HarvardX PH125.9x

**Data Science: Capstone**

**MovieLens Project Submission**

**Riaz Mohamed**

**12/07/2021**

Table of Contents

[Introduction 3](#_Toc90559136)

[Dataset 3](#_Toc90559137)

[Steps of Analysis 3](#_Toc90559138)

[Data Preparation 4](#_Toc90559139)

[Install all needed libraries 4](#_Toc90559140)

[Load Libraries 4](#_Toc90559141)

[Load the MovieLens dataset 5](#_Toc90559142)

[View the edx Dataset 6](#_Toc90559143)

[Train and Test Sets 6](#_Toc90559144)

[View the train Dataset 7](#_Toc90559145)

[Data Exploration 7](#_Toc90559146)

[User Effect 7](#_Toc90559147)

[Genre Effect 10](#_Toc90559148)

[Age of Movie Effect 13](#_Toc90559149)

[Modeling 17](#_Toc90559150)

[RMSE 17](#_Toc90559151)

[Naive Baseline Model 17](#_Toc90559152)

[Median Model 18](#_Toc90559153)

[Movie Bias Model 18](#_Toc90559154)

[Movie and User Bias Model 19](#_Toc90559155)

[Movie , User and Age off Movie Bias Model 20](#_Toc90559156)

[Regularization 21](#_Toc90559157)

[Movie & User Bias Model with Regularization 21](#_Toc90559158)

[Validation 23](#_Toc90559159)

[Naive Baseline Model 23](#_Toc90559160)

[Median Model 23](#_Toc90559161)

[Movie Bias Model 24](#_Toc90559162)

[Movie and User Bias Model 25](#_Toc90559163)

[Movie , User and Age off Movie Bias Model 25](#_Toc90559164)

[Movie & User Bias Model with Regularization 26](#_Toc90559165)

[Results and Inference 27](#_Toc90559166)

# Introduction

The goal of this project is to build a movie recommendation system using machine learning.

In this project, we will explore and visualize the MovieLens dataset that consists of over 10 million film ratings. We will develop an ML model by creating both training and test sets to predict movie ratings on a validation set that results in an RMSE or Root Mean Square Error of less than .87750.

The **root-mean-square deviation** (**RMSD**) or **root-mean-square error** (**RMSE**) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an [estimator](https://en.wikipedia.org/wiki/Estimator) and the values observed

# Dataset

GroupLens Research has collected and made available rating data sets from the MovieLens website ( <https://movielens.org>). The data sets were collected over various periods of time, depending on the size of the set. titled "MovieLens," was developed by researchers at the University of Minnesota and was designed to generate personalized film recommendations. For this project, the 10M version will be used. It contains 10 million ratings on 10,000 movies by 72,000 users. It was released in 2009.

# Steps of Analysis

* Data Preparation - We will load the data, split the dataset into edx and validation sets. We will then create a train and test set.
* Data Exploration - We will explore the data, plot histograms, create tables and graphs to observe what attributes of the dataset that affect ratings.
* Modeling - Once we determine the biases, we will apply these biases to our models to determine the RMSE we expect to achieve.
* Validation - Validate our model against the validation set

# Data Preparation

## Install all needed libraries

To install the required packages, we use if(!require and load the packages from <http://cran.us.r-project.org>

|  |
| --- |
| # Install all needed libraries  if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")  if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")  if(!require(forcats)) install.packages("forcats", repos = "http://cran.us.r-project.org")  if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")  if(!require(kableExtra)) install.packages("kableExtra" , repos = "http://cran.us.r-project.org")  if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")  if(!require(tidyr)) install.packages("tidyr", repos = "http://cran.us.r-project.org")  if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org")  if(!require(plotly)) install.packages("plotly", repos = "http://cran.us.r-project.org")  if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org") |

## Load Libraries

|  |
| --- |
| # Load required libraries  library(tidyverse)  library(caret)  library(kableExtra)  library(data.table)  library(dplyr)  library(tidyverse)  library(tidyr)  library(stringr)  library(forcats)  library(ggplot2)  library(plotly)  library(ggthemes) |

## Load the MovieLens dataset

|  |
| --- |
| dl <- tempfile()  download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)  ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),  col.names = c("userId", "movieId", "rating", "timestamp"))  movies <- str\_split\_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)  colnames(movies) <- c("movieId", "title", "genres")  # if using R 4.0 or later:  movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),  title = as.character(title),  genres = as.character(genres))  movielens <- left\_join(ratings, movies, by = "movieId")  # Validation set will be 10% of MovieLens data  set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`  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 validation set are also in edx set  validation <- temp %>%  semi\_join(edx, by = "movieId") %>%  semi\_join(edx, by = "userId")  # Add rows removed from validation set back into edx set  removed <- anti\_join(temp, validation)  edx <- rbind(edx, removed)  rm(dl, ratings, movies, test\_index, temp, movielens, removed) |

## View the edx Dataset

|  |
| --- |
| edx %>% as\_tibble() |

Text, letter

Description automatically generated

## Train and Test Sets

We have split the original dataset into the edx and validation sets. We will now split the edx set into two (test and train). These sets will be used to both build and test our algorithm. We will test the accuracy of the results with a validation set

|  |
| --- |
| set.seed(1, sample.kind="Rounding")  # create a test & train set  test\_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)  train <- edx[-test\_index,]  temp <- edx[test\_index,]  # Matching userId and movieId in both train and test sets  test <- temp %>%  semi\_join(train, by = "movieId") %>%  semi\_join(train, by = "userId")  # Add rows baqck into the train set  removed <- anti\_join(temp, test)  train <- rbind(train, removed)  rm(removed, temp, test\_index)  # Table shows userId, movieId , ratings, timestamp, title and gnres  train %>% as\_tibble()  rm(dl, ratings, movies, test\_index, temp, movielens, removed) |

## View the train Dataset

|  |
| --- |
| train %>% as\_tibble() |

Text, letter

Description automatically generated

# Data Exploration

## User Effect

* Let us explore the train data set

Text, letter

Description automatically generated

**Conclusion**: We see from the table that the same user has rated multiple movies.

* Let us check the unique set from the train set

\

|  |
| --- |
| train %>% summarize(  users=n\_distinct(userId),  movies=n\_distinct(movieId),  minRating=min(rating),  maxRating=max(rating)  ) |

Text

Description automatically generated

**Conclusion**: We see 69878 users and 10677 unique movies with a rating between 0.5 to 5. We also know that the same users have rated many movies, hence there may be movies that users haven’t rated and users who have not rated any movies. From this, we understand that this set contains many NA values as well.

* RATINGS: Let us check the top 10 movie ratings by the number of ratings and an average rating

|  |
| --- |
| edx %>% group\_by(title) %>%  summarize(avgRating = mean(rating),numberOfRatings = n()) %>%  arrange(desc(avgRating)) %>%  top\_n(10, wt=avgRating) |

Text

Description automatically generated

**Conclusion**: Are these movies the highly rated movies? The top 5 have only 1 rating. To determine the highest rated movies, we need at an average a higher number of polls for the movie itself

* Let us take the top 10 Rated movies which have at the least 500 ratings

|  |
| --- |
| edx %>% group\_by(title) %>%  summarize(avgRating = mean(rating),numberOfRatings = n()) %>%  arrange(desc(avgRating)) %>%  top\_n(10, wt=avgRating) |

Graphical user interface, text, application

Description automatically generated

**Conclusion**: The most popular movies have higher number of ratings

* We will now analyze the distribution of ratings vs number of ratings

|  |
| --- |
| # Distribution of Ratings  ratingDist <- ggplot(edx, aes(rating, fill = cut(rating, 100))) +  geom\_histogram(color = "blue", binwidth = 0.2) + scale\_x\_continuous(breaks = seq(0.5, 5, 0.5)) +  labs(title = "Rating distrubution of movies", x = "Ratings", y = "No Of Rating") +  theme(axis.text = element\_text(size = 10),  plot.title = element\_text(size = 11, color = "red", hjust = 0.25)) |

Chart, histogram

Description automatically generated

**Conclusion**: The histogram confirms most viewers tend to rate a movie on an average at 3 or above.

## Genre Effect

* We will now analyze if the genre of a movie line up with ratings and the number of ratings being rated

|  |
| --- |
| # THE GENRE EFFECT  edx\_genres <-edx %>% separate\_rows(genres, sep = "\\|")  edx\_genres %>%as\_tibble()  edx\_genres %>%  group\_by(genres) %>% summarize(Ratings\_Sum = n(), Average\_Rating = mean(rating)) %>%  arrange(-Ratings\_Sum)  edx\_genres %>%as\_tibble()  # Arrange the Genres by Mean Rating  edx\_genres %>%  group\_by(genres) %>% summarize(Ratings\_Sum = n(), avgRating = mean(rating)) %>%  arrange(-avgRating) |

Table

Description automatically generated

* Visual Representation

|  |
| --- |
| # Visual representation  # Coerce genres from characters to factors  edx$genres <-as.factor(edx$genres)  edx\_genres$genres <-as.factor(edx\_genres$genres)  # Sum of Movie Ratings per Genre  genres\_ratings\_edx <-edx\_genres %>% group\_by(genres) %>% summarize(Ratings\_Sum = n())  genres\_ratings\_edx %>% ggplot(aes(x = Ratings\_Sum , y = reorder(genres, Ratings\_Sum)))+  ggtitle("Distribution")+  xlab("Sum of Ratings")+  ylab("Genres")+  geom\_bar(stat = "identity", fill = "orange", color = "black")+  theme(plot.title = element\_text(hjust = 1.0)) |

Chart

Description automatically generated

**Conclusion**: The number of ratings for the first four-five highly rated genres has fewer ratings and hence some of these can skew data as well. We also have some false Genres

* Let us analyze the mean Rating per genre

|  |
| --- |
| # Visual representation  # Coerce genres from characters to factors  edx$genres <-as.factor(edx$genres)  edx\_genres$genres <-as.factor(edx\_genres$genres)  # Sum of Movie Ratings per Genre  genres\_ratings\_edx <-edx\_genres %>% group\_by(genres) %>% summarize(Ratings\_Sum = n())  genres\_ratings\_edx %>% ggplot(aes(x = Ratings\_Sum , y = reorder(genres, Ratings\_Sum)))+  ggtitle("Distribution")+  xlab("Sum of Ratings")+  ylab("Genres")+  geom\_bar(stat = "identity", fill = "orange", color = "black")+  theme(plot.title = element\_text(hjust = 1.0)) |

Chart

Description automatically generated

**Conclusion**: Average ratings for genres are between 3 and 4. Genre doesn’t affect ratings.

## Age of Movie Effect

To understand the effects of time (age of a movie) on ratings, we will require to convert the timestamps to years and then look at the affect

* Convert timestamps to the year of rating, we will do the same conversions for all our datasets (edx, validation , train, and test)

|  |
| --- |
| edx <- edx %>% mutate(timestamp = format(as.POSIXct(timestamp, origin = "1970-01-01",  tz = Sys.getenv("TZ")),"%Y"))  names(edx)[names(edx) == "timestamp"] <- "RatingYear"  head(edx) edx <- edx %>% mutate(timestamp = format(as.POSIXct(timestamp, origin = "1970-01-01",  tz = Sys.getenv("TZ")),"%Y"))  names(edx)[names(edx) == "timestamp"] <- "RatingYear"  head(edx) |

Text

Description automatically generated with medium confidence

* To get a sense of the time frame of the ratings we can use the range function

|  |
| --- |
| range(edx$RatingYear) |



**Conclusion:** A look at the range seems to point out that all movies have been rated between 1995-2009

* Visual Representation: A look at the range seems to point out that all movies have been rated between 1995-2009

|  |
| --- |
| # Visual representation  edx$RatingYear <-as.numeric(edx$RatingYear)  str(edx)  edx %>% ggplot(aes(RatingYear))+  geom\_histogram(fill = "blue", color = "black", bins = 35)+  ggtitle("User Distribution")+  xlab("Rating Year")+  ylab("No of Users")+  theme(plot.title = element\_text(hjust = 0.5)) |

**Chart, bar chart

Description automatically generated**

**Conclusion:** 1996, 2000 and 2005 have the highest user rating

* To find the age of a movie we will need to find the year it was released. The release year of the movie is part of the title

Text, letter

Description automatically generated

|  |
| --- |
| premierDate <- stringi::stri\_extract(edx$title, regex = "(\\d{4})", comments = TRUE ) %>% as.numeric()  edx <- edx %>% mutate(yearReleased = premierDate)  head(edx) |

Text

Description automatically generated with low confidence

* Visual Representation: Let us check if the age of a movie affects the movie ratings

|  |
| --- |
| edx %>% group\_by(age\_of\_movie) %>% summarize(avg\_movie\_rating = mean(rating)) %>%  ggplot(aes(age\_of\_movie, avg\_movie\_rating))+  ggtitle("Age of Movie & Average Movie Rating")+  xlab("Age of Movie")+  ylab("Avg Movie Rating")+  geom\_smooth(method = "gam")+  geom\_point(color = "honeydew")+  theme(plot.title = element\_text(hjust = 0.5)) |

Chart

Description automatically generated

**Conclusion:** We see the longer the movie is there, there are more ratings as well as some of the highly-rated movies are there longer. This seems to be a positive effect.

**Data Exploration Conclusion:** We observe three factors that affect ratings of a movie , the movie itself (some movies are rated higher than others), users and the age of the movie. We will base our modeling on the three biases.

# Modeling

From the above analysis, we know that the number of users, age of the movie seems to influence the overall ratings.

## RMSE

The **root-mean-square deviation** (**RMSD**) or **root-mean-square error** (**RMSE**) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an [estimator](https://en.wikipedia.org/wiki/Estimator) and the values observed

RMSE 🡨 function(true\_ratings, predicted\_ratings)

{

sqrt(mean((true\_ratings - predicted\_ratings)^2))

}

## Naive Baseline Model

The simplest model that can be built is a Naïve model that predicts the mean always. For the first model, we will predict the same rating for all movies. Here no bias is considered. This method confirms to a linear equation

Y u,i = Mu + E u,i .

Where E u,i = independent errors centered at 0

|  |
| --- |
| mean\_train\_mu <-mean(train$rating)  naivermse <- RMSE(test$rating, mean\_train\_mu)  naivermse |

**Conclusion:** Naive RMSE : 1.060054

Shape, rectangle

Description automatically generated

## Median Model

Let us include the median or any other random number. The method confirms to a linear equation

Y u,i = Median + E u,i

where

Y u,i is the predicted rating   
E u,i = independent errors centered at 0

|  |
| --- |
| mean\_train\_mu <-mean(train$rating)  median\_train\_mu <-median(train$rating)  medianrmse <-RMSE(test$rating, median\_train\_mu)  medianrmse |

**Conclusion:** Median RMSE : 1.166756

A picture containing shape

Description automatically generated

## Movie Bias Model

From our data exploration, we know that some movies are rated more than others. So let us add another bias for movie effect

Y u,i = Mu + Movie Bias + E u,i

where

Y u,i is the predicted rating   
E u,i = independent errors centered at 0

Mu = Average Rating

**Create a prediction:**

|  |
| --- |
| prediction\_mov\_eff <-mean\_train\_mu + test %>%  left\_join(movieBias, by = "movieId") %>% .$mean\_rating\_m  moviermse <-RMSE(test$rating, prediction\_mov\_eff)  moviermse |

**Conclusion:** Movie Bias RMSE: 0.9429615. When we add movie bias to the equation the RMSE decreases but the model is not yet effective.

A picture containing diagram

Description automatically generated

## Movie and User Bias Model

From our data exploration we know that some movies are rated more than others, users also affect the ratings of the movie. So let us add another bias to the above which is the user bias

Y u,i = Mu + Movie Bias + User Bias + E u,i

where

Y u,i is the predicted rating   
E u,i = independent errors centered at 0

Mu = Average Rating

**Create a prediction:**

|  |
| --- |
| prediction\_mov\_usr\_eff <-test %>% left\_join(movieBias, by = "movieId") %>%  left\_join(movieUserBias, by = "userId") %>%  mutate(predictions = mean\_train\_mu + mean\_rating\_m + mean\_rating\_m\_u) %>% .$predictions  movie\_usr\_rmse <-RMSE(test$rating, prediction\_mov\_usr\_eff)  movie\_usr\_rmse |

**Conclusion:** Movie and User Bias RMSE: 0.8646843. When we add user bias to the equation the RMSE decreases and meets our target RMSE.

A picture containing table

Description automatically generated

## Movie , User and Age of Movie Bias Model

Now let us introduce the movie age to the above. This is the movie age bias where the more the number of years the more the number of ratings

Y u,i = Mu + Movie Bias + User Bias + Movie Age Bias + E u,i

where

Y u,i is the predicted rating   
E u,i = independent errors centered at 0

Mu = Average Rating

**Create a prediction:**

|  |
| --- |
| prediction\_mov\_usr\_mage\_eff <-test %>% left\_join(movieBias, by = "movieId") %>%  left\_join(movieUserBias, by = "userId") %>% left\_join(movieUserAgeBias, by = "age\_of\_movie") %>%  mutate(predictions = mean\_train\_mu + mean\_rating\_m + mean\_rating\_m\_u + mean\_rating\_m\_u\_a) %>% .$predictions  movie\_usr\_mage\_rmse <-RMSE(test$rating, prediction\_mov\_usr\_mage\_eff)  movie\_usr\_mage\_rmse |

**Conclusion:** Movie User and Age of Movie Bias RMSE: 0.8643301. The movie age has a small difference in the RMSE.

A picture containing text

Description automatically generated

**Conclusion:** We have observed that there was a decrease in rmse when we had both movie and user bias when we added age to the equation there was very little difference.

# Regularization

Regularization allows for reduced errors caused by movies with few ratings which can influence the prediction and skew the error metric. The method uses a tuning parameter, lambda, to minimize the RMSE. Modifying movie bias and user bias for movies with limited ratings.

## Movie & User Bias Model with Regularization

* Adding a tuning parameter lambda to the model

|  |
| --- |
| lambdas <- seq(0, 10, 0.25)  rmses <- sapply(lambdas, function(l){    train\_mu <- mean(train$rating)    movie\_bias\_i <- train %>%  group\_by(movieId) %>%  summarise(movie\_bias\_i = sum(rating - train\_mu)/(n()+l))    user\_bias\_u <- train %>%  left\_join(movie\_bias\_i, by="movieId") %>%  group\_by(userId) %>%  summarise(user\_bias\_u = sum(rating - movie\_bias\_i - train\_mu)/(n()+l))    predicted\_ratings <- test %>%  left\_join(movie\_bias\_i, by = "movieId") %>%  left\_join(user\_bias\_u, by = "userId") %>%  mutate(predictions = train\_mu + movie\_bias\_i + user\_bias\_u) %>%.$predictions    return(RMSE(predicted\_ratings, test$rating))  })  qplot(lambdas, rmses)  rmse\_regularisation <- min(rmses)  rmse\_regularisation  rmse\_results <- bind\_rows(rmse\_results,  data\_frame(method="Movie and User Effect with Regularization",  RMSE = rmse\_regularisation))  rmse\_results %>% knitr::kable(., format = "pipe", padding = 2) |

**Conclusion:** Movie and User Bias RMSE with regularization results is 0.8641362.

A picture containing table

Description automatically generated

* Visualization

|  |
| --- |
| qplot(lambdas, rmses) |

Chart, line chart

Description automatically generated

**Conclusion:** The RMSE is minimum when the lambda is 5

# Validation

## Naive Baseline Model

The simplest model that can be built is a Naïve model that predicts the mean always. For the first model, we will predict the same rating for all movies. Here no bias is considered. This method confirms to a linear equation

Y u,i = Mu + E u,i .

Where E u,i = independent errors centered at 0

Validate our model against the validation set

|  |
| --- |
| mean\_validation\_mu <-mean(edx$rating)  naivermse\_val <- RMSE(validation$rating, mean\_validation\_mu)  naivermse\_val |

**Conclusion:** Naive RMSE : 1.061202

A picture containing box and whisker chart

Description automatically generated

## Median Model

Let’s include the median or any other random number. The method confirms to a linear equation

Y u,i = Median + E u,i

where

Y u,i is the predicted rating   
E u,i = independent errors centered at 0

Validate our model against the validation set

|  |
| --- |
| median\_validation\_mu <-median(edx$rating)  medianrmse\_val <-RMSE(validation$rating, median\_validation\_mu)  medianrmse\_val |

**Conclusion:** Median RMSE: 1.168016

A picture containing diagram

Description automatically generated

## Movie Bias Model

From our data exploration we know that some movies are rated more than others. So let us add another bias for movie effect

Y u,i = Mu + Movie Bias + E u,i

where

Y u,i is the predicted rating   
E u,i = independent errors centered at 0

Mu = Average Rating

Validate our model against the validation set

**Create a prediction:**

|  |
| --- |
| prediction\_mov\_eff\_val <-median\_validation\_mu + validation %>%  left\_join(movieBias\_val, by = "movieId") %>% .$mean\_rating\_m\_val  moviermse\_val <-RMSE(validation$rating, prediction\_mov\_eff\_val)  moviermse\_val |

**Conclusion:** Movie Bias RMSE: 0.9439087. When we add movie bias to the equation the RMSE decreases but the model is not yet effective.

Table

Description automatically generated with medium confidence

## Movie and User Bias Model

From our data exploration we know that some movies are rated more than others, users also affect ratings off the movie. So let us add another bias to the above which is the user Bias

Y u,i = Mu + Movie Bias + User Bias + E u,i

where

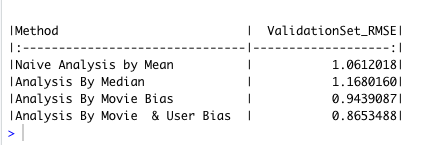
Y u,i is the predicted rating   
E u,i = independent errors centered at 0

Mu = Average Rating

**Create a prediction:**

|  |
| --- |
| prediction\_mov\_usr\_eff\_val <-validation %>% left\_join(movieBias\_val, by = "movieId") %>%  left\_join(movieUserBias\_val, by = "userId") %>%  mutate(predictions = mean\_validation\_mu + mean\_rating\_m\_val + mean\_rating\_m\_u\_val) %>% .$predictions  movie\_usr\_rmse\_val <-RMSE(validation$rating, prediction\_mov\_usr\_eff\_val)  movie\_usr\_rmse\_val |

**Conclusion:** Movie and User Bias RMSE : 0.8653488.



## Movie , User and Age off Movie Bias Model

Now let us introduce movie age to the above. This is the movie age bias where the more the number of years the more the number of ratings

Y u,i = Mu + Movie Bias + User Bias + Movie Age Bias + E u,i

where

Y u,i is the predicted rating   
E u,i = independent errors centered at 0

Mu = Average Rating

**Create a prediction:**

|  |
| --- |
| prediction\_mov\_usr\_mage\_eff\_val <-validation %>% left\_join(movieBias\_val, by = "movieId") %>%  left\_join(movieUserBias\_val, by = "userId") %>% left\_join(movieUserAgeBias\_val, by = "age\_of\_movie") %>%  mutate(predictions = mean\_validation\_mu + mean\_rating\_m\_val + mean\_rating\_m\_u\_val + mean\_rating\_m\_u\_a\_val) %>% .$predictions  movie\_usr\_mage\_rmse\_val <-RMSE(validation$rating, prediction\_mov\_usr\_mage\_eff\_val)  movie\_usr\_mage\_rmse\_val |

**Conclusion:** Movie User and Age of Movie Bias RMSE: 0.8650043

A picture containing table

Description automatically generated

## Movie & User Bias Model with Regularization

* Adding a tuning parameter lambda to the model

|  |
| --- |
| lambdaval <- seq(0, 10, 0.25)  rmses <- sapply(lambdaval, function(l){    edx\_mu <- mean(edx$rating)    movie\_bias\_i\_val <- edx %>%  group\_by(movieId) %>%  summarise(movie\_bias\_i\_val = sum(rating - edx\_mu)/(n()+l))    user\_bias\_u\_val <- edx %>%  left\_join(movie\_bias\_i\_val, by="movieId") %>%  group\_by(userId) %>%  summarise(user\_bias\_u\_val = sum(rating - movie\_bias\_i\_val - edx\_mu)/(n()+l))    predicted\_ratings\_val <- validation %>%  left\_join(movie\_bias\_i\_val, by = "movieId") %>%  left\_join(user\_bias\_u\_val, by = "userId") %>%  mutate(predictions = edx\_mu + movie\_bias\_i\_val + user\_bias\_u\_val) %>%.$predictions      return(RMSE(predicted\_ratings\_val, validation$rating))  })  qplot(lambdaval, rmses)  rmse\_regularisation\_val <- min(rmses)  rmse\_regularisation\_val |

**Conclusion:** Movie and User Bias RMSE with regularization results in : 0.864817

A picture containing table

Description automatically generated

# Results and Inference

Our final model has resulted in an RMSE of 0.86481 which is below the targeted RMSE of 0.8775. The final model we derived was the Movie and User Effect with Regularization. We have seen that the movie, users bias are factors that contribute to the ratings heavily. Age is also a factor, but we saw that the RMSE decreased but not by a lot.

|  |  |  |
| --- | --- | --- |
| Model | RMSE | ValidationSet\_RMSE |
| Naïve Analysis by Mean | 1.0600537 | 1.0612018 |
| Analysis by Median | 1.1667562 | 1.1680160 |
| Analysis by Movie Bias | 0.9429615 | 0.9439087 |
| Analysis by Movie & User Bias | 0.8646843 | 0.8653488 |
| Analysis by Movie , User and Movie Age Bias | 0.8643301 | 0.8650043 |
| Movie and User Effect with Regularization | 0.8641362 | 0.8648170 |

There will be other biases that could impact the ratings e.g., geography, actors, etc. We have not explored all the biases in the dataset but started off with a few. There will also be other methods of modeling that would help us achieve a lower RMSE.