HarvardX PH125.9x Data Science Capstone: MovieLens Predictive Model

Anwar A. AbuAlhussein

2025-03-26

Introduction

This project is part of the HarvardX PH125.9x Data Science Capstone, where we aim to build a predictive model for movie ratings using the MovieLens dataset.

The MovieLens dataset is a widely used benchmark dataset in the field of recommendation systems. It contains user-movie interactions in the form of explicit ratings, ranging from 1 to 5 stars, where users express their preferences for movies. The dataset also includes metadata such as movie titles and genres, which can be leveraged for content-based recommendations.

The objective is to develop an algorithm that accurately predicts user ratings for movies while minimizing the Root Mean Square Error (RMSE).

Step 1: Download and Extract Data

```r
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

movielens <- left_join(ratings, movies, by = "movieId")

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
# set.seed(1) # if using R 3.5 or earlier
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
```

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

```r
edx <- rbind(edx, removed)
```

Step 2: Exploratory Data Analysis (EDA)

To see the basic information about the data, dimension, types and to see number of missing entries on each column.

DataSet Structure

```r
class(edx)
```

```
## [1] "data.frame"
```

```r
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   <int> 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~
```

```r
colnames(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

There are 9,000,055 obs of 6 variables “userID”, “movieID”, “rating”, “timestamp”, “title”, and “genres”.

The edx dataset contains the following columns:

Quantitative features
-userId : discrete, Unique ID for the user.
-movieId: discrete, Unique ID for the movie.
-timestamp : discrete , Date and time the rating was given.

Qualitative features
-title: nominal , movie title (not unique)
-genres: nominal, genres associated with the movie.

Outcome,y
-rating : continuous, a rating between 0 and 5 for the movie.

```r
edx %>%
  head(5) %>%
  kable(booktabs = TRUE, caption = "\\textcolor{blue}{Head Rows}", align = "c") %>%
  kable_styling(position = "center", latex_options = c("striped", "scale_down"))
```

```
## Warning in styling_latex_scale(out, table_info, "down"): Longtable cannot be
## resized.
```

Table 1: Head Rows

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy&#124;Romance
2	1	185	5	838983525	Net, The (1995)	Action&#124;Crime&#124;Thriller
4	1	292	5	838983421	Outbreak (1995)	Action&#124;Drama&#124;Sci-Fi&#124;
5	1	316	5	838983392	Stargate (1994)	Action&#124;Adventure&#124;Sci-
6	1	329	5	838983392	Star Trek: Generations (1994)	Action&#124;Adventure&#124;Drama&#1

Check basic statistics

```r
summary_edx <- summary(edx)

kable(summary_edx, booktabs = TRUE, caption = "Summary Statistics") %>%
  kable_styling(position = "center", latex_options = c("scale_down", "striped"))
```

```
## Warning in styling_latex_scale(out, table_info, "down"): Longtable cannot be
## resized.
```

Table 2: Summary Statistics

userId	movieId	rating	timestamp	title	genres
Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:9000055	Length:9000055
1st Qu.:18124	1st Qu.: 648	1st Qu.:3.000	1st Qu.:9.468e+08	Class :character	Class :character
Median :35738	Median : 1834	Median :4.000	Median :1.035e+09	Mode :character	Mode :character
Mean :35870	Mean : 4122	Mean :3.512	Mean :1.033e+09	NA	NA
3rd Qu.:53607	3rd Qu.: 3626	3rd Qu.:4.000	3rd Qu.:1.127e+09	NA	NA
Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09	NA	NA

Count total missing values in the dataset

```r
total_missing <- sum(is.na(edx))
colSums(is.na(edx))
```

```
##    userId   movieId    rating timestamp     title    genres
##         0         0         0         0         0         0
```

```r
cat("Total missing values in the dataset:", total_missing, "\n")
```

```
## Total missing values in the dataset: 0
```

The dataset is clean with no missing values.

Count unique values for the "users", "movies" and "genre" variables.

```r
num_users <- n_distinct(edx$userId)
num_movies <- n_distinct(edx$movieId)
num_genres<- n_distinct(edx$genres)
cat("Number of users:", num_users, "\n")
```

```
## Number of users: 69878
```

```
cat("Number of movies:", num_movies, "\n")
```

```
## Number of movies: 10677
```

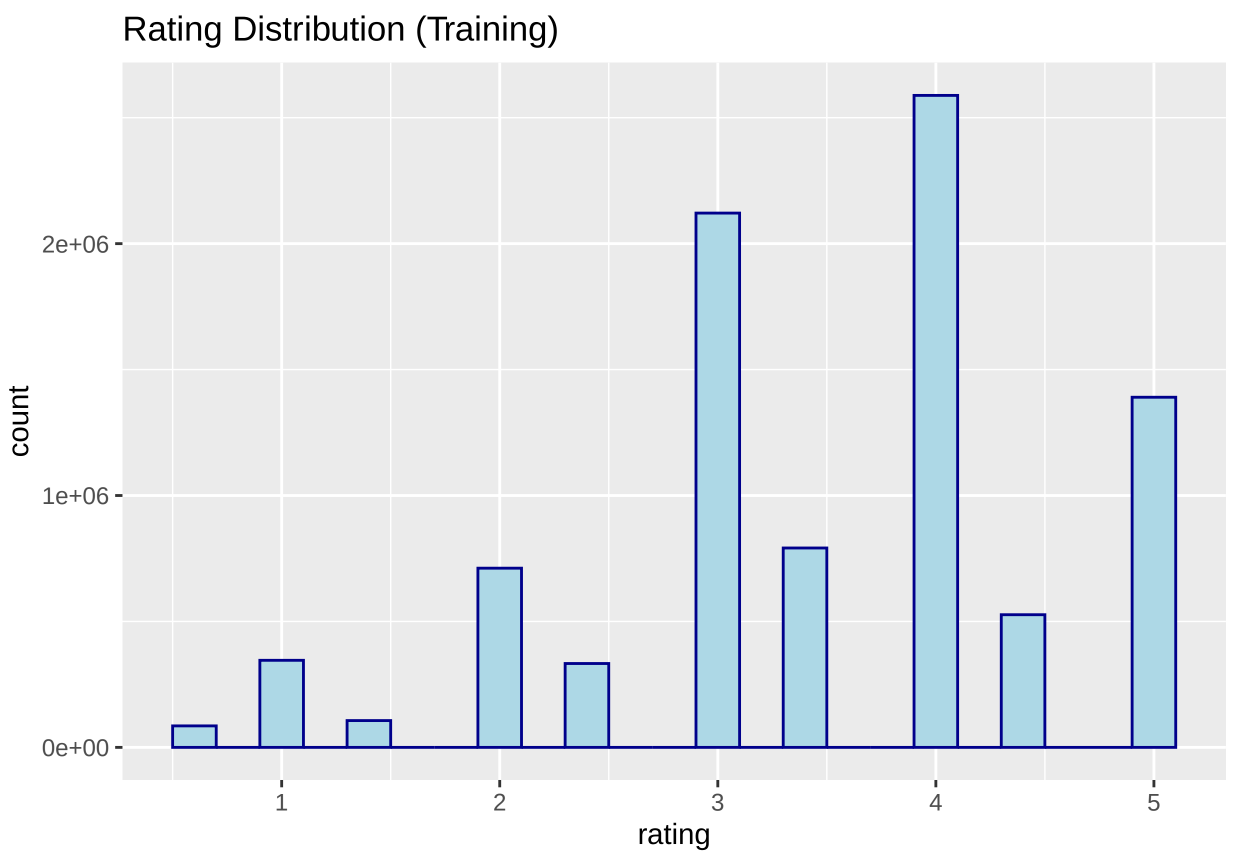
```
cat("Number of genres:", num_genres, "\n")
```

```
## Number of genres: 797
```

The data contains 9000055 ratings applied to 10677 different movies of 797 different unique or combination of genre from 69878 single users.

Following is a bar chart showing the distribution of rating.

```
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth=0.2, color="darkblue", fill="lightblue") +
  ggtitle("Rating Distribution (Training)")
```



```
summary(edx$rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

```
edx %>% group_by(rating) %>% summarize(number_rows=n())
```

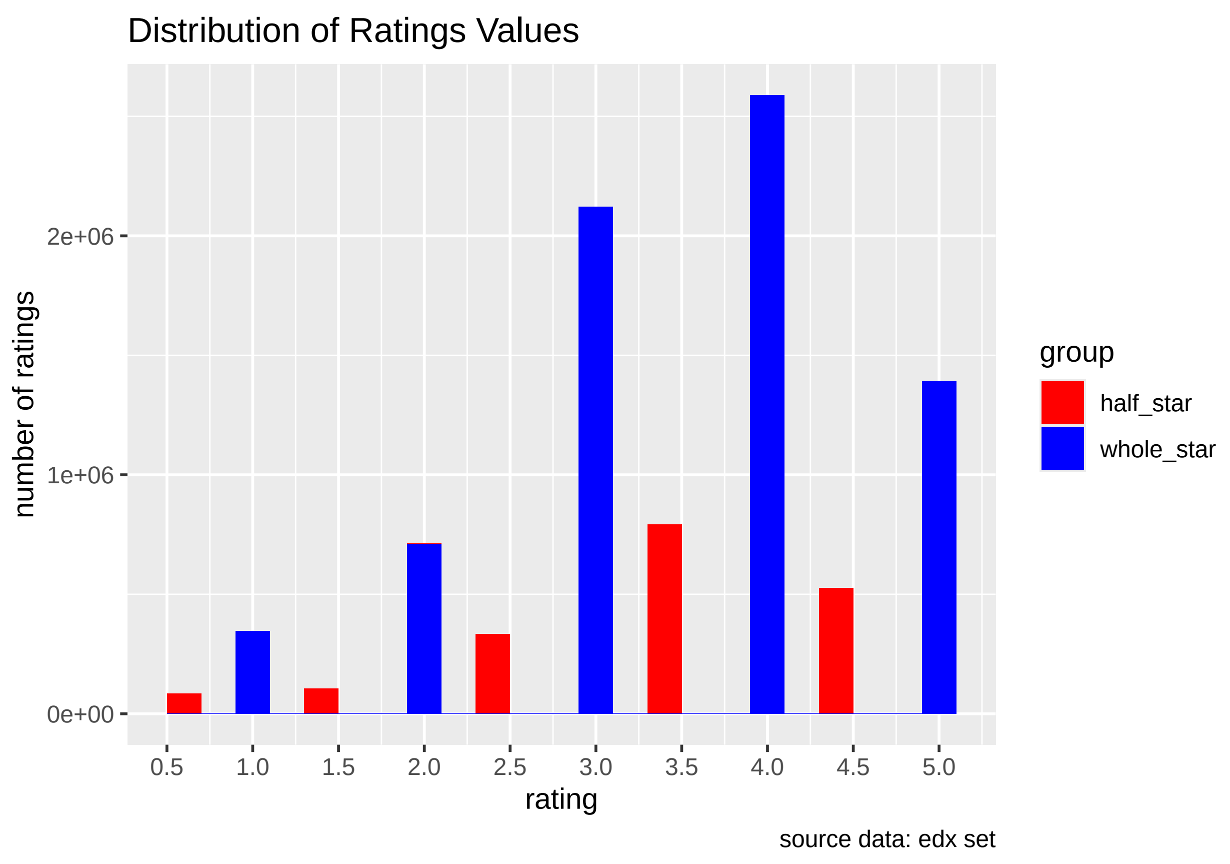
```
## # A tibble: 10 x 2
##    rating number_rows
##     <dbl>       <int>
##  1    0.5       85374
##  2    1        345679
##  3    1.5      106426
##  4    2        711422
##  5    2.5      333010
##  6    3       2121240
##  7    3.5      791624
##  8    4       2588430
##  9    4.5      526736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

Create a dataframe "explore_edx_ratings" which contains half star and whole star ratings from the edx

```
group <- ifelse((edx$rating == 1 |edx$rating == 2 |edx$rating == 3 |
                   edx$rating == 4 | edx$rating == 5),
                "whole_star","half_star")
explore_edx_ratings <- data.frame(edx$rating, group)
```

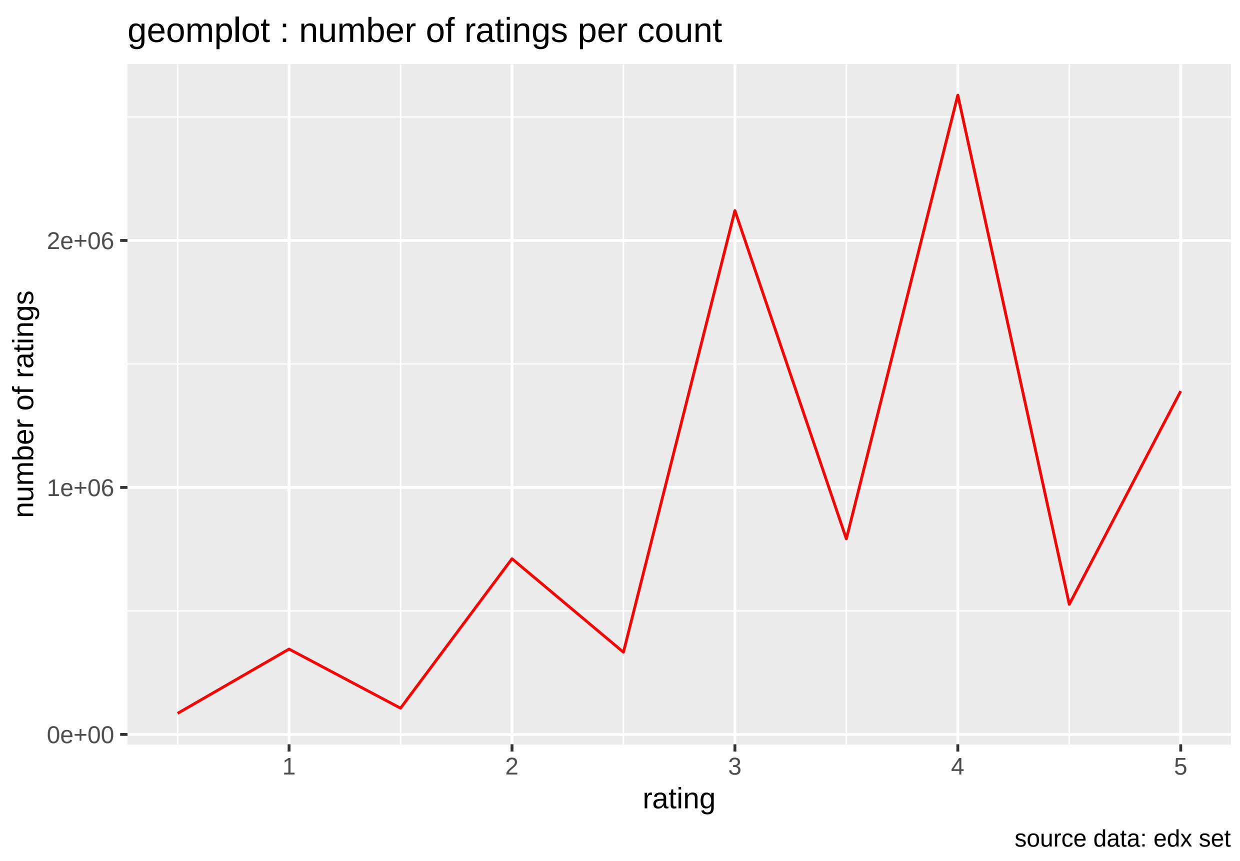
Distribution of Ratings Values

```
ggplot(explore_edx_ratings, aes(x= edx.rating, fill = group)) +
  geom_histogram( binwidth = 0.2) +
  scale_x_continuous(breaks=seq(0, 5, by= 0.5)) +
  scale_fill_manual(values = c("half_star"="red", "whole_star"="blue")) +
  labs(x="rating", y="number of ratings", caption = "source data: edx set") +
  ggtitle("Distribution of Ratings Values ")
```



Geom_line showing rating

```
edx %>%group_by(rating) %>%summarize(count = n()) %>%ggplot(aes(x = rating, y = count)) +geom_line(colo
  labs(x="rating", y="number of ratings", caption = "source data: edx set") +ggtitle("geomplot : number
```



Summary of five rating count

```
edx %>% group_by(rating) %>% summarize(count = n()) %>% top_n(5, count) %>%arrange(desc(count))
```

```
## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1    4   2588430
## 2    3   2121240
## 3    5   1390114
## 4    3.5  791624
## 5    2    711422
```

Exploring ratings of the edx dataset, it is observed that the average user's activity reveals that no user gives 0 as rating, the top 5 ratings from most to least are : 4, 3, 5, 3.5 and 2.

The Distribution of Ratings Values shows that the halfstar ratings are less common than whole star ratings.

Qualitative features: genres, title

Now, we are going to explore the features "genres" and "title" of our edx set.

##########Genres#########

Using str_detect for selected genres.

```
drama <- edx %>% filter(str_detect(genres,"Drama"))
comedy <- edx %>% filter(str_detect(genres,"Comedy"))
thriller <- edx %>% filter(str_detect(genres,"Thriller"))
romance <- edx %>% filter(str_detect(genres,"Romance"))
cat('Drama has', nrow(drama), 'movies\n')
```

```
## Drama has 3910127 movies
```

```
cat('Comedy has', nrow(comedy), 'movies\n')
```

```
## Comedy has 3540930 movies
```

```
cat('Thriller has', nrow(thriller), 'movies\n')
```

```
## Thriller has 2325899 movies
```

```
cat(' Romance has', nrow( romance), 'movies\n')
```

```
##  Romance has 1712100 movies
```

```
rm(drama, comedy, thriller, romance)
```

Find the top 10 most rated movies:

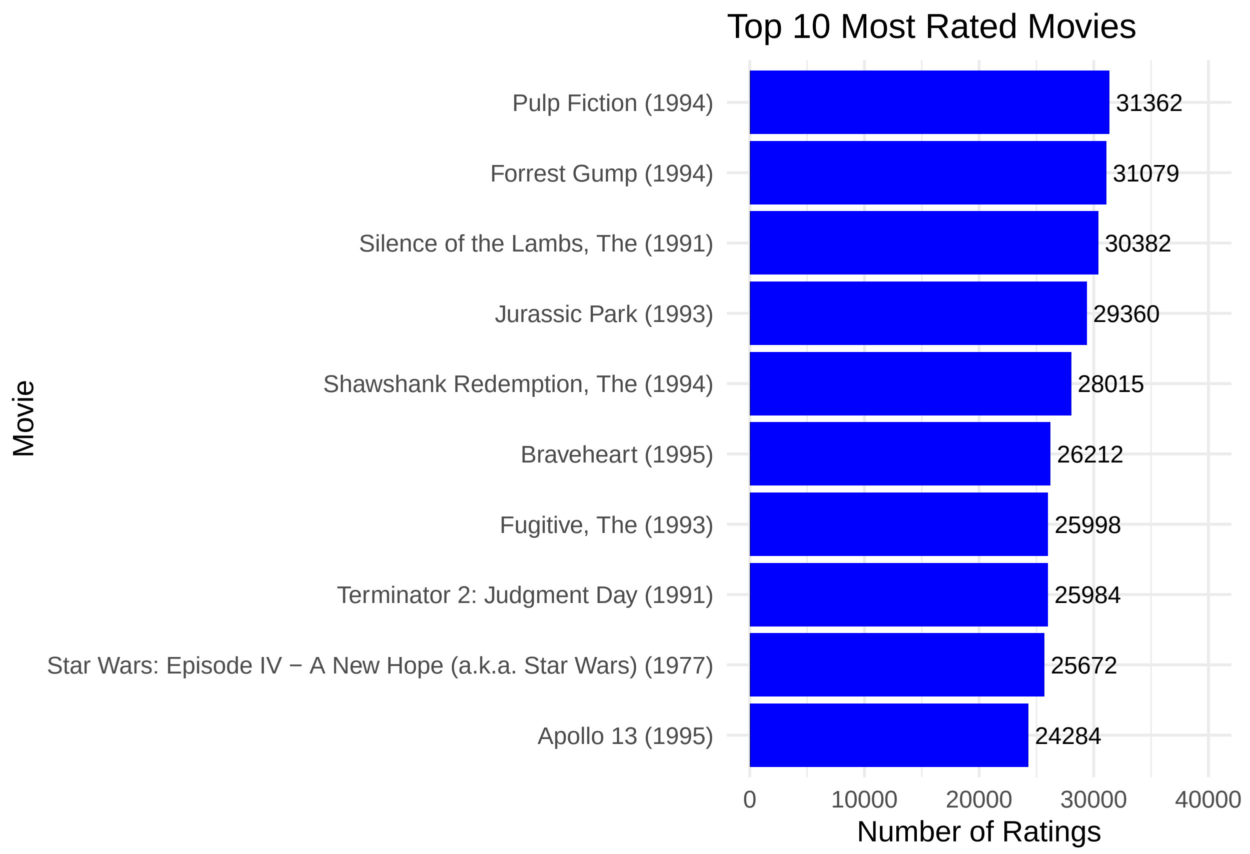
```
top_movies <- edx %>%
  group_by(title) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  head(10)

print(top_movies)
```

```
## # A tibble: 10 x 2
##    title                                                   count
##    <chr>                                                   <int>
##  1 Pulp Fiction (1994)                                     31362
##  2 Forrest Gump (1994)                                     31079
##  3 Silence of the Lambs, The (1991)                        30382
##  4 Jurassic Park (1993)                                    29360
##  5 Shawshank Redemption, The (1994)                        28015
##  6 Braveheart (1995)                                       26212
##  7 Fugitive, The (1993)                                    25998
##  8 Terminator 2: Judgment Day (1991)                       25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10 Apollo 13 (1995)                                        24284
```

Bar chart of top_title

```r
ggplot(top_movies, aes(x = reorder(title, count), y = count)) +
  geom_bar(stat = "identity", fill = "blue") +
  coord_flip(y=c(0, 40000)) +labs(x="", y="Number of ratings") +
  geom_text(aes(label= count), hjust=-0.1, size=3) +
  labs(title = "Top 10 Most Rated Movies", x = "Movie", y = "Number of Ratings") +
  theme_minimal()
```



Top ten title and genres

```r
edx %>%
  group_by(title, genres) %>%
  summarize(count = n(), .groups = "drop") %>%
  slice_max(order_by = count, n = 10) %>%
  arrange(desc(count))
```

```
## # A tibble: 10 x 3
##    title                                                                  genres     count
##    <chr>                                                                  <chr>      <int>
##  1 Pulp Fiction (1994)                                                    Comedy|Cr~ 31362
##  2 Forrest Gump (1994)                                                    Comedy|Dr~ 31079
##  3 Silence of the Lambs, The (1991)                              Crime|Hor~ 30382
##  4 Jurassic Park (1993)                                          Action|Ad~ 29360
##  5 Shawshank Redemption, The (1994)                              Drama      28015
##  6 Braveheart (1995)                                             Action|Dr~ 26212
##  7 Fugitive, The (1993)                                          Thriller   25998
##  8 Terminator 2: Judgment Day (1991)                             Action|Sc~ 25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) Action|Ad~ 25672
## 10 Apollo 13 (1995)                                              Adventure~ 24284
```

```
kable(head(edx %>%
             group_by(title, genres) %>%
             summarize(count = n(), .groups = "drop") %>%  # Explicitly drop grouping
             top_n(10, count) %>%
             arrange(desc(count)), 5)) %>%
  kable_styling(bootstrap_options = "bordered", full_width = F, position = "center") %>%
  column_spec(1, bold = T) %>%
  column_spec(2, bold = T) %>%
  column_spec(3, bold = T)
```

title	genres	count
Pulp Fiction (1994)	**Comedy&#124;Crime&#124;Drama**	**31362**
Forrest Gump (1994)	**Comedy&#124;Drama&#124;Romance&#124;War**	**31079**
Silence of the Lambs, The (1991)	**Crime&#124;Horror&#124;Thriller**	**30382**
Jurassic Park (1993)	**Action&#124;Adventure&#124;Sci-Fi&#124;Thriller**	**29360**
Shawshank Redemption, The (1994)	**Drama**	**28015**

The movies which have the highest number of ratings are in the top genres categories : movies like Pulp fiction (1994), ForrestGump(1994) or Jurrasic Park(1993) which are in the top 5 of movie's ratings number , are part of theDrama, Comedy or Action genres.

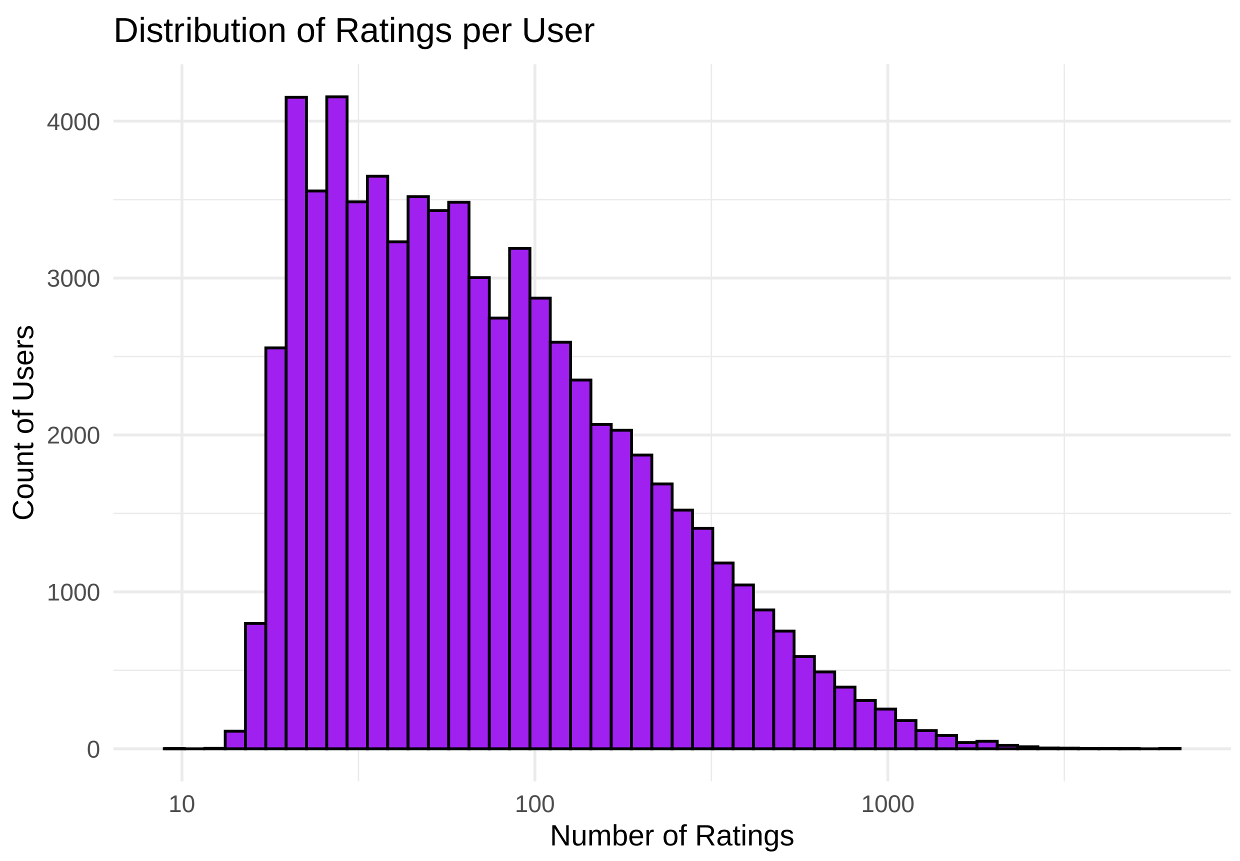
Number of Ratings per User

Check how many ratings each user has given:

```
# Ratings per user
user_ratings <- edx %>%
  group_by(userId) %>%
  summarise(count = n())
```

Histogram of user ratings count

```
ggplot(user_ratings, aes(x = count)) +
  geom_histogram(bins = 50, fill = "purple", color = "black") +
  scale_x_log10() +  # Log scale for better visualization
  labs(title = "Distribution of Ratings per User", x = "Number of Ratings", y = "Count of Users") +
  theme_minimal()
```

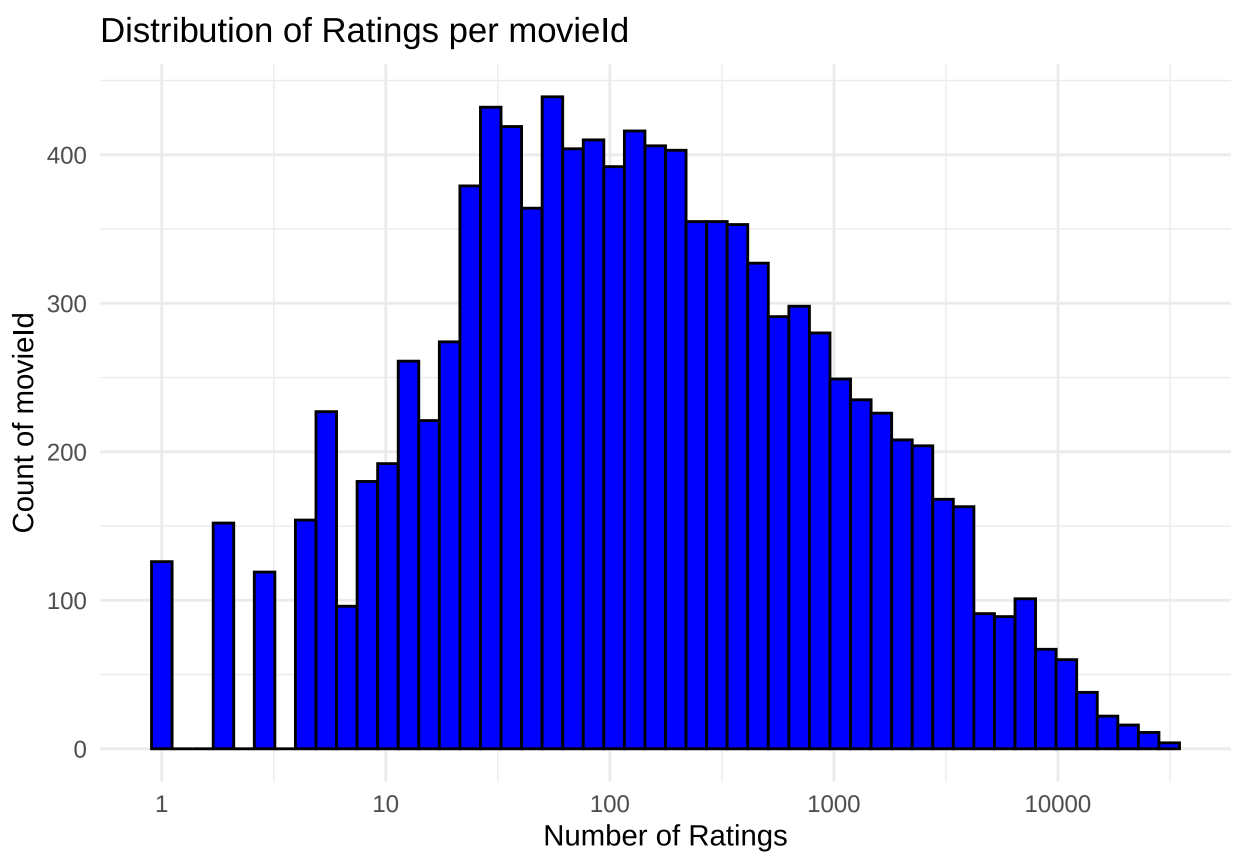


Ratings per movieId

```r
movieId_ratings <- edx %>%
  group_by( movieId ) %>%
  summarise(count = n())
```

Histogram of movieId ratings count

```r
ggplot(movieId_ratings, aes(x = count)) +
  geom_histogram(bins = 50, fill = "blue", color = "black") +
  scale_x_log10() +  # Log scale for better visualization
  labs(title = "Distribution of Ratings per movieId", x = "Number of Ratings", y = "Count of movieId")
  theme_minimal()
```



The distribution of the number of ratings by movieId and by userId shows the following relationships : some movies get rated more than others, and some users are more active than others at rating movies. This explains the presence of movies and users effect.

Ratings Over Time

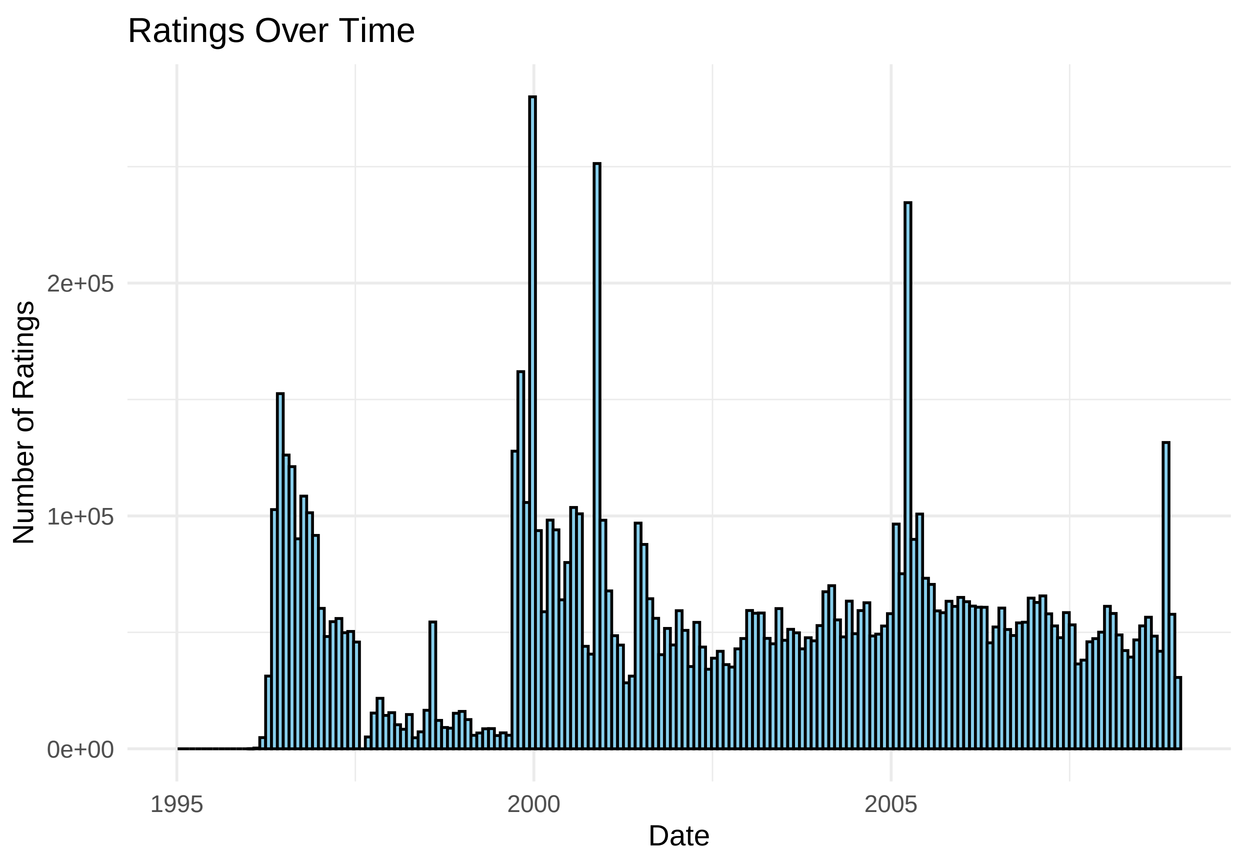
Ratings change over time:

Convert timestamp to date format.

```r
edx <- edx %>%
  mutate(date = as.Date(as.POSIXct(timestamp, origin = "1970-01-01")))
```

Plot ratings over time

```r
ggplot(edx, aes(x = date)) +
  geom_histogram(binwidth = 30, fill = "skyblue", color = "black") +
  labs(title = "Ratings Over Time", x = "Date", y = "Number of Ratings") +
  theme_minimal()
```



This analysis helps us understand if time-based effects influence movie ratings. If we notice strong time trends, we might consider adding a time-based feature to our model later.

"The Ratings over time", shows that time has a weak effect on the ratings.

Yearly Trends in Ratings

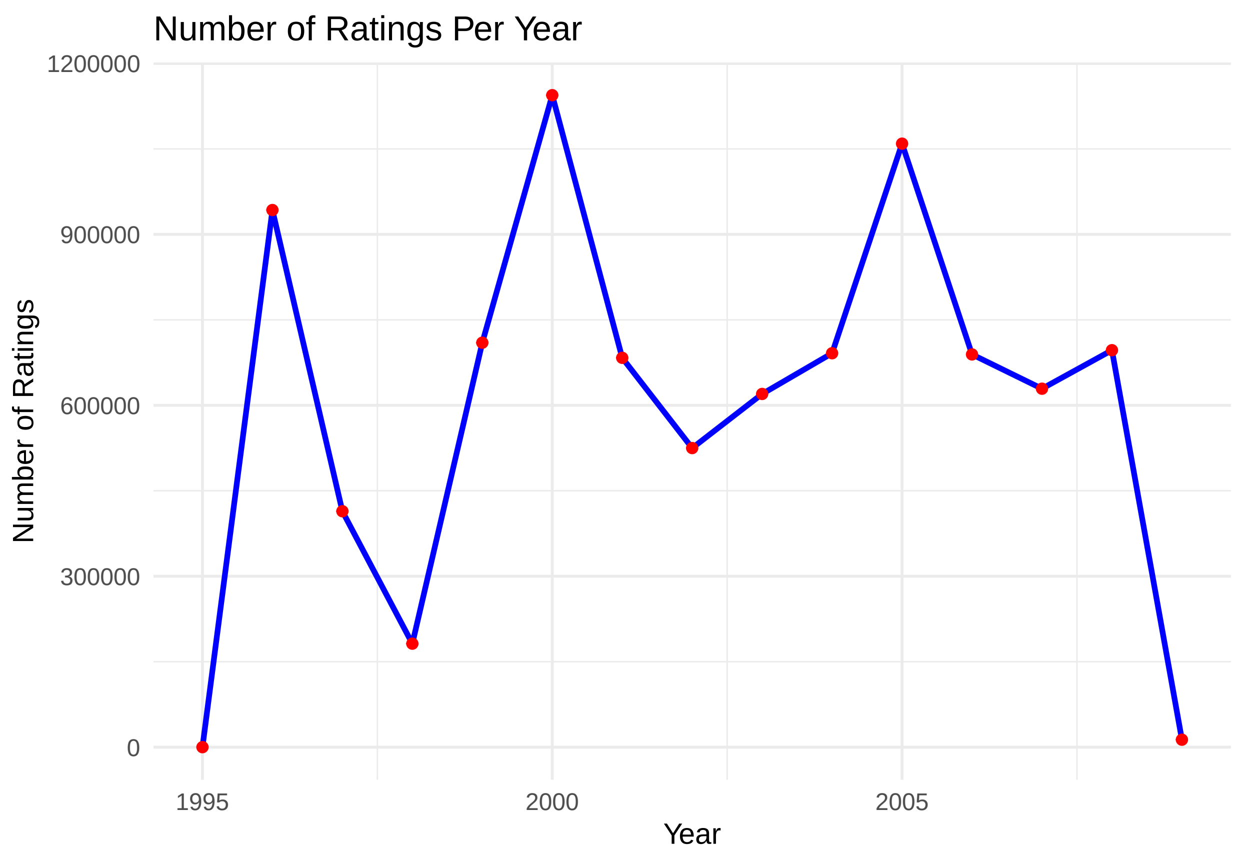
Extract year from the date column

```r
edx <- edx %>%
  mutate(year = format(date, "%Y"))
```

Aggregate ratings per year

```r
yearly_ratings <- edx %>%
  group_by(year) %>%
  summarise(count = n()) %>%
  arrange(year)
```

```r
ggplot(yearly_ratings, aes(x = as.numeric(year), y = count)) +
  geom_line(color = "blue", linewidth = 1) +
  geom_point(color = "red") +
  labs(title = "Number of Ratings Per Year", x = "Year", y = "Number of Ratings") +
  theme_minimal()
```



Despite the number of ratings fluctuating, the rating distribution has not changed significantly over time.

Step 3: Modelling and Regularization

The goal is to predict the rating column for unseen data in the final_holdout_test set.

Following the EDA we conducted, we will focus on the the effects in relation with the "users" and "movies".

Define RMSE function

Root Mean Squared Error (RMSE) is our evaluation metric.

```
RMSE <- function(true_ratings, predicted_ratings) {
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Split edx into training and validation set

```r
set.seed(1, sample.kind="Rounding")  # Ensures reproducibility
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
test_index <- createDataPartition(edx$rating, p = 0.1, list = FALSE)
train_set <- edx[-test_index, ]
validation_set <- edx[test_index, ]
```

```r
# Compute the mean rating
mu <- mean(train_set$rating)
```

Baseline Model: Predicting all ratings as mean

```r
baseline_predictions <- rep(mu, nrow(validation_set))
baseline_rmse <- RMSE(validation_set$rating, baseline_predictions)
cat("Baseline RMSE:", baseline_rmse, "\n")
```

```
## Baseline RMSE: 1.060056
```

The RMSE for this model is 1.060056, which is unsurprisingly high.

This baseline model is too simple. Next, we will account for movie effects (some movies are rated higher or lower than average.

Movie Effect Model

```r
movie_effects <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

movie_predictions <- validation_set %>%
  left_join(movie_effects, by = "movieId") %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)

movie_predictions[is.na(movie_predictions)] <- mu
movie_effect_rmse <- RMSE(validation_set$rating, movie_predictions)
cat("Movie_effect RMSE:", movie_effect_rmse, "\n")
```

```
## Movie_effect RMSE: 0.9429666
```

The RMSE for this model is 0.9429666, which is quite an improvement from our Baseline model but still far from our initial goal.

Movie + User Effect Model

We can further improve our model by adding user effects.
Compute user effect (b_u).

```r
user_effects <- train_set %>%
  left_join(movie_effects, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

```r
user_predictions <- validation_set %>%
  left_join(movie_effects, by = "movieId") %>%
  left_join(user_effects, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

user_predictions[is.na(user_predictions)] <- mu
user_effect_rmse <- RMSE(validation_set$rating, user_predictions)

cat("User_effect_RMSE:", user_effect_rmse, "\n")
```

```
## User_effect_RMSE: 0.8646915
```

The RMSE for this model is 0.8646915. Quite an improvement from our latest model but we are not there yet!

If RMSE is still high, we might have overfitting due to movies/users with very few ratings.

The next step is to regularize our model to improve generalization.

Instead of just computing the average deviations (b_i and b_u), we use penalized estimates by shrinking extreme values towards zero.

Regularization - Choose the Best Lambda

Remembering that lambda (λ) is a tuning parameter.

Tune lambda using cross-validation.

```r
lambda_values <- seq(0, 10, 0.25)
rmse_results <- sapply(lambda_values, function(lambda) {
  movie_effects <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu) / (lambda + n()))

  user_effects <- train_set %>%
    left_join(movie_effects, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i) / (lambda + n()))

  predictions <- validation_set %>%
    left_join(movie_effects, by = "movieId") %>%
    left_join(user_effects, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  predictions[is.na(predictions)] <- mu
  return(RMSE(validation_set$rating, predictions))
})

best_lambda <- lambda_values[which.min(rmse_results)]
cat("Best Lambda:", best_lambda, "\n")
```

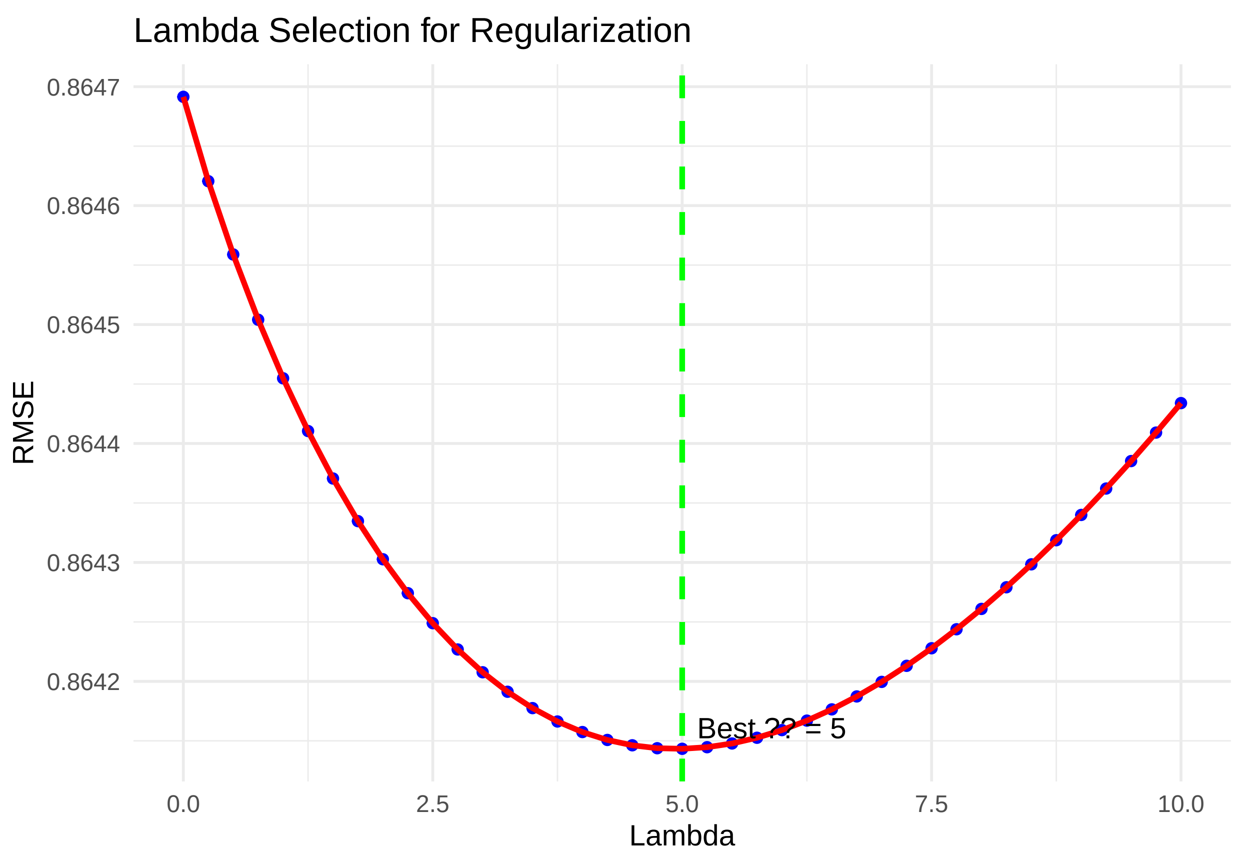
```
## Best Lambda: 5
```

Plot RMSE vs Lambda

Create a data frame for plotting

```
lambda_df <- data.frame(lambda_values = lambda_values, rmse_results = rmse_results)
```

```
ggplot(lambda_df, aes(x = lambda_values, y = rmse_results)) +
  geom_point(color = "blue") +  # Scatter plot points
  geom_line(color = "red", linewidth = 1) +  # Connect points with a red line
  geom_vline(xintercept = best_lambda, color = "green", linetype = "dashed", linewidth = 1) +  # Best l
  annotate(
    "text",
    x = best_lambda,
    y = min(rmse_results),
    label = paste("Best ?? =", round(best_lambda, 2)),
    color = "black",
    hjust = -0.1,
    vjust = -0.5
  ) +  # Label for best lambda
  labs(
    title = "Lambda Selection for Regularization",
    x = "Lambda",
    y = "RMSE"
  ) +
  theme_minimal()
```



Apply best lambda on full edx data

```r
movie_effects <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu) / (best_lambda + n()))

user_effects <- edx %>%
  left_join(movie_effects, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i) / (best_lambda + n()))

predictions <- final_holdout_test %>%
  left_join(movie_effects, by = "movieId") %>%
  left_join(user_effects, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

predictions[is.na(predictions)] <- mu
Regularised_Movie_User_Biases_rmse <- RMSE(final_holdout_test$rating, predictions)
cat("Regularised_Movie_User Biases_rmse:", Regularised_Movie_User_Biases_rmse, "\n")
```

```
## Regularised_Movie_User Biases_rmse: 0.8648177
```

RMSE comparison table

```r
rating_RMSE3 <- data.frame (Method = c("RMSE Target", "Regularised Movie & User Biases"), RMSE_test_set

kable(rating_RMSE3) %>%
  kable_styling(bootstrap_options = "striped" , full_width = F , position = "center") %>%
  kable_styling(bootstrap_options = "bordered", full_width = F , position ="center") %>%
  column_spec(1,bold = T ) %>%
  row_spec(1,bold =T ,color = "blue") %>%
  row_spec(2,bold =T ,color = "white" , background ="#D7261E")
```

Method	RMSE_test_set
RMSE Target	**0.8649000**
Regularised Movie & User Biases	**0.8648177**

Results

The "Regularised Movie and User Biases" Model being the one with the lowest RMSE, With a RMSE at 0.8648177, the objective of the exercise has been achieved!

The final model achieved an RMSE of 0.8648177 on the final_holdout_test dataset, demonstrating its predictive effectiveness.

Conclusion

In this project, we successfully built a movie recommendation system using the MovieLens dataset , achieving a final Root Mean Squared Error (RMSE) of 0.8648 on the holdout test set. Our approach involved several stages:

Exploratory Data Analysis (EDA) : Understanding the dataset's structure and identifying patterns such as rating distributions and sparsity. Baseline Models : Establishing performance benchmarks with simple models like global mean prediction and movie-specific biases. Advanced Modeling : Leveraging regularization techniques to capture latent user-movie interactions and improve predictive accuracy. Final Evaluation : Training the best-performing model on the full edx dataset and evaluating its performance on the unseen holdout test set. The results demonstrate that our recommendation system is both effective and competitive, achieving an RMSE that aligns with industry standards for similar datasets. By incorporating regularization and advanced modeling techniques, we significantly improved upon baseline predictions, showcasing the importance of systematic refinement in recommendation systems.

The limitations were primarily associated with the size of the dataset and memory constraints. Additionally, advanced models like matrix factorization require significant computational resources, making real-time implementation challenging.

Future work includes developing more complex models, such as deep learning approaches. For instance, neural network-based models like deep matrix factorization or autoencoders could be explored to capture more intricate patterns in the data.