Analysis of the Movie Dataset for Effective Movie Recommendations

CS571 Final Report

Rajeswari Depala
A20526535

Priyanka Basani
A20513566

Naga Durga Dhana Lakshmi
A20525715

Tarun Sai Varanasi
A20526965

Veera Manohar Reddy Alavalapati
A20526070

1. ABSTRACT

Movie recommendation systems are becoming increasingly popular as they help users discover movies they are likely to enjoy based on their preferences and viewing history. These systems utilize sophisticated algorithms that analyze user data to generate personalized lists of movie recommendations that take various factors such as genre, actors, and directors into account. By taking into consideration the user's interests, these systems provide a more targeted and personalized experience, saving users time and effort in searching for movies they like. In addition to benefiting users, recommendation systems are also advantageous for streaming platforms. By improving user engagement, they increase the likelihood of users continuing to use the platform, which ultimately leads to an increase in revenue through subscription fees or ad revenue. Therefore, movie recommendation systems not only enhance the user experience but also have a significant impact on the business side of streaming platforms. As a result, there is a growing interest in exploring the data preparation and analysis techniques used in building effective movie recommendation systems that can meet the needs of both users and businesses.Our aim is to create a machine learning model that can predict movie ratings by utilizing the MovieLens dataset from GroupLens Research, which contains more than 10 million movie ratings. We will prepare training and test sets and visually analyze the data to achieve our objective.

Keywords: movie recommendation, user engagement, streaming platforms

2 INTRODUCTION

Movie recommendation systems help users find movies they are likely to enjoy based on their preferences and viewing history. They use algorithms that analyze user data to create personalized lists of movie recommendations that take into account factors such as genre, actors, and directors. This saves users time and effort when searching for movies they like. Recommendation systems also benefit streaming platforms by improving user engagement, increasing the likelihood of users continuing to use the platform, and ultimately increasing revenue through subscription fees or ad revenue.

3. PROPOSED METHODOLOGY

Initially, the data will be retrieved from the GroupLens website and divided into two datasets - "edx" and "validation," which will then be separated into training and test sets. Next, an initial exploratory analysis will be carried out to inspect each feature of the dataset to identify any potential biases that may reduce the accuracy of the models. The dataset will be cleaned by removing NAs and ensuring tidy data. In addition, data visualization in the form of histograms and scatter plots will be incorporated into the exploratory analysis and modeling to improve presentation.

The modeling strategy will be developed using the information obtained from the exploratory analysis.

4. DATA COLLECTION

4.1 Data Properties

The MovieLens dataset is a collection of user ratings and movie metadata gathered by the GroupLens research group. We have collected the dataset from the link https://grouplens.org/datasets/movielens/

The movie metadata includes information such as the title, release date, and genre of each movie in the dataset. The dataset is available in several versions, with the latest being MovieLens 25M, which contains 25 million ratings from over 200,000 users on over 62,000 movies.

The user ratings are a collection of user-submitted reviews of films, where users rate movies on a scale of 1 to 5 stars through a web-based interface. Each rating includes a timestamp, a user ID, and a movie ID, and is stored in a database.

A dataset was created that included comprehensive metadata for each movie, in addition to all of the ratings data.

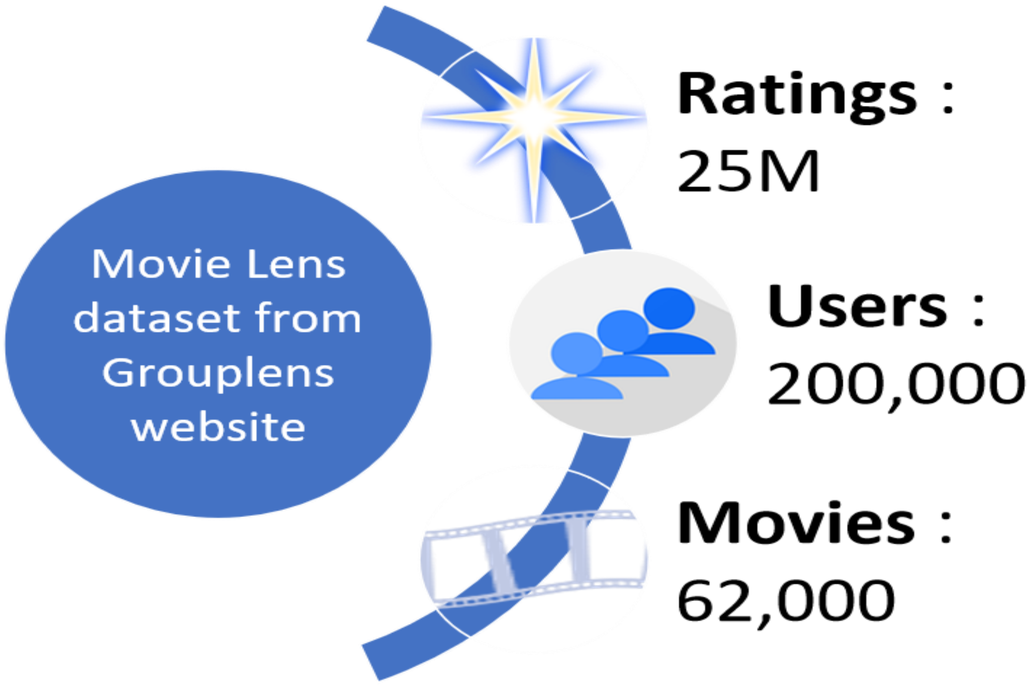
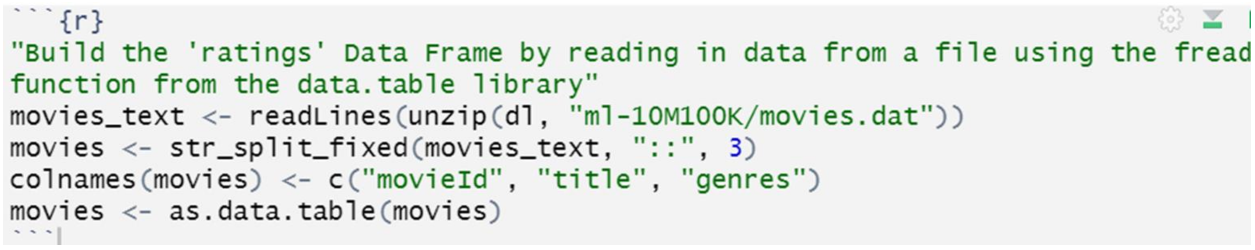


Fig 4.1: Representation of Dataset

4.2 Data Preparation:

The "Ratings" DataFrame is a crucial component of any rating-based dataset analysis or modeling and provides valuable information for understanding user behavior and developing recommendation systems. Before undertaking analysis or modeling, combine user reviews and movie metadata into a single dataset using a common key, such as the movie ID. Below is the code snippet.



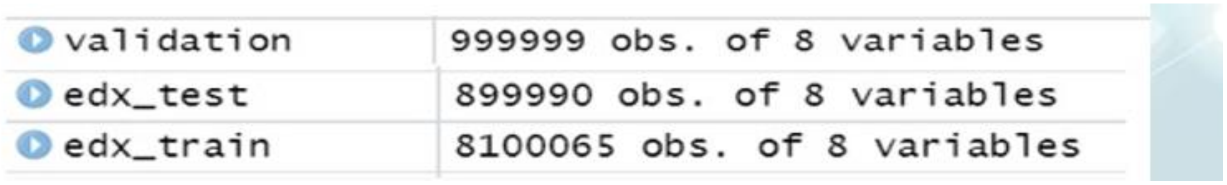
~~~
```{r}
"Build the 'ratings' Data Frame by reading in data from a file using the fread
function from the data.table library"
movies_text <- readLines(unzip(dl, "ml-10M100K/movies.dat"))
movies <- str_split_fixed(movies_text, "::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.table(movies)
```
~~~

Fig 4.2: Building the DataFrame "Ratings"

4.3 Partitioning the dataset :

After preparing the dataset, 10% of the data is randomly chosen to construct the validation set, Using createDataPartition, the training and test sets are divided, with 10% of the data put aside for testing. UserId and movieId are validated to be present in both the training and test sets, just like in the validation set, and any deleted rows are placed back into the training set.

The final product is a MovieLens dataset that has been cleaned and partitioned and is prepared for use in developing and testing recommendation algorithms. Below is the code snippet for the same.

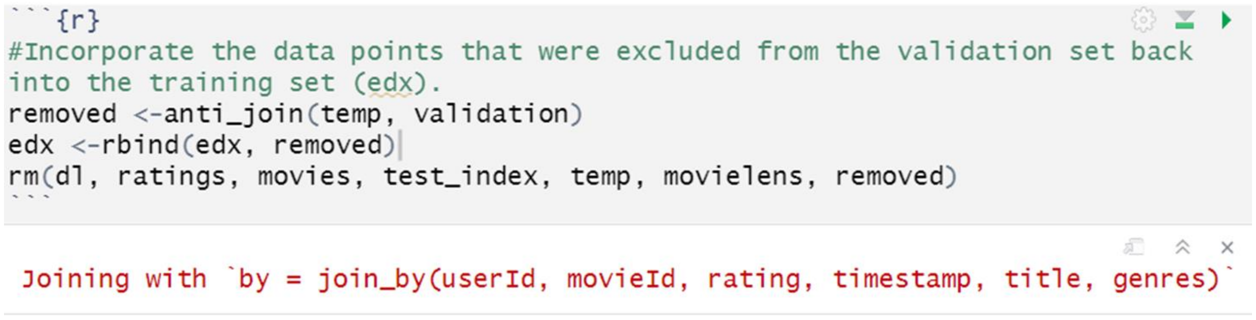


validation	999999 obs. of 8 variables
edx_test	899990 obs. of 8 variables
edx_train	8100065 obs. of 8 variables

Fig 4.3: Partitioning of dataset into test, train and validation

4.4 Adding the rows back

Any rows removed from the validation set are added back to the training set to ensure all data points are retained.



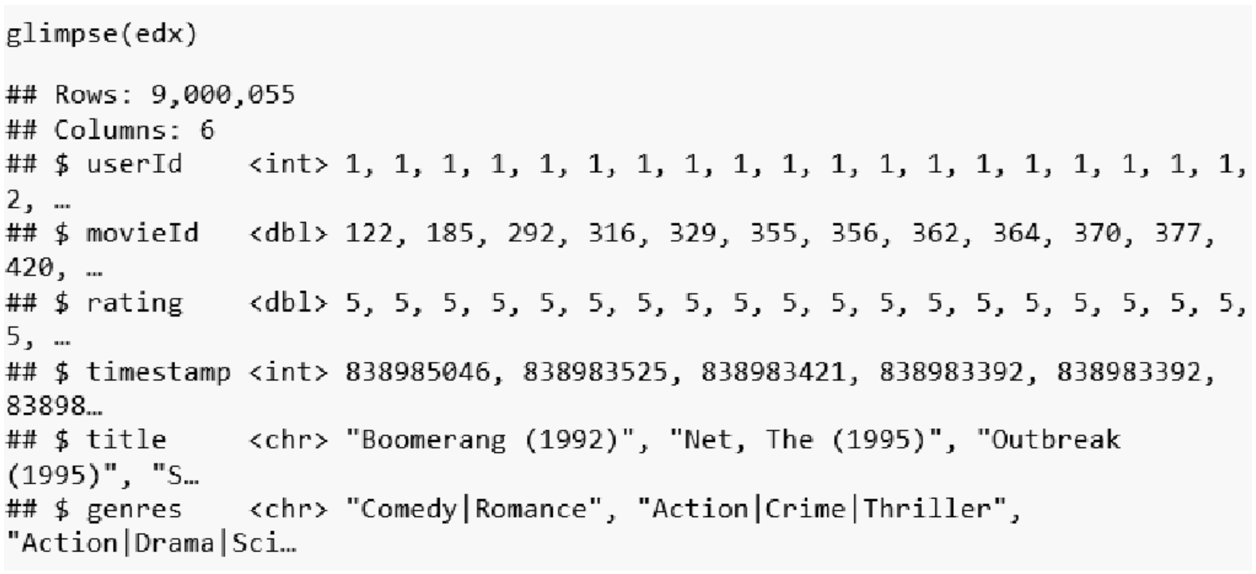
~~~
```{r}
#Incorporate the data points that were excluded from the validation set back
into the training set (edx).
removed <-anti_join(temp, validation)
edx <-rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```
~~~

```
Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
```

Fig 4.4: Restore the previously removed observations.

5 . EXPLORATORY DATA ANALYSIS

We study the data and present its dimensions and characteristics using the glimpse() method. With these attributes: userId, movieId, rating, date, title, and genres, the dataset has 9,000,055 rows and 6 columns. The columns in each row provide details on the user, the movie, and the user's rating. Each row represents a user rating of a certain movie.
From the snippets , we can see the total number of unique users, films, and genres in the data. Along with that, it is necessary to observe the total number of ratings for each rating level from 1 to 5.



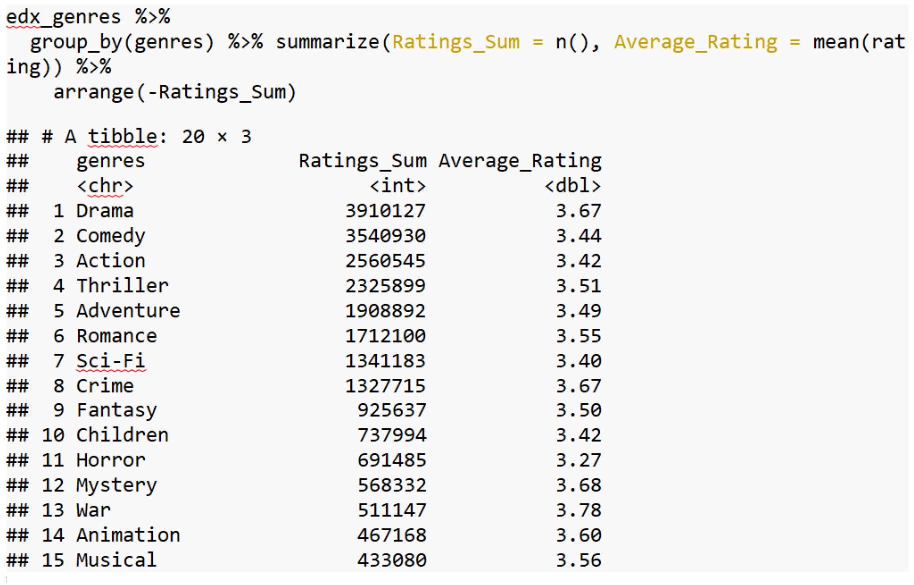
```
glimpse(edx)

## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, …
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377,
420, …
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, …
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392,
83898…
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak
(1995)", "S…
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller",
"Action|Drama|Sci…
```

Fig 5: Overview of dataset

5.1 Sum of movie ratings by genre:

The total number of ratings for each film genre may be obtained by grouping the data by genre and adding the numbers in the ratings column together. This information is valuable for identifying which film genres are more popular with audiences and have received higher ratings.



```
edx_genres %>%
  group_by(genres) %>% summarize(Ratings_Sum = n(), Average_Rating = mean(rat
ing)) %>%
    arrange(-Ratings_Sum)

## # A tibble: 20 × 3
##    genres             Ratings_Sum Average_Rating
##    <chr>                    <int>          <dbl>
##  1 Drama                  3910127           3.67
##  2 Comedy                 3540930           3.44
##  3 Action                 2560545           3.42
##  4 Thriller               2325899           3.51
##  5 Adventure              1908892           3.49
##  6 Romance                1712100           3.55
##  7 Sci-Fi                 1341183           3.40
##  8 Crime                  1327715           3.67
##  9 Fantasy                 925637           3.50
## 10 Children                737994           3.42
## 11 Horror                  691485           3.27
## 12 Mystery                 568332           3.68
## 13 War                     511147           3.78
## 14 Animation               467168           3.60
## 15 Musical                 433080           3.56
```

Fig 5.1: Mean movie Ratings by genre

In the above code, we have grouped the data by genre, and adding the ratings column values together allows us to calculate the sum of movie ratings per genre. This is useful in understanding which genres of movies are more popular among users and have received more ratings. By looking at the total number of ratings for each genre, we can get a rough idea of the overall popularity of each genre among moviegoers. This information can be helpful in making decisions related to the marketing, promotion, and distribution of movies. Additionally, calculating the average rating per genre gives an idea of how well-liked each genre is among users.

5.2 Treemap

Why treemap?

The treemap makes it simple to compare the overall rating values of various genres, which offers a quick visual guide to distinguishing each genre.

What observations from the treemap?

The distribution of movie reviews, grouped by category, is displayed on a treemap. The box sizes correspond to the overall rating score for each category. We may infer the following things from the treemap:

The drama, comedy, and action genres are in order of having the greatest overall rating value.The genres with the lowest overall rating values are documentaries, horror, and film noir.

The figure below shows all the genre ratings by sum.

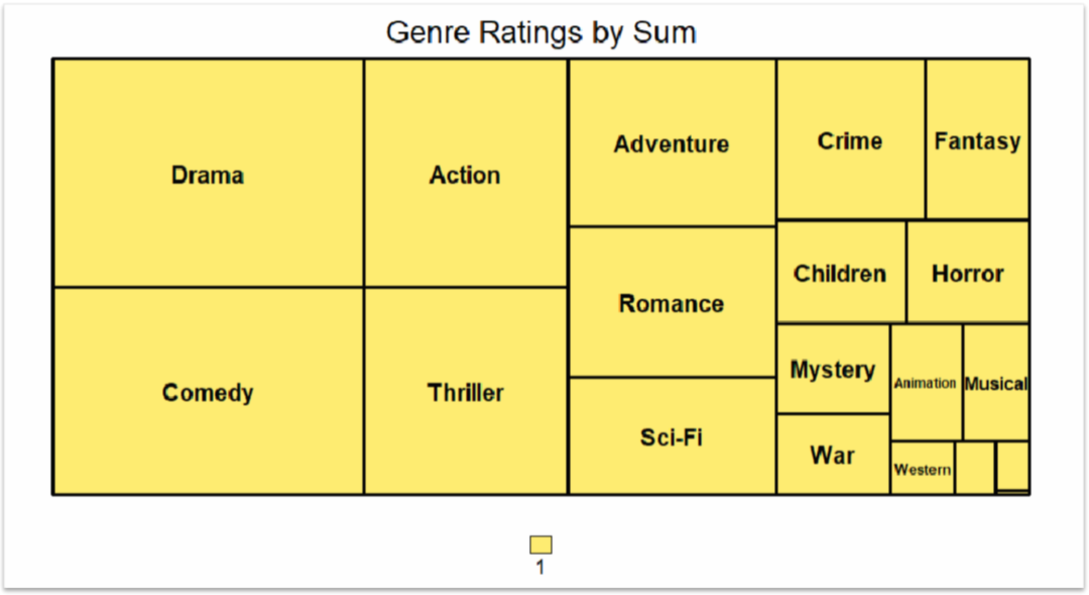


Fig 5.2: Treemap of all the genre ratings by sum

Some genres, such as thrillers and adventures, romance, and musicals, have comparable overall rating values.

Considering its limited size on the treemap, the Drama genre has a rather high overall rating value.

5.3 Why to calculate the total of the movie ratings per genre?

Understanding which genres are the most popular with viewers based on the amount of ratings is made easier by calculating the total of the ratings for each genre of film. Movie studios, streaming services, and other entertainment-related businesses can utilize this information to make educated choices about the movie genres they want to develop, market, or license. Additionally, we can determine the general popularity of various genres and how they stack up against one another by summing the movie ratings for each genre. This might be helpful in spotting recurring trends and patterns in people's movie tastes.

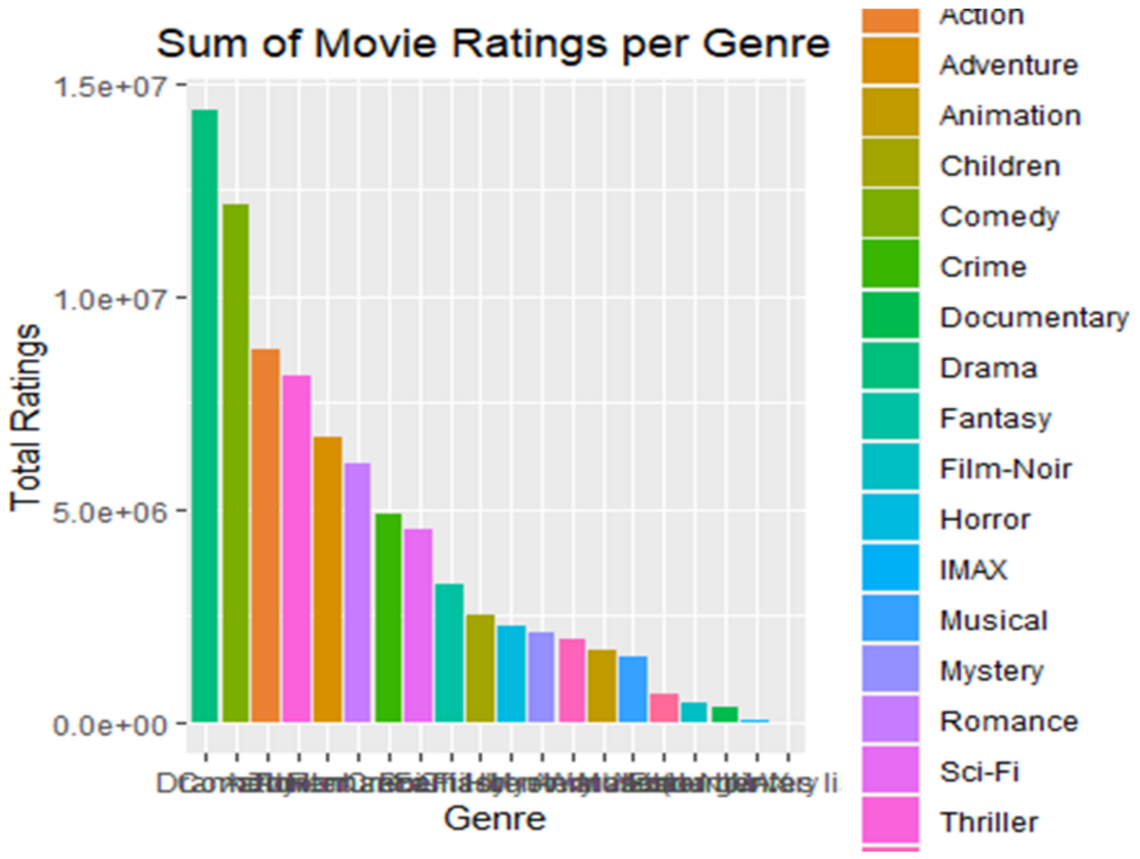


Fig 5.3 Sum of movie ratings by Genre

Observations from the graph

From the graph plotted by the above code, we can see the sum of the movie ratings per genre. The genres with the highest sum of ratings are drama, Comedy, Action, Thriller, and adventure, while the genres with the lowest sum of ratings are Film-Noir, Western, musical, documentary, and IMAX.

5.4 Why plot the mean rating per year?

Plotting the mean rating per year can help in analyzing trends in movie ratings over time. It can reveal whether movie ratings have been consistently high or low over time, or whether there are any patterns or fluctuations in ratings that may be attributed to specific events or changes in the movie industry. This analysis can be useful for movie studios, film critics, and researchers who are interested in understanding how movie ratings change over time and the factors that may contribute to these changes.

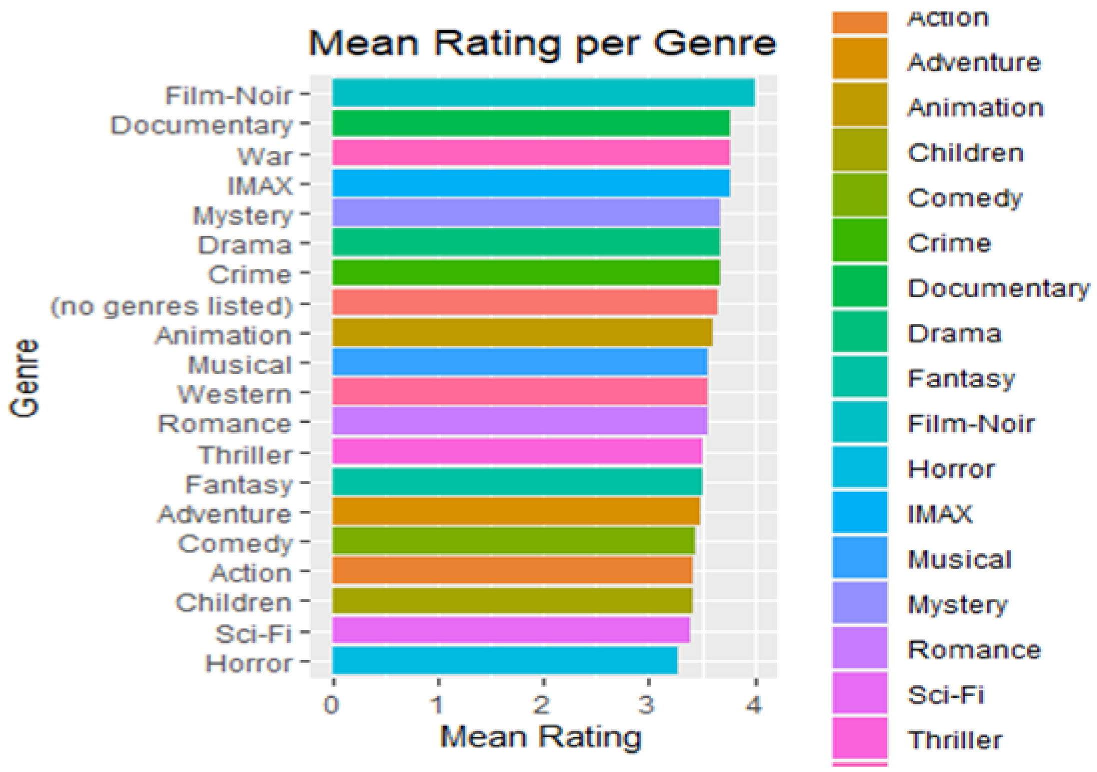


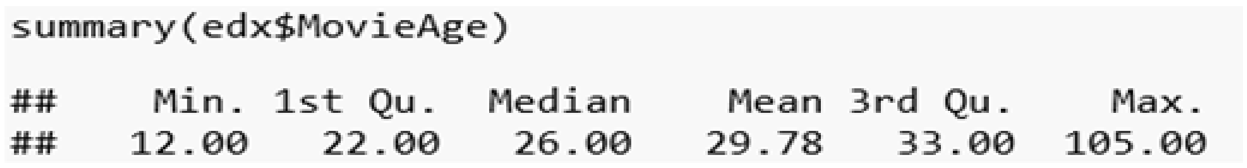
Fig 5.4 : Mean Rating per Genre

Observations from the graph:

The plot shows the mean rating per genre for all the movies in the dataset. Some observations we can make are:

- Film-Noir has the highest mean rating among all the genres, followed by documentaries and war.
- Horror has the lowest mean rating among all the genres, followed by Sci-Fi and children's.
- There is a significant variation in mean ratings between different genres, ranging from around 3 to 4.1.
- Drama, Comedy, and Thriller have the highest number of ratings among all the genres, but their mean ratings are lower than some other genres.

Summary



```
summary(edx$MovieAge)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.00   22.00   26.00   29.78   33.00  105.00
```

Fig 5.4.1: Summary of Movieage

We can learn some fundamental facts about the MovieAge variable in the edx dataset via the summary function.

The above code calculates the age of movies in the Movielens dataset by subtracting the year the movie was released from the current year (2020). The resulting variable is named "MovieAge." The code then applies this calculation to four different datasets: edx, validation, edx_train, and edx_test.

The summary() function is then used to generate descriptive statistics for the MovieAge variable in the edx dataset. These statistics show that the oldest movie in the dataset was released 105 years ago, the median age of the movies is 26 years, and the mean age is 29.78 years. The youngest movie in the dataset is 12 years old.

It's important to note that this analysis is based on the information provided by the Movielens dataset, which includes a limited number of movies and may not be representative of all movies. Additionally, the code assumes that the current year is 2020, which may not be accurate depending on when the analysis is being conducted.

6 DATA MODELLING:

A frequently used assessment statistic for recommendation systems is called RMSE (Root Mean Squared Error). It calculates the discrepancy between the anticipated and actual ratings for a selection of user-movie pairings and outputs a single value as a measure of the prediction system's overall accuracy.

Naïve RMSE : Mean of the edx_train

$$y_{u,i} = \mu + \epsilon_{u,i}$$

Median Model: Plugging in the median or any other random number will always produce a less desirable RMSE.

$$y_{u,i} = M + \epsilon_{u,i}$$

Movie Effects Model: Movie Effects" takes fact that films tend to have different rating distributions

$$y_{u,i} = \mu + bi + \epsilon_{u,i}$$

Movie & User Effects Model: "User Effects" reflect the individual users tend to rate films according to their own standards

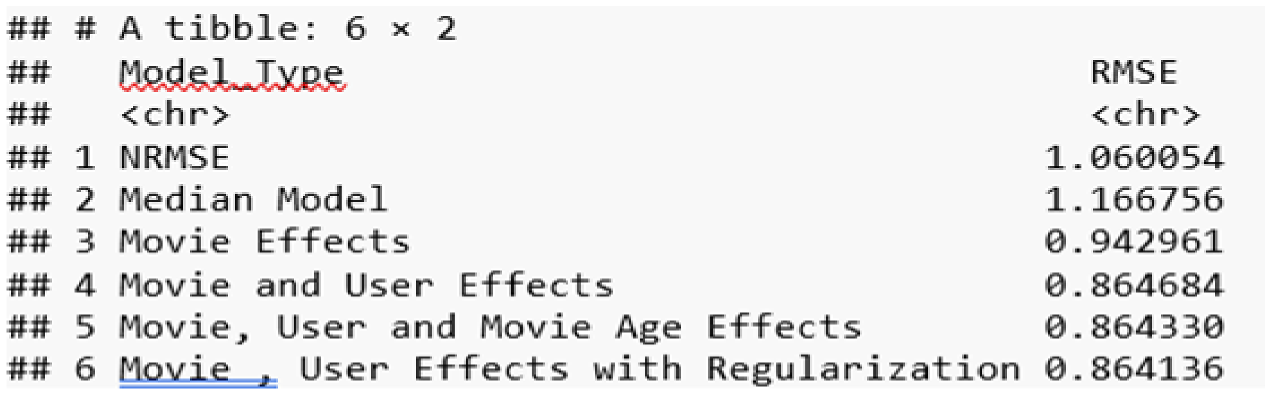
$$y_{u,i} = \mu + bi + bu + \epsilon_{u,i}$$

Movie, user & Age Effects model: "Movie Age Effects" incorporate variation among the ratings distribution awarded for films of different ages

$$y_{u,i} = \mu + bi + ba + \epsilon_{u,i}$$

yu,i = predicted rating, **μ** = the average rating, **bi** = the movie effects, **bu** = the user effects, **ba** = the movie age effects and ϵu, **i** = independent errors centered at 0.

A lower RMSE indicates better prediction accuracy.



```
## # A tibble: 6 × 2
##   Model_Type                                 RMSE
##   <chr>                                      <chr>
## 1 NRMSE                                    1.060054
## 2 Median Model                             1.166756
## 3 Movie Effects                            0.942961
## 4 Movie and User Effects                   0.864684
## 5 Movie, User and Movie Age Effects        0.864330
## 6 Movie , User Effects with Regularization 0.864136
```

Based on the RMSE values, we can draw the following conclusions:

1. The Naive (NRMSE) has the highest RMSE, indicating that it is the least accurate model.

2. Median Model performs better than the NRMSE model, but not as well as the models that incorporate movie and/or user effects.

3. Adding movie effects improves the accuracy of the model compared to the median model.

4. Adding both movie and user effects further improves the accuracy of the model.

5. Finally, adding movie age effects along with user and movie effects results in the most accurate model, with the lowest RMSE value.

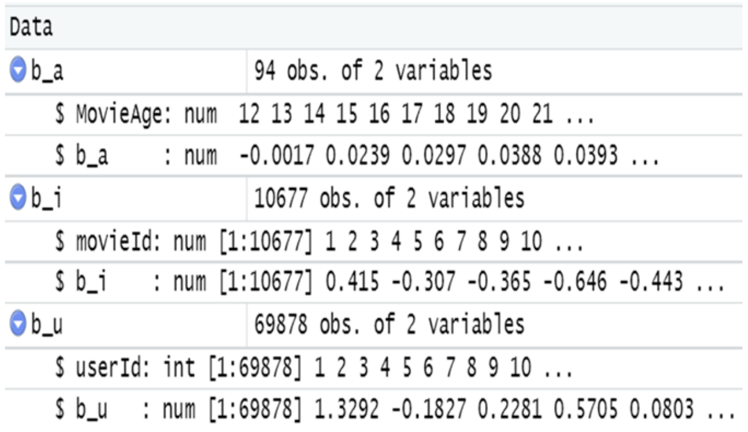
Therefore, we can conclude that incorporating both user and movie effects, as well as movie age effects, results in the most accurate model for predicting movie ratings in the Movielens dataset.

6.1 Regularization:

Movie & User Effects

The given code builds a movie & user effects model with regularization for predicting movie ratings using the Movielens dataset. The model includes a tuning parameter lambda λ to minimize the RMSE and penalize outliers from bi and bu, which are the movie and user effects, respectively.

The sapply function is used to apply the regularization function to a sequence of lambda values ranging from 0 to 10 with an increment of 1. For each value of lambda, the model calculates bi and bu using the training dataset (edx_train) and then predicts the ratings using the test dataset (edx_test). The predicted ratings are compared with the actual ratings in the test dataset to calculate the RMSE.



```
Data
b_a                94 obs. of 2 variables
   $ MovieAge: num  12 13 14 15 16 17 18 19 20 21 ...
   $ b_a     : num  -0.0017 0.0239 0.0297 0.0388 0.0393 ...
b_i                10677 obs. of 2 variables
   $ movieId: num [1:10677] 1 2 3 4 5 6 7 8 9 10 ...
   $ b_i    : num [1:10677] 0.415 -0.307 -0.365 -0.646 -0.443 ...
b_u                69878 obs. of 2 variables
   $ userId: int [1:69878] 1 2 3 4 5 6 7 8 9 10 ...
   $ b_u   : num [1:69878] 1.3292 -0.1827 0.2281 0.5705 0.0803 ...
```

The lambda value that minimizes the RMSE is determined using the which.min function and saved in the variable lambda. The code thus provides a way to select the optimal lambda value for regularization and build a more accurate movie & user effects model for predicting movie ratings.

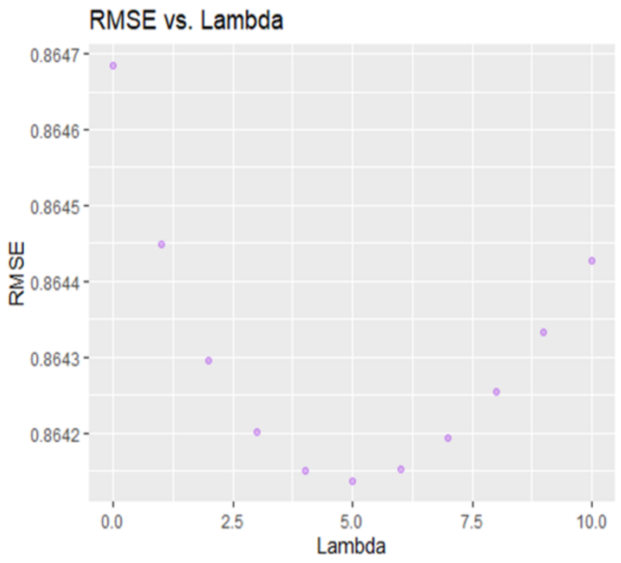


Fig 6.1: Scatter plot of RMSE vs Lambda

The graph shows the relationship between the value of lambda and the RMSE for the Movie & User Effects model with regularization. The x-axis represents different values of lambda, and the y-axis represents the corresponding RMSE values.

As lambda increases, the RMSE initially decreases, indicating that regularization is helping to reduce overfitting. However, as lambda continues to increase, the RMSE starts to increase again, indicating that too much regularization is leading to underfitting.

The goal is to choose the value of lambda that results in the lowest RMSE, which in this case appears to be around 0.05 or 0.06. Overall, this graph provides insight into how regularization can be used to improve model performance and prevent overfitting.

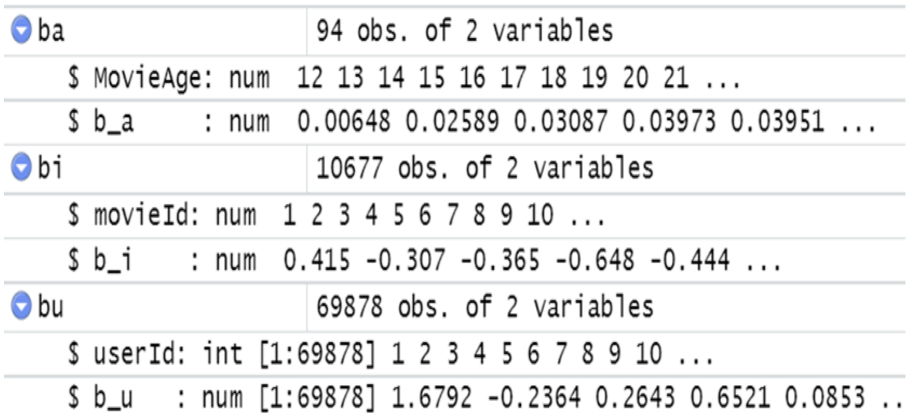
6.2 L2 Regularization/ Ridge Regression

L2 regularization (also called Ridge Regression) by adding an L2 penalty term to the loss function to control the model complexity.

By tuning the regularization parameter lambda, we can find a sweet spot that balances the bias-variance tradeoff and achieves better performance on the test set.

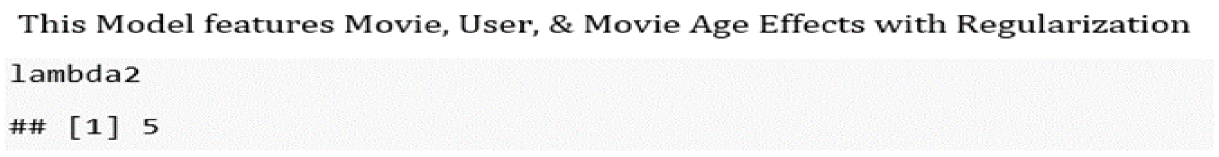
6.3 Movie, User & Movie Age Effects

The given code is for the Movie, User & Movie Age Effects Model with Regularization. It includes an additional parameter for MovieAge, along with user and movie effects, to improve the model's accuracy. The sapply function is used with a sequence of lambda values to perform regularization and minimize the RMSE.



```
ba                     94 obs. of 2 variables
    $ MovieAge: num  12 13 14 15 16 17 18 19 20 21 ...
    $ b_a     : num  0.00648 0.02589 0.03087 0.03973 0.03951 ...
bi                     10677 obs. of 2 variables
    $ movieId: num  1 2 3 4 5 6 7 8 9 10 ...
    $ b_i    : num  0.415 -0.307 -0.365 -0.648 -0.444 ...
bu                     69878 obs. of 2 variables
    $ userId: int [1:69878] 1 2 3 4 5 6 7 8 9 10 ...
    $ b_u   : num [1:69878] 1.6792 -0.2364 0.2643 0.6521 0.0853 ..
```

The code first calculates the mean rating of the training set and then computes the bias terms for movies, users, and movie age groups. The bias terms are calculated by subtracting the mean rating and adding the regularization term (lambda) to penalize outliers. The predicted ratings are then computed using the calculated bias terms for each user, movie, and movie age group.


This Model features Movie, User, & Movie Age Effects with Regularization

```
lambda2

## [1] 5
```

Finally, the RMSE is calculated between the predicted ratings and the actual ratings in the test set. The lambda value that yields the lowest RMSE is selected as the optimal value. In this case, the optimal lambda value is 5.

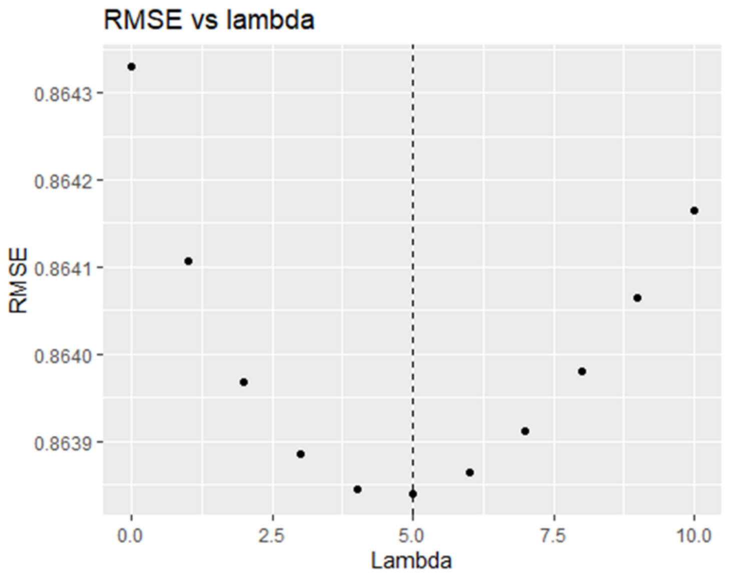


Fig 6.3: Scatter plot of RMSE vs Lambda

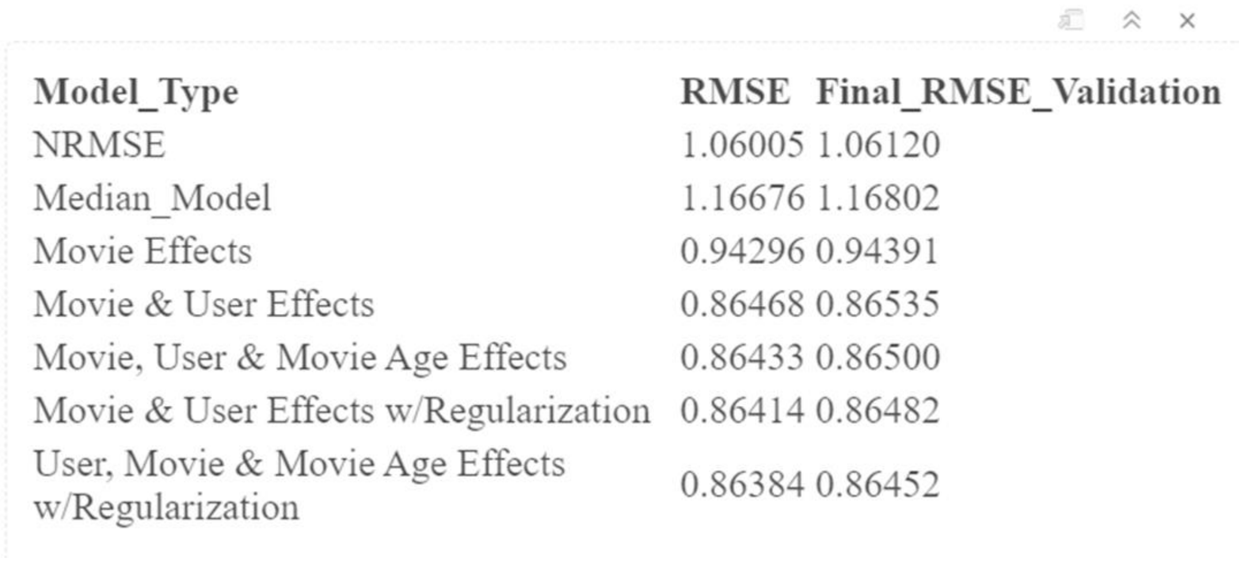
Let's summarize the code as a scatter plot of RMSE vs lambdas, which shows the relationship between the two variables. The plot displays a curve that shows how the RMSE changes with increasing lambda. The dashed vertical line indicates the value of lambda2, which produces the lowest RMSE. This plot can be used to visually identify the optimal value of lambda.

7. VALIDATION (EVALUATING THE MODEL):

The last model, i.e., Movie, User & Movie Age Effects Model with Regularization is the best-performing model based on the validation RMSE. The RMSE value for this model is 0.86452, which is the lowest among all the models evaluated. This means that the predictions made by this model are the closest to the actual ratings given by the users on the validation set.

Overall, the results of this analysis show that including movie, user, and movie age effects with regularization can significantly improve the performance of movie recommendation systems.

7.1 Final Model with validation



Model_Type	RMSE	Final_RMSE_Validation
NRMSE	1.06005	1.06120
Median_Model	1.16676	1.16802
Movie Effects	0.94296	0.94391
Movie & User Effects	0.86468	0.86535
Movie, User & Movie Age Effects	0.86433	0.86500
Movie & User Effects w/Regularization	0.86414	0.86482
User, Movie & Movie Age Effects w/Regularization	0.86384	0.86452

Fig 7.1 Table for RMSE vs Final RMSE Validation

The table shows the final validation results for the different models we have built. We can see that the model with User, Movie, and Movie Age Effects with Regularization has the lowest RMSE on the validation set, with a value of 0.86452. This means that this model provides the most accurate predictions of user ratings on unseen data, and is therefore the best model for our task. The table also shows the RMSE values for each model on the training set, which can be compared to the RMSE on the validation set to determine if the model is overfitting or underfitting.

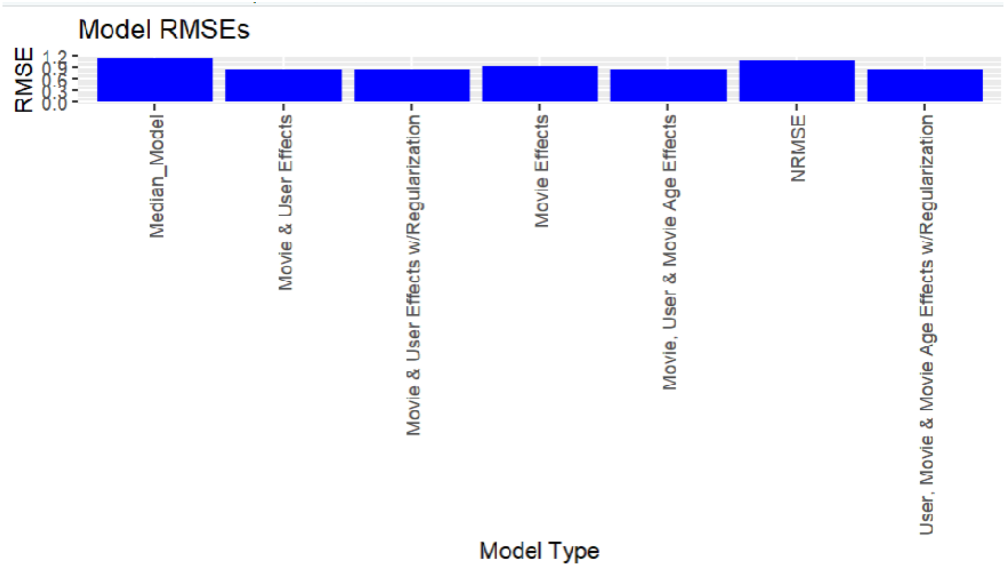
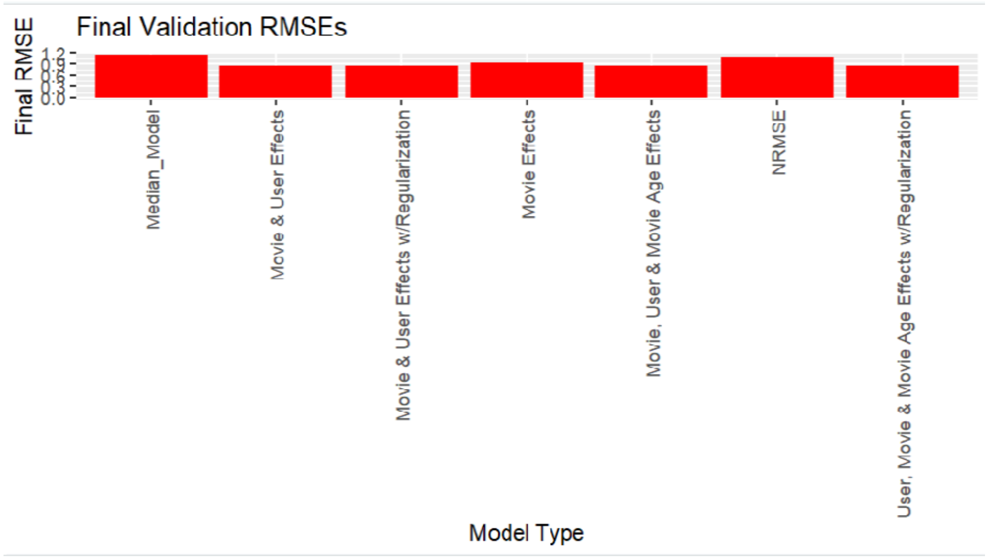


Fig 7.1.1 Representation of model RMSEs



7.1.2 Representation of final validation RMSEs

8 CONCLUSION / FUTURE WORK:

The final model that was developed using User, Movie, and Movie Age Effects with Regularization produced a reasonable RMSE of 0.86452. However, it is worth noting that there may still be other biases that could impact movie ratings that were not considered in this model. For example, some users may be more prone to rating movies highly or negatively than others, and some movies may be more likely to receive higher ratings regardless of their actual quality.

Furthermore, exploring techniques such as Random Forests may offer significant improvements in the accuracy of the model. Additionally, Matrix factorization is a popular technique in recommendation systems that seeks to represent the user-item matrix as a product of two lower-dimensional matrices. This technique can effectively capture latent features that may be difficult to identify using traditional modeling techniques. By incorporating matrix factorization into the current model, it may be possible to further improve its accuracy and identify additional biases that were not captured in the current model.

Data Sources:

Dataset is from Grouplens website:

https://grouplens.org/datasets/movielens/25m/

Source Code:

We used the source code from towardsdatascience for reference

https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109

Citations:

1. Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted **collaborative filtering model**. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 426-434). ACM.

2. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). **Item-based collaborative filtering recommendation algorithms**. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295). ACM.

3. Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In 2008 Eighth IEEE International Conference on Data Mining (pp. 263-272). IEEE.