# Movielens-Project-Report

Harshad B.

## Introduction

Machine learning helps us to describe data and deduce useful patterns from the same. The aim of machine learning is to process data into helpful information and naturally intuitive solutions. In 2006, Netflix placed a seven-figure bounty on a verified improvemnent to their movie recommendation system.

The following project is based on the Netflix Challenge. In this project, we use the MovieLens 10M dataset that consists of 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users.

The dataset is divided into edx and validation sets in a 90-10 ratio.

## Approach:

First, the edx set has been divided into two sets: edx_train and edx_test. Various models will be created using the *edx_train* set and their RMSEs will be calculated on the *edx_test* set. When a model with RMSE close to the expected RMSE is achieved, then the *edx* set will be used to train and predict on the *vaildation* set.

### Partitioning of *edx* dataset

```
testindex <- createDataPartition(edx$rating, times = 1, p = 0.2, list = FALSE)
edx_train <- edx[-testindex,]
edx_test <- edx[testindex,]
edx_test <- edx_test %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")
```

## Method :

### Calculating rmse of Average Method on *edx_test* set

```
mean_tt <- mean(edx_train$rating)
rmse_avg <- RMSE(edx_test$rating, mean_tt)
```

## Results

| Method | RMSE |
|---|---|
| Just the Average (edx_test) | 1.060448 |

## Method :

**Calculating rmse of movie effect on *edx_test* set**

```
movie_tt <- edx_train %>%
  group_by(movieId) %>%
  summarize(bi = mean(rating - mean_tt), .groups = 'drop')
pred_bi <- mean_tt + edx_test %>%
  left_join(movie_tt, by='movieId') %>%
  .$bi
rmse_movie <- RMSE(pred_bi, edx_test$rating)
```

## Results

| Method | RMSE |
|---|---|
| Just the Average (edx_test) | 1.0604483 |
| Movie Effect Model (edx_test) | 0.9437588 |

## Method :

**Calculating rmse of movie and user model on *edx_test* set**

```
user_tt <- edx_test %>%
  left_join(movie_tt, by='movieId') %>%
  group_by(userId) %>%
  summarize(bu = mean(rating - mean_tt), .groups = 'drop')
pred_bu <- edx_test %>%
  left_join(movie_tt, by='movieId') %>%
  left_join(user_tt, by='userId') %>%
  mutate(pred = mean_tt + bi + bu) %>%
  .$pred
rmse_user <- RMSE(pred_bu, edx_test$rating)
```

## Results

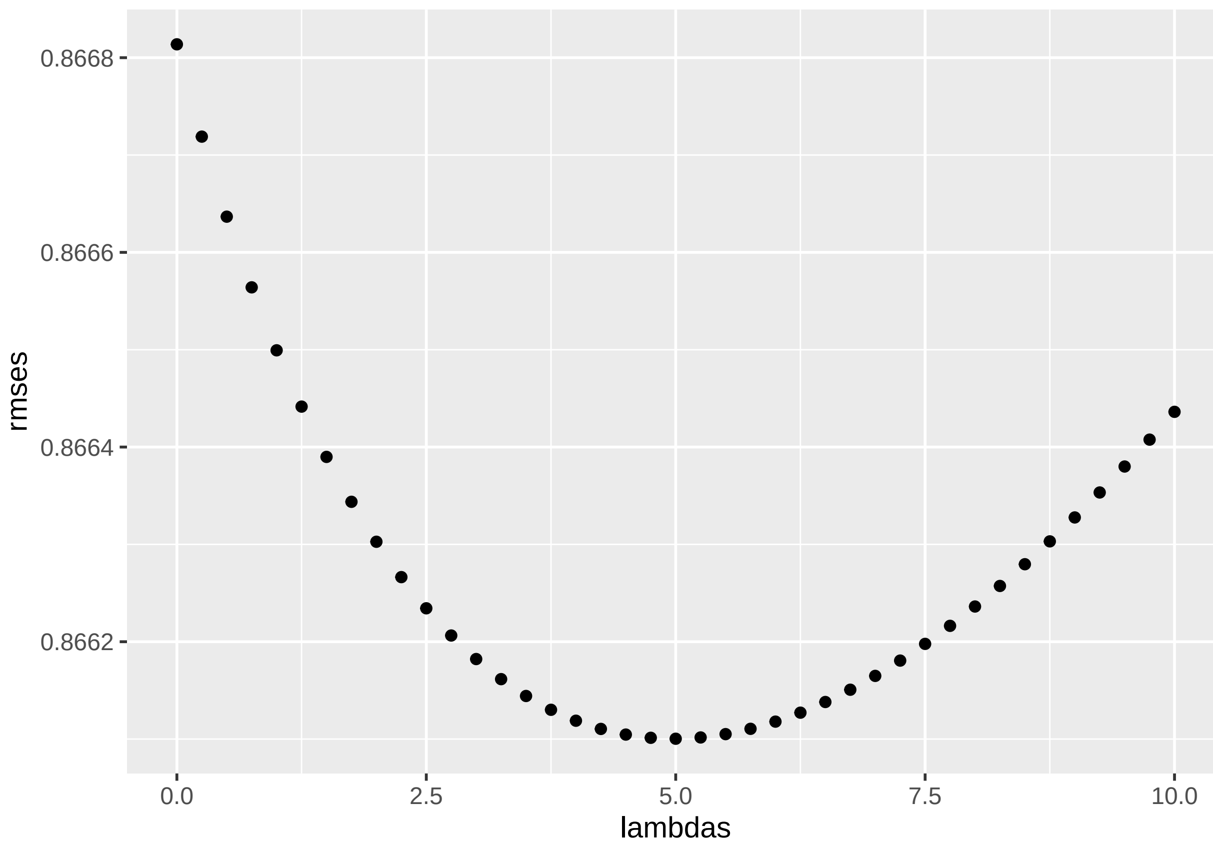
| Method | RMSE |
|---|---|
| Just the Average (edx_test) | 1.0604483 |
| Movie Effect Model (edx_test) | 0.9437588 |
| User + Movie Effect Model (edx_test) | 0.8678670 |

### Method :

#### Calculating rmse of regularized movie and user model on *edx_test* set

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx_train$rating)
  bi <- edx_train %>%
    group_by(movieId) %>%
    summarize(bi = sum(rating - mean_tt)/(n()+l), .groups = 'drop')
  bu <- edx_train %>%
    left_join(bi, by="movieId") %>%
    group_by(userId) %>%
    summarize(bu = sum(rating - bi - mean_tt)/(n()+l), .groups = 'drop')
  pred <-
    edx_test %>%
    left_join(bi, by = "movieId") %>%
    left_join(bu, by = "userId") %>%
    mutate(pred = mean_tt + bi + bu) %>%
    .$pred
  return(RMSE(pred, edx_test$rating))
})
```

The plot below shows us qq-plot of *lambdas vs. rmses* for the regularized movie and user model on **edx_test**




set

```
## [1] 5
```

## Results

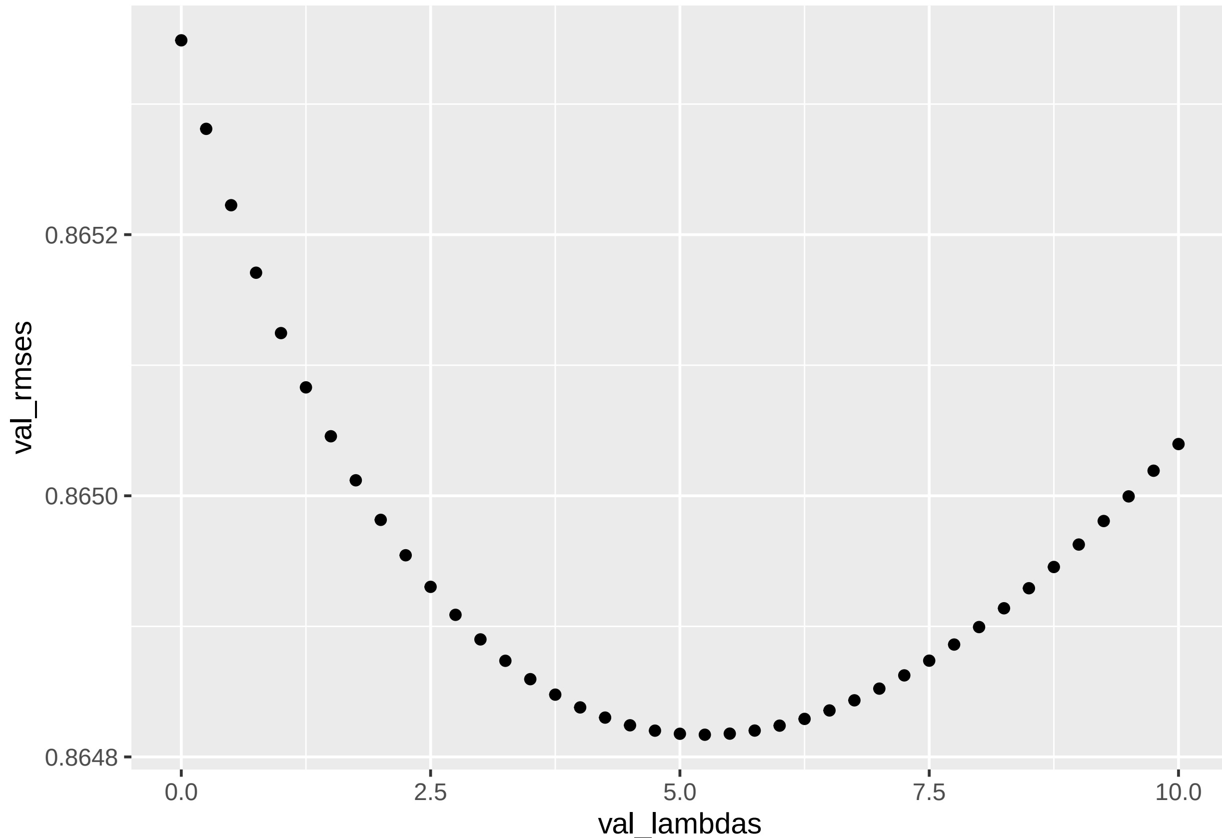
| Method | RMSE |
|---|---|
| Just the Average (edx_test) | 1.0604483 |
| Movie Effect Model (edx_test) | 0.9437588 |
| User + Movie Effect Model (edx_test) | 0.8678670 |
| Regularized Movie + User Effect Model (edx_test) | 0.8661004 |

## Method

**calculating rmse of regularized movie and user model on *validation* set**

```
val_lambdas <- seq(0, 10, 0.25)
val_rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
  bi <- edx %>%
    group_by(movieId) %>%
    summarize(bi = sum(rating - mu)/(n()+l), .groups = 'drop')
  bu <- edx %>%
    left_join(bi, by="movieId") %>%
    group_by(userId) %>%
    summarize(bu = sum(rating - bi - mu)/(n()+l), .groups = 'drop')
  pred <- validation %>%
    left_join(bi, by = "movieId") %>%
    left_join(bu, by = "userId") %>%
    mutate(pred = mu + bi + bu) %>%
    .$pred
  return(RMSE(pred, validation$rating))
})
final_rmse <- min(val_rmses)
```

The plot below shows us qq-plot of *val_lambdas vs. val_rmses* for the regularized movie and user model on **validation** set




```
## [1] 5
```

## Results

| Method | RMSE |
|---|---|
| Just the Average (edx_test) | 1.0604483 |
| Movie Effect Model (edx_test) | 0.9437588 |
| User + Movie Effect Model (edx_test) | 0.8678670 |
| Regularized Movie + User Effect Model (edx_test) | 0.8661004 |
| Regularized Movie + User Effect Model (validation set) | 0.8648170 |

## Conclusion

From the above table, we can see incremental improvements to the RMSE as we supplant our model with bias terms and regularization. Because of the simplicity of the linear model, we are able to predict movie ratings without a serious toll on the computer resources.