title: "MovieLens Project - 3" author: "Tarun Ch. Bordoloi" date: "3/4/2020" output:pdf_document toc: true toc_depth: 2 number_sections: true highlight: pygments keep_tex: true

html_document: default

```{r setup, include=FALSE} knitr::opts_chunk$set(echo = TRUE, fig.align = 'center', cache=FALSE, cache.lazy = FALSE)

```
1.0 Executive summary:

1.1 The Data set: Overview:
This project is a part of the HarvardX: PH125.9x: Data Science: Capstone.
The primary objective of the project is to create a movie recommendation system using th

1.2 Summary goals:
The purpose of the recommender system being developed in this project is to predict user

1.3 Key steps performed :
•   Downloaded the dataset
$ Ensured that the required packages and libraries are installed
$ Splitted the data set into ‘edx’ and ‘validation’ set

•   Carried out exploration of the data and performed feature engineering
$ Included data visualization tools as required
$ Incorporated insights gained

•   Models were developed and those were evaluated
$ Results tabulated
$ The performance of our final model was evaluated based on the ‘Penalized Root Mean Squ
This algorithm achieved a RMSE of 0.86482 while testing on the 'validation set'
•   Conclusion stated


```{r include=FALSE, echo=FALSE}}
### 2.0 The Data set

### 2.1 Downloading the data

################################
## This code is provided by the edx staff to download and create an edx set, validation
################################

## Loading required libraries

## Install all needed packages if not present
```

```
## Install all needed packages if not present
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(kableExtra)) install.packages("kableExtra")
if(!require(tidyr)) install.packages("tidyr")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(stringr)) install.packages("stringr")
if(!require(forcats)) install.packages("forcats")
if(!require(ggplot2)) install.packages("ggplot2")
## Loading all needed libraries
library(dplyr)
library(tidyverse)
library(kableExtra)
library(tidyr)
library(stringr)
library(forcats)
library(ggplot2)
library(lubridate)
library(caret)
library(magrittr)

## Downloading files

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

dl <- tempfile()
 download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.da
                      col.names = c("userId", "movieId", "rating", "timestamp"))



## Build the data set

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
 colnames(movies) <- c("movieId", "title", "genres")
 movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieI
                                            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)` instead
```

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE
 edx <- movielens[-test_index,]
 temp <- movielens[test_index,]

## To make sure that the userId and movieId in validation set are also in edx set

 validation <- temp %>%
      semi_join(edx, by = "movieId") %>%
      semi_join(edx, by = "userId")

## To 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)
```

2.2 Data exploration & Feature Egineering

```{r echo=FALSE, include=TRUE} glimpse(edx)

```
 It would appear that the'edx' data set has 9,000,055 observations and 6 variables
```{r echo=FALSE, include=TRUE}
glimpse(validation)
```

The 'validation' data set has 999,999 observations and same 6 variables.

These variables are :

$ userId `<integer>` which contains an unique identification number of each user $ movieId `<numeric>` which contains an unique identification number for each movie $ timestamp `<integer>` which contains timestamp for a specific rating provided by one user $ title `<character>` which contains the title of each movie with the year of the release $ genres `<character>` which contains the list of pipe-delimited genre of each movie $ rating `<numeric>` which contains raing for each movie by one user.Movies are rated in a 5 star scale in an increment of half star

```{r echo=FALSE, include=TRUE}

# How many unique users, movies and genres we are dealing with .

edx %>% summarize(Unique_Users = n_distinct(userId), Unique_Movies = n_distinct(movieId), Unique_Genres = n_distinct(genres))
```

```
There are 69878 unique users, 10677 unique movies and 797 unique genres



```{r echo=FALSE, include=TRUE}


## converting 'timestamp' to human readable form and creating 'year_rated' column

edx <- mutate(edx, year_rated = year(as_datetime(timestamp)))
head(edx)


validation <- mutate(validation, year_rated = year(as_datetime(timestamp)))
head(validation)
```

Extracting the year of release of each movie and creating 'year' column .As would be observed release date of each movie is included with the "title'

```{r echo=FALSE, include=TRUE}

edx <- edx %>% mutate(year = as.numeric(str_sub('title',-5,-2))) head(edx)

```
```{r echo=FALSE, include=TRUE}

validation <- validation %>% mutate(year = as.numeric(str_sub('title',-5,-2)))
head(validation)
```

Checking for missing value per column

```{r echo=FALSE, include=TRUE} edx <- edx %>% select(-X)

validation <- validation %>% select(-X)

```
```{r echo=FALSE, include=TRUE}
sapply(edx, function(x) sum(is.na(x))) %>%
kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```
```

```{r echo=FALSE, include=TRUE} sapply(validation, function(x) sum(is.na(x))) %>% kable() %>% kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"), position = "center", font_size = 10, full_width = FALSE)

```
 ## It appears there is no missing value in any column

```{r echo=FALSE, include=TRUE}
## Now , let us remove unncessary columns from both 'edx' as well as 'validation' sets

edx <- edx %>% select( - title, - timestamp)
head(edx)


validation <- validation %>% select( - title, - timestamp)
head(validation)
```

```{r echo=FALSE, include=TRUE} summary(edx)

```
 ## Let us create a data frame 'rating_distribution' with half star and whole star rating

```{r echo=FALSE, include=TRUE}

group <- ifelse((edx$rating == 1|edx$rating == 2|edx$rating == 3|edx$rating == 4|edx$rat

rating_distribution <- data.frame(edx$rating,group)

head(rating_distribution)
```

Histogram of ratings

```{r echo=FALSE, include=TRUE} rating_distribution %>% ggplot(aes(edx$rating)) + geom_histogram(fill = "blue")+ labs(title = "number _of_ratings-for_each_rating", x = "rating", y = "number_of_ratings")
```

```
 It is observed that :
$ No user gives a 0 rating
$ The distribution of 'rating' is left skewed
$ Number of ratings are in the descending order are 4,3,5,3.5and 2
$ Higher ratings are more than the lower ratings
$ Half star ratings are less common
It is likely that an user habitually recommends a movie only if he/she likes it somewhat

```{r echo=FALSE, include=TRUE}
## Let us now see what response different movies attract going by their count of ratings
edx %>% count(movieId)%>%
ggplot(aes(n))+
geom_histogram(bin = 30, fill= "green")+
scale_x_log10()+
ggtitle("Movies")+
labs(subtitle = "number_of_ratings_by_movieId",
x = "movieId",
y = "number_of_ratings", caption ="source data : edx set")
```

```{r echo=FALSE, include=TRUE}

# Let us now see how each user rates different movies

edx %>% count(userId)%>% ggplot(aes(n))+ geom_histogram(bins = 30,fill = "green")+ scale_x_log10()+ ggtitle("Users")+ labs(subtitle = "number_of_ratings_by_userId", x = "userId", y = " number_of_ratings",caption ="source data : edx set")

```
 From the above analysis it appears that some movies are rated significantly more than tl

```{r echo=FALSE, include=TRUE}
edx %>% ggplot(aes(year_rated))+
geom_histogram(fill = "green")+
labs(title = "Distribution of yearwise ratings",
subtitle = "Year the rating was given",
x = "Year",
y = "number_of_ratings")
```

$ Year wise ratings appear to be irregular $ 1998 and 2002 have fewer ratings Having observed such behaviour I would not consider the feature 'year_rated' of the data to be a reliable predictor.

```{r echo=FALSE, include=TRUE} edx %>% ggplot(aes(year))+ geom_histogram(fill = "darkgreen")+ labs(title = "Distribution of movie ratings by release year", subtitle = "number of ratings by release year", x = "year", y = "number of ratings")

```
$ Ratings by release year distribution is clearly left skewed
$ Movies were rated most which were released during the period 1990 and 2006 which seeme
Due to such inconsistency this feature, 'year_release', may not be a reliable feature of

```{r echo=FALSE, include=TRUE}

## Rating distribution per genre for top 20 genres by ratings
edx %>%
   group_by(genres) %>%
   summarise(count = n()) %>%
   top_n(20,count) %>%
   arrange(desc(count)) %>%
   kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

```{r echo=FALSE, include=TRUE}

edx %>% group_by(genres) %>% summarise(count = n()) %>% top_n(20,count) %>% ggplot(aes(genres, count)) + theme_classic() + geom_col(fill = "green") + theme(axis.text.x = element_text(angle = 90, hjust = 1)) + labs(title = "Number of ratings Per Genre", x = "Genre", y = "Number of ratings", caption ="source data : edx set")

```
$ Most rated category appears to be 'Drama' , 'Comedy' and 'Comedy|Romance' meriting the
$ We ,however, need to note that data provided is not distinctly seperated category wise
$ Considering the category wise rating pattern we , perhaps, would do well to consider t

3.0 Building and evaluating model

3.1 Loss function

The performance of our final model will be evaluated based on the Residual Mean Squared
Simply defined , if $$y_{u,i}$$ as the rating given to a movie 'i'by user 'u' and denote
$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}^{} (\hat{y}_{u,i} - y_{u,i})^2}$$
with *N* being the number of user/movie combinations and the sum occurring over all thes
Let’s write a function that computes the RMSE for vectors of ratings and their correspon

```{r echo=FALSE, include=TRUE}

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

3.2 Train and Test sets

First task is to split the 'edx' set to 'train ' and 'test' sets We are taking 'test' set as 10% of the 'edx'

```{r echo=FALSE, include=TRUE}

set.seed(1, sample.kind="Rounding") test_index <- createDataPartition(edx$rating,times = 1, p = 0.1, list = FALSE) # Created 'test_index' train <- edx[- test_index , ] # Created 'train' set test <- edx[test_index , ] # Created 'test'set #

# Need to make sure that the 'userId' and 'movieId' in the 'test' set are also there in the 'train' set , not exactly the same cases though.

test <- temp %>% semi_join(edx, by = "movieId") %>% semi_join(edx, by = "userId")

## To add rows removed from validation set back into edx set

removed <- anti_join(temp, test) train <- rbind(edx, removed)

```
Checking the sets

```{r echo=FALSE, include=TRUE}

glimpse(train)
glimpse(test)
```

3,3 Baseline model

In its simplest form this model is generated by considering the same rating for all the movies irrespective of the 'userId' and the 'movieId'.All the differences explained by random variation. The formula would look like this: $Y_{u,i} = \hat{\mu} + \varepsilon_{u,i}$ With $\hat{\mu}$ is the mean and $\varepsilon_{u,i}$ is the independent errors sampled from the same distribution centered at 0.

```{r echo=FALSE, include=TRUE}

# Calculating the average accross all movies

mu_hat <- mean(train$rating) mu_hat

```
If we predict all the unknown ratings with $\hat{\mu}$ our RMSE will be as follows

```{r echo=FALSE, include=TRUE}

base_rmse <- RMSE(test$rating,mu_hat)
base_rmse
```

Let us now prepare a data frame to record all the RMSEs here after for all our evaluations

```{r echo=FALSE, include=TRUE}

rmse_results <- data.frame(method = "Baseline approach", RMSE = base_rmse) rmse_results%>% knitr::kable()%>% kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"), position = "center", font_size = 10, full_width = FALSE)

```
3.4 Movie effect model
The 'base_rmse' of 1.06 as seen above is by no means acceptable.
Hence , we need to attempt to improve this 'RMSE' and as a first step we are trying to
The intuition that different movies are rated differently are confirmed by the data.The
We shall now augment the previous model as shown in the following formula :
$$Y_{u,i} = \hat{\mu} + b_i + \varepsilon_{u,i}$$
With $\hat{\mu}$ is the mean and $\varepsilon_{u,i}$ is the independent errors sampled f

```{r echo=FALSE, include=TRUE}

## Computing average of 'rating' accross all the movies 'mu_hat'

mu_hat <- mean(train$rating)

## Computing averages by movie 'b_i'

b_i <- train %>%
group_by(movieId) %>%
summarize(b_i = mean(rating - mu_hat))


## Computing the predicted ratings on test set

predicted_ratings_movie <-test %>%
left_join(b_i, by='movieId') %>%
replace_na(list(b_i=0))%>%
mutate(pred = mu_hat + b_i)
rmse_movie <- RMSE(test$rating,predicted_ratings_movie$pred)
rmse_results <- bind_rows(rmse_results,data_frame(method="Movie Effect ",RMSE = rmse_mov
rmse_results
rmse_results %>% knitr::kable()%>%
kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
position = "center",
font_size = 10,
full_width = FALSE)
```

We have achieved some improvement of RMSE at 0.9429 over that of 'Baseline model'. But it is still from the target RMSE of 0.8649

3.5 User and Movie effect model

We shall now be trying to improve further our earlier RMSE by 'Movie effect model' incorporating the 'User + Movie' effect, which will be in the following form : $$Y_{u,i} = \hat{\mu} + b_i + b_u + \varepsilon_{u,i}$$ With $\hat{\mu}$ is the mean and $\varepsilon_{u,i}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the user's bias for the movie i. The b_u is a measure for the
```

user's rating behaviour u.

```{r echo=FALSE, include=TRUE}

# Mean accross all the movies 'mu_hat'

mu_hat <- mean(train$rating)

# Calculating the average by movie 'b_i'

b_i <- train %>% group_by(movieId) %>% summarize(b_i = mean(rating - mu_hat))

# Calculating the averages by user 'b_u'

b_u <- train %>% left_join(b_i , by = "movieId") %>% group_by(userId) %>% summarize(b_u = mean(rating - mu_hat - b_i))

# Computing the predicted ratings on test dataset

predicted_ratings_user <- test %>% left_join(b_i, by = "movieId") %>% left_join(b_u,by = "userId") %>% replace_na(list(b_i=0,b_u=0))%>% mutate(pred = mu_hat + b_i + b_u ) %>% .$pred user_movie_rmse <- RMSE(test$rating,predicted_ratings_user) user_movie_rmse rmse_results <- rbind(rmse_results, data.frame(method = "User & Movie effect", RMSE = user_movie_rmse )) rmse_results rmse_results%>% knitr::kable()%>% kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"), position = "center", font_size = 10, full_width = FALSE)

```
 It is encouraging that we have succeeded in improving the RMSE further to 0.8646 which

However , we intend trying further to see if we can achieve further improvement with our

3.6 User , Movie and Genre effect model

We shall now be trying to improve our earlier RMSE by 'User + Movie' effect incorporatin

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_{u,g} + \epsilon_{u,i}$$
With $\hat{\mu}$ is the mean and $\varepsilon_{u,i}$ is the independent errors sampled f

```{r echo=FALSE, include=TRUE}

## Calculating the average accross all the movies 'mu_hat'

mu_hat <- mean(edx$rating)

## Calculating the average by movie 'b_i'
b_i <- train %>%
   group_by(movieId) %>%
```

```
  summarize(b_i = mean(rating - mu_hat))

## Calculating the average by user 'b_u'
b_u <- train %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - b_i))

## Calculate the average by genre 'b_u_g'
genre_avgs <- train %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  group_by(genres) %>%
  summarize(b_u_g = mean(rating - mu_hat - b_i - b_u))


## Computing the predicted ratings on test dataset
movie_user_genre_rmse <- test %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  left_join(b_u_g, by='genres') %>%
  replace_na(list(b_i=0,b_u=0,b_u_g = 0))%>%
  mutate(pred = mu_hat + b_i + b_u + b_u_g) %>%
  pull(pred)
movie_user_genre_rmse_result <- RMSE(test$rating, movie_user_genre_rmse)

## Adding the results to the results dataset
rmse_results <- rmse_results %>% add_row(method ="Movie+User+Genre effectl", RMSE = movi
rmse_results
rmse_results %>%
knitr::kable()%>%
kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
position = "center",
font_size = 10,
full_width = FALSE)
```

We have achieved some further improvement at 0.8643

Aithough this is our best achievment interms of RMSE so far we shall now be testing both our models 'user & movie effect' and the 'Movie+User+Genre effectl' on the validation set.

3.7 Models - testing on validation set.

```{r echo=FALSE, include=TRUE}

# Testing 'user_and_movie_model' on the validation set
```

Mean accross all the movies 'mu_hat'

mu_hat <- mean(train$rating)

Calculating the average by movie 'b_i'

b_i <- train %>% group_by(movieId) %>% summarize(b_i = mean(rating - mu_hat))

Calculating the averages by user 'b_u'

b_u <- train %>% left_join(b_i , by = "movieId") %>% group_by(userId) %>% summarize(b_u = mean(rating - mu_hat - b_i))

Computing the predicted ratings on the validation dataset

predicted_ratings_user_movie <- validation %>% left_join(b_i, by = "movieId") %>% left_join(b_u,by = "userId") %>% replace_na(list(b_i=0,b_u=0))%>% mutate(pred = mu_hat + b_i + b_u) %>% .$pred user_movie_valid_rmse <- RMSE(validation$rating,predicted_ratings_user_movie) user_movie_valid_rmse

Adding the results to the results dataset

rmse_results <- rbind(rmse_results, data.frame(method = "User & Movie effect on validation set", RMSE = user_movie_valid_rmse)) rmse_results rmse_results%>% knitr::kable()%>% kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"), position = "center", font_size = 10, full_width = FALSE)

```
It would seem the RMSE has declined to 0.8654 while testing on the unseen validation dat

We shall now be testing our 'user_movie_genre_model' on the validation set


```{r echo=FALSE, include=TRUE}

## Testing on the validation set

## Calculating the average accross all the movies 'mu_hat'
mu_hat <- mean(edx$rating)

## Calculating the average by movie 'b_i'
b_i <- train %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu_hat))

## Calculating the average by user 'b_u'
b_u <- train %>%
   left_join(b_i, by='movieId') %>%
```
```

```
   group_by(userId) %>%
   summarize(b_u = mean(rating - mu_hat - b_i))

## Calculating the average by genre 'b_u_g'
b_u_g <- train %>%
   left_join(b_i, by='movieId') %>%
   left_join(b_u, by='userId') %>%
   group_by(genres) %>%
   summarize(b_u_g = mean(rating - mu_hat - b_i - b_u))


## Computing the predicted ratings on the validation dataset
predicted_ratings_user_movie_genre  <- validation %>%
   left_join(b_i, by='movieId') %>%
   left_join(b_u, by='userId') %>%
   left_join(b_u_g, by='genres') %>%
   replace_na(list(b_i=0, b_u=0,b_u_g = 0))%>%
   mutate(pred = mu_hat + b_i + b_u + b_u_g) %>%
   pull(pred)
movie_user_genre_valid_rmse <- RMSE(validation$rating,predicted_ratings_user_movie_genre
movie_user_genre_valid_rmse

## Adding the results to the results dataset
rmse_results <- rbind(rmse_results, data.frame(method = "User & Movie & genre effect on
rmse_results
rmse_results%>%
knitr::kable()%>%
kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
position = "center",
font_size = 10,
full_width = FALSE)
```

It would seem the RMSE has declined to 0.8658 while testing on the unseen validation data. As mentioned earlier at this stage all we can assume that it is possible and we need to prod along further.

3.8 Regularization based approach(Penalized RMSE)

It has come to light during our data exploration above, that some users have more actively participated in movie reviewing. At the same time there are some who have rated very few movies . Again there are instances where some movies are rated very few times . These are basically misleading noisy estimates . Further, RMSEs are sensitive to large errors. Large errors can increase our RMSE. So such issues necessitate putting a penalty term to give less importance to such effect.The regularisation method allows us to add a penalty λ to penalise movies with large estimates from small sample size.Let us call it Penalized RMSE approach Although we have accomplished a significant improvement over the 'Baseline model','Movie effect model' through the 'User and Movie effect model'and 'Movie+User+Genre effect model while testing these models on the test set RMSEs of these models showed a decine while testing on the unseen validation set. However We shall now be dealing with these models with the regularisation(Penalized) apprach.

```{r echo=FALSE, include=TRUE}

# Testing regularised user_movie model on the validation set

# Defining a table of lambdas

lambdas <- seq(0, 10, 0.25)

# For each lambda we shall be working on b_i, b_u , predict rating and test accuracy.

RMSE_function_reg <- sapply(lambdas,function(l){

# Calculating average accross all the movies 'mu_hat'

mu_hat <- mean(train$rating)

# Calculating the average by movie 'b_i'

b_i <- train %>% group_by(movieId)%>% summarize(b_i = sum(rating - mu_hat)/(n()+ l))

# Calculating the average by user 'b_u'

b_u <- train%>% left_join(b_i, by = "movieId")%>% group_by(userId)%>% summarize(b_u = sum(rating - b_i - mu_hat)/(n()+ l) )

# Computing the predicted ratings on the validation dataset

predicted_ratings_reg <- validation %>% left_join(b_i , by = "movieId")%>% left_join(b_u , by = "userId")%>% replace_na(list(b_i=0, b_u=0))%>% mutate(pred = mu_hat+b_i+b_u)%>% .$pred
return(RMSE(validation$rating,predicted_ratings_reg))

})

# Plot RMSE_function_reg vs. lambdas to select optimal lambda

qplot(lambdas,RMSE_function_reg)
```

```
Getting the lambda value that minimises the RMSE

```{r echo=FALSE, include=TRUE}
lambda <- lambdas[which.min(RMSE_function_reg)]
lambda
```

Now we shall be predicting the RMSE on the validation set with this mininised lambda value

```{r echo=FALSE, include=TRUE}

# Calculating average accross all the movies 'mu_hat'

mu_hat <- mean(train$rating)

# Computing the regularised estimate by movie 'b_i' using lambda

b_i <- train %>% group_by(movieId)%>% summarize(b_i = sum(rating - mu_hat)/(n() + lambda),n_i = n())

# Computing regularised estimate by user 'b_u' using lambda

b_u <- train %>% left_join(b_i, by = "movieId")%>% group_by(userId)%>% summarize(b_u = sum(rating - mu_hat - b_i)/(n() + lambda), n_u = n())

# Computing the predicted ratings on the validation dataset

predicted_ratings_reg <- validation %>% left_join(b_i, by = "movieId")%>% left_join(b_u, by = "userId")%>% replace_na(list(b_i=0, b_u=0))%>% mutate(pred = mu_hat + b_i +b_u)%>% .$pred

# Test and save results

user_movie_reg_rmse <- RMSE(validation$rating,predicted_ratings_reg) user_movie_reg_rmse rmse_results <- rbind(rmse_results, data.frame(method = "Regularised User & Movie effect model on validation set" , RMSE = user_movie_reg_rmse )) rmse_results%>% knitr::kable()%>% kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"), position = "center", font_size = 10, full_width = FALSE)

```
 Although there has been some improvement from 0.8658 to 0.86522 this is still some dista

Now we shall be trying the regularised 'user_movie_genre model' on the validation set

```{r echo=FALSE, include=TRUE}
##  Testing the regularized 'User and Movie and genre' effect model on the validation se

## Defining a table of lambdas
lambdas <- seq(0, 10, 0.25)
```
```

```
## For each lambda we shall be working on b_i, b_u , predict rating and test accuracy.
RMSE_function_reg  <- sapply(lambdas,function(l){
## Calculating average accross all the movies 'b_i'
mu_hat <- mean(train$rating)

## Calculating the average by movie 'b_i'
b_i <- train %>%
group_by(movieId)%>%
summarize(b_i = sum(rating - mu_hat)/(n()+ l))

## Calculating the average by user 'b_u'
b_u <- train%>%
left_join(b_i, by = "movieId")%>%
group_by(userId)%>%
summarize(b_u = sum(rating - b_i - mu_hat)/(n()+ l) )

## Calculating the average by genre'b_U_g'

b_u_g <- train %>%
   left_join(b_i, by='movieId') %>%
   left_join(b_u, by='userId') %>%
   group_by(genres) %>%
   summarize(b_u_g = sum(rating - mu_hat - b_i - b_u)/(n()+ l))

## Computing the predicted ratings on the validation dataset
predicted_ratings_reg <- validation %>%
left_join(b_i , by = "movieId")%>%
left_join(b_u , by = "userId")%>%
left_join(b_u_g, by = "genres") %>%
replace_na(list(b_i=0, b_u=0,b_u_g = 0))%>%
mutate(pred = mu_hat+b_i+b_u+b_u_g)%>% .$pred
return(RMSE(validation$rating, predicted_ratings_reg))
})
## Plot RMSE_function_reg vs. lambdas to select optimal lambda
qplot(lambdas,RMSE_function_reg)
```

Getting the lambda value that minimises the RMSE

```{r echo=FALSE, include=TRUE}

lambda <- lambdas[which.min(RMSE_function_reg)] lambda
```

```
Now we shall be predicting the RMSE on the validation set with this mininised lambda val

```{r echo=FALSE, include=TRUE}

## Calculating average accross all the movies 'mu_hat'
mu_hat <- mean(train$rating)

## Computing regularised estimate of b_i using lambda
b_i <- train %>%
group_by(movieId)%>%
summarize(b_i = sum(rating - mu_hat)/(n() + lambda),n_i = n())

## Computing regularised estimate of b_u using lambda
b_u <- train %>%
left_join(b_i, by = "movieId")%>%
group_by(userId)%>%
summarize(b_u = sum(rating - mu_hat - b_i)/(n() + lambda), n_u = n())

## Computing regularised estimate of b_u_g using lambda
b_u_g <- train %>%
   left_join(b_i, by='movieId') %>%
   left_join(b_u, by='userId') %>%
   group_by(genres) %>%
   summarize(b_u_g = sum(rating - mu_hat - b_i - b_u)/(n()+ lambda), n_u_g = n())


predicted_ratings_reg <- validation %>%
left_join(b_i, by = "movieId")%>%
left_join(b_u, by = "userId")%>%
left_join(b_u_g, by = "genres")%>%
replace_na(list(b_i=0, b_u=0,b_u_g = 0))%>%
mutate(pred = mu_hat + b_i +b_u + b_u_g)%>% .$pred
## Test and save results
user_movie_genre_reg_rmse  <- RMSE(validation$rating,predicted_ratings_reg)
user_movie_genre_reg_rmse
rmse_results <- rbind(rmse_results, data.frame(method = "Regularised User & Movie & genr
rmse_results%>%
knitr::kable()%>%
kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
position = "center",
font_size = 10,
full_width = FALSE)
```

We seem to have finally achieved an RMSE of 0.86485 .
```

4.0 Conclusion

The primary objective of this project was to predict user movie ratings based on the other user's ratings using a 10M version of the MovieLens dataset. The exploration of the dataset and the key revelations of their visualization has lead us to believe that the features strongly suggesting an influence on prediction would be the movie(movieId),the user(userId) and the genre of the movie(genres).Accordingly algorithms with different combinations of these features were trained and tested to evaluate the accuracy of the RMSE(prediction). Results and performance of each of those models have been individually tabulated and discussed under the relevant sections of the detailed report. Finally, we have achieved the highest RMSE accuracy of 0.86482 with a lambda of 5 on the validation set with the Regularised(Penalized) Root Mean Square Error approach. Talking of the future work, we have good scope of utilizing Matrix factorization in the context of this movie recommendation system.Our final model leaves out an important source of variation related to the fact that groups of movies have similar rating patterns and groups of users have similar rating patterns as well. We could also train different models, recommender engines and ensemble methods in our endeavour to better our accuracy .Considering (in my perception though) that, limited scope of this project does not call for further exhaustive work than what has been done here.