MovieLens Capstone Project

Anshuman Dash

9/30/2020

Introduction:

One of the many applications of machine learning is in the use of recommendation systems. Following the 2006 Netflix challenge, we set out to use the MovieLens 10 million data set to create our own recommendation system. Our goal for the MovieLens Capstone project is to utilize the MovieLens dataset to predict user ratings for movies. We utilize techniques learned in class, and expand upon them to train and finally test our final model.

Our specific version of the MovieLens dataset contains 10 million ratings from 72,000 users rating 10,0000 movies. In the data set we get access to the userId, movieId, the rating given by that user, timestamp, title, and the movie genre. This data is the basis to which we are to start our project.

Now our goal is to create a model to predict ratings, accounting for different biases, the sample sizes, etc. We start the project by first downloading, processing, and prepping the data for our modelling. We then start by slowly developing our model, and at each step we check the RMSE value to see how our improvements are going. We then do a final test on our validation set to see how successful we were.

Method/Analysis:

Our methodology can be broken down into stages. The first stage is getting the data and preparing it to be in a format for building our data. We call this the **Data Ingestion Stage**. Our second stage is to create datasets to train and test our data. At this stage we will split the data into as many sets we want to work on. We call this the **Preparing Datasets Stage**. Our next stage is where we start actually modelling our data, we build as many models as we need and see how the RMSE decreases/increases, and we improve the models as we need to. This is the **Modelling Stage**. Our final stage is our **Validation Stage**. During the Validation stage we will see if our final model passes and achieves a RMSE value <0.8490.

As we develop our model, I will be explaining some of the reasons behind my decision making. After our **Methods/Analysis** section we will finish with a presentation of our final results, showcasing how our model performed. Followed my our **Conclusion** section.

We will now proceed through each stage.

Data Ingestion Stage:

In this stage we need to download the data and put it in a form to start creating data sets for developing our model in our next stage. We download the data from GroupLens website, and then format it so that we can create our edx and validation sets. This is also the stage that we download any libraries we may need to run our code. The base code for this portion was provided from the course on downloading and pre-processing the data.

After doing this, we are ready for the next stage.

Preparing Datasets Stage:

We are now ready start creating our data sets to train and test our models. The first step is to split the data set into two parts, our training set and our test set. Our training set will be called *edx*, and our test set will be called *validation*. The edx set will be 90% of our original data set, and validation will therefore be the remaining 10%.

We also made sure that the userId's and movieId's in the validation set are also in the edx set. But we aren't done yet. We need to separate our edx dataset into another train and test set. The reason we do this is to have a test set that is not our validation set. This is to avoid overtraining our model. So we create *edx_train* and *edx_validation* to be the data sets that we use to train, and the test our model to avoid overtraining. The following code showcases how I did this.

```
##### Split Edx Again #####

# This is to avoid over-training, we want to split our set again.

set.seed(1)
edx_test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_train <- edx[-edx_test_index,]
edx_temp <- edx[edx_test_index,]

# Make sure userId and movieId in edx_validation set are also in edx_train set
edx_validation <- edx_temp %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

# Add rows removed from validation set back into edx set
edx_removed <- anti_join(edx_temp, edx_validation)
edx_train <-  rbind(edx_train, edx_removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Now that our datasets are ready, we will begin developing our models in the next stage.

Modelling Stage:

Our modelling can also be broken down into stages, and each stage our focus is to improve the RMSE until we feel comfortable to do a final validation set. Our goal is to create a baseline model, and the simplest model is to take the average of the ratings in the edx_train dataset and check the RMSE against our edx_validation set. In this model we are predicting the same rating for all movies regardless of each user. Here is how I coded it:

```
##### Base Mean Model #####

# First create data frame to store our RMSE Results

rmse_results <- data_frame()

# Base Model
# Just using the mean
mu <- mean(edx_train$rating)
mu
```

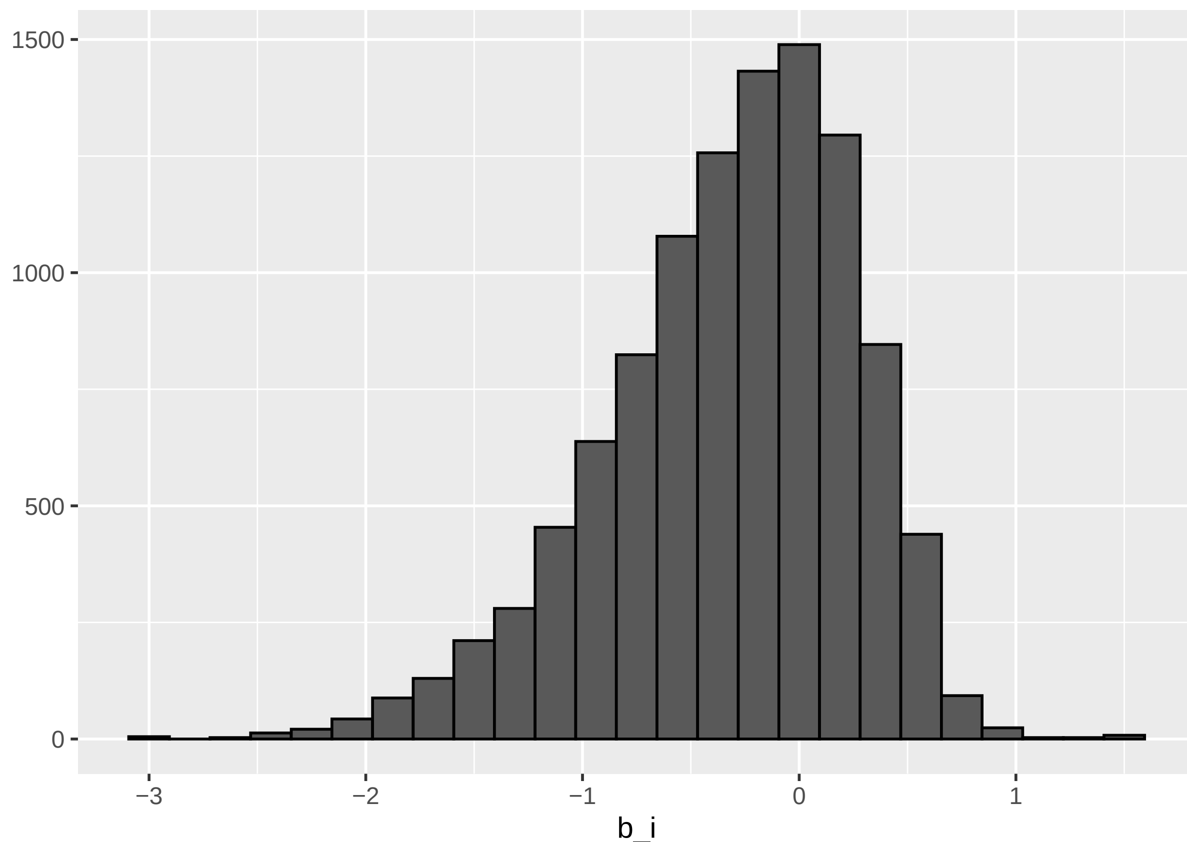
```
## [1] 3.512354
```

```r
base_rmse <- RMSE(edx_validation$rating, mu)
rmse_results <- data_frame(Model = "Base Mean Model", RMSE = base_rmse)

rmse_results %>% knitr::kable() # To look at our results
```

Model	RMSE
Base Mean Model	1.059

Movie Effect Model:

As you can see we get an initial RMSE of 1.06, meaning we have a lot of work to do. To improve our model we need to look at how movies have an affect. We can look at the biases in movies, since different movies will have higher ratings, and others will have lower ratings on average. Lets first take a look at the histogram of our movie bias.



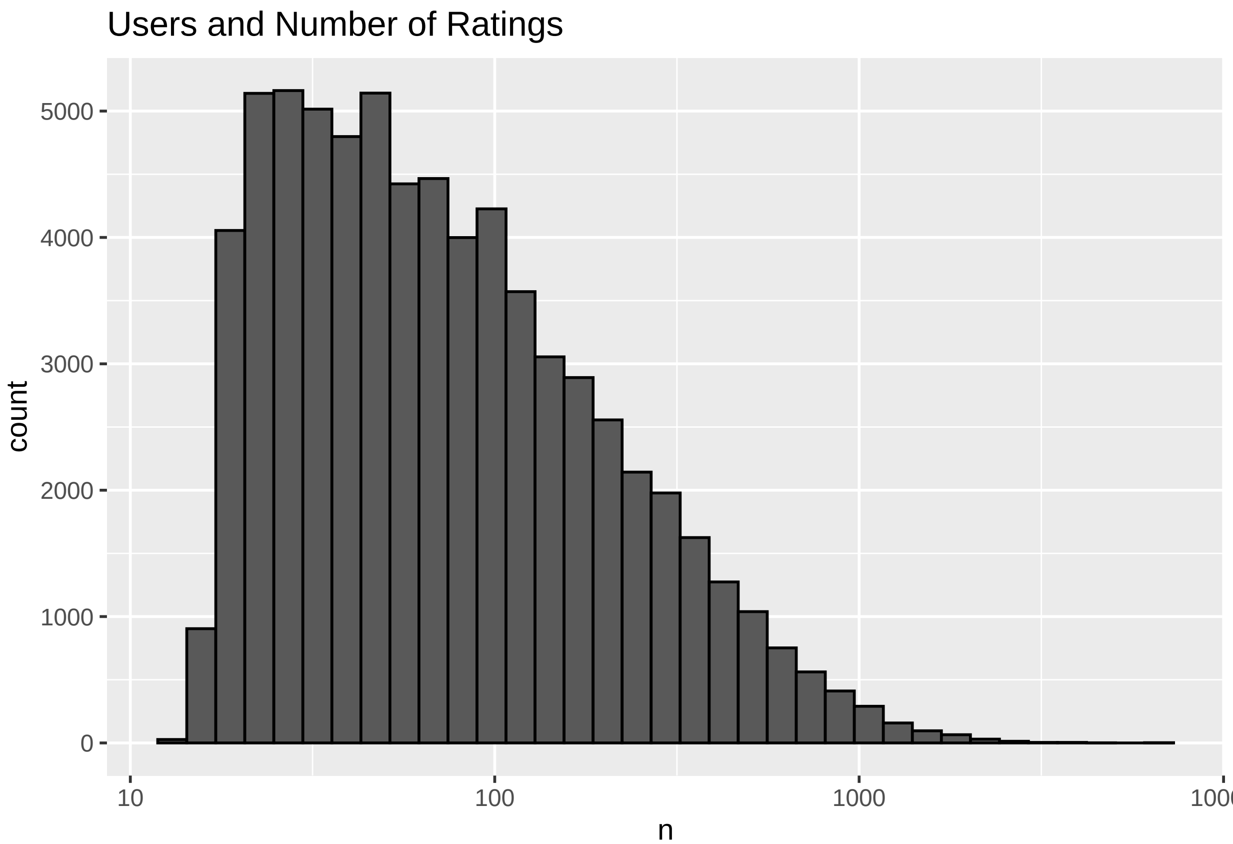
What you will notice is that our histogram is skewed to the left, which is the negative rating effect. So lets go ahead and build the rest of our movie effect model, and check the RMSE.

Model	RMSE
Base Mean Model	1.0590002
Movie Effect Model	0.9426564

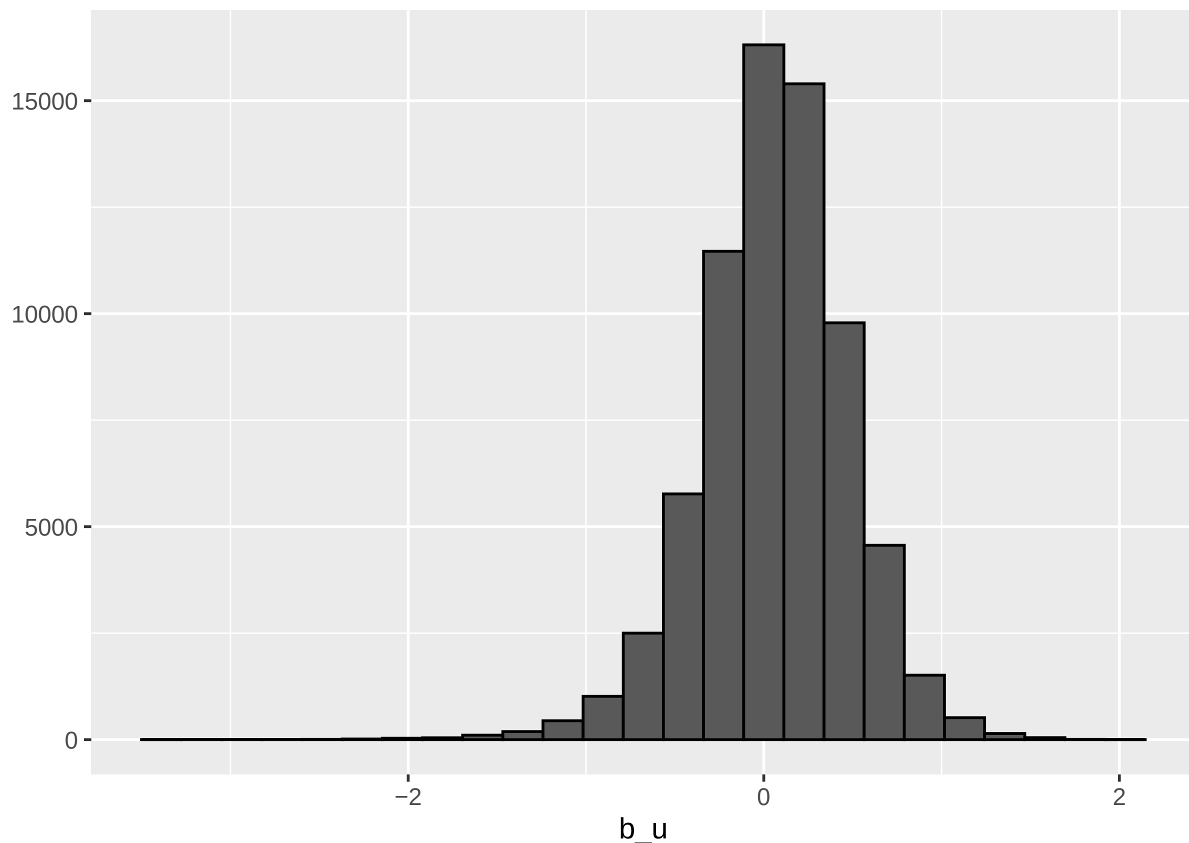
With an RMSE value of 0.94296, we have improved our model. But it is still not enough. Lets take a look at the user effect.

User and Movie Effect Model:

Let us first take a look at the user breakdown. What you will notice is that some users are a lot more active than others. But from a general stand point we see most users fall between 10 to 100 reviews.



Now we can look at a plot of the user bias. We can do that by creating another histogram.



We can see the user bias is actually has a much better distribution compared to the movie bias histogram. So let us go ahead and build our model and look at the RMSE generated.

```
pred_ratings_user <- edx_validation %>%
  left_join(movie_avg, by = 'movieId') %>%
  left_join(user_avg, by = 'userId') %>%
  mutate(prediction = mu + b_i + b_u) %>%
  pull(prediction)

model_user_movie <- RMSE(edx_validation$rating, pred_ratings_user)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model = "Movie and User Effect Model",
                                     RMSE = model_user_movie))

rmse_results %>% knitr::kable()
```

Model	RMSE
Base Mean Model	1.0590002
Movie Effect Model	0.9426564
Movie and User Effect Model	0.8646047

We see that our RMSE has improved again. We could stop here and try to test our model on the final validation test, but that wouldn't be a good idea. Our edx_train set and edx_validation set being a smaller population than the original edx and validation could give us better results. Therefore we will want to really improve our model before doing a final validation test. We are also continuing to add more and more factors and next we will add our b_g term genre.

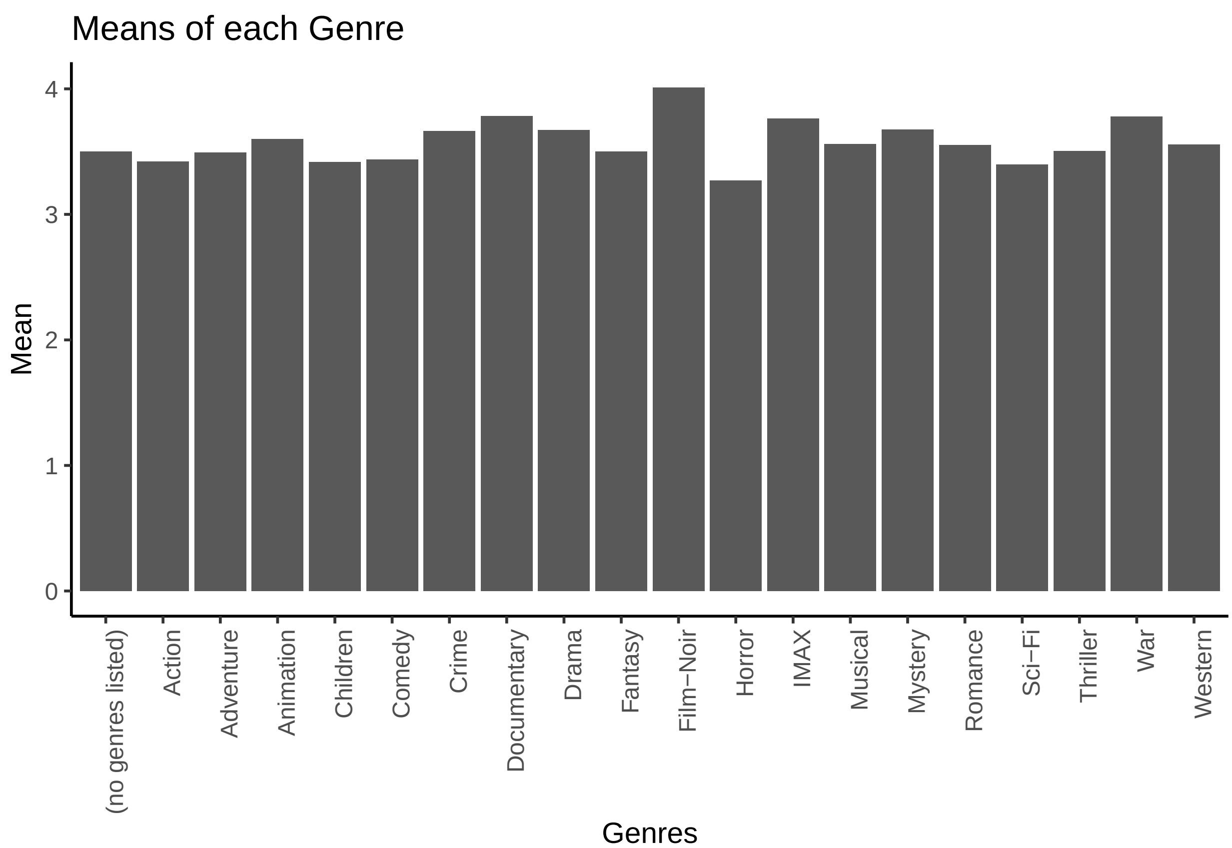
Movie, User, and Genre Model:

Now we want to further improve our model by applying the effect of genres. Let's explore some genre graphs to see how genres can impact the ratings. To do this we first need to separate the genres since some movies have multiple genre tags.

```
### WARNING: This step takes a long time
# This is to seperate the genres out into individual tags to then graph the mean rating of each genre

edx_genre <- edx %>%
   mutate(genre = fct_explicit_na(genres,
                                    na_level = "(no genres listed)")
          ) %>%
   separate_rows(genre,
                 sep = "\\|")
```

The following graph shows how different genres produce average ratings. This is important as we want to utilize genres as an effect in our model.

```
# Here we access the dataset we created with the individual genres and calculatate the means for each g

edx_genre %>%
   group_by(genre) %>%
   summarise(mean = mean(rating)) %>%
   ggplot(aes(genre, mean)) +
   theme_classic()  +
   geom_col() +
   theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
   labs(title = "Means of each Genre",
        x = "Genres",
        y = "Mean")
```



To make our model we will be adding the b_g which is the bias associated with the genre. In my case I did not carry out the effect on individual genres, that is an improvement that can be added on future models. I will talk more about improvements later on.

```
##### Movie, User, and Genre Model #####
genre_avg <- edx_train %>%
  left_join(movie_avg, by = "movieId") %>%
  left_join(user_avg, by = "userId") %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))

pred_ratings_genre <- edx_validation %>%
  left_join(movie_avg, by = "movieId") %>%
  left_join(user_avg, by = "userId") %>%
  left_join(genre_avg, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  pull(pred)

model_user_movie_genre <- RMSE(edx_validation$rating, pred_ratings_genre)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model = "Movie, User. and Genre Effect Model",
                                     RMSE = model_user_movie_genre))

rmse_results %>% knitr::kable()
```

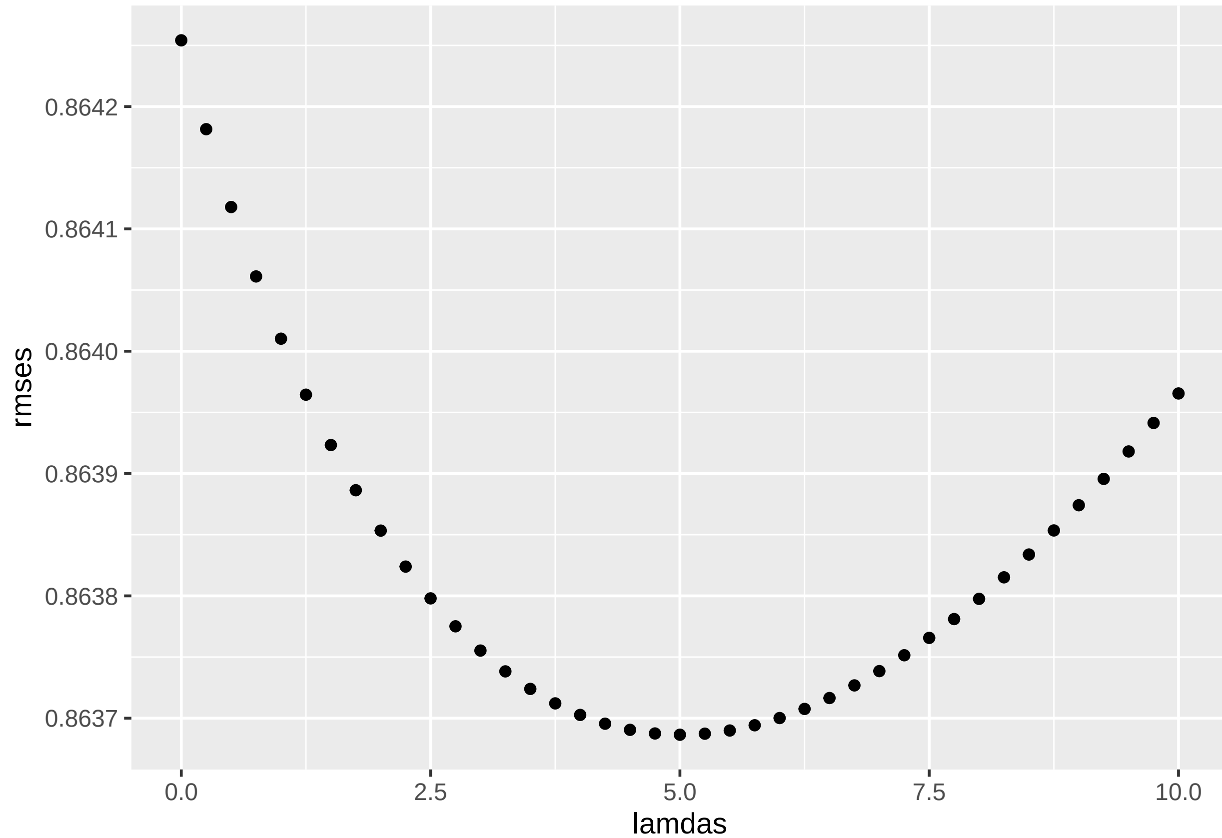
Model	RMSE
Base Mean Model	1.0590002
Movie Effect Model	0.9426564
Movie and User Effect Model	0.8646047
Movie, User. and Genre Effect Model	0.8642542

What we notice is another improvement in our RMSE, from 0.864683 to 0.864321, which is great. But we are not done just yet. We still want to ensure that when we do our final validation test that we are sure we will get a RMSE value below 0.8649, therefore we want to keep creating a cushion. Our next step is to apply regularization methods to our model.

Regularization

Regularization is a method that adds a penalty term to samples with larger estimates from a smaller sample size. For my model I have applied regularization to create a Regularized Movie + User + Genre Model. Lets see how our model improves with this implementation.

```
# Generate a Lambda plot to find our lowest RMSE

lamdas <- seq(0, 10, 0.25)

# Create a function to generate lambdas and their respective RMSE value
# Will use the "equation" again to double check the RMSE with our found lamda later

rmses <- sapply(lamdas, function(l){
  mu <- mean(edx_train$rating)

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

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

  b_g <- edx_train %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u) / (n() + l), n_g = n())

  predicted_ratings <- edx_validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    .$pred

  return(RMSE(edx_validation$rating, predicted_ratings))

})

qplot(lamdas, rmses) # Plot to see the shape of graph and estimate our lowest RMSE
```




```
min_lamda <- lamdas[which.min(rmses)] # assigns lowest RMSE so we can use it later
min_lamda
```

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

```
##### Find RMSE using Lamda Value #####

# Using the lambda value generated with the estimated lowest RMSE we can actually plug it back in

mu <- mean(edx_train$rating)

b_i <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu) / (n() + min_lamda))

b_u <- edx_train %>%
  left_join(b_i, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)  / (n() + min_lamda))

b_g <- edx_train %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u) / (n() + min_lamda), n_g = n())
```

```r
reg_predicted_ratings <- edx_validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  .$pred

reg_RMSE <- RMSE(edx_validation$rating, reg_predicted_ratings)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model = "Reg: Movie, User. and Genre Effect Model",
                                     RMSE = reg_RMSE))
rmse_results %>% knitr::kable()
```

Model	RMSE
Base Mean Model	1.0590002
Movie Effect Model	0.9426564
Movie and User Effect Model	0.8646047
Movie, User. and Genre Effect Model	0.8642542
Reg: Movie, User. and Genre Effect Model	0.8636865

The goal of the code is to create an array of lambda values and the predicted RMSE so that we can identify where our lowest RMSE value is and the associated lambda value. We can then use this lambda value later to get our RMSE value at the predicted lowest value.

Now that we see another successful lowering of the RMSE value we can do a final validation test to see if our model truly works. This will be the only time that we will use the validation data set, up to this point we have been using edx_validation to test our data.

Validation Stage:

Now that we have a created a model that we think is ready, we will finally test it using the validation dataset. Again we will utilize the lambda value we found earlier.

```r
##### Final Validation Test #####

# Using the lamda value we found we remove the function and just see what our RMSE value is
# This time we will be using our final validation set to see what our lowest RMSE is

mu <- mean(edx$rating)

b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu) / (n() + min_lamda))

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

b_g <- edx %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  group_by(genres) %>%
```

```r
  summarize(b_g = sum(rating - mu - b_i - b_u) / (n() + min_lamda), n_g = n())

# Utilizing final Validation set to see what our RMSE is

final_reg_predicted_ratings <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  .$pred

final_reg_RMSE <- RMSE(validation$rating, final_reg_predicted_ratings)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model = "Final Validation",
                                     RMSE = final_reg_RMSE))
rmse_results %>% knitr::kable()
```

Model	RMSE
Base Mean Model	1.0590002
Movie Effect Model	0.9426564
Movie and User Effect Model	0.8646047
Movie, User. and Genre Effect Model	0.8642542
Reg: Movie, User. and Genre Effect Model	0.8636865
Final Validation	0.8646798

With our final validation test done we can see that we have successfully lowered the RMSE to be below 0.8490, which is our project objective.

Conclusion:

In conclusion we were successfully able to achieve a RMSE score below 0.86490 on our final validation test of our model. To achieve this we took our original dataset and split it into the edx and validation set, then split the edx set into edx_train and edx_validation. This was done to avoid overtraining our model. With the edx_train set we designed our model, and then tested it with the edx_validation set. We started with a simple mean model and slowly created subsequent models adding more and more effects. We first started with the movie, then user, and finally genre effect. From a basic overview the movie and user effects were a lot more profound than the genre effect. Afterwards we use our regularization method on the movie, user and genre to create a regularized model. It was with this final model that we tested our validation set and saw our RMSE was below the threshold. One thing to note is our validation RMSE was higher than our regularization model using the edx_validation set. This was a prediction we made, and the reason we continued to develop our model further even when the edx_validation was meeting the course minimum of 0.8649. We knew because we were using a slightly smaller dataset compared to the edx and edx_validation, that we might see slight better results. This is why we wanted to build a model with some cushion on our RMSE results.

Overall there were some limitations that under different circumstances make a future model better. One limitation is that we were using the 10 million MovieLens Dataset, if we had the computational power using the larger 100M dataset would have been better. More importantly we might have seen the genre effect have a stronger effect than what it had on our 10M set. Talking about the genre effect what you will notice is that I actually kept the genres clustered, and did not parse them for our genre effect model. The reason behind this was my computer was having a hard time with the computation. Instead I only parsed the genres and made the edx_genres dataset to show some graphs of what the genre effect was having when split up.

We can further improve our model using more effects such as time, or using new techniques like matrix factorization. Matrix factorization is a powerful technique that is very useful in recommendation systems like this.

Overall this project gave great insight into how to use techniques learned in our class/book and build upon them to create a recommendation system to learn machine learning techniques.