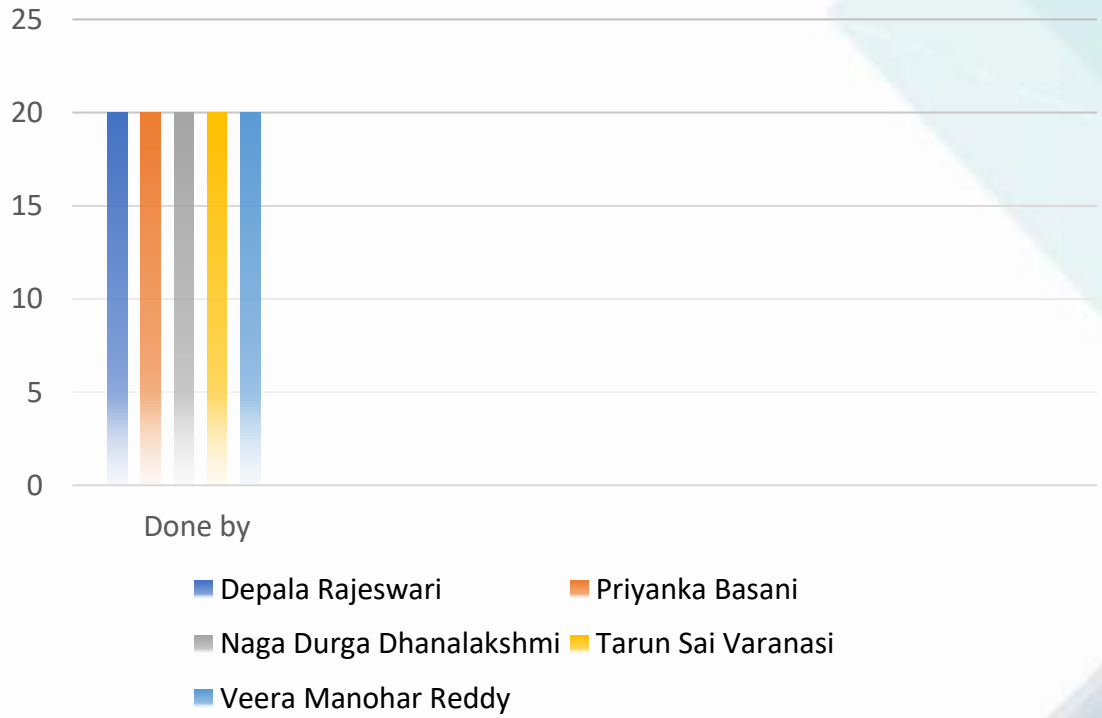
Movie Recommendation System



Agenda

- Introduction
- Data collection
- Data Preparation
- Data exploration/ analysis
- Data Modelling
- Evaluation
- Conclusion

Introduction

Why movie recommendation system ?

- Movie recommendation algorithms help individuals find new films based on their interests and watching habits, saving time and effort.

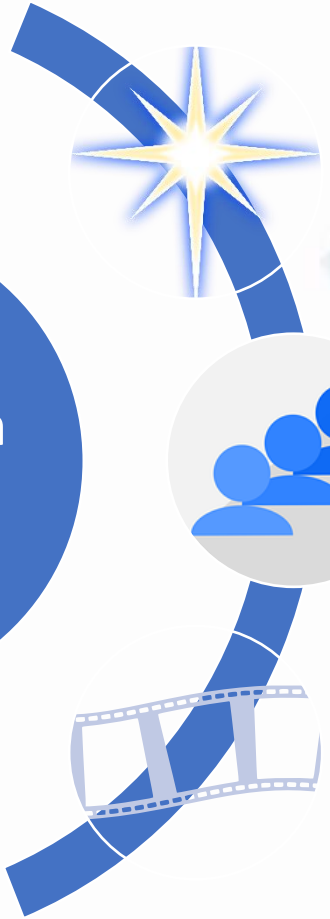
- Enhances user engagement and increase the likelihood of continued platform usage, leading to higher revenue from membership fees or advertising for movie streaming services.

Introduction

- **Specific Questions to address:**
- Which movie is best rated based on the user reviews?
- Which genres are the most popular among the movies in the dataset?
- Which users have rated the most movies in the dataset?
- Are there any relationships between the age or genders of users and their ratings?
- Can we predict the rating a user will give to a movie based on their previous ratings and other factors?
- Which movies are most similar to each other based on user ratings?

Data Collection

Movie Lens dataset from Grouplens website



Ratings : 25M

Users : 200,000

Movies : 62,000

Movie Metadata

- title
- release date
- genre

User Ratings

- timestamp
- user ID
- Movie ID

Data Preparation

- **Building the DataFrame “ Ratings”**

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.

```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)
```

Continued

- **Merging two datasets**

Before undertaking analysis or modeling, combine movie information and user ratings into a single dataset using a common key, such as the movie ID.

```
colnames(movies) <- c("movieId", "title", "genres")

movies <- as.data.frame(movies) %>%
  mutate(movieId = as.numeric(unique(movieId)),
         title = as.character(title),
         genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
```

Continued

- **Partitioning the dataset into training and testing sets.**

Make a validation set for model validation by randomly picking 10% of the data. The validation set is cleaned to guarantee that its userId and movieId are present in the training data.

```
test_index <-createDataPartition(y = movielens$rating, times = 1, p = 0.1, list
= FALSE)
edx <-movielens[-test_index,]
temp <-movielens[test_index,]
#Check userId and movieId present in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

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

Continued

- **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)`
```

Analyzing the Data

Overview of data

```
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…
```

Determining the total number of **distinct** users, films, and genres in the data.

distinct_users <int>	distnict_movies <int>	distinct_genres <int>
69878	10677	797

Showing the number of ratings for each rating level is displayed, along with an analysis of the dataset's **rating distribution**

```
edx %>% group_by(rating) %>% summarize(ratings_sum = n()) %>%
  arrange(desc(ratings_sum))

## # A tibble: 10 × 2
##    rating ratings_sum
##     <dbl>       <int>
##  1    4       2588430
##  2    3       2121240
##  3    5       1390114
##  4    3.5      791624
##  5    2        711422
##  6    4.5      526736
##  7    1        345679
##  8    2.5      333010
##  9    1.5      106426
## 10    0.5       85374
```

Sum of movie ratings per genre

Grouping the data by genre is useful in determining which movie genres are more well-liked by viewers and have garnered more ratings.

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

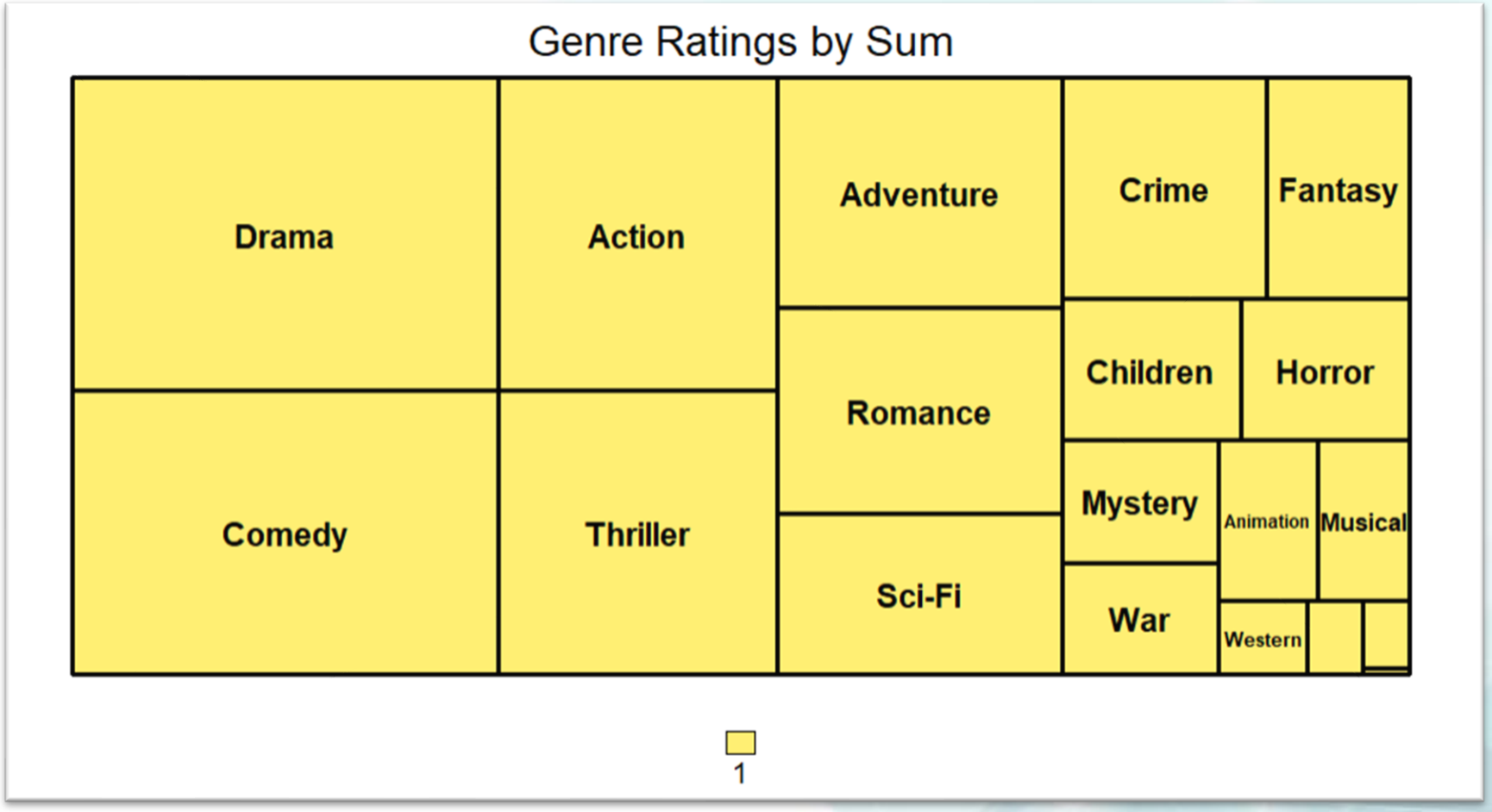
## # A tibble: 20 × 3
##    genres              Ratings_Sum Average_Rating
##    <chr>                     <int>          <dbl>
##  1 Drama                   3910127           3.67
##  2 Comedy                  3540930           3.44
##  3 Action                  2560545           3.42
##  4 Thriller                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
```

Why treemap?

Simple to compare the overall rating values of various genres

- Determines the most well-liked genres and how many ratings each one has earned compares to the others.

- Examining the connections between genres and the general rating distribution.



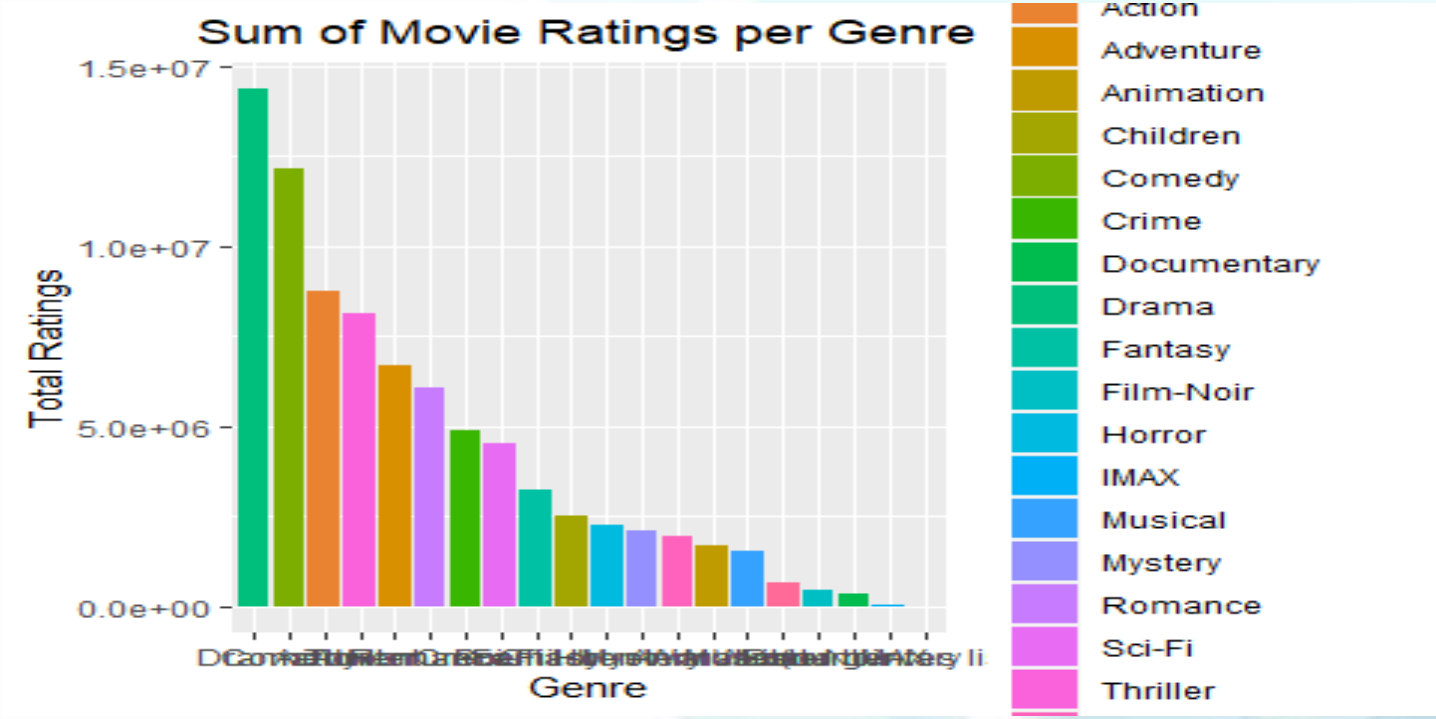
The genre with the most ratings is drama, followed by comedy and action. Based on the ratings added together, the treemap offers a rapid visual comparison of the popularity of various genres.

Why Calculating the total movie ratings per genre?

- To get an idea of the general popularity of various genres and how they contrast with one another.

- To view trends and patterns in viewers' tastes for movies through time and enables us to comprehend which genres are the most well-liked by audiences depending on the quantity of ratings.

Comparing Sum of Ratings Across Genres

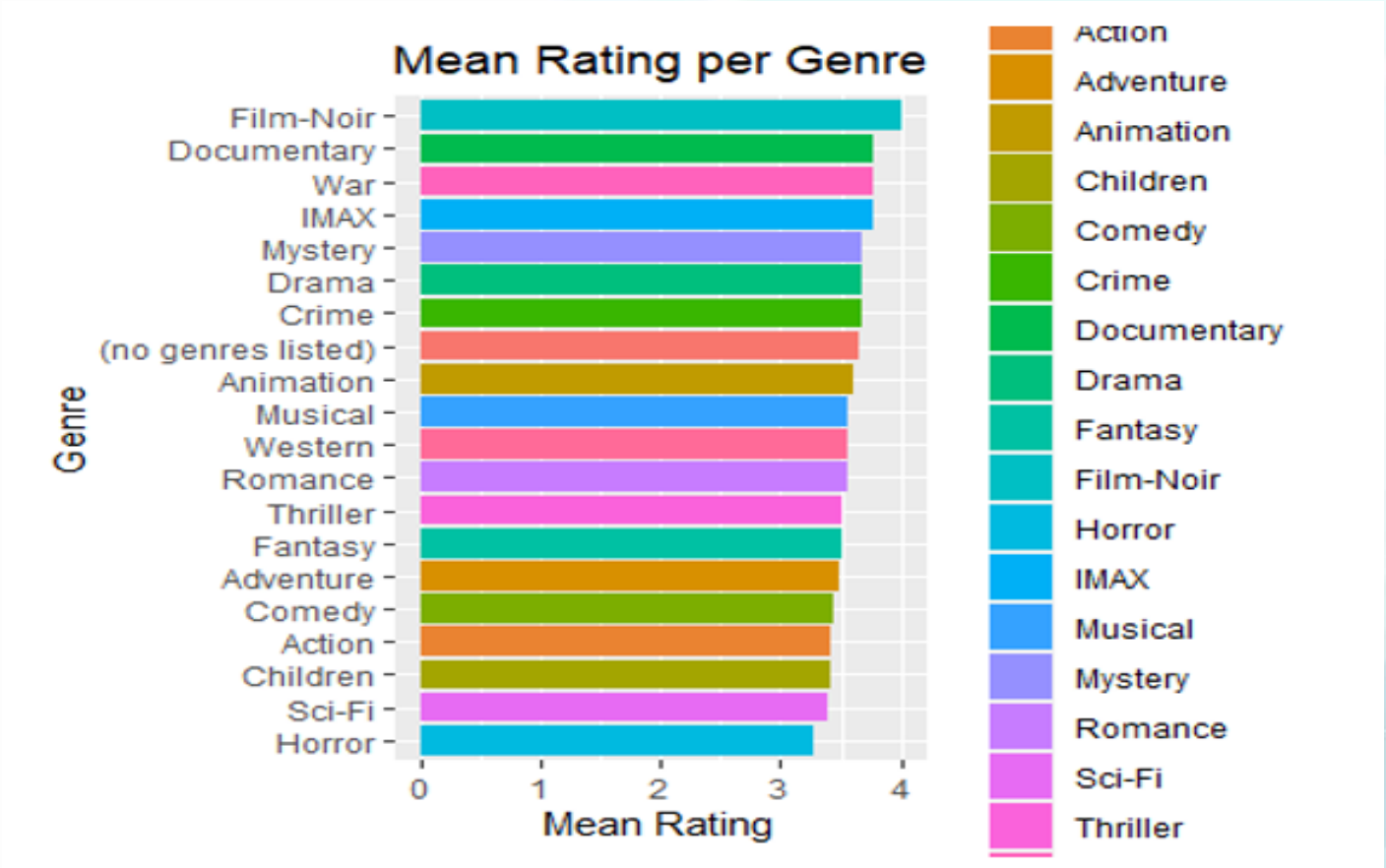


The genres with the highest sum of ratings are Drama, Comedy, Action, Thriller, and Adventure, while the genres with the lowest sum of ratings are Film-Noir, Western, Musical, Documentary, and IMAX.

(Baseline model) Mean rating per year

- Examining historical patterns in movie ratings.
- It can show if movie ratings have been consistently high or low throughout time, as well as whether there are any trends or swings in ratings that might be connected to certain occasions or adjustments in the movie business.

Exploring Trends Over Time



Summary of Exploring Trends Over Time

- The greatest mean ratings are for Film-Noir, Documentary, and War, while the lowest mean ratings are for Horror, Sci-Fi, and Children.

- There are large differences in the mean scores for various genres, with a range of around **3 to 4.1**

- Drama, Comedy, and Thriller have lower mean ratings than certain other genres while receiving the most ratings.

Summary of MovieAge

```
summary(edx$MovieAge)

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

- The MovieAge variable in the edx dataset is then given descriptive statistics using the summary() method.
- The median age of the films is 26 years, and the mean age is 29.78 years.
- The oldest film in the collection was released 105 years ago.
- The dataset's newest film is twelve years old.

Data Modelling

Collaborative Filtering Technique

Implementing a **matrix factorization** *model*

- The **Alternating Least Squares (ALS) algorithm** which is a collaborative filtering technique would utilize user-provided prior movie ratings to forecast how probable it is that they will rate other movies in the future.

- The model is evaluated on the edx_test dataset, which comprises a collection of ratings that were with held from the training set, after having been trained on the edx_train dataset, which contains a set of user-item interactions.

Data Modelling

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.

Data Modelling

RMSE/Loss Funtion

We can assess how effectively the recommendation algorithm predicts user ratings and generates reliable recommendations. A **lower RMSE** suggests **more accurate** prediction.

$$\sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

```
##   <chr>                                                <chr>
## 1 NRMSE                                                1.06005
## 2 Median_Model                                         1.16676
## 3 Movie Effects                                        0.94296
## 4 Movie & User Effects                                 0.86468
## 5 User, Movie & Movie Age Effects                      0.86433
## 6 Movie & User Effects w/Regularization                0.86414
## 7 User, Movie & Movie Age Effects Regularization       0.86384
```

Data Modelling

- The Non-Personalized Model is the least accurate model with the greatest RMSE.
- The Median Model performs better than the Non-Personalized Model but not as well as the models that consider user and/or movie impacts.
- Adding movie effects improves the accuracy of the model, while including both user and movie effects further enhances the accuracy.
- The **most accurate model** is obtained by **including movie age effects** along **with user and movie effects**. Therefore, the best model for predicting movie ratings includes both user and movie effects as well as movie age effects.

Regularization

- *Movie & User Effects*
- *Movie, User & Movie Age Effects*

❑ It is an extension of the Movie & User Effects Model, which takes into account both the average rating of a movie and the average rating of a user.

❑ The model with regularization is used to prevent overfitting, which is a common problem when fitting models to large datasets.

❑ The model with regularization is trained using a training set and then tested on a validation set.

❑ The regularization parameter is chosen using **cross-validation,** which involves splitting the training set into several subsets and using each subset as a validation set in turn.

Regularization

- *Movie & User Effects*

- The model includes a tuning parameter lambda λ tominimize the RMSE and penalize outliers from bi and bu, which are the movie and user effects, respectively.

```
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 ...
```

Movie & User Effects with Regularization (Validation):

```
lambda

## [1] 5
```

- The sapply function is used to apply the regularization function.
- The lambda value that minimizes the RMSE is determined using the which .min function and saved in the variable lambda.

L2 Regularization

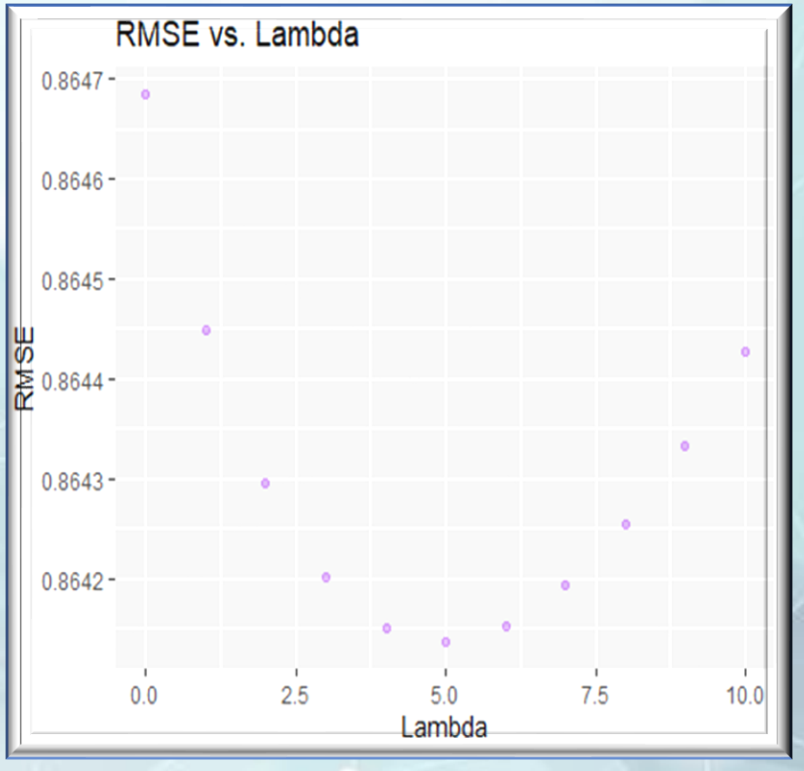
- L2 regularization (also called **Ridge Regression)** by adding an L2 penalty term to the loss function to control the model complexity.

- 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.

Regularization

- *Movie & User Effects*
  - ❑ Regularization can help reduce overfitting by penalizing outliers in the model, as evident from the drop in RMSE as lambda rises. Lambda=5
  - ❑ **Why to select optimal lambda value?**
    - Minimizes RMSE
    - Improving model performance while avoiding overfitting

Excessive regularization can cause underfitting, leading to an increase in RMSE.



Regularization

- Movie, User & Movie Age Effects

  - ❑ Calculates bias terms for movies, users, and movie age groups and uses them to predict ratings.

```
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 ..
```

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

```
lambda2

## [1] 5
```

  - ❑ The lambda value that yields the lowest RMSE is selected as the optimal value. In this case, the optimal lambda2 value is 5.

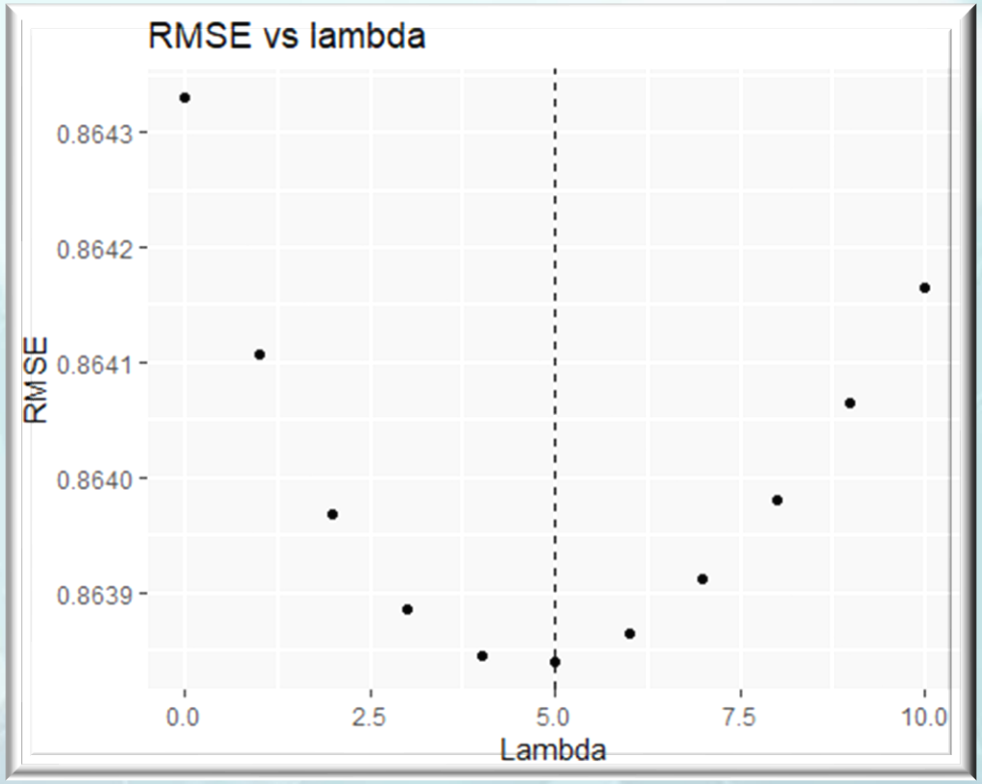
Regularization

- Movie, User & Movie Age Effects

❑ Depicts the correlation between the two variables

❑ Curve illustrates how the RMSE varies as lambda increases

❑ Value of lambda, which results in the lowest RMSE, is shown by the dashed vertical line



Validation(Evaluating the model)

*Training & test set results against those of the **validation set**.*

Model_Type	Final_RMSE_Validation
<chr>	<chr>
1 NRMSE	1.06120
2 Median_Model	1.16802
3 Movie Effects	0.94391
4 Movie & User Effects	0.86535
5 Movie, User, & Movie Age Effects	0.86500
6 Movie & User Effects w/Regularization	0.86482
7 Movie, User & Movie Age Effects w/Regularization	0.86452

Validation(Evaluating the model)

.

- According to the validation RMSE, the final model—the Movie, User, and Movie Age Effects Model with Regularization—performs the best.

- The lowest of all the models assessed, this model's RMSE score is **0.86452.**

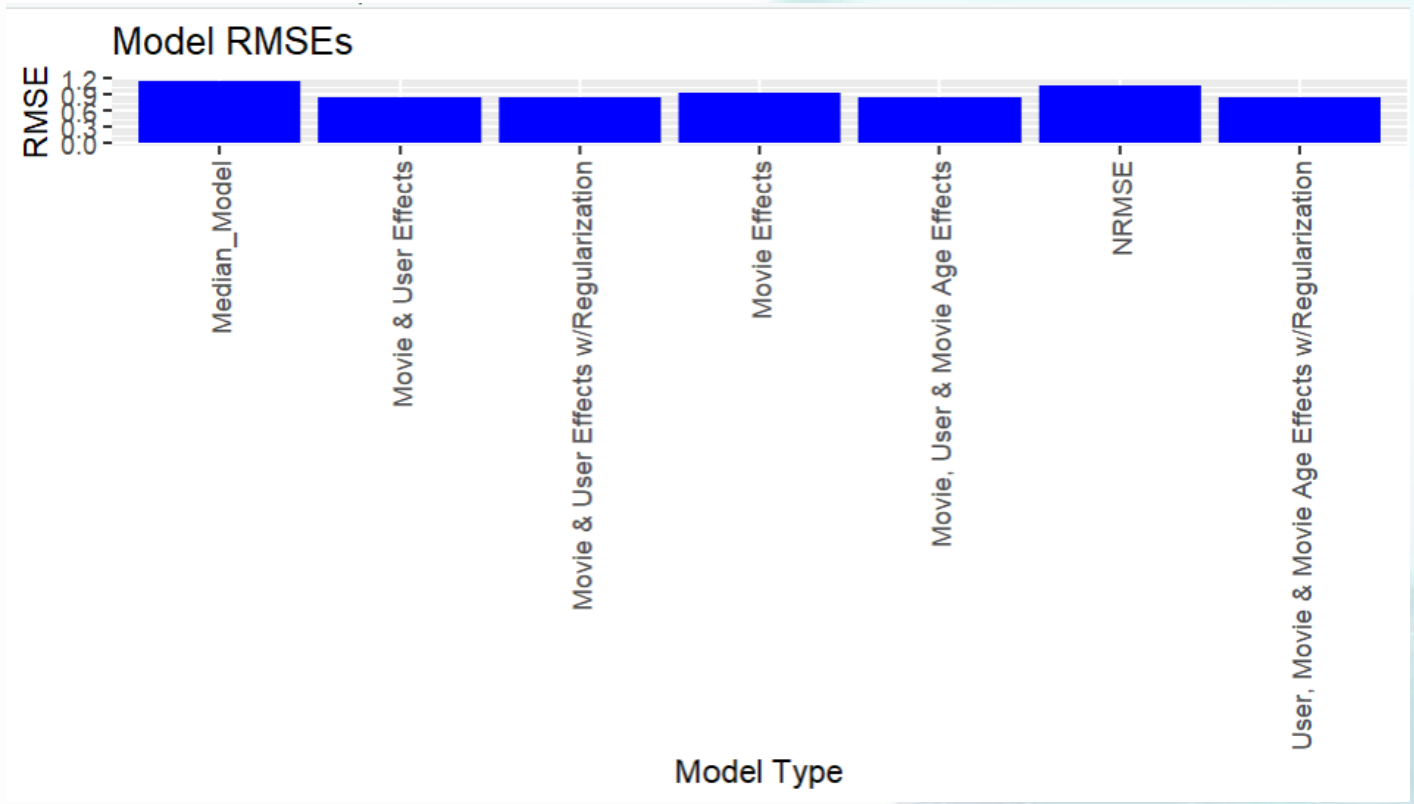
- This indicates that this model's forecasts are the ones that correspond most closely to the actual ratings provided by the users on the validation set

Final Model with Validation

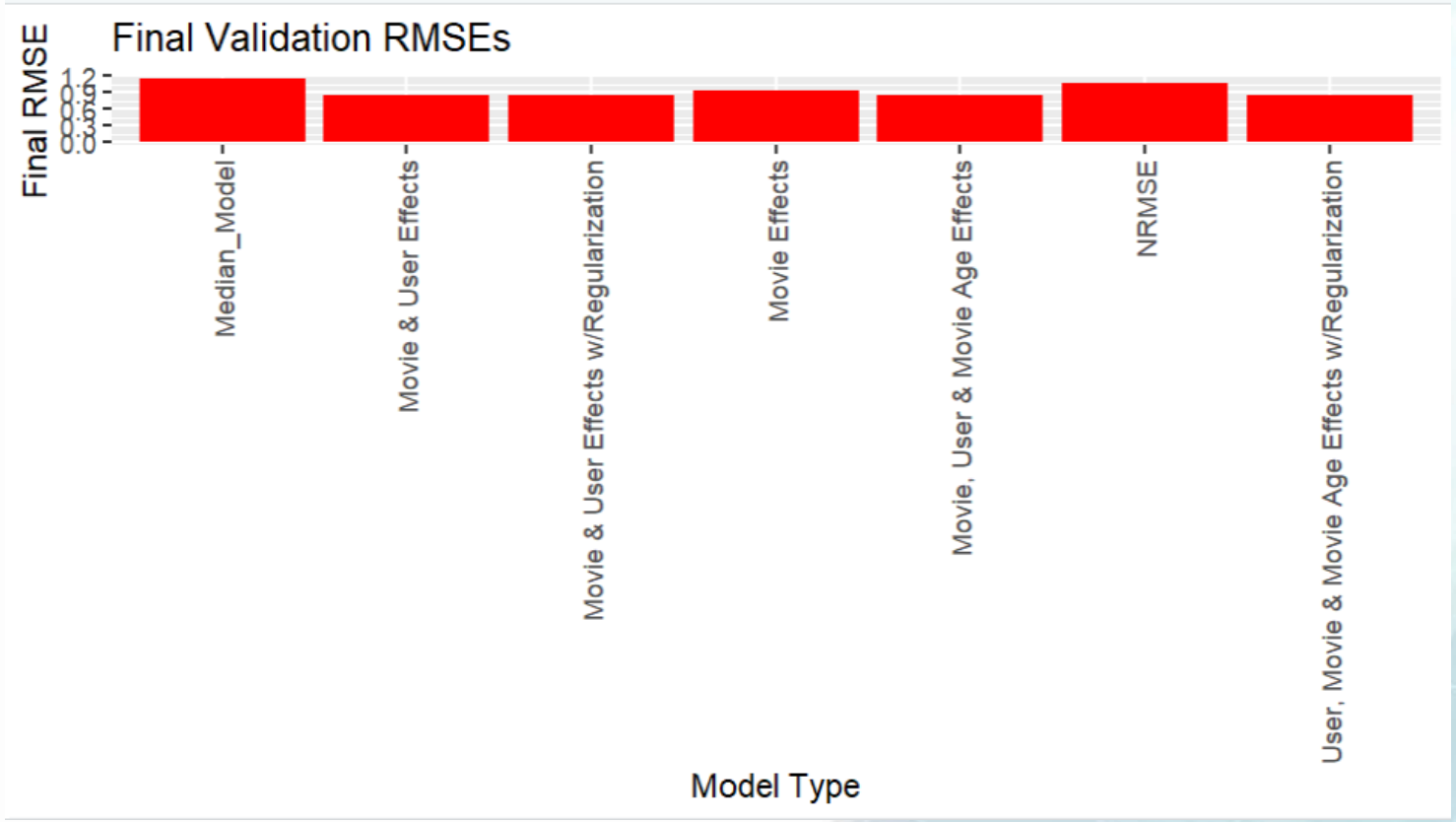
Outcomes of our final model validation for the various models

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

Representation of Model RMSEs



Representation of final Validation RMSEs



Final Model with Validation

Value of 0.86452, the model with User, Movie, and Movie Age Effects with Regularization has the lowest RMSE on the validation set.

```
## 7 User, Movie & Movie Age Effects w/Regularization 0.86384 0.86452
```

As a result, this model is the best model for our objective since it predicts user ratings on unobserved data the most accurately.

This Final Model achieves an RMSE of .86452

Conclusion

- Although the final model, which included User, Movie, and Movie Age Effects with Regularization, produced a satisfactory RMSE of 0.86452, it may be worthwhile to investigate additional biases that could enhance the accuracy of the model.

- Moreover, using more advanced techniques like Random Forests may offer significant improvements in the RMSE.

- Despite achieving an RMSE of .86452, the final model that incorporated User, Movie, and Movie Age Effects with Regularization could benefit from exploring other biases to further enhance its accuracy.