MovieLens Project Report

Arezou Tufekci

4/20/2021

1. Introduction

In today's digital age, recommendation systems are among the most popular Machine Learning technologies used in business to personalize content for its consumers. As Steve Jobs said: "a lot of times, people don't know what they want until you show it to them." [1] Recommendation systems play an important role in online streaming services, such as Netflix among others. According to McKinsey, 75% of what users watch on Netflix come from movie recommendations [2]. In the paper "The Netflix Recommender System: Algorithms, Business Value, and Innovation" [3] written by Netflix executives (Carlos A. Gomez-Uribe and Neil Hunt) state that the recommendation system saves the company around $1 billion each year. So the economic impact of recommendation systems is clear as matching recommendations can help improve overall user satisfaction which increases the likelihood of the consumers to return to website or application again.

Machine Learning is an integral part of these modern recommendation systems as it describes data about customers as a whole and personalizes an experience for a single user. The goal of Machine Learning is to process data into useful information and naturally intuitive solutions.

2. Summary

2.1 Objectives

Our goal is to create a movie recommendation system using the MovieLens dataset and apply the lessons learned during *HarvardX's Data Science Professional Certificate program*. In this paper, we will predict movie ratings for users in a large dataset. We do this by training a linear model and generating predicted movie ratings, and then calculating the Root Mean Square Error (RMSE) of the predicted ratings versus the actual ratings. **Our project goal is to be less than RMSE $<$ 0.86490.**

2.2 Structure

This document is structured as follows:

1. **Prepare Data:** Describes the dataset and summarizes the goal of the project and key steps that were performed.
2. **Define Methods and Explain Results:** Defines the model as we improve upon the model and explains the results for each method.
3. **Final Validation:** Evaluates our final model.
4. **Conclusion:** Concludes with a brief summary of the report.

We begin by utilizing MovieLens dataset that contains 10 million movie ratings. Below, we quickly explore the most rated movie to be Pulp Fiction (1994) with **31362** movie ratings; whereas the count of movies rated single time is at **126**. This leads us to believe that there is wide varying numbers of ratings for each movie.

```r
# Most rated movies
edx %>%
     group_by(title)   %>%
     summarize(number_ratings = n())   %>%
     arrange(desc(number_ratings))
```

```
## # A tibble: 10,676 x 2
##    title                                                        number_ratings
##    <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
## # ... with 10,666 more rows
```

```
# Number of single rated movies
edx %>%
    group_by(title) %>%
    summarize(number_ratings = n()) %>%
    filter(number_ratings==1) %>%
    count() %>%
    pull()
```

```
## [1] 126
```

Since the size of the dataset is large, a built-in Machine Learning algorithm using R packages may require many resources for a laptop. Therefore, we decided to investigate Linear Models and use RMSE function as the measure of accuracy.

3. Approach

3.1 Data Preparation

We are going to train a Machine Learning algorithm using the inputs in one subset to predict movie ratings in the validation set. In order to ease computation, a smaller set of data is used (10 million) version of the MovieLens dataset). We downloaded the dataset from MovieLens website and split it into two subsets of data for training and evaluation. The training dataset is called **edx** and the evaluation dataset is called **validation** with 90% and 10% of the original dataset respectively. Then we split the edx set in two parts, the train set, and the test set again with 90% and 10% of edx respectively.

The model is built in the training set, and the test set will be used to test this model until the RMSE target is achieved (RMSE < 0.8649). When the model is complete, next step we will use the validation set to calculate the final RMSE.

The following code sets up the training and test datasets.

```
###################################
# Create Training & Test Datasets
###################################
set.seed(1, sample.kind="Rounding")

# The training set will be 90% of edx data and the test set will be the remaining 10%
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
```

```r
train_set <- edx[-test_index,]
temp <-  edx[test_index,]

# Ensure movieId and userId in test set are also in train set
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Add rows removed from test set back into train set
removed <-  anti_join(temp, test_set)
train_set <- rbind(train_set, removed)
rm(test_index, temp, removed)
```

After exploring the data, we decided to remove some of the features (columns) in order to make the model less complex & save computer resources.

```r
train_set <- train_set  %>%
                    select(userId, movieId, rating, title)
test_set  <- test_set  %>%
                    select(userId, movieId, rating, title)
```

3. Define Methods & Explain Results

In this step, we have examined three main methods to improve upon each step. Three main methods are examined and the RMSE is calculated for each method. The target is to get closer to the RMSE of **0.868** value.

1. **Random Prediction**
2. **Linear Model**
3. **Regularization**

3.1 Random Prediction

3.1.1 Define Method

The simplest method is to randomly predict the movie ratings. We look at the distribution of the **10,677** different movies in the edx set as shown in figure below to find out that the distribution is fairly symmetric. We can then use probability distribution to make predictions. If for example we know that the probably of all users giving a movie a rating 3 is 20%, we may predict that 20% of the movie ratings will have a rating of 3.

```r
edx %>%   group_by(movieId) %>%
          summarize(n = n()) %>%
          ggplot(aes(n)) +
          geom_histogram(color = "pink") +
          scale_x_log10()  +
          xlab("# of Ratings") +
          ylab("# of Movies")
```

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

# Build the probability for each movie rating
pr <- function(x, y) mean(y == x)
rating <- seq(0.5,5,0.5)

# Guess the probability of each rating utilizing Monte Carlo simulation
```

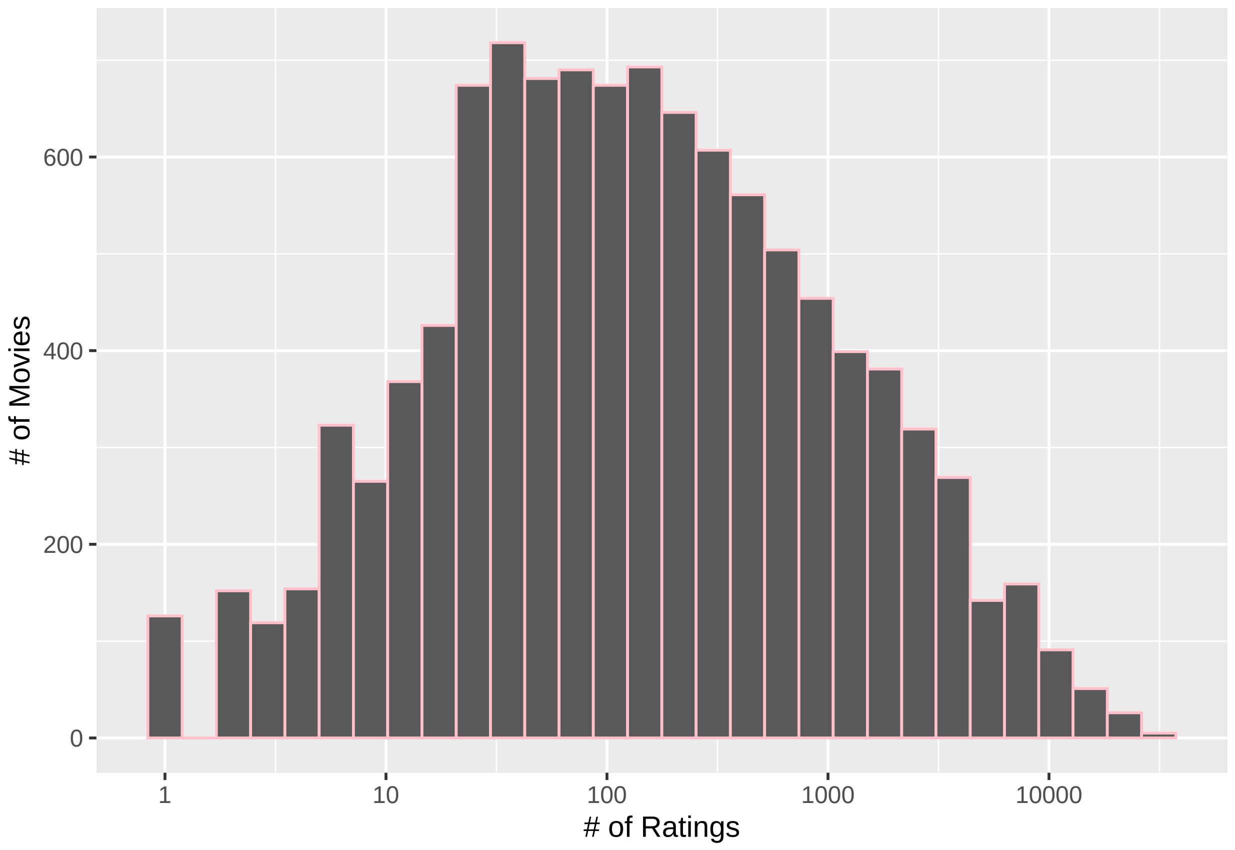


Figure 1: Random Distribution Fairly Symetric

```r
B <- 10000
Mont <- replicate(B, {
  samp <- sample(train_set$rating, 100, replace = TRUE)
  sapply(rating, pr, y= samp)
})
prob <- sapply(1:nrow(Mont), function(x) mean(Mont[x,]))

# Predict random ratings
y_hat_random <- sample(rating,  size = nrow(test_set),  replace = TRUE,  prob = prob)

#Use test set and then calculate RMSE
RMSE(test_set$rating, y_hat_random)
```

```
## [1] 1.499797
```

3.1.2 Explain Results

RMSE 1.499797 is pretty high. Since these types of predictions can be very erroneous, we will look into methods with better predictions.

Method	RMSE
Goal	0.864900
Random Prediction	**1.499797**

3.2 Linear Model

3.2.1 Linear Method Using Mean

3.2.1.1 Define Method

For initial estimation, we use the average across every user and every movie as all of our predicted movie ratings. Statistical theory explains that the average of observed ratings minimizes the RMSE. Therefore, this is where we start. The equation takes the form of,

$Y_{u,i} = \mu$,

where $Y_{u,i}$ is the predicted rating of user u and movie i and μ is the average rating across all entries. This is computed as **3.512** (`mean(edx$rating)`).

```r
mu <- mean(edx$rating)
RMSE(test_set$rating, mu)
```

```
## [1] 1.060054
```

3.2.1.2 Explain Results

RMSE is **1.060054** which is lower than method 1 at 1.499797. We know that the average of observed ratings minimizes the RMSE so this makes sense. Can we improve upon this? We will answer that next:

Method	RMSE
Goal	0.864900
Random Prediction	1.499797
Linear Model - Mean	**1.060054**

3.2.2 Linear Method Using Mean and Movie Bias

3.2.2.1 Define Method

Now, we will look into other features to improve the model. We know that different movies have different rating distribution (popular ones have higher and more ratings). This is movie effect or bias. In order to count for this bias, we add the term bi for movie bias. This term averages the rankings for any movie i because some are favored or unfavored more than others. So the new model becomes:

$Y_{u,i} = \mu + b_i$.

where $Y_{u,i}$ is the predicted rating of user u, movie i, movie bias b_i and μ is the average rating across all entries.

```
 # movie bias term = bi
bi <- train_set %>%
group_by(movieId) %>%
summarize(bi = mean(rating - mu))

# All (unknown) movie ratings prediction using mu and bi
predicted_ratings <- test_set    %>%
      left_join(bi, by='movieId')   %>%
      mutate(predict = mu + bi)     %>%
      pull(predict)

# Now calculate RMSE of movie affect
RMSE(test_set$rating, predicted_ratings)
```

```
## [1] 0.9429615
```

3.2.2.2 Explain Results

The RMSE is now lowered to **0.9429615** compare to the previous RMSE value of 1.060054. We hope to be making better predictions now that we consider movie effect.

Method	RMSE
Goal	0.864900
Random Prediction	1.499797
Linear Model:Mean	1.060054
Linear Model:Mean+Movie Bias	**0.9429615**

3.2.3 Linear Method Using Mean, Movie Bias, and User Bias

3.2.3.1 Define Method

Similar to movie affect, different users have different rating pattern. For example, some users like most movies and consistently rate 4 and 5, while other users dislike most movies rating 1 or 2. Now we introduce the user bias term bu in order to further improve our model. This term minimizes the effect of extreme ratings made by users that love or hate every movie. Each user u is given a bias term that sways their predicted movies. This is user bias and here is the formula for it:

$Y_{u,i} = \mu + b_i + b_u$.

where $Y_{u,i}$ is the predicted rating of user u, movie i, movie bias b_i, user bias b_u and μ is the average rating across all entries.

```
# Now we add user bias term = bu
bu <- train_set %>%
```

```r
left_join(bi, by='movieId')  %>%
group_by(userId)  %>%
summarize(bu = mean(rating - mu - bi))

# predicting new movie ratings bringing in movie and user bias
predicted_ratings <- test_set   %>%
     left_join(bi, by='movieId')  %>%
     left_join(bu, by='userId')   %>%
     mutate(predict = mu + bi + bu) %>%
     pull(predict)

# calculate RMSE of movie + user bias effect
RMSE(predicted_ratings, test_set$rating)
```

```
## [1] 0.8646843
```

3.2.3.2 Explain Results

RMSE is now lowered to **0.8646843** compare to 0.9429615. We hope that we can make better predictions with our new model now that we bring into account user bias.

Method	RMSE
Goal	0.864900
Random Prediction	1.499797
Linear Model:Mean	1.060054
Linear Model:Mean+Movie Bias	0.9429615
Linear Model:Mean+Movie+User Bias	**0.8646843**

3.3 Regularization

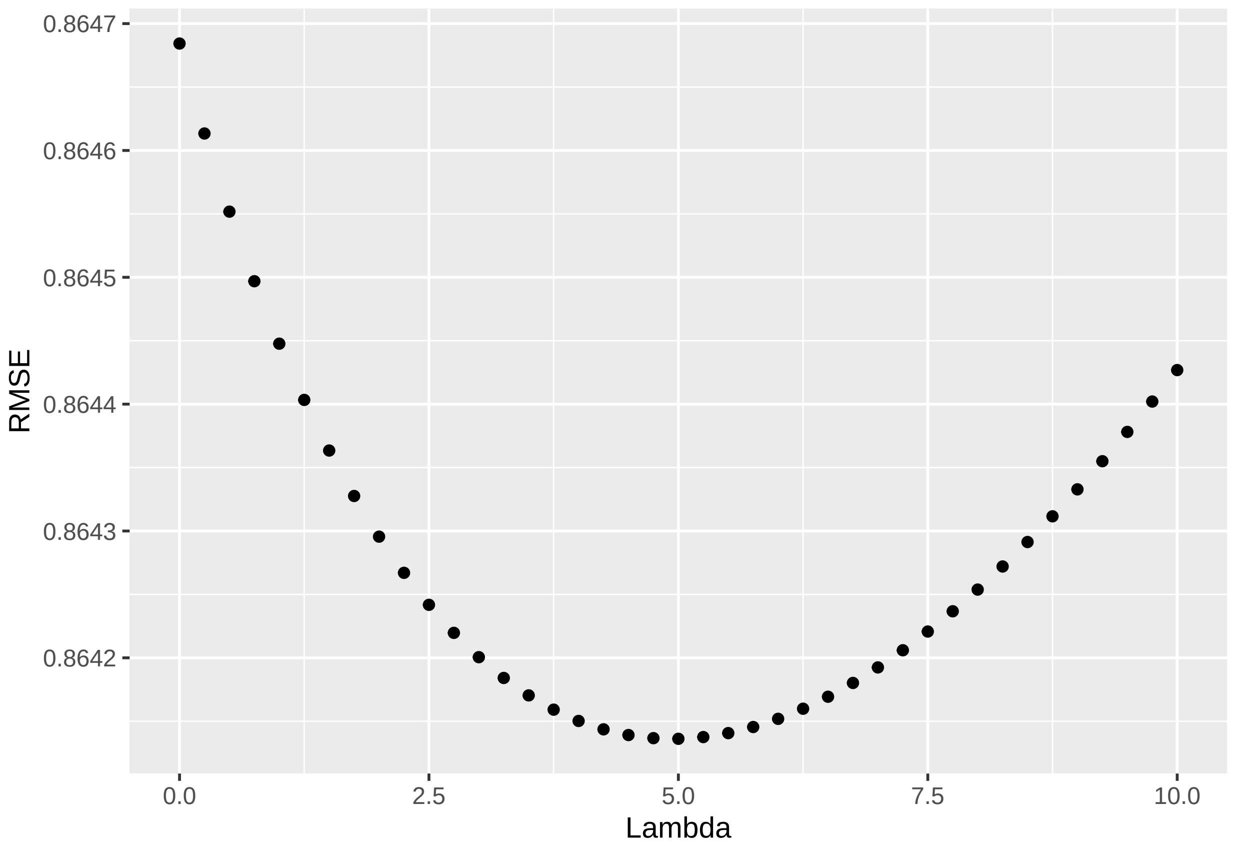
Finally, we employ regularization to reduce the effect of large errors in our predictions. Regularization penalizes incorrect estimates on small sample sizes. For instance, our bi term accounts for the average deviation on all ratings of a movie, whether there is 1 or 100 ratings to the movie. We use regularization to reduce the dramatic effect that a exceptionally extreme rating will have on our bi term. This method is also applied to the user bias term bu to reduce large anomalies in the ratings of users. Regularization achieves the same goal as confidence intervals when you are only able to predict a single number, not an interval. Our new model is:

Regularization achieves the same goal as confidence intervals when you are only able to predict a single number, not an interval. Our new model is:

$\frac{1}{N}\sum_{u,i}(Y_{u,i} - \mu - b_i - b_u)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2)$,

where the first term is our previous least squares equation and the last term is the penalty with large bias terms. Minimizing the biases using a single λ is the goal to our model shown above. We test `lamda <- seq(from=0, to=10, by=0.25)` and plot the results below:

```r
# Write a regularization function to fine tune using lambda
regularization <- function(lambda, train, test) {

  # Average rating data on trainset
  mu <- mean(train$rating)

  # Movie effect, bi
  b_i <- train %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu)/(n()+lambda))

  # User effect, bu
  b_u <- train %>%
        left_join(b_i, by = "movieId")  %>%
        filter(!is.na(b_i))  %>%
        group_by(userId)  %>%
        summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

  # Rating prediction using mu + bi + bu
  predicted_ratings <- test   %>%
                     left_join(b_i, by = "movieId")  %>%
                     left_join(b_u, by = "userId")   %>%
                     filter(!is.na(b_i), !is.na(b_u))  %>%
                     mutate(pred = mu + b_i + b_u) %>%
                    pull(pred)

  return(RMSE(predicted_ratings, test$rating))
}
# An effective method to choose lambda that minimizes the RMSE is running simulations with several valu
# Define a set (lambdas)
lambdas <- seq(0, 10, 0.25)

# Tuning
RMSEs <- sapply(lambdas, regularization, train = train_set, test = test_set)

# Plot the lambda vs RMSE
tibble(Lambda = lambdas, RMSE = RMSEs) %>%
  ggplot(aes(x = Lambda, y = RMSE)) +
    geom_point()
```



```
# Pick lambda returns the minimum RMSE
lambda <- lambdas[which.min(RMSEs)]
mu <- mean(train_set$rating)

# bring in Movie effect, bi
b_i <- train_set    %>%
       group_by(movieId)      %>%
       summarize(b_i = sum(rating - mu)/(n()+lambda))

# bring in user effect, bu
b_u <- train_set   %>%
       left_join(b_i, by = "movieId")  %>%
       group_by(userId)   %>%
       summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

# Prediction now that we have fine tuned using lambda
y_hat_reg <- test_set %>%
             left_join(b_i, by = "movieId") %>%
             left_join(b_u, by = "userId") %>%
             mutate(pred = mu + b_i + b_u) %>%
             pull(pred)

RMSE(test_set$rating, y_hat_reg)
```

```
## [1] 0.8641362
```

Method	RMSE
Goal	0.864900
Random Prediction	1.499797
Linear Model:Mean	1.060054
Linear Model:Mean+Movie Bias	0.9429615
Linear Model:Mean+ Movie + User Bias	0.8646843
Regularization	**0.8641362**

4. Final Validation Results

4.1 Linear Model With Regularization

During the training and testing phases, the linear model with regularization achieved the target RMSE with a small margin. Here we do the final validation with the validation set.

```r
mu_edx <- mean(edx$rating)

# Movie effect (bi)
b_i_edx <- edx %>%
           group_by(movieId) %>%
           summarize(b_i = sum(rating - mu_edx)/(n()+lambda))

# User effect (bu)
b_u_edx <- edx %>%
           left_join(b_i_edx, by="movieId") %>%
           group_by(userId) %>%
           summarize(b_u = sum(rating - b_i - mu_edx)/(n()+lambda))

# Prediction
y_hat_edx <- validation %>%
             left_join(b_i_edx, by = "movieId") %>%
             left_join(b_u_edx, by = "userId") %>%
             mutate(pred = mu_edx + b_i + b_u) %>%
             pull(pred)

RMSE(validation$rating, y_hat_edx)
```

```
## [1] 0.8648177
```

Method	RMSE
Goal	0.864900
Random Prediction	1.499797
Linear Model- Mean	1.060054
Linear Model- Mean+Movie Bias	0.9429615
Linear Model- Mean+Movie Bias+User Bias	0.8646843
Regularization (edx)	0.8641362
Regularization (validation)	**0.8648177**

As expected, the RMSE calculated on the validation set **0.8648177** is lower than the target of 0.864900 and slightly higher than the RMSE of the test set 0.8641362.

5. Conclusion

Today, more and more online companies use recommendation systems to improve user interaction with their services. Recommendation systems are efficient Machine Learning solutions which can help increase customer satisfaction and retention, and also lead to significant increase of revenues. We began with preparing the dataset and perform data exploration to help us build our model. Next, we created a random model that predicts the rating based on the probability distribution of each rating. This model gives the worst result. Next model was linear model using the mean of the observed ratings. We observed imporovement. From there, we added movie and user effects to our linear model, that models the user behavior and movie distribution. With regularization we added a penalty value for the movies and users with few number of ratings. The linear model achieved the RMSE of **0.8648177**, successfully passing the target of **0.864900**.

Some Machine Learning algorithms are computationally expensive to run on a laptop and therefore were unable to use for testing. Because of the simplicity of the linear model, we are able to predict movie ratings without a serious toll on the computer resources. We used the movie and user feature as our predictors. Modern recommendation systems obviously use many predictors such as genres, playlist, etc.

6. Sources

- [1] Edson, J., 2012. Design Like Apple: Seven Principles for Creating Insanely Great Products, Services, and Experiences. John Wiley & Sons. pp. 47
- [2] Alex Castrounis, 2019. AI for People and Business. A Framework for Better Human Experiences and Business Success. (ebook)
- [3] Gomez-Uribe, C.A. and Hunt, N., 2015. The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 6(4), pp.1–19