	178
5	tt0000005	6.2	2727
6	tt0000006	5.0	184

Fig 3: Ratings dataset

Data Preparation/ Processing:
Data preparation involves cleaning, transforming, and structuring raw data to make it suitable for analysis. This process often includes handling missing values, removing duplicates, standardizing formats, and feature engineering to extract relevant information.

Merging Datasets: We have joined these two datasets using the "tconst" feature as a key using the merge operation and named it as a final_dataset

Final dataset (final_dataset):

	tconst <chr>	titleType <chr>	primaryTitle <chr>	originalTitle <chr>	isAdult <int>	startYear <int>	endYear <int>	runtimeMinutes <int>	genres <chr>	averageRating <dbl>	numVotes <int>
1	tt0000001	short	Carmencita	Carmencita	0	1894	NA	1	Documentary,Short	5.7	2024
2	tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	NA	5	Animation,Short	5.7	272
3	tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	NA	4	Animation,Comedy,Romance	6.5	1962
4	tt0000004	short	Un bon bock	Un bon bock	0	1892	NA	12	Animation,Short	5.4	178
5	tt0000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	NA	1	Comedy,Short	6.2	2727
6	tt0000006	short	Chinese Opium Den	Chinese Opium Den	0	1894	NA	1	Short	5.0	184

6 rows

Fig 4: Final dataset

Dropping unnecessary columns: keeping the unnecessary data in dataset increases uncertainty while building the model. So here we have removed the unwanted column that are tconst, primaryTitle, originalTitle, endYear.

	tconst <chr>	titleType <chr>	isAdult <int>	startYear <int>	runtimeMinutes <int>	genres <chr>	averageRating <dbl>	numVotes <int>
1	tt0000001	short	0	1894	1	Documentary,Short	5.7	2024
2	tt0000002	short	0	1892	5	Animation,Short	5.7	272
3	tt0000003	short	0	1892	4	Animation,Comedy,Romance	6.5	1962
4	tt0000004	short	0	1892	12	Animation,Short	5.4	178
5	tt0000005	short	0	1893	1	Comedy,Short	6.2	2727
6	tt0000006	short	0	1894	1	Short	5.0	184

Fig 5: Final dataset after removing unnecessary columns

The most common problem in every machine learning project are null values. Accordingly, we have checked for null values within the dataset and found null value in isAdult, startYear, runtimeMinutes, genres, averageRating, and numVotes.

tconst	titleType	isAdult	startYear	runtimeMinutes	genres	averageRating	numVotes
0	0	3574702	3857911	8050542	291312	9162163	9162163

Fig 6: Null values in final dataset

Handling these null values effectively is a crucial task.

- For the "isAdult" feature, we assigned a value of 0 to fill the null entries, assuming they represent non-adult movies since the majority fall into this category.
- For the "startYear" feature, we filled null values with the year of the preceding entry, presuming the movie was released in the same year.
- Regarding the "runtimeMinutes" feature, we filled null values with the average runtime corresponding to its title type.
- For the "genre" feature, we substituted null values with "other".
- As for "averageRating" and "numVotes" features, we have observed that both have same number of null values. The entry with no numVotes value also doesn't have averageRating. Filling some value can cause uncertainity in dataset. So we have dropped all the row where averageRating contains null value.

tconst	titleType	isAdult	startYear	runtimeMinutes	genres	averageRating	numVotes
0	0	0	0	0	0	0	0

Fig 7: Handled null values in final dataset

Feature Encoding: We have performed label encoding in this stage. It is a technique used to convert categorical data into numerical format. Each unique category is assigned a unique integer, allowing algorithms to work with categorical variables. Every categorical data in the dataset is converted to numerical.

titleType <int>	isAdult <dbl>	startYear <int>	runtimeMinutes <dbl>	genres <int>	averageRating <dbl>	numVotes <int>
2	0	1894	1	7	5.7	2024
2	0	1892	5	3	5.7	272
2	0	1892	4	3	6.5	1962
2	0	1892	12	3	5.4	178
2	0	1893	1	5	6.2	2727
2	0	1894	1	28	5.0	184

Fig 8: Final dataset after feature encoding

Data Exploring/ Analysis:
Data exploration stage involves examining and understanding the structure and patterns within the movie dataset through various techniques such as summary statistics and data visualization. It aims to uncover insights, identify trends, and patterns, providing a foundational understanding that informs subsequent analysis and decision-making processes.

First, we analyze the titleType feature. titleType feature is all about the type/format of the movie. Let's count the number of movies in each category of titleType.

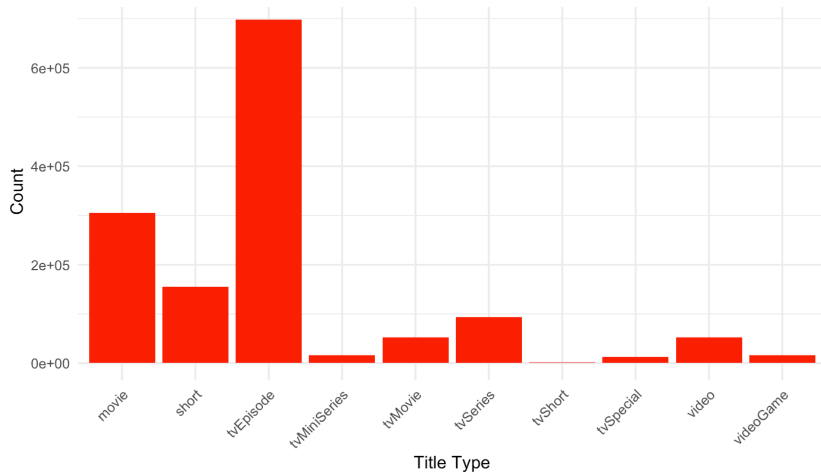


Fig 9: Count of each titleType

From the above plot, we can say that most of the movies belongs to tvEpisodes (698229).
Let's analyze the isAdult feature. This feature tells about the whether the movie contains adult content or not. Let's count the number of movies in each category of isAdult feature.

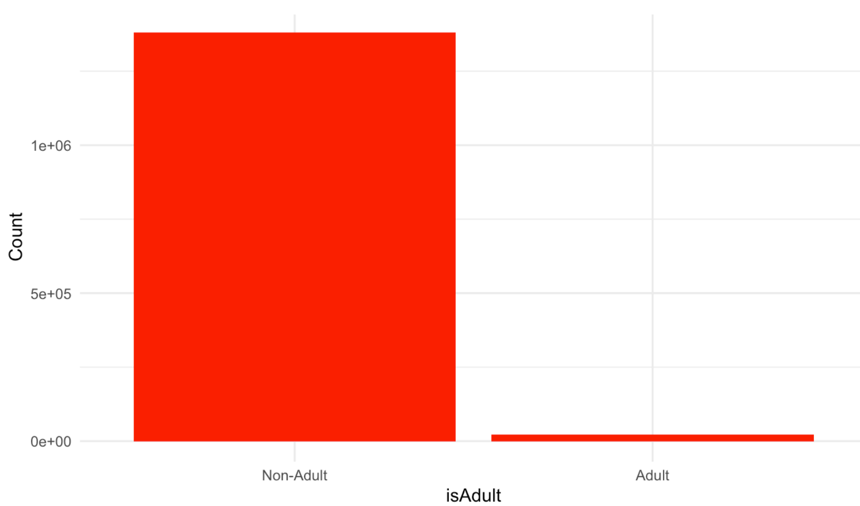


Fig 10: Count of each value in isAdult feature

From the above plot, we can say that most of the movies are non-Adult based (1381769).

Let’s analyze the genre feature. This feature tells about the genre of the movie. Let’s count the number of movies in each category of genre feature.

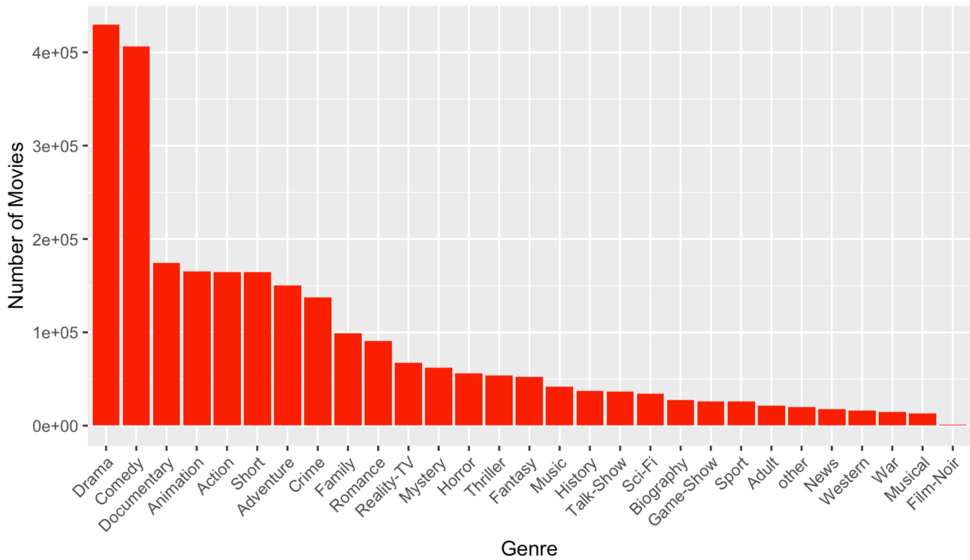


Fig 11: Number of movies in each genre

From the above plot, we can say that most of the movies given in the dataset are belongs to Drama genre (429382) and least number of movies are Film-Noir (880)

Let's find the average ratings received for each genre in the dataset.

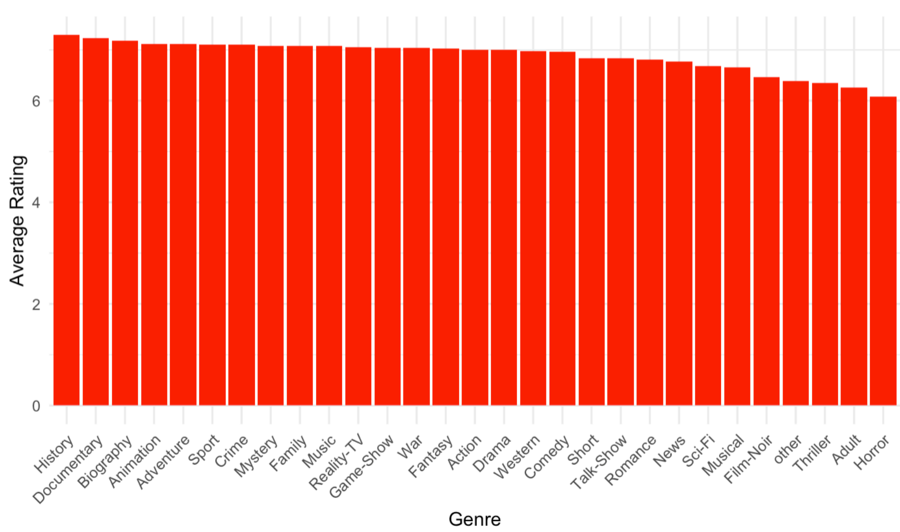


Fig 12: Average rating for each genre

From the above plot, we can say that the average rating for all the genres is around 6 to 7. Movies belongs to History genre got highest average rating (7.289976) and movies belongs to Horror genre got least average rating (6.080079)

Understanding the distribution of the values in a dataset is so crucial. It helps to identify the patterns and skewness of the features. Here we have performed and visual represented the distribution of the averageRating feature.

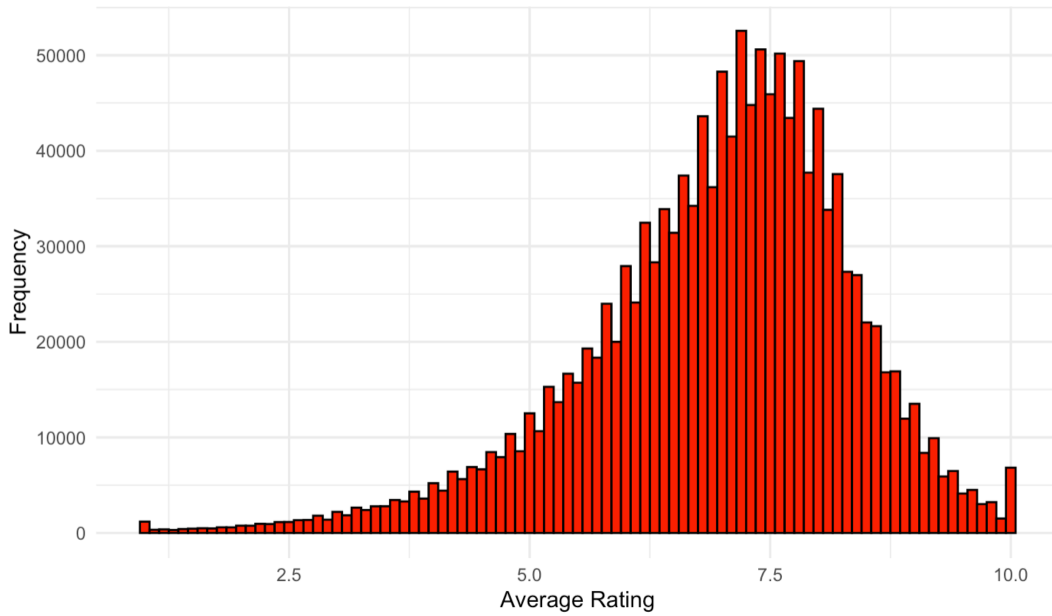


Fig 13:Distribution of Average ratings across all movies

Here, most of the movies are rated around 7.5

Let's check the outliers for the averageRating feature.

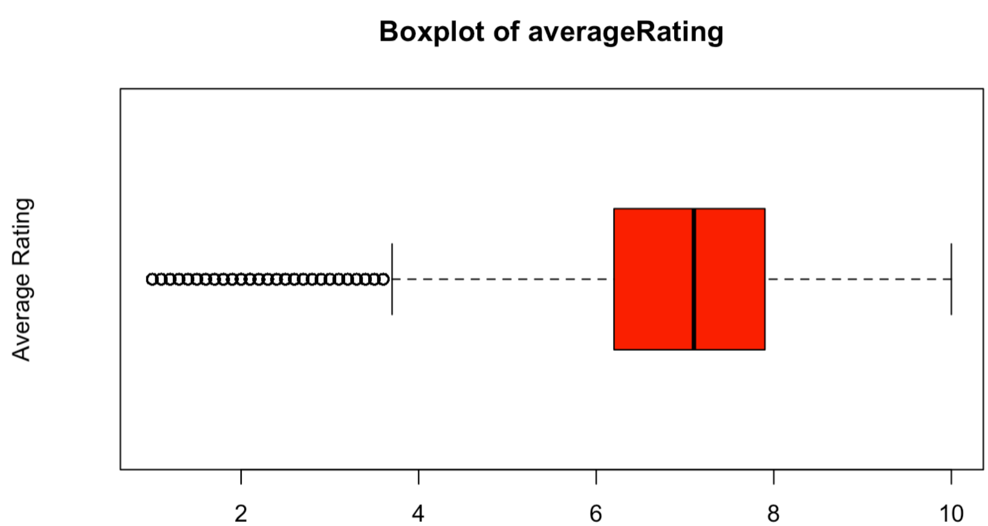


Fig 14:Boxplot of averageRating feature

Here, there are very less number of outliers. Most of the ratings in between 6 to 8

Let's know which genres receive the highest number of votes on average? This analysis tells which genre is most watched by the audience.

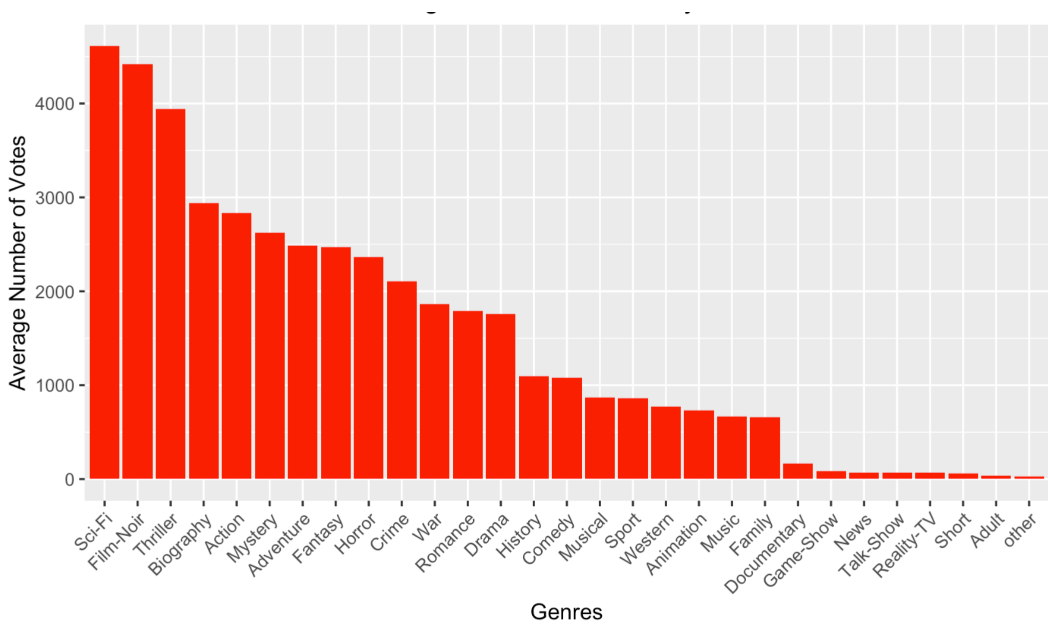


Fig 15: Average number of votes received by each genre

From the graph, we can say that, on average, Sci-Fi genre received highest number of votes (4610) and other genre received lowest number of votes (27).

Let's know what kind of movies are most produced in yearly based. Typically, different genres tend to trend each year. This analysis enables us to discern which genres have garnered the most hype over time.

startYear <int>	genres <chr>	count <int>
1874	Documentary	2
1877	Animation	4
1878	Short	3
1881	Short	2
1882	Documentary	2
1883	Documentary	1
1885	Animation	1
1887	Short	45
1888	Short	5
1889	Short	2

Fig 16: Kind of movies most produced in each year

The above given genres are most produced in corresponding year.

Let's analyze what kind of movies are most classified as adult content? This analysis enables us to know which genre contains most of the adult content.

genres <chr>	num_adult_movies <int>
Adult	21211
Drama	2277
Comedy	2042
Romance	1897
Crime	643
Fantasy	609
Animation	431
Short	406
other	362
Horror	330

1–10 of 27 rows

Fig 17: Kind of movies most classified as adult content

From the above data, we can say that most classified movies as adult content are belongs to adult genre

Now let's know the top-rated movies based on the average rating and number of votes.

```
##       tconst titleType isAdult startYear runtimeMinutes                 genres
## 1  tt2301451 tvEpisode       0      2013             47 Crime, Drama, Thriller
## 2 tt30643438     short       0      2023              2                  Short
## 3 tt29902774 tvEpisode       0      2021             37        News, Talk-Show
## 4 tt13688764 tvEpisode       0      2020             37
## 5 tt31029309     movie       0      2024            102            Documentary
## 6 tt29466076     short       0      2021             13                  Short
##   averageRating numVotes
## 1            10   212163
## 2            10     1153
## 3            10      986
## 4            10      961
## 5            10      769
## 6            10      742
```

Fig 18: Top – 10 rated movies based on average rating and number of votes

Correlation: It measures the strength and direction of the relationship between two variables. A correlation value close to 1 indicates a strong positive correlation, while a value close to -1 indicates a strong negative correlation. A value near 0 suggests little to no linear relationship between the variables. It helps in understanding how changes in one variable affect the other, aiding feature selection and model interpretability.

In the realm of movies, the runtime is a significant factor that impacts movie ratings. For instance, some viewers prefer shorter films as lengthy ones can lead to boredom and result in lower ratings. So let's analyze the correlation between runtime and average ratings.

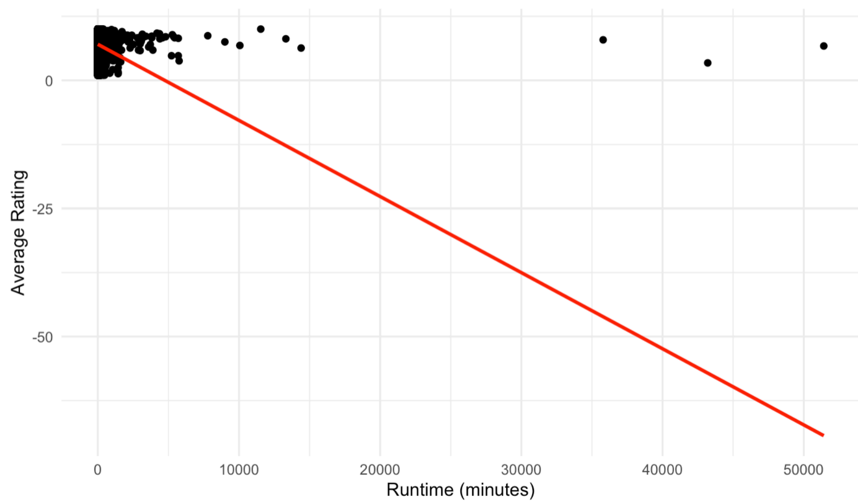


Fig 19: Correlation between runtime and average ratings

The above graph shows the correlation between runtime and average ratings. We got the correlation -0.08715879. It says this feature is negatively corelated.
Now let’s analyze the correlation between average rating and number of votes. This helps us to know how votes impact the ratings.

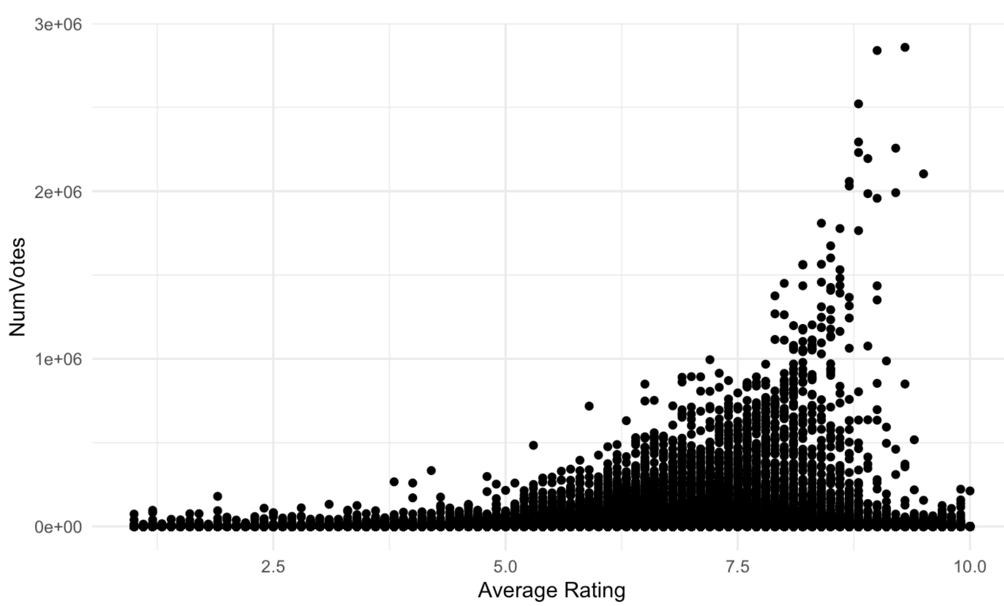


Fig 20: Correlation between average ratings and number of votes

The above graph shows the correlation between number of votes and average ratings. We got the correlation 0.01041723. It says this feature is positively corelated.

Model development:

Now the data is clean and ready for the model development. Created training and testing datasets by using createDataPartition function based on the "averageRating" column using an 80/20 split. The training set contains 80% of the data, while the testing set contains the remaining 20%.

```
print(nrow(training_data))
print(nrow(testing_data))
```

```
[1] 1122989
[1] 280747
```

Fig 21: Size of training and testing dataset

And further divided these training and testing datasets into x_train, y_train, x_test and y_test.

x_train:

	titleType <int>	isAdult <dbl>	startYear <int>	runtimeMinutes <dbl>	genres <int>	numVotes <int>
1	2	0	1894	1	7	2024
2	2	0	1892	5	3	272
3	2	0	1892	4	3	1962
5	2	0	1893	1	5	2727
6	2	0	1894	1	28	184
7	2	0	1894	1	28	847

Fig 22: x_train dataset

y_train:

```
head(y_train)
```

```
[1] 5.7 5.7 6.5 6.2 5.0 5.4
```

Fig 23: y_train dataset

x_test:

	titleType <int>	isAdult <dbl>	startYear <int>	runtimeMinutes <dbl>	genres <int>	numVotes <int>
4	2	0	1892	12	3	178
10	2	0	1895	1	7	7449
17	2	0	1895	1	7	339
21	2	0	1895	1	7	1127
23	2	0	1895	1	18	126
43	2	0	1896	1	28	49

Fig 24: x_test dataset

y_test:

```
head(y_test)
```

```
[1] 5.4 6.8 4.6 5.1 3.9 4.0
```

Fig 25: y_test dataset

Usually, predicting ratings is a regression problem. Upon analyzing the ratings in this dataset, it was observed that they range between 0.0 and 10.0, with a total of 100 distinct values for the output feature (averageRating). Consequently, this dataset could also be regarded as a classification problem. Therefore, we have built three models in this project. They are linear regression, KNN regression, and decision tree.

Linear regression model:
Multiple linear regression is a statistical technique used to model the relationship between multiple independent variables and a single dependent variable. It extends simple linear regression by allowing for the analysis of more complex relationships involving multiple predictors. The model assumes a linear relationship between the independent variables and the dependent variable, aiming to predict the latter based on the former while minimizing the sum of squared differences between observed and predicted values.

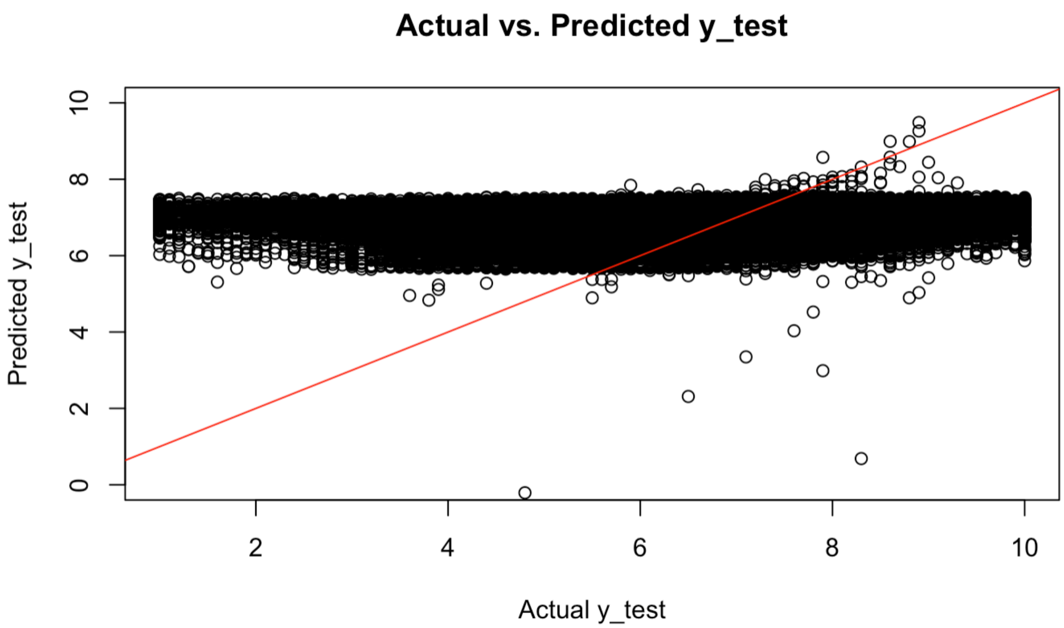


Fig 26: Linear regression model

KNN regression model:
K-Nearest Neighbors (KNN) regression is a non-parametric algorithm used for regression tasks. It predicts the value of a continuous target variable by averaging the values of its k nearest neighbors. The algorithm computes distances between data points and selects the k closest neighbors based on a chosen distance metric, typically Euclidean distance. The predicted value is then determined by averaging the target variable values of these neighbors.

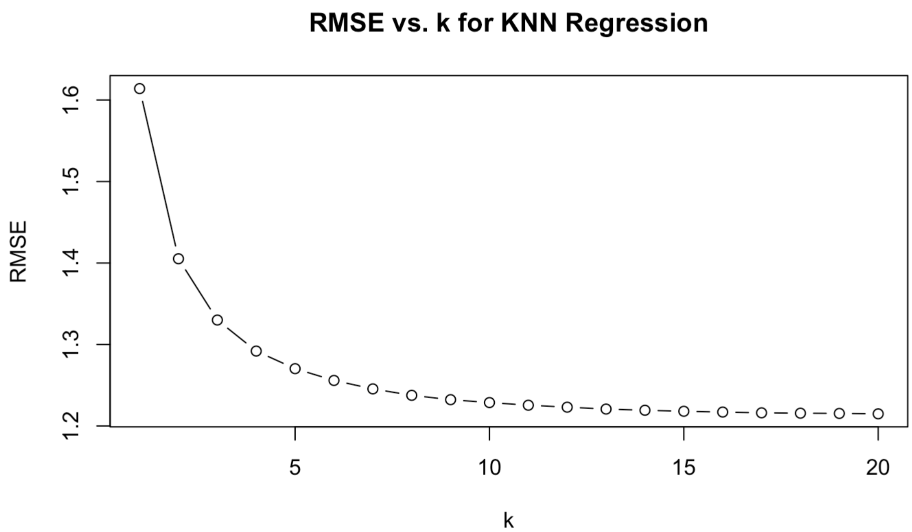


Fig 27: RMSE vs K value for KNN regression

Selecting an optimal value for k in KNN is crucial for achieving the right balance between bias and variance. Lower values of k may lead to overfitting, capturing noise in the data, while higher values of k may result in underfitting, oversimplifying the model. After evaluating the model's performance for various k values, we observed that the RMSE values plateaued after k = 10. This indicates that increasing the value of k beyond 10 does not significantly improve the model's performance. It suggests that the model starts to overfit the data beyond k = 10. Therefore, based on this analysis, we have chosen k = 10 as the optimal value for our KNN model. This value strikes a balance between capturing the underlying patterns in the data and avoiding overfitting.

Decision tree model:

Decision tree is a supervised learning algorithm used for both classification and regression tasks. It recursively partitions the data into subsets based on features that best separate the target variable. Each split aims to maximize information gain or minimize impurity, resulting in a tree-like structure where leaf nodes represent the final predicted outcomes. Decision trees are interpretable and can handle both numerical and categorical data.

```
{r}
summary(tree_model)

Call:
rpart(formula = y_train ~ ., data = x_train)
  n= 1122989

          CP nsplit rel error    xerror        xstd
1 0.08852954      0 1.0000000 1.0000020 0.001672211
2 0.02154786      1 0.9114705 0.9114743 0.001634129
3 0.01926800      2 0.8899226 0.8899288 0.001592396
4 0.01000000      3 0.8706546 0.8707121 0.001586265
```

Fig 28: Summary of decision tree model

Model Evaluation:
In this stage, we evaluated the performance of our models. We used a variety of evaluation metrics, such as Mean Absolute Error, Mean squared error (loss function), Root Mean Squared Error, and R-squared to assess the performance of our models

For Linear regression model,

```
Mean Absolute Error (MAE): 1.029465
Mean Squared Error (MSE): 1.82996
Root Mean Squared Error (RMSE): 1.35276
R-squared: 0.04318821
```

Fig 29: Evaluation metrics for linear regression model

For KNN regression model,

```
Mean Absolute Error (MAE): 0.9068456
Mean Squared Error (MSE): 1.509794
Root Mean Squared Error (RMSE): 1.228737
R-squared: 0.2105901
```

Fig 30: Evaluation metrics for KNN regression model

For Decision tree model,

```
Mean Absolute Error (MAE): 0.9740413
Mean Squared Error (MSE): 1.665022
Root Mean Squared Error (RMSE): 1.290357
R-squared: 0.1294278
```

Fig 31: Evaluation metrics for decision tree model

Based on these metrics, the KNN Regression Model appears to perform the best among the three models. Because,

- Lower Error Metrics: The KNN Regression Model has the lowest values of MAE, MSE, and RMSE compared to the Linear Regression and Decision Tree models. This indicates that the KNN model has smaller errors in prediction, suggesting better accuracy and precision.
- Higher R-squared: The KNN Regression Model also has the highest R-squared value among the three models. R-squared measures the proportion of the variance in the dependent variable that is predictable from the independent variables. A higher R-squared value (closer to 1) indicates a better fit of the model to the data. Therefore, the KNN model explains more variance in the dependent variable compared to the other models.

- Overall Performance: While the Decision Tree Model has comparable performance in terms of MAE and RMSE, it has a lower R-squared value compared to the KNN model. The Linear Regression Model performs the worst among the three models across all metrics.

In summary, based on the provided evaluation metrics, the KNN Regression Model is the best choice for this dataset as it demonstrates superior accuracy, precision, and explanatory power compared to the Linear Regression and Decision Tree models.

Results

Based on the KNN regression model, here are the recommended the top 10 movies based on predicted ratings

	originalTitle <chr>	predicted_ratings <dbl>
283831	Along for the Ride	10
321748	The Art of Biting	10
351916	Comenzando De Nuevo	10
351917	El Baile	10
391103	Jalkapuussa	10
547198	Bolum 1	10
558662	Fear and Falling in Montana	10
558664	Making Allowances	10
558666	Solar Mates	10
567757	Jennifer's Instinct; Sailors Angel; OR Miracle; High School Reunion	10

1–10 of 10 rows

Fig 32: Top 10 movies recommended by the model

Conclusion

In conclusion, this project successfully developed an efficient recommendation system for movies based on ratings analysis using the IMDB dataset. Through comprehensive data exploration and preparation, key insights into movie genres, ratings, and audience preferences were uncovered. Utilizing machine learning models including linear regression, KNN regression, and decision tree, we demonstrated the effectiveness of the KNN regression model in predicting movie ratings. The findings provide valuable guidance for streaming platform companies in understanding factors contributing to movie success and making informed decisions regarding future acquisitions, ultimately enhancing user experience, and driving business growth in the competitive landscape of online streaming platforms.

Source Code

We have successfully implemented this project and uploaded in GitHub. This project is available in GitHub. Here's the link,

Link – https://github.com/PenumarthiSushmanth/Movie-Recommendation-system

References

[1] C. -S. M. Wu, D. Garg and U. Bhandary, "Movie Recommendation System Using Collaborative Filtering," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 11-15 - https://ieeexplore.ieee.org/abstract/document/8663822

[2] "Content-Based Movie Recommendation System Using Genre Correlation" Smart Intelligent Computing and Applications, 2019, Volume 105 ISBN : 978-981-13-1926-6 SRS Reddy, Sravani Nalluri, Subramanyam Kunisetti, S. Ashok, B. Venkatesh - https://link.springer.com/chapter/10.1007/978-981-13-1927-3_42

[3] "Modeling and prediction for movies" Susan Li - https://susanli2016.github.io/Modeling-Prediction-Movies/

[4] "linear regression model R documentation" - https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/lm

[5] "KNN regression model R documentation" - https://www.rdocumentation.org/packages/FNN/versions/1.1.4/topics/knn.reg

[6] "Decision Trees in Machine Learning Using R" - https://www.datacamp.com/tutorial/decision-trees-R

```
<!--
DPA Project - Movie Recommendation System

Group members:
--------------
Mohana uma sushmanth Penumarthi - A20525576
Vinnapala Sai Ramya - A20526253
Harini Vaidya - A20525926
-->
```

Importing Libraries

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(caret)
```

```
## Loading required package: lattice
```

```
library(FNN)
library(rpart)
library(ROCR)
library(class)
```

```
##
## Attaching package: 'class'
```

```
## The following objects are masked from 'package:FNN':
##
##     knn, knn.cv
```

Importing Dataset

```
# Loading movies dataset
movies_dataset <- read.table("title.basics.tsv",
                   header = TRUE,
                   sep = "\t",
                   quote = "",
                   na.strings = "\\N",
                   stringsAsFactors = FALSE,
                   fill = TRUE)
```

```
head(movies_dataset)
```

```
##      tconst titleType           primaryTitle          originalTitle isAdult
## 1 tt0000001     short             Carmencita             Carmencita       0
## 2 tt0000002     short Le clown et ses chiens Le clown et ses chiens       0
## 3 tt0000003     short         Pauvre Pierrot         Pauvre Pierrot       0
## 4 tt0000004     short            Un bon bock            Un bon bock       0
## 5 tt0000005     short       Blacksmith Scene       Blacksmith Scene       0
## 6 tt0000006     short      Chinese Opium Den      Chinese Opium Den       0
##   startYear endYear runtimeMinutes                   genres
## 1      1894      NA              1        Documentary,Short
## 2      1892      NA              5          Animation,Short
## 3      1892      NA              4 Animation,Comedy,Romance
## 4      1892      NA             12          Animation,Short
## 5      1893      NA              1             Comedy,Short
## 6      1894      NA              1                    Short
```

```
# Summary Statistics of movies dataset
summary(movies_dataset)
```

```
##     tconst           titleType         primaryTitle       originalTitle     
##  Length:10565899    Length:10565899    Length:10565899    Length:10565899   
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                            
##                                                                            
##                                                                            
##                                                                            
##     isAdult          startYear         endYear         runtimeMinutes   
##  Min.   :0         Min.   :1874      Min.   :1906      Min.   :    0    
##  1st Qu.:0         1st Qu.:2003      1st Qu.:1999      1st Qu.:   15    
##  Median :0         Median :2013      Median :2013      Median :   30    
##  Mean   :0         Mean   :2006      Mean   :2007      Mean   :   45    
##  3rd Qu.:0         3rd Qu.:2018      3rd Qu.:2019      3rd Qu.:   60    
##  Max.   :1         Max.   :2031      Max.   :2030      Max.   :54321    
##  NA's   :3574702   NA's   :3857911   NA's   :10447068   NA's   :8050542  
##     genres         
##  Length:10565899   
##  Class :character  
##  Mode  :character  
##                    
##                    
##                    
##                    
```

```
# Structure of the movies dataset
str(movies_dataset)
```

```
## 'data.frame':    10565899 obs. of  9 variables:
##  $ tconst        : chr  "tt0000001" "tt0000002" "tt0000003" "tt0000004" ...
##  $ titleType     : chr  "short" "short" "short" "short" ...
##  $ primaryTitle  : chr  "Carmencita" "Le clown et ses chiens" "Pauvre Pierrot" "Un bo
n bock" ...
##  $ originalTitle : chr  "Carmencita" "Le clown et ses chiens" "Pauvre Pierrot" "Un bo
n bock" ...
##  $ isAdult       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ startYear     : int  1894 1892 1892 1892 1893 1894 1894 1894 1894 1895 ...
##  $ endYear       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ runtimeMinutes: int  1 5 4 12 1 1 1 1 45 1 ...
##  $ genres        : chr  "Documentary,Short" "Animation,Short" "Animation,Comedy,Roman
ce" "Animation,Short" ...
```

```
# Loading ratings dataset
ratings_dataset <- read.table("title.ratings.tsv", header = TRUE, fill = TRUE)
```

```
head(ratings_dataset)
```

```
##      tconst averageRating numVotes
## 1 tt0000001           5.7     2024
## 2 tt0000002           5.7      272
## 3 tt0000003           6.5     1962
## 4 tt0000004           5.4      178
## 5 tt0000005           6.2     2727
## 6 tt0000006           5.0      184
```

```
# Summary Statistics of ratings dataset
summary(ratings_dataset)
```

```
##     tconst          averageRating       numVotes
##  Length:1403737     Min.   : 1.000   Min.   :      5
##  Class :character   1st Qu.: 6.200   1st Qu.:     11
##  Mode  :character   Median : 7.100   Median :     26
##                     Mean   : 6.956   Mean   :   1037
##                     3rd Qu.: 7.900   3rd Qu.:    101
##                     Max.   :10.000   Max.   :2858177
```

```
# Structure of the ratings dataset
str(ratings_dataset)
```

```
## 'data.frame':    1403737 obs. of  3 variables:
##  $ tconst       : chr  "tt0000001" "tt0000002" "tt0000003" "tt0000004" ...
##  $ averageRating: num  5.7 5.7 6.5 5.4 6.2 5 5.4 5.4 5.3 6.8 ...
##  $ numVotes     : int  2024 272 1962 178 2727 184 847 2172 209 7449 ...
```

Droping unnecessary columns

```
final_movies_dataset <- subset(movies_dataset, select = -c(primaryTitle, originalTitle,
endYear))
head(final_movies_dataset)
```

```
##      tconst titleType isAdult startYear runtimeMinutes                  genres
## 1 tt0000001     short       0      1894              1       Documentary,Short
## 2 tt0000002     short       0      1892              5         Animation,Short
## 3 tt0000003     short       0      1892              4 Animation,Comedy,Romance
## 4 tt0000004     short       0      1892             12         Animation,Short
## 5 tt0000005     short       0      1893              1            Comedy,Short
## 6 tt0000006     short       0      1894              1                   Short
```

Merging two datasets

```
final_dataset <- merge(final_movies_dataset, ratings_dataset, by = "tconst", all.x = TRU
E)
```

```
head(final_dataset)
```

```
##      tconst titleType isAdult startYear runtimeMinutes                   genres
## 1 tt0000001     short       0      1894              1       Documentary,Short
## 2 tt0000002     short       0      1892              5         Animation,Short
## 3 tt0000003     short       0      1892              4 Animation,Comedy,Romance
## 4 tt0000004     short       0      1892             12         Animation,Short
## 5 tt0000005     short       0      1893              1            Comedy,Short
## 6 tt0000006     short       0      1894              1                   Short
##   averageRating numVotes
## 1           5.7     2024
## 2           5.7      272
## 3           6.5     1962
## 4           5.4      178
## 5           6.2     2727
## 6           5.0      184
```

Handling null values

```
null_counts <- sapply(final_dataset, function(x) sum(is.na(x)))
print(null_counts)
```

```
##         tconst      titleType        isAdult      startYear runtimeMinutes
##              0              0        3574702        3857911        8050542
##         genres  averageRating       numVotes
##         291312        9162163        9162163
```

Null values in isAdult

```
#Considering those null values as not adult movies
final_dataset$isAdult <- ifelse(is.na(final_dataset$isAdult), 0, final_dataset$isAdult)
```

Null values in startYear

```
#Filling the null values in startYear field with the previous non-null entry value. Cons
idering the movie is relased in same year
final_dataset$startYear <- na.locf(final_dataset$startYear)
```

Null values in runtimeMinutes

```
# Remove rows with missing runtime values
temp_ds <- final_dataset[!is.na(final_dataset$runtimeMinutes), ]

# Calculate average runtime for each title type
average_runtime <- tapply(temp_ds$runtimeMinutes, temp_ds$titleType, mean)

# Convert average runtime to integer
average_runtime <- round(average_runtime)

# Replacing null values with the average value of corresponding titletype

for(tt in unique(final_dataset$titleType)) {
  null_indices <- is.na(final_dataset$runtimeMinutes) & final_dataset$titleType == tt
  final_dataset$runtimeMinutes[null_indices] <- average_runtime[tt]
}
```

```
# Check for null values in the runtimeMinutes column
null_indices <- which(is.na(final_dataset$runtimeMinutes))

# Print rows with null values in the runtimeMinutes column
print(final_dataset[null_indices, ])
```

```
##             tconst titleType isAdult startYear runtimeMinutes genres
## 3834477 tt15258334   tvPilot       0      1991             NA   <NA>
##         averageRating numVotes
## 3834477            NA       NA
```

```
unique_runtimes <- unique(final_dataset$titleType)
# Print the unique values
print(unique_runtimes)
```

```
##  [1] "short"        "movie"        "tvShort"      "tvMovie"      "tvSeries"
##  [6] "tvEpisode"    "tvMiniSeries" "tvSpecial"    "video"        "videoGame"
## [11] "tvPilot"
```

```
final_dataset <- final_dataset[final_dataset$titleType != "tvPilot", ]
```

Null values in AverageRating and nuumVotes

```
# Remove rows with null values in the averageRating column
final_dataset <- final_dataset[complete.cases(final_dataset$averageRating), ]
```

Null values in Genres

```
# Replacing null values with other in the genres column
final_dataset$genres[is.na(final_dataset$genres)] <- 'other,'
```

```
null_counts <- sapply(final_dataset, function(x) sum(is.na(x)))
print(null_counts)
```

```
##         tconst      titleType        isAdult      startYear runtimeMinutes
##              0              0              0              0              0
##         genres  averageRating       numVotes
##              0              0              0
```

```
num_rows <- nrow(final_dataset)

# Print the number of rows
print(num_rows)
```

```
## [1] 1403736
```

Data Exploration

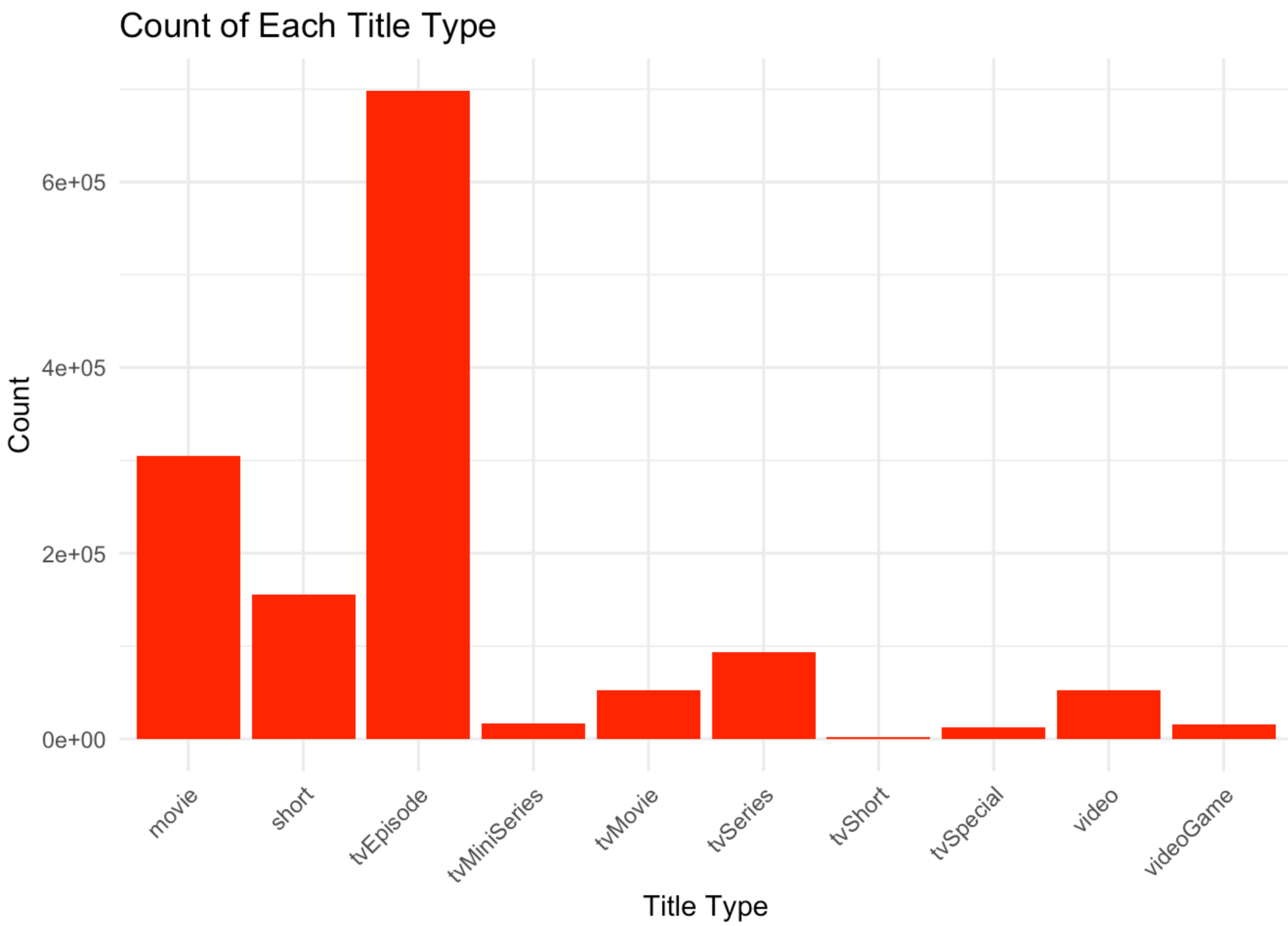
```
# Unique values of feature titleType
distinct_title_types <- unique(final_dataset$titleType)
print(distinct_title_types)
```

```
##  [1] "short"        "movie"        "tvShort"      "tvMovie"      "tvSeries"
##  [6] "tvEpisode"    "tvMiniSeries" "tvSpecial"    "video"        "videoGame"
```

```
# Count the occurrences of each value in the titleType column
title_type_counts <- as.data.frame(table(final_dataset$titleType))

print(title_type_counts)
```

```
##            Var1   Freq
## 1         movie 304790
## 2         short 155700
## 3     tvEpisode 698229
## 4  tvMiniSeries  16689
## 5       tvMovie  52263
## 6      tvSeries  93344
## 7       tvShort   2279
## 8     tvSpecial  12100
## 9         video  52437
## 10    videoGame  15905
```

```
# Plot the graph using ggplot2
ggplot(title_type_counts, aes(x = Var1, y = Freq)) +
  geom_bar(stat = "identity", fill = "red") +
  labs(title = "Count of Each Title Type",
       x = "Title Type",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

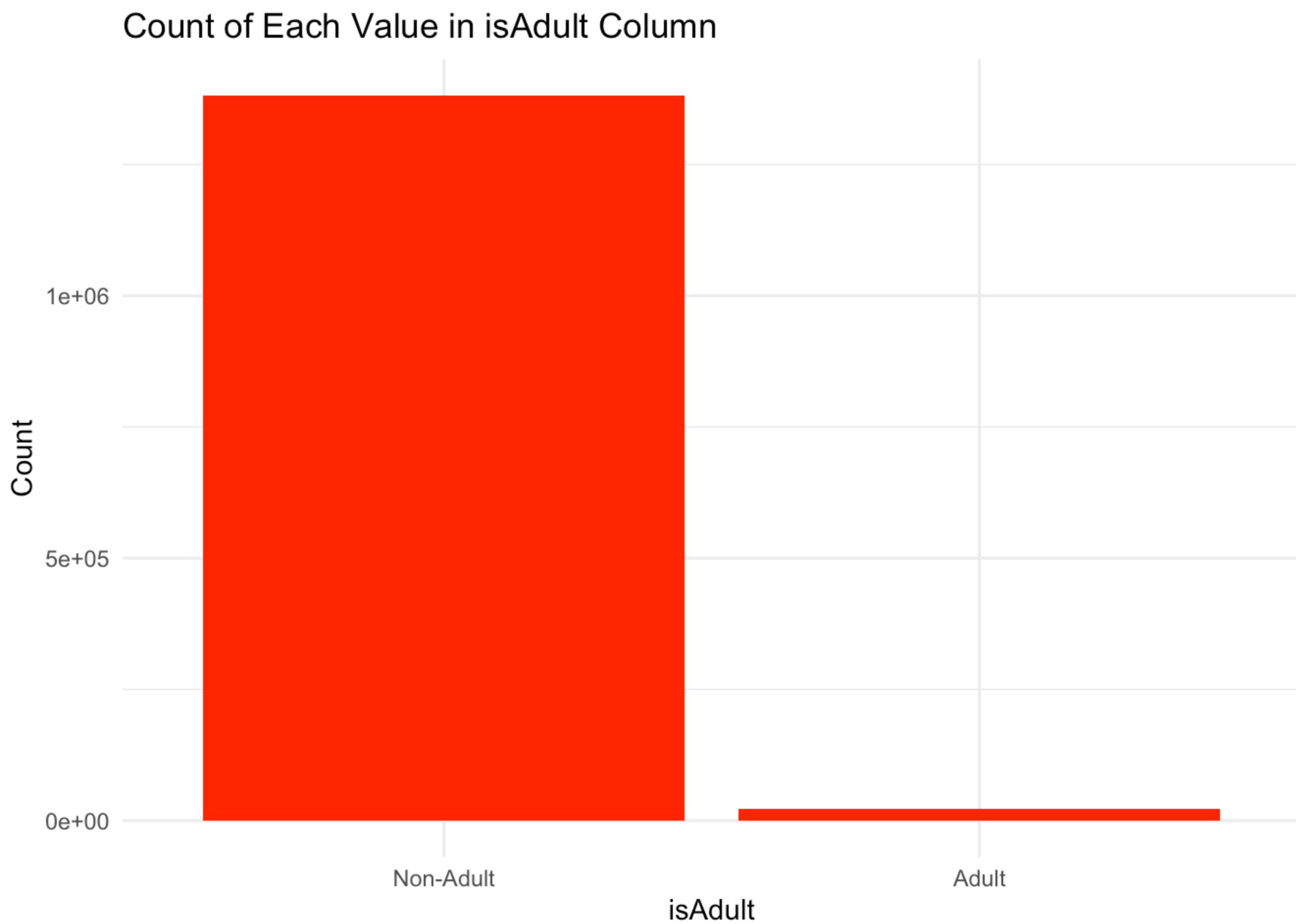


Most of the movies belongs to tvEpisodes

```
# Unique values of feature titleType
is_adult_counts <- as.data.frame(table(final_dataset$isAdult))

print(is_adult_counts)
```

```
##   Var1    Freq
## 1    0 1381769
## 2    1   21967
```

```
# Plot the graph using ggplot2
ggplot(is_adult_counts, aes(x = factor(Var1), y = Freq)) +
  geom_bar(stat = "identity", fill = "red") +
  labs(title = "Count of Each Value in isAdult Column",
       x = "isAdult",
       y = "Count") +
  scale_x_discrete(labels = c("Non-Adult", "Adult")) + theme_minimal()
```




```
# Split the strings in the genres column by comma and convert to list
final_dataset$genres <- strsplit(final_dataset$genres, ",")
```

```
head(final_dataset)
```

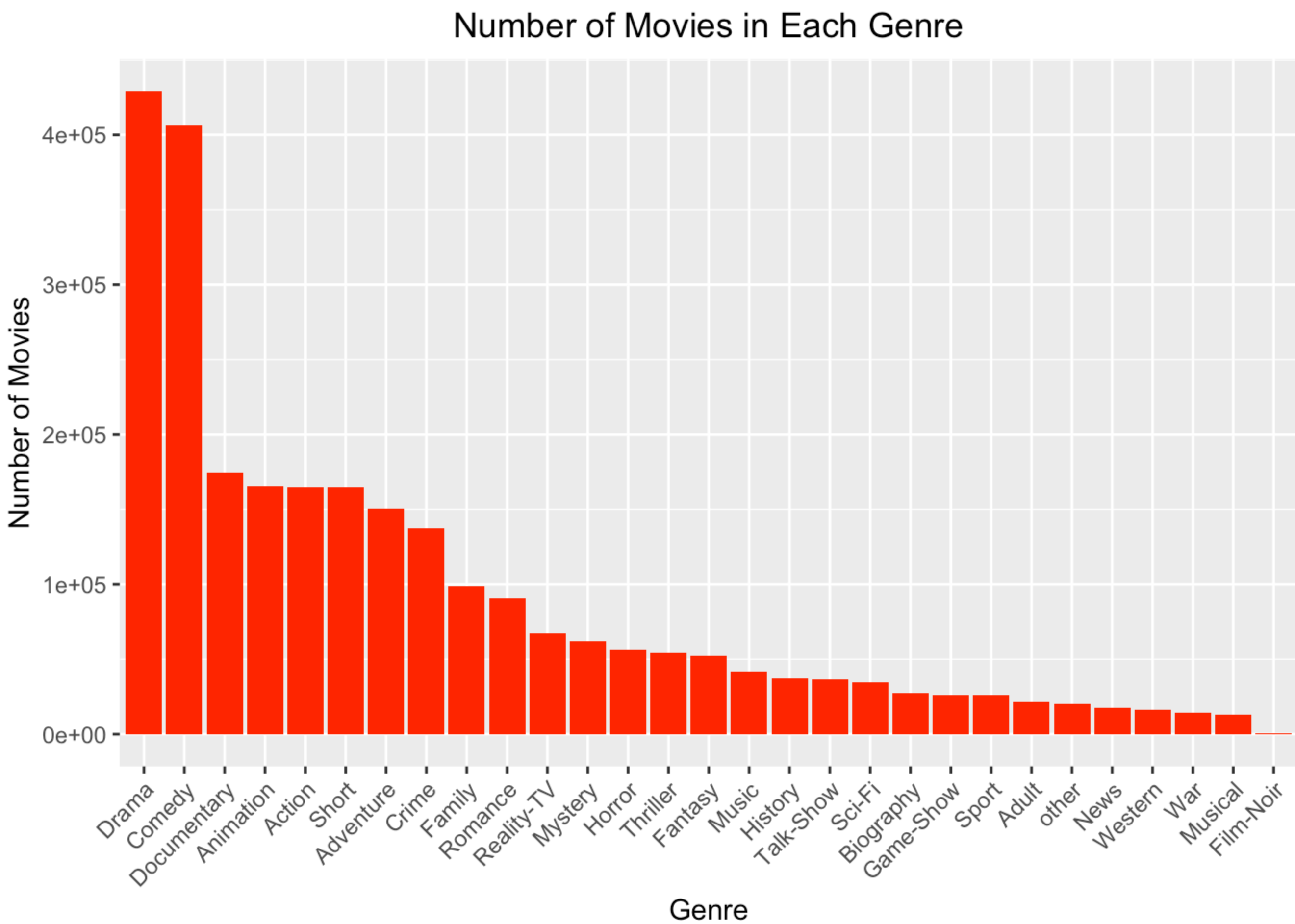
```
##      tconst titleType isAdult startYear runtimeMinutes
## 1 tt0000001     short       0      1894              1
## 2 tt0000002     short       0      1892              5
## 3 tt0000003     short       0      1892              4
## 4 tt0000004     short       0      1892             12
## 5 tt0000005     short       0      1893              1
## 6 tt0000006     short       0      1894              1
##                       genres averageRating numVotes
## 1          Documentary, Short           5.7     2024
## 2            Animation, Short           5.7      272
## 3 Animation, Comedy, Romance           6.5     1962
## 4            Animation, Short           5.4      178
## 5               Comedy, Short           6.2     2727
## 6                       Short           5.0      184
```

Number of movies in each genre

```
# Unnest the genres column to create separate rows for each genre
unnested_gendata <- final_dataset %>%
  unnest(genres)

# Count the number of movies in each genre
genre_counts <- unnested_gendata %>%
  group_by(genres) %>%
  summarise(num_movies = n()) %>%
  arrange(desc(num_movies))

print(genre_counts)
```

```
## # A tibble: 29 × 2
##    genres      num_movies
##    <chr>            <int>
##  1 Drama           429382
##  2 Comedy          406012
##  3 Documentary     174480
##  4 Animation       165657
##  5 Action          164897
##  6 Short           164682
##  7 Adventure       150133
##  8 Crime           137656
##  9 Family           98860
## 10 Romance          91074
## # ℹ 19 more rows
```

```
# Plot the graph
ggplot(genre_counts, aes(x = reorder(genres, -num_movies), y = num_movies)) +
  geom_bar(stat = "identity", fill = "red") +
  labs(title = "Number of Movies in Each Genre",
       x = "Genre",
       y = "Number of Movies") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))
```



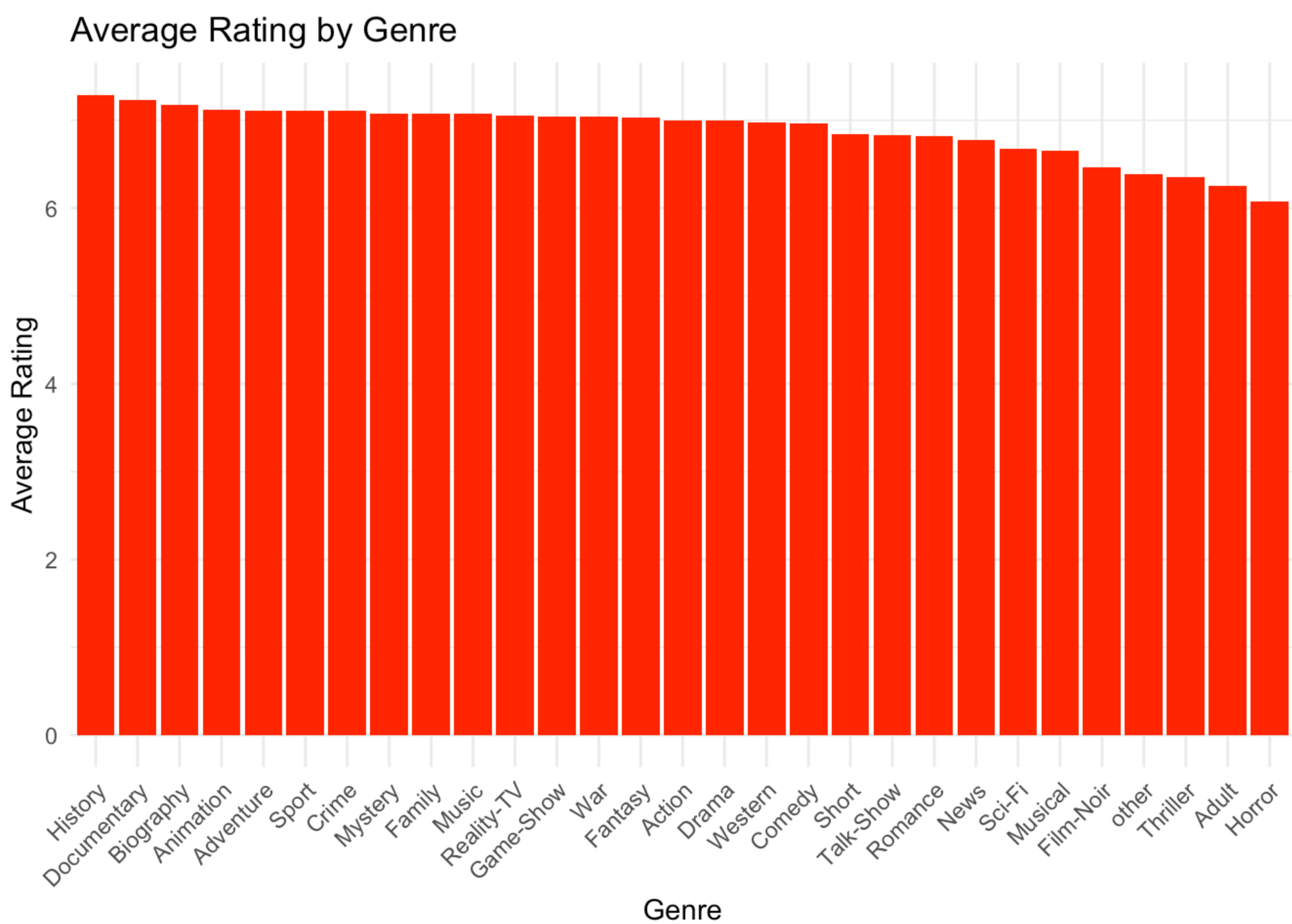
Most of the movies given in the dataset are belongs to Drama genre (429382) and least number of movies are Film-Noir (880)

Average ratings for each genres

```
# Group by genre and calculate the average rating
average_ratings_by_genre <- unnested_gendata %>%
  group_by(genres) %>%
  summarise(average_rating = mean(averageRating, na.rm = TRUE)) %>%
  arrange(desc(average_rating))

# Print the top-rated genres
print(average_ratings_by_genre)
```

```
## # A tibble: 29 × 2
##    genres      average_rating
##    <chr>                <dbl>
##  1 History               7.29
##  2 Documentary           7.24
##  3 Biography             7.18
##  4 Animation             7.12
##  5 Adventure             7.11
##  6 Sport                 7.11
##  7 Crime                 7.11
##  8 Mystery               7.08
##  9 Family                7.08
## 10 Music                 7.08
## # ℹ 19 more rows
```

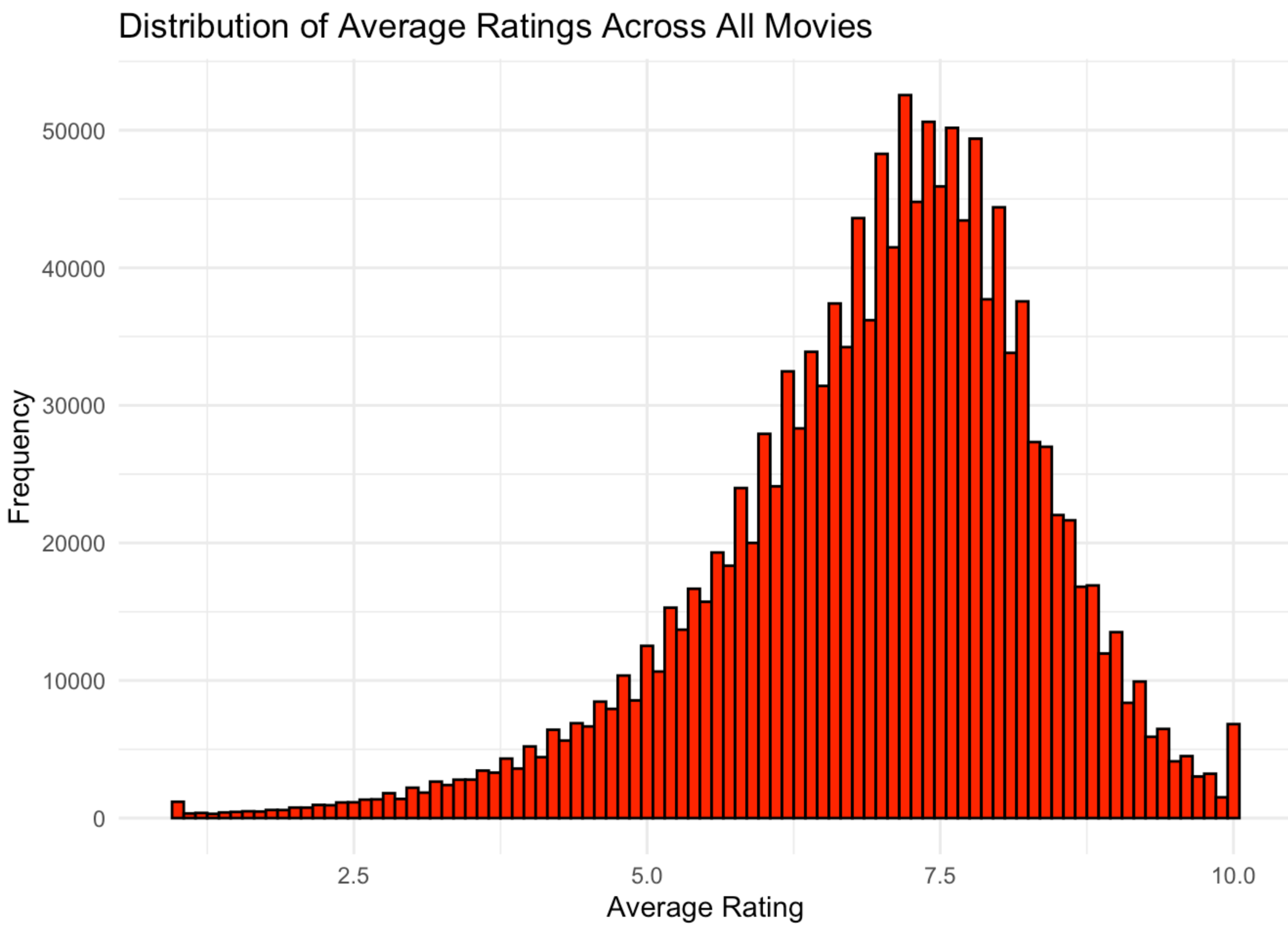
```
# Plot the graph
ggplot(average_ratings_by_genre, aes(x = reorder(genres, -average_rating), y = average_r
ating)) +
  geom_bar(stat = "identity", fill = "red") +
  labs(title = "Average Rating by Genre",
       x = "Genre",
       y = "Average Rating") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Here, the average rating for all the genres is around 6 to 7 Movies belongs to History genre got highest average rating (7.289976) and movies belongs to Horror genre got least average rating (6.080079)

Distribution of average ratings across all movies

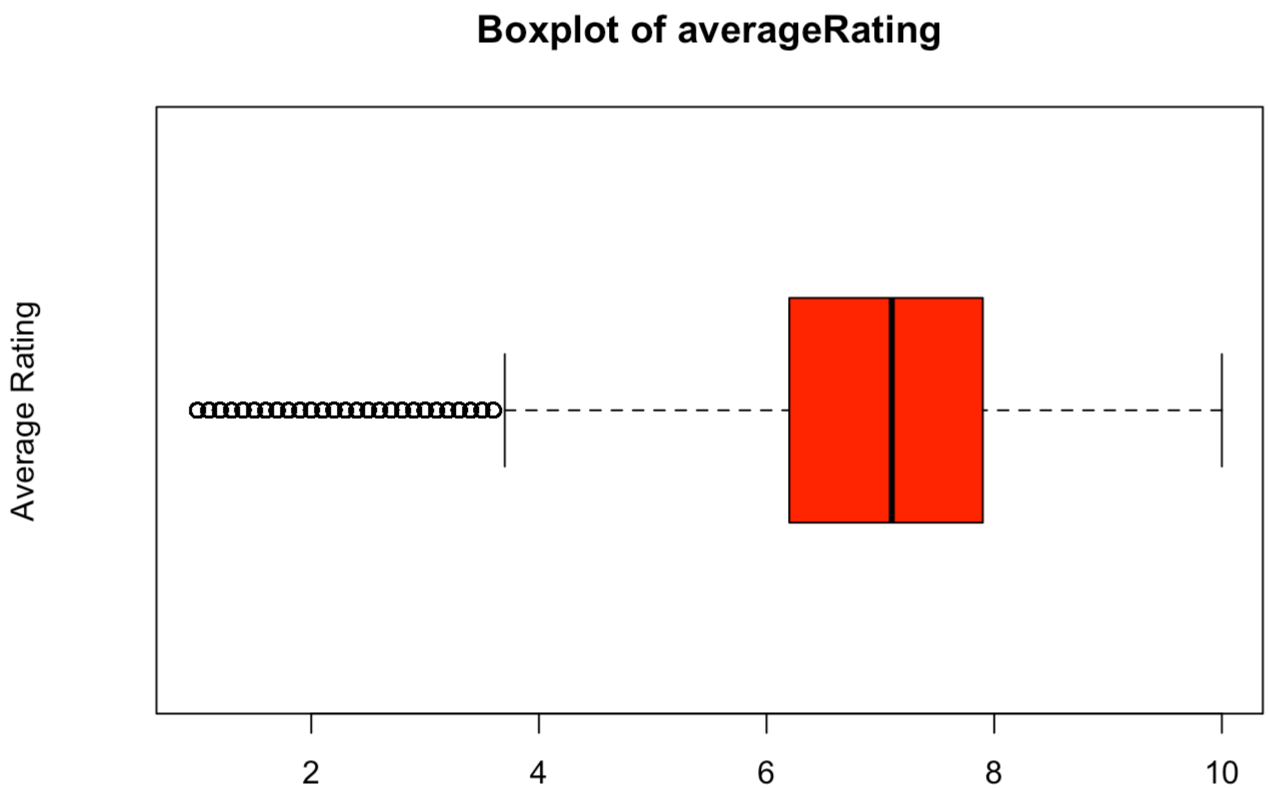
```
# Plot histogram of average ratings to distribution of average ratings across all movies
ggplot(final_dataset, aes(x = averageRating)) +
  geom_histogram(binwidth = 0.1, fill = "red", color = "black") +
  labs(title = "Distribution of Average Ratings Across All Movies",
       x = "Average Rating",
       y = "Frequency") +
  theme_minimal()
```



Here, most of the movies are rated around 7.5

Checking outliers for the averageRating feature

```
# Plot a boxplot for the averageRating feature
boxplot(final_dataset$averageRating,
        main = "Boxplot of averageRating",
        ylab = "Average Rating",
        col = "red",
        border = "black",
        horizontal = TRUE)
```



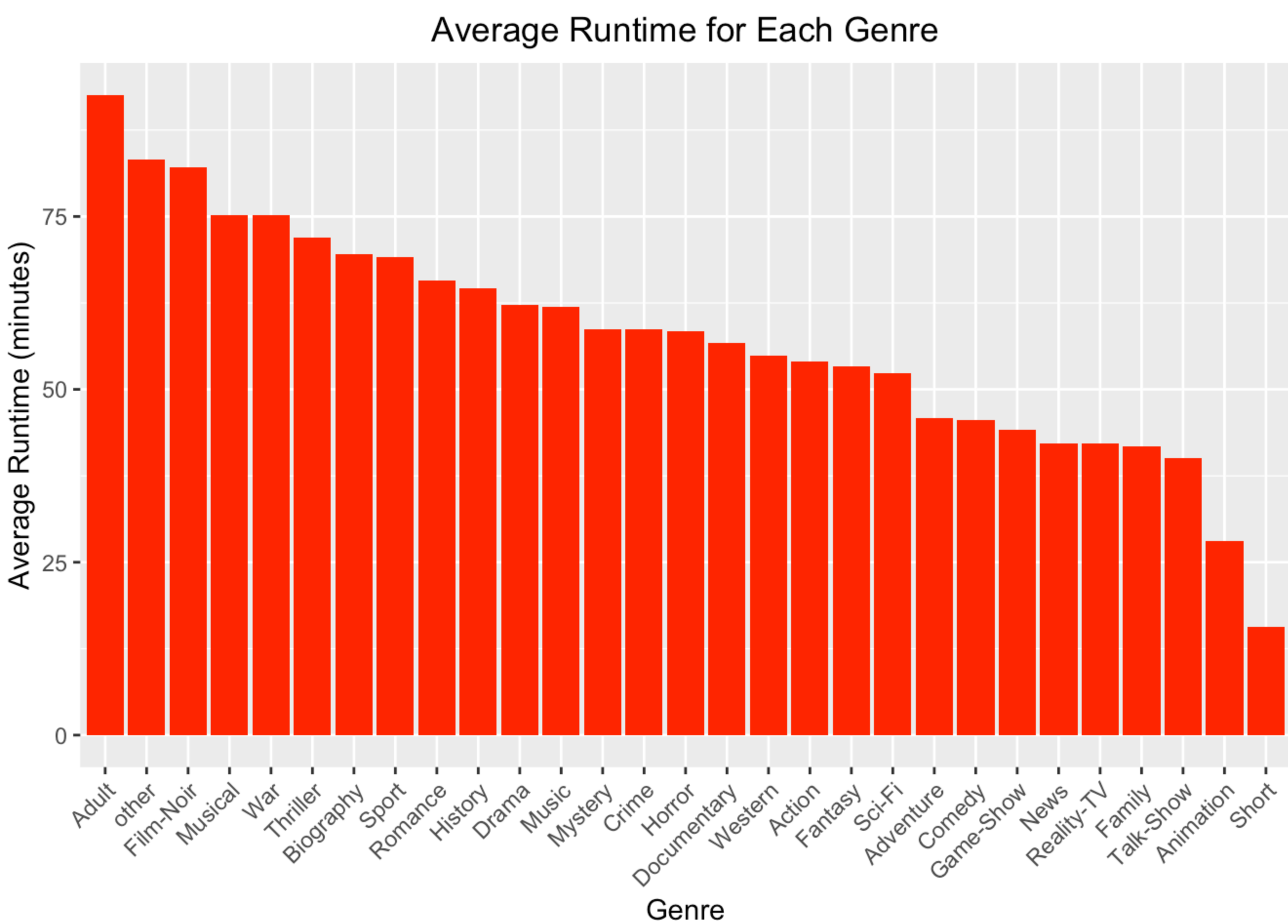
There are very less amount of outliers. Most of the ratings in between 6 to 8

Average Runtime for each genres

```
# Group by genre and calculate the average runtime
average_runtime_by_genre <- unnested_gendata %>%
  group_by(genres) %>%
  summarise(average_runtime = mean(runtimeMinutes, na.rm = TRUE)) %>%
  arrange(desc(average_runtime))

print(average_runtime_by_genre)
```

```
## # A tibble: 29 × 2
##    genres     average_runtime
##    <chr>                <dbl>
##  1 Adult                 92.6
##  2 other                 83.3
##  3 Film-Noir             82.1
##  4 Musical               75.2
##  5 War                   75.2
##  6 Thriller              72.0
##  7 Biography             69.5
##  8 Sport                 69.1
##  9 Romance               65.7
## 10 History               64.6
## # ℹ 19 more rows
```

```
# Plot graph
ggplot(average_runtime_by_genre, aes(x = reorder(genres, -average_runtime), y = average_
runtime)) +
  geom_bar(stat = "identity", fill = "red") +
  labs(title = "Average Runtime for Each Genre",
       x = "Genre",
       y = "Average Runtime (minutes)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))
```



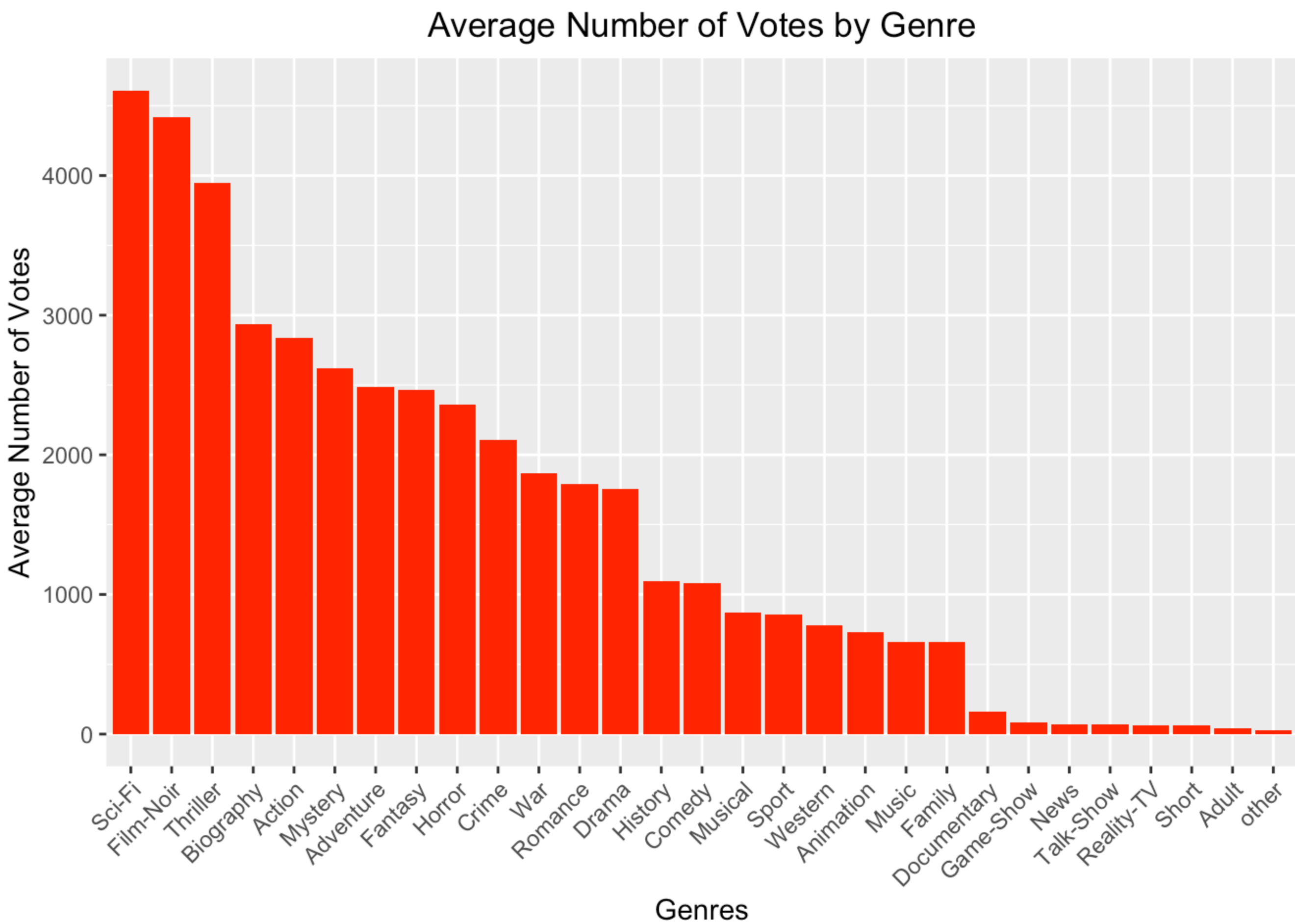
Longest average rumtime movies are belongs to Adult genre (92.60358 mins) and shortest average rumtime movies are belongs to short genre (15.66562 mins)

Which genres receive the highest number of votes on average?

```
# Group by genre and calculate the average number of votes
average_votes_by_genre <- unnested_gendata %>%
  group_by(genres) %>%
  summarise(average_votes = mean(numVotes))

# Arrange in descending order based on the average number of votes
average_votes_by_genre <- average_votes_by_genre %>%
  arrange(desc(average_votes))

average_votes_by_genre$average_votes = round(average_votes_by_genre$average_votes)
# Print the result
print(average_votes_by_genre)
```

```
## # A tibble: 29 × 2
##    genres    average_votes
##    <chr>             <dbl>
##  1 Sci-Fi             4610
##  2 Film-Noir          4417
##  3 Thriller           3944
##  4 Biography          2937
##  5 Action             2835
##  6 Mystery            2621
##  7 Adventure          2486
##  8 Fantasy            2467
##  9 Horror             2361
## 10 Crime              2109
## # ℹ 19 more rows
```

```
# Plot a graph
ggplot(average_votes_by_genre, aes(x = reorder(genres, -average_votes), y = average_vote
s)) +
  geom_bar(stat = "identity", fill = "red") +
  labs(title = "Average Number of Votes by Genre",
       x = "Genres",
       y = "Average Number of Votes") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))
```



On average, Sci-Fi genre recieved highest number of votes (4610) and other genre recieved lowest number of votes (27).

what kind of movies are most produced in yearly based

```
# Unnest the genres column to create separate rows for each genre
unnested_data <- final_dataset %>%
  unnest(genres)

# Group by startYear and genre, then count the occurrences
genre_counts <- unnested_data %>%
  group_by(startYear, genres, .drop = FALSE) %>%
  summarise(count = n(), .groups = "keep")

# Find the most produced genre for each year
most_produced_genre <- genre_counts %>%
  group_by(startYear) %>%
  slice(which.max(count)) %>%
  ungroup()

# Print the result
print(most_produced_genre)
```

```
## # A tibble: 145 × 3
##    startYear genres      count
##        <int> <chr>       <int>
##  1      1874 Documentary     2
##  2      1877 Animation       4
##  3      1878 Short           3
##  4      1881 Short           2
##  5      1882 Documentary     2
##  6      1883 Documentary     1
##  7      1885 Animation       1
##  8      1887 Short          45
##  9      1888 Short           5
## 10      1889 Short           2
## # ℹ 135 more rows
```

These are the genres that are most produced in each year

what kind of movies are most classified as adult content

```
# Count the occurrences of adult content for each genre
adult_genre_counts <- final_dataset %>%
  filter(isAdult == 1) %>%
  unnest(genres) %>%
  group_by(genres) %>%
  summarise(num_adult_movies = n()) %>%
  arrange(desc(num_adult_movies))

# Print the result
print(adult_genre_counts)
```

```
## # A tibble: 27 × 2
##    genres    num_adult_movies
##    <chr>                <int>
##  1 Adult                21211
##  2 Drama                 2277
##  3 Comedy                2042
##  4 Romance               1897
##  5 Crime                  643
##  6 Fantasy                609
##  7 Animation              431
##  8 Short                  406
##  9 other                  362
## 10 Horror                 330
## # ℹ 17 more rows
```

Most classified movies as adult content are belongs to Adult genre

Top-rated movies interms of avg rating & Numvotes

```
# Rank movies based on average rating and numVotes
top_rated_movies <- final_dataset %>%
  arrange(desc(averageRating), desc(numVotes)) %>%
  slice(1:10)  # Select the top 10 movies

# Print the top-rated movies
head(top_rated_movies)
```

```
##      tconst titleType isAdult startYear runtimeMinutes                  genres
## 1  tt2301451 tvEpisode       0      2013             47 Crime, Drama, Thriller
## 2 tt30643438     short       0      2023              2                  Short
## 3 tt29902774 tvEpisode       0      2021             37        News, Talk-Show
## 4 tt13688764 tvEpisode       0      2020             37                       
## 5 tt31029309     movie       0      2024            102            Documentary
## 6 tt29466076     short       0      2021             13                  Short
##   averageRating numVotes
## 1            10   212163
## 2            10     1153
## 3            10      986
## 4            10      961
## 5            10      769
## 6            10      742
```

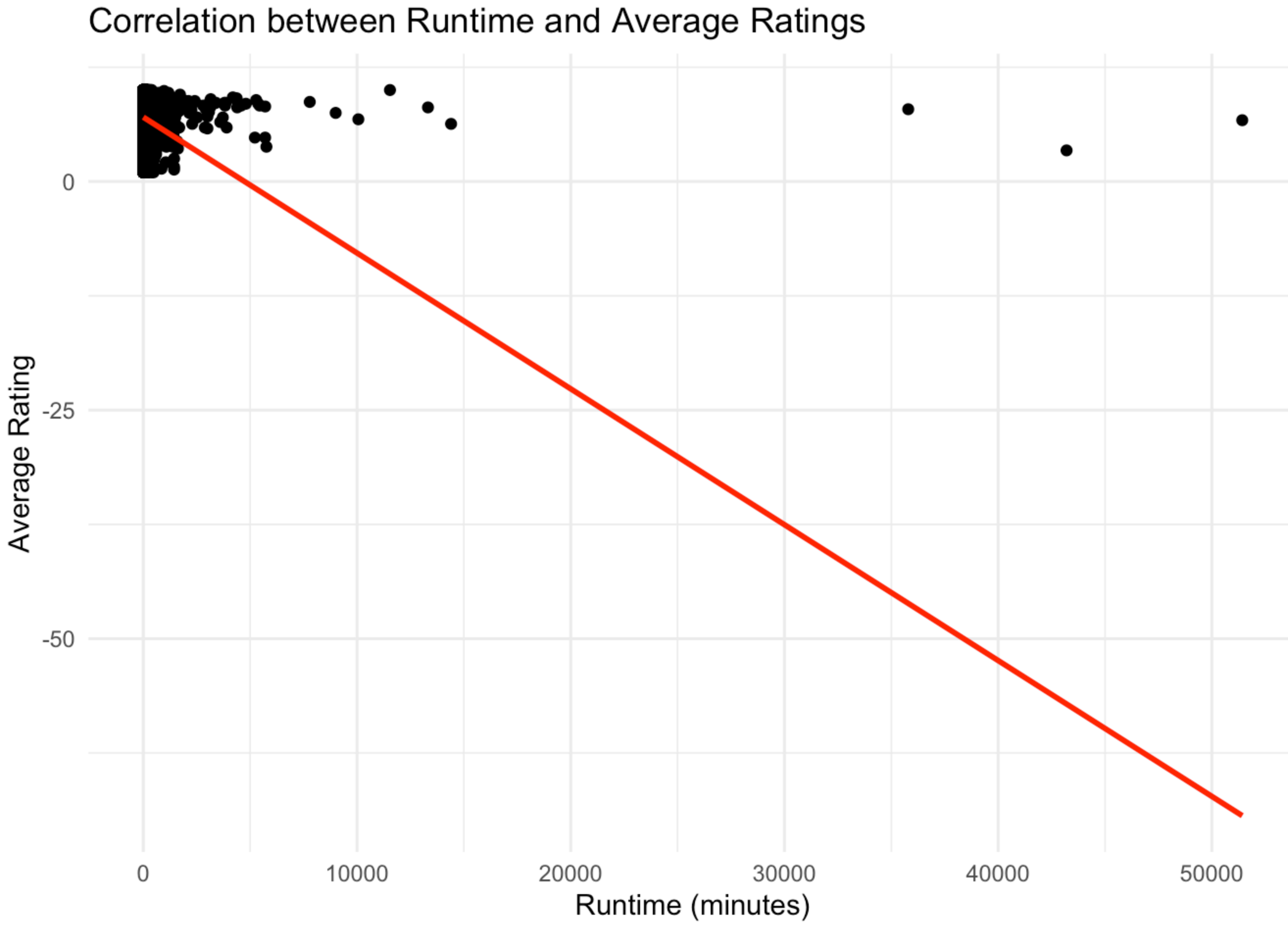
Correlation between runtime and average ratings

```
# Calculate correlation between runtime and average ratings
correlation <- cor(final_dataset$runtimeMinutes, final_dataset$averageRating)

# Print correlation
print(correlation)
```

```
## [1] -0.08715879
```

```
# Plot correlation graph
ggplot(final_dataset, aes(x = runtimeMinutes, y = averageRating)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Correlation between Runtime and Average Ratings",
       x = "Runtime (minutes)",
       y = "Average Rating") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



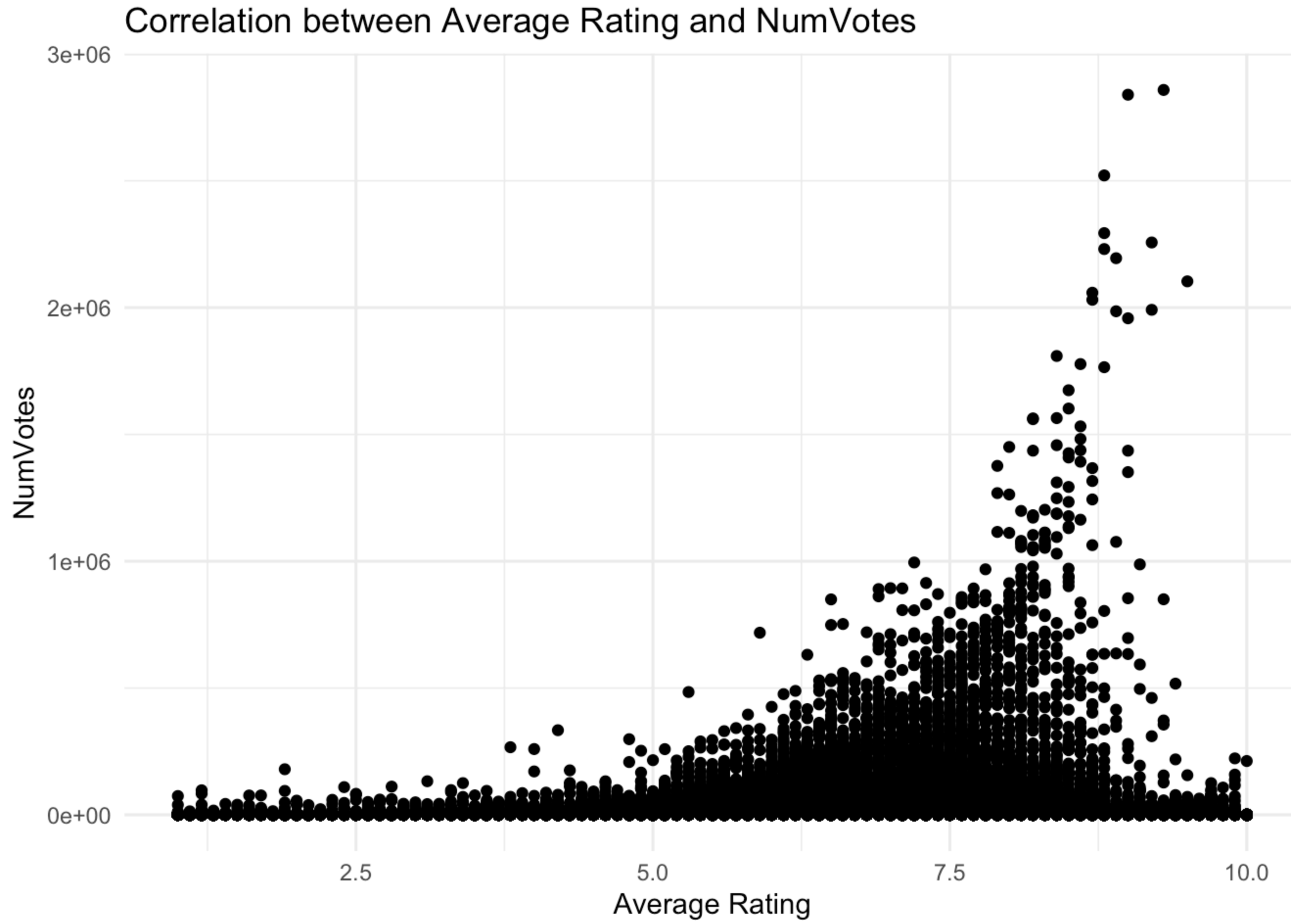
The features runtimeMinutes and averageRating are negatively corelated

Correlation between average rating and numVotes

```
# Calculate correlation between average rating and numVotes
correlation <- cor(final_dataset$averageRating, final_dataset$numVotes)

# Print correlation
print(correlation)
```

```
## [1] 0.01041723
```

```
# Plot correlation graph
ggplot(final_dataset, aes(x = averageRating, y = numVotes)) +
  geom_point() +
  labs(title = "Correlation between Average Rating and NumVotes",
       x = "Average Rating",
       y = "NumVotes") +
  theme_minimal()
```



The features numVotes and averageRating are positively corelated

Data preparation

```
head(final_dataset)
```

```
##      tconst titleType isAdult startYear runtimeMinutes
## 1 tt0000001     short       0      1894              1
## 2 tt0000002     short       0      1892              5
## 3 tt0000003     short       0      1892              4
## 4 tt0000004     short       0      1892             12
## 5 tt0000005     short       0      1893              1
## 6 tt0000006     short       0      1894              1
##                       genres averageRating numVotes
## 1         Documentary, Short           5.7     2024
## 2           Animation, Short           5.7      272
## 3 Animation, Comedy, Romance           6.5     1962
## 4           Animation, Short           5.4      178
## 5              Comedy, Short           6.2     2727
## 6                      Short           5.0      184
```

Converting all feature to integer datatype. Lets do feature encoding…

```
# Convert titleType to integer using factor
final_dataset$titleType <- as.integer(factor(final_dataset$titleType))
```

```
# Unnest the genres column to create a vector of all genres
all_genres <- unlist(final_dataset$genres)

# Get unique values of genres
unique_genres <- unique(all_genres)

# Print unique genres
print(unique_genres)
```

```
##  [1] "Documentary" "Short"       "Animation"   "Comedy"      "Romance"
##  [6] "Sport"       "News"        "Drama"       "Fantasy"     "Horror"
## [11] "Biography"   "Music"       "War"         "Crime"       "Western"
## [16] "Family"      "Adventure"   "Action"      "History"     "Mystery"
## [21] "other"       "Sci-Fi"      "Musical"     "Thriller"    "Film-Noir"
## [26] "Game-Show"   "Talk-Show"   "Reality-TV"  "Adult"
```

```
# Define a function to map genres to integers
genre_to_integer <- function(genre_list) {
  # Define a mapping of genres to integers
  genre_mapping <- c("Action" = 1, "Adventure" = 2, "Animation" = 3, "Biography" = 4,
                     "Comedy" = 5, "Crime" = 6, "Documentary" = 7, "Drama" = 8,
                     "Family" = 9, "Fantasy" = 10, "Film-Noir" = 11, "Game-Show" = 12,
                     "History" = 13, "Horror" = 14, "Music" = 15, "Musical" = 16,
                     "Mystery" = 17, "News" = 18, "Reality-TV" = 19, "Romance" = 20,
                     "Sci-Fi" = 21, "Sport" = 22, "Talk-Show" = 23, "Thriller" = 24,
                     "War" = 25, "Western" = 26, "Adult" = 27,  "Short" = 28, "other" =
29)

  # Map each genre to its corresponding integer value
  integer_list <- sapply(genre_list, function(genre) genre_mapping[genre])

  return(integer_list)
}

# Perform feature encoding on the genre feature
final_dataset$genres <- lapply(final_dataset$genre, genre_to_integer)
```

```
first_genre <- sapply(final_dataset$genre, function(x) ifelse(length(x) > 0, x[1], 29))

# Convert to factors for creating dummy variables
first_genre <- as.factor(first_genre)

temp <- final_dataset

final_dataset$genres <- as.integer(first_genre)
```

```
head(final_dataset)
```

```
##      tconst titleType isAdult startYear runtimeMinutes genres averageRating
## 1 tt0000001         2       0      1894              1      7           5.7
## 2 tt0000002         2       0      1892              5      3           5.7
## 3 tt0000003         2       0      1892              4      3           6.5
## 4 tt0000004         2       0      1892             12      3           5.4
## 5 tt0000005         2       0      1893              1      5           6.2
## 6 tt0000006         2       0      1894              1     28           5.0
##   numVotes
## 1     2024
## 2      272
## 3     1962
## 4      178
## 5     2727
## 6      184
```

```
unique_genres <- unique(final_dataset$genres)

print(unique_genres)
```

```
##  [1]  7  3  5 28 20 18  8 10 14  4 15  6  9  2  1 13 29 17 25 21 26 24 16 11 22
## [26] 12 23 27 19
```

Let’s save a copy of dataset for reccomendations

```
reccon_dataset <- final_dataset
```

```
head(reccon_dataset)
```

```
##      tconst titleType isAdult startYear runtimeMinutes genres averageRating
## 1 tt0000001         2       0      1894              1      7           5.7
## 2 tt0000002         2       0      1892              5      3           5.7
## 3 tt0000003         2       0      1892              4      3           6.5
## 4 tt0000004         2       0      1892             12      3           5.4
## 5 tt0000005         2       0      1893              1      5           6.2
## 6 tt0000006         2       0      1894              1     28           5.0
##   numVotes
## 1     2024
## 2      272
## 3     1962
## 4      178
## 5     2727
## 6      184
```

Here we don’t require the “tconst” feature. So lets drop it.

```r
# Drop tconst column using subset()
final_dataset <- subset(final_dataset, select = -tconst)
```

```r
head(final_dataset)
```

```
##   titleType isAdult startYear runtimeMinutes genres averageRating numVotes
## 1         2       0      1894              1      7           5.7     2024
## 2         2       0      1892              5      3           5.7      272
## 3         2       0      1892              4      3           6.5     1962
## 4         2       0      1892             12      3           5.4      178
## 5         2       0      1893              1      5           6.2     2727
## 6         2       0      1894              1     28           5.0      184
```

Now the data is ready for model development stage

Model development

```r
# Set the seed for reproducibility
set.seed(42)

# Determine the number of rows for the training set (80%)
train_size <- 0.8

# Create an index vector for partitioning the data
train_indices <- createDataPartition(final_dataset$averageRating, p = train_size, list =
FALSE)

# Create the training and testing sets
training_data <- final_dataset[train_indices, ]
testing_data <- final_dataset[-train_indices, ]
```

```r
print(nrow(training_data))
```

```
## [1] 1122989
```

```r
print(nrow(testing_data))
```

```
## [1] 280747
```

```r
x_train <- training_data[, !names(training_data) %in% c("averageRating")]
y_train <- training_data$averageRating

x_test <- testing_data[, !names(training_data) %in% c("averageRating")]
y_test <- testing_data$averageRating
```

```
head(x_train)
```

```
##   titleType isAdult startYear runtimeMinutes genres numVotes
## 1         2       0      1894              1      7     2024
## 2         2       0      1892              5      3      272
## 3         2       0      1892              4      3     1962
## 5         2       0      1893              1      5     2727
## 6         2       0      1894              1     28      184
## 7         2       0      1894              1     28      847
```

```
head(y_train)
```

```
## [1] 5.7 5.7 6.5 6.2 5.0 5.4
```

```
head(x_test)
```

```
##    titleType isAdult startYear runtimeMinutes genres numVotes
## 4          2       0      1892             12      3      178
## 10         2       0      1895              1      7     7449
## 17         2       0      1895              1      7      339
## 21         2       0      1895              1      7     1127
## 23         2       0      1895              1     18      126
## 43         2       0      1896              1     28       49
```

```
head(y_test)
```

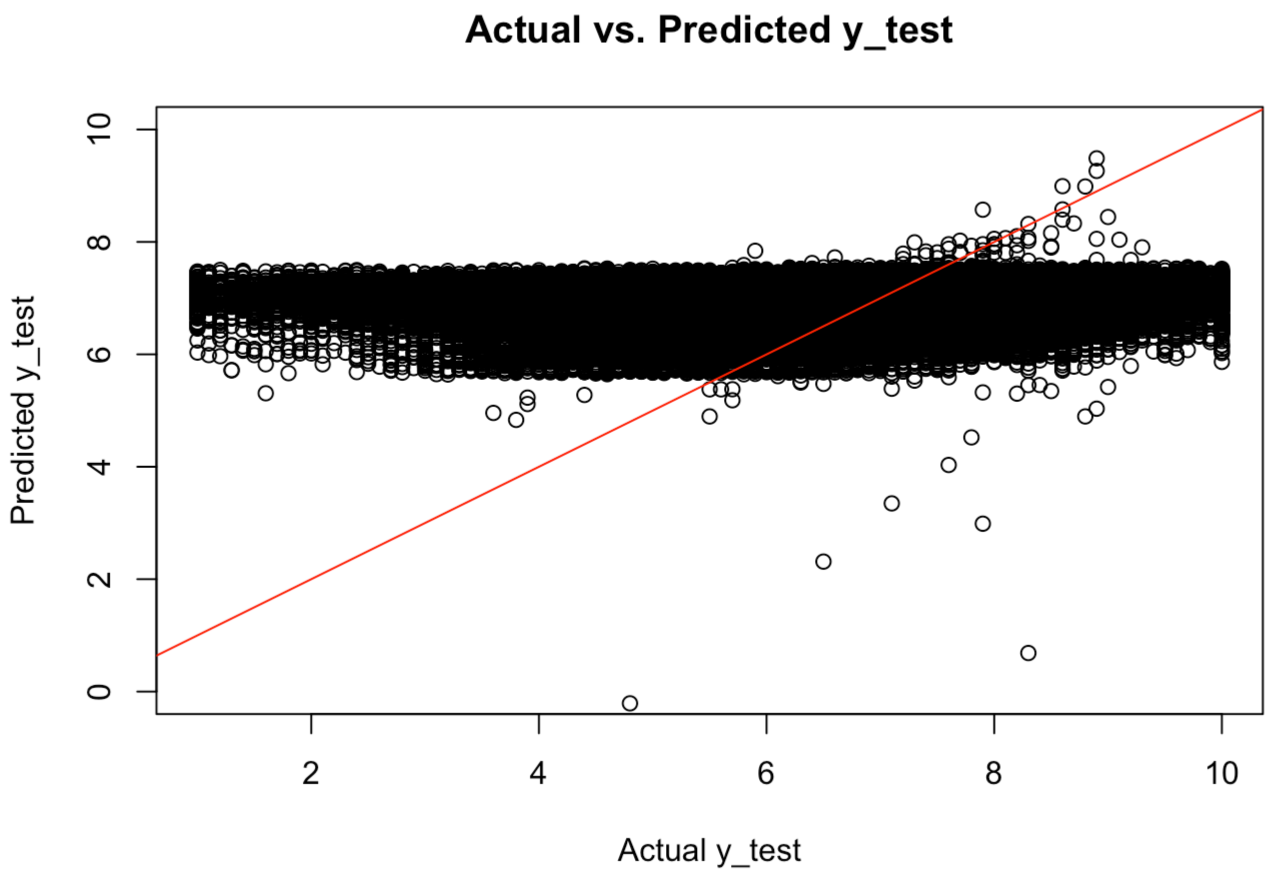
```
## [1] 5.4 6.8 4.6 5.1 3.9 4.0
```

Linear regression model

```
# Build the multiple linear regression model using training data
lm_model <- lm(averageRating ~ ., data = training_data)
```

```
# Predict y values using the model
lm_y_pred <- predict(lm_model, newdata = testing_data)
```

```
plot(y_test, lm_y_pred,
     xlab = "Actual y_test", ylab = "Predicted y_test",
     main = "Actual vs. Predicted y_test",
     ylim = c(0, 10))
abline(0, 1, col = "red")
```




```
summary(lm_model)
```

```
##
## Call:
## lm(formula = averageRating ~ ., data = training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.496 -0.714  0.177  0.898 68.531
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.269e+01  1.214e-01 -104.49   <2e-16 ***
## titleType       5.141e-02  6.788e-04   75.73   <2e-16 ***
## isAdult        -7.939e-01  1.103e-02  -71.95   <2e-16 ***
## startYear       9.778e-03  6.074e-05  160.99   <2e-16 ***
## runtimeMinutes -1.339e-03  1.578e-05  -84.86   <2e-16 ***
## genres         -2.208e-03  1.581e-04  -13.96   <2e-16 ***
## numVotes        1.295e-06  7.266e-08   17.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.357 on 1122982 degrees of freedom
## Multiple R-squared:  0.04205,    Adjusted R-squared:  0.04204
## F-statistic:  8215 on 6 and 1122982 DF,  p-value: < 2.2e-16
```

```
# Calculate Mean Absolute Error (MAE)
MAE <- mean(abs(lm_y_pred - y_test))
cat("Mean Absolute Error (MAE):", MAE, "\n")
```

```
## Mean Absolute Error (MAE): 1.029465
```

```
# Calculate Mean Squared Error (MSE)
MSE <- mean((lm_y_pred - y_test)^2)
cat("Mean Squared Error (MSE):", MSE, "\n")
```

```
## Mean Squared Error (MSE): 1.82996
```

```
# Calculate Root Mean Squared Error (RMSE)
RMSE <- sqrt(MSE)
cat("Root Mean Squared Error (RMSE):", RMSE, "\n")
```

```
## Root Mean Squared Error (RMSE): 1.35276
```

```
# Calculate R-squared
SS_res <- sum((lm_y_pred - y_test)^2)
SS_tot <- sum((y_test - mean(y_test))^2)
R_squared <- 1 - (SS_res / SS_tot)
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.04318821
```

KNN regression model

```
# Build the KNN model using training data
knn_final_model <- knn.reg(train = x_train, test = x_test, y = y_train, k = 10)  # Adjus
t the value of k as needed
```

```
# Predict average ratings for testing data
knn_pred_ratings <- knn_final_model$pred

# Print the predicted ratings
head(knn_pred_ratings)
```

```
## [1] 4.98 7.38 5.34 5.78 4.85 4.99
```

```
# Calculate Mean Absolute Error (MAE)
MAE <- mean(abs(knn_pred_ratings - y_test))
cat("Mean Absolute Error (MAE):", MAE, "\n")
```

```
## Mean Absolute Error (MAE): 0.9068456
```

```
# Calculate Mean Squared Error (MSE)
MSE <- mean((knn_pred_ratings - y_test)^2)
cat("Mean Squared Error (MSE):", MSE, "\n")
```

```
## Mean Squared Error (MSE): 1.509794
```

```
# Calculate Root Mean Squared Error (RMSE)
RMSE <- sqrt(MSE)
cat("Root Mean Squared Error (RMSE):", RMSE, "\n")
```

```
## Root Mean Squared Error (RMSE): 1.228737
```

```
# Calculate R-squared
SS_res <- sum((knn_pred_ratings - y_test)^2)
SS_tot <- sum((y_test - mean(y_test))^2)
R_squared <- 1 - (SS_res / SS_tot)
cat("R-squared:", R_squared, "\n")
```

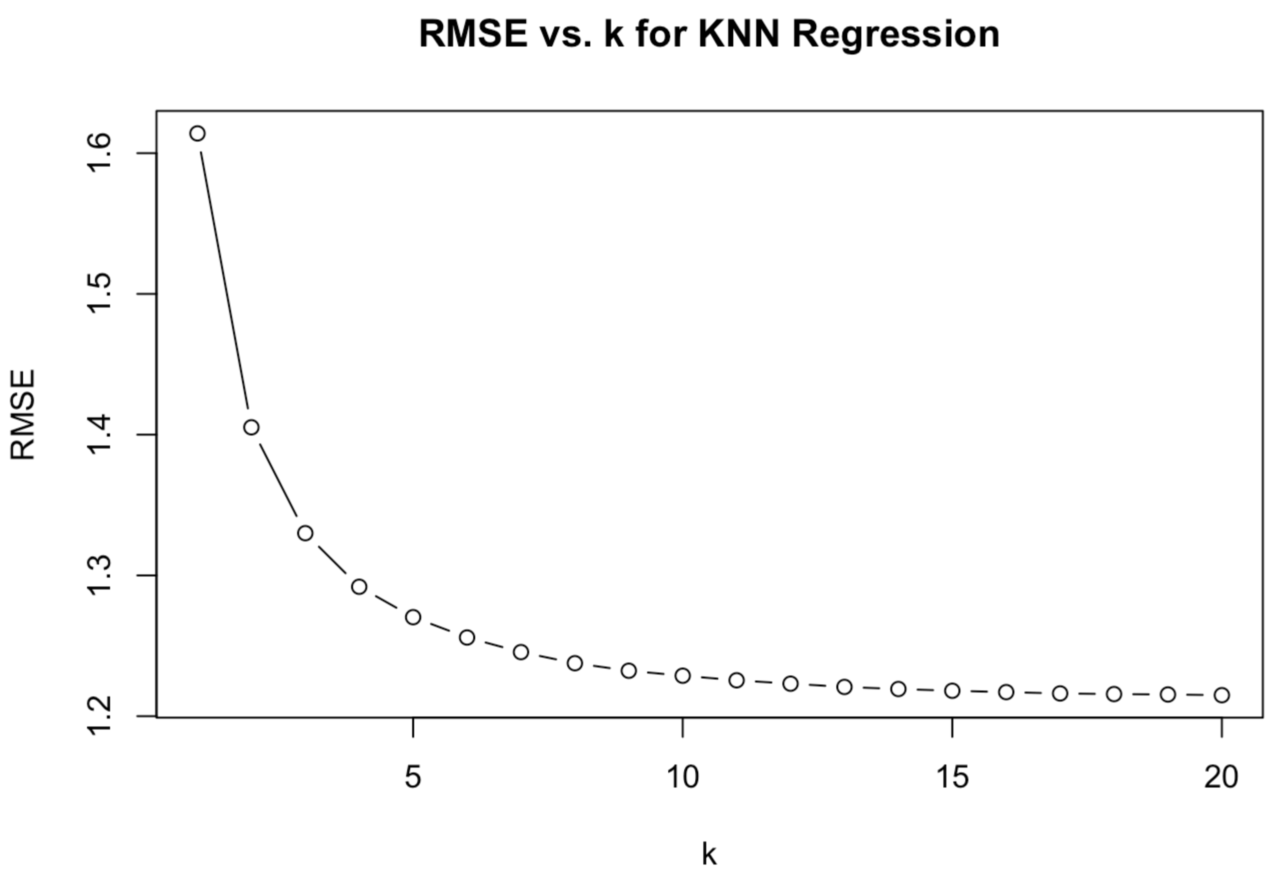
```
## R-squared: 0.2105901
```

```r
# Define a function to calculate RMSE for KNN regression
knn_rmse <- function(k, train_data, test_data) {

  # Train KNN regression model
  knn_model <- knn.reg(train = x_train, test = x_test, y = y_train, k = k)

  # Predict on test data
  predictions <- knn_model$pred

  cat("For the k = ", k, "\n")
  # Calculate Root Mean Squared Error (RMSE)
  RMSE <- sqrt(mean((predictions - y_test)^2))
  cat("Root Mean Squared Error (RMSE):", RMSE, "\n")
  print("-------------------------------------")

  return(RMSE)
}


# Perform grid search to find the best k value
k_values <- 1:20
rmse_values <- sapply(k_values, function(k) knn_rmse(k, training_data, testing_data))
```

```
## For the k =  1
## Root Mean Squared Error (RMSE): 1.614034
## [1] "-------------------------------------"
## For the k =  2
## Root Mean Squared Error (RMSE): 1.405222
## [1] "-------------------------------------"
## For the k =  3
## Root Mean Squared Error (RMSE): 1.330015
## [1] "-------------------------------------"
## For the k =  4
## Root Mean Squared Error (RMSE): 1.291941
## [1] "-------------------------------------"
## For the k =  5
## Root Mean Squared Error (RMSE): 1.270332
## [1] "-------------------------------------"
## For the k =  6
## Root Mean Squared Error (RMSE): 1.255893
## [1] "-------------------------------------"
## For the k =  7
## Root Mean Squared Error (RMSE): 1.245517
## [1] "-------------------------------------"
## For the k =  8
## Root Mean Squared Error (RMSE): 1.237622
## [1] "-------------------------------------"
## For the k =  9
## Root Mean Squared Error (RMSE): 1.232273
## [1] "-------------------------------------"
## For the k =  10
## Root Mean Squared Error (RMSE): 1.228737
## [1] "-------------------------------------"
## For the k =  11
## Root Mean Squared Error (RMSE): 1.225487
## [1] "-------------------------------------"
## For the k =  12
## Root Mean Squared Error (RMSE): 1.223079
## [1] "-------------------------------------"
## For the k =  13
## Root Mean Squared Error (RMSE): 1.220765
## [1] "-------------------------------------"
## For the k =  14
## Root Mean Squared Error (RMSE): 1.21929
## [1] "-------------------------------------"
## For the k =  15
## Root Mean Squared Error (RMSE): 1.218051
## [1] "-------------------------------------"
## For the k =  16
## Root Mean Squared Error (RMSE): 1.217086
## [1] "-------------------------------------"
## For the k =  17
## Root Mean Squared Error (RMSE): 1.216145
## [1] "-------------------------------------"
## For the k =  18
```

```
## Root Mean Squared Error (RMSE): 1.215699
## [1] "--------------------------------------"
## For the k =  19
## Root Mean Squared Error (RMSE): 1.215407
## [1] "--------------------------------------"
## For the k =  20
## Root Mean Squared Error (RMSE): 1.214908
## [1] "--------------------------------------"
```

```
# Plot RMSE values for different k values
plot(k_values, rmse_values, type = "b", xlab = "k", ylab = "RMSE", main = "RMSE vs. k fo
r KNN Regression")
```



Selecting an optimal value for k in KNN is crucial for achieving the right balance between bias and variance. Lower values of k may lead to overfitting, capturing noise in the data, while higher values of k may result in underfitting, oversimplifying the model. After evaluating the model's performance for various k values, we observed that the RMSE values plateaued after k = 10. This indicates that increasing the value of k beyond 10 does not significantly improve the model's performance. It suggests that the model starts to overfit the data beyond k = 10. Therefore, based on this analysis, we have chosen k = 10 as the optimal value for our KNN model. This value strikes a balance between capturing the underlying patterns in the data and avoiding overfitting.

Decision Tree

```
# Build the decision tree model using training data
tree_model <- rpart(formula = y_train ~ ., data = x_train)
```

```
# Predict average ratings for testing data
tree_pred_ratings <- predict(tree_model, newdata = x_test)

# Print the predicted ratings
head(tree_pred_ratings)
```

```
##        4       10       17       21       23       43
## 5.726709 5.726709 5.726709 5.726709 5.726709 5.726709
```

```
summary(tree_model)
```

```
## Call:
## rpart(formula = y_train ~ ., data = x_train)
##   n= 1122989
##
##           CP nsplit rel error    xerror        xstd
## 1 0.08852954      0 1.0000000 1.0000036 0.001672212
## 2 0.02154786      1 0.9114705 0.9114745 0.001634130
## 3 0.01926800      2 0.8899226 0.8899275 0.001592392
## 4 0.01000000      3 0.8706546 0.8706605 0.001586000
##
## Variable importance
##      titleType runtimeMinutes      startYear       numVotes        isAdult
##             59             28             11              1              1
##
## Node number 1: 1122989 observations,    complexity param=0.08852954
##   mean=6.955654, MSE=1.922095
##   left son=2 (243631 obs) right son=3 (879358 obs)
##   Primary splits:
##       titleType      < 1.5    to the left,  improve=0.088529540, (0 missing)
##       runtimeMinutes < 62.5   to the right, improve=0.088311050, (0 missing)
##       startYear      < 1951.5 to the left,  improve=0.018360130, (0 missing)
##       numVotes       < 13.5   to the right, improve=0.005344171, (0 missing)
##       genres         < 7.5    to the right, improve=0.004320781, (0 missing)
##   Surrogate splits:
##       runtimeMinutes < 72.5   to the right, agree=0.894, adj=0.510, (0 split)
##       numVotes       < 5170.5 to the right, agree=0.789, adj=0.028, (0 split)
##       startYear      < 1951.5 to the left,  agree=0.786, adj=0.013, (0 split)
##
## Node number 2: 243631 observations
##   mean=6.171957, MSE=1.890874
##
## Node number 3: 879358 observations,    complexity param=0.02154786
##   mean=7.172781, MSE=1.713439
##   left son=6 (182658 obs) right son=7 (696700 obs)
##   Primary splits:
##       titleType      < 4.5    to the right, improve=0.030868840, (0 missing)
##       startYear      < 1949.5 to the left,  improve=0.023493090, (0 missing)
##       runtimeMinutes < 66.5   to the right, improve=0.018464580, (0 missing)
##       isAdult        < 0.5    to the right, improve=0.005183296, (0 missing)
##       numVotes       < 529.5  to the left,  improve=0.005126594, (0 missing)
##   Surrogate splits:
##       runtimeMinutes < 47.5   to the right, agree=0.858, adj=0.316, (0 split)
##       isAdult        < 0.5    to the right, agree=0.806, adj=0.068, (0 split)
##       numVotes       < 8273.5 to the right, agree=0.793, adj=0.004, (0 split)
##
## Node number 6: 182658 observations
##   mean=6.723624, MSE=2.161057
##
## Node number 7: 696700 observations,    complexity param=0.019268
##   mean=7.290539, MSE=1.529325
##   left son=14 (16601 obs) right son=15 (680099 obs)
##   Primary splits:
```

```
##       startYear      < 1950.5 to the left,  improve=0.039033850, (0 missing)
##       titleType      < 2.5    to the left,  improve=0.029819600, (0 missing)
##       runtimeMinutes < 18.5   to the left,  improve=0.026089870, (0 missing)
##       numVotes       < 287.5  to the left,  improve=0.012490180, (0 missing)
##       genres         < 8.5    to the right, improve=0.002029804, (0 missing)
##
## Node number 14: 16601 observations
##   mean=5.726709, MSE=1.457866
##
## Node number 15: 680099 observations
##   mean=7.328712, MSE=1.469917
```

```
# Calculate Mean Absolute Error (MAE)
MAE <- mean(abs(tree_pred_ratings - y_test))
cat("Mean Absolute Error (MAE):", MAE, "\n")
```

```
## Mean Absolute Error (MAE): 0.9740413
```

```
# Calculate Mean Squared Error (MSE)
MSE <- mean((tree_pred_ratings - y_test)^2)
cat("Mean Squared Error (MSE):", MSE, "\n")
```

```
## Mean Squared Error (MSE): 1.665022
```

```
# Calculate Root Mean Squared Error (RMSE)
RMSE <- sqrt(MSE)
cat("Root Mean Squared Error (RMSE):", RMSE, "\n")
```

```
## Root Mean Squared Error (RMSE): 1.290357
```

```
# Calculate R-squared
SS_res <- sum((tree_pred_ratings - y_test)^2)
SS_tot <- sum((y_test - mean(y_test))^2)
R_squared <- 1 - (SS_res / SS_tot)
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.1294278
```

Based on these metrics, the KNN Regression Model appears to perform the best among the three models.

Reccomendations - Top 10 Reccomended movies

Let’s apply the KNN regression model on total dataset and predit the ratings. Based on these ratings, reccommend the top rated movies

```
# Join reccon_dataset and movie_dataset using tconst as the key to get the originalTitle
feature from movie_dataset
merged_dataset <- merge(reccon_dataset, movies_dataset[, c("tconst", "originalTitle")],
by = "tconst", all.x = TRUE)
#Considering whole dataset as a testing dataset
new_test_dataset <- subset(merged_dataset, select = c(titleType, isAdult,startYear, runt
imeMinutes, genres, numVotes))


# Building knn regression model
knn_recc_model <- knn.reg(train = x_train, test = new_test_dataset, y = y_train, k = 10)
# Predict average ratings for testing data
knn_recc_predratings <- knn_recc_model$pred


# Attach predicted ratings to the dataset
merged_dataset$predicted_ratings <- knn_recc_predratings
# Sort entries by predicted_ratings in descending order
merged_dataset <- merged_dataset[order(merged_dataset$predicted_ratings, decreasing = TR
UE), ]
print(merged_dataset[1:10, c("originalTitle", "predicted_ratings")])
```

```
##                                                                  originalTitle
## 283831                                                      Along for the Ride
## 321748                                                       The Art of Biting
## 351916                                                      Comenzando De Nuevo
## 351917                                                                 El Baile
## 391103                                                              Jalkapuussa
## 547198                                                                  Bolum 1
## 558662                                             Fear and Falling in Montana
## 558664                                                       Making Allowances
## 558666                                                              Solar Mates
## 567757 Jennifer's Instinct; Sailors Angel; OR Miracle; High School Reunion
##        predicted_ratings
## 283831                10
## 321748                10
## 351916                10
## 351917                10
## 391103                10
## 547198                10
## 558662                10
## 558664                10
## 558666                10
## 567757                10
```

Here are the top 10 reccomended movie by the KNN regression model.