HarvardX - BEER!!! - Capstone IDV Project

Chris Mann

6/17/2019

Introduction

As part of the HarvardX Data Science Certificate program, this Individual Learner- Capstone project focuses on the creation of a recommendation system using the **Kaggle - Beer Review** dataset (publicly available at Kaggle).

This datset was discussed in the very interesting Strata 2017 talk: "How to hire and test for data skills: A one-size-fits-all interview kit" Strata 2017

All files can be downloaded from GitHub.

The goal of this project is to utilize the tools and techniques covered in the course to develop a recommender system, which will be judged against the simplest "Guess the Mean" approach available.

As with the MovieLens 10M project, we will use the **Root Mean Squared Error (RMSE)** on a segregated test dataset as our benchmark.

I found the MovieLens 10M project frustrating… a 10 Million record dataset is clearly beyond the capabilities of R and Caret. Yes, I could custom compile some code to enable GPU support for some libraries, but why should I have to? Part of the reason that I took this course was to brush up on my "R." I thought that the language was getting a bad rap when compared to Python/Pandas or other Data Science programs…

Sadly, I was wrong.

For this individual project, I selected a dataset that is MUCH smaller hoping that I could just write an lapply statement and magically have R and Caret churn through 20-30 algorithims. Sadly, that was not the case. The **Beer Review** Data has approximately 1.5M records. I did a random split of 60/40 for Training and Testing in order to ensure that the Training dataset had fewer than 1M records.

- Training: 972,788 reviews
- Testing: 613,826 reviews

Essentially, R pukes on even 1M records and the **caret** package is literally useless unless you are fiddling with very small datasets. So, unfortunately, R does not appear to be ready for prime time.

Given that, what is our target? We will explore the data shortly, but the overall **rating** average is:

- Overall Mean: 3.813921
- RMSE: 0.7159839

So our goal is to see how much better we can do than an RMSE of **0.716**

I will take several approaches:

- Caret - *Useless and no Results*
- Regularized Linear Regression
- XGB – Extreme Parallel Tree Boosting (xgBoost)
- RECO - Recommender w/ Matrix Factorization (recosystem)
- Salford Predictive Modeler V8.0

```
## Loading required package: tidyverse
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## -- Attaching packages --------------------------------------------------------------------------- tid
```

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.2     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------ tidyverse
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
## Loading required package: lubridate
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
## Loading required package: knitr
```

```
## Loading required package: scales
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
## Loading required package: kableExtra
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

Kaggle - Beer Reviews Dataset

Load and Create the Train & Test Datasets

NOTE: if using R version 3.6.0, be sure to use: set.seed(1, sample.kind = "Rounding") instead of set.seed(1)

```r
# Read in the CSV file
beer <- read.csv("beer_reviews.csv")

# Drop the additional "Rating" columns and only keep the Overall rating
beer <- beer %>% select(-one_of("review_aroma", "review_appearance",
                                "review_palate", "review_taste"))
colnames(beer)
```

```
## [1] "brewery_id"         "brewery_name"       "review_time"
## [4] "review_overall"     "review_profilename" "beer_style"
## [7] "beer_name"          "beer_abv"           "beer_beerid"
```

```r
# Rename some columns to make them easier to use
colnames(beer)[3]  <- "timestamp"          # Convert "review_time" to "timestamp"
colnames(beer)[4]  <- "rating"             # Convert "review_overall" to "rating"
colnames(beer)[5]  <- "user_id"            # Convert "review_profilename" to "user_id"
colnames(beer)[9]  <- "beer_id"            # Convert "beer_beer_id" to "beer_id"

# Convert from a Reviewer Name Factor to a Numeric ID
beer$user_id <- as.numeric(beer$user_id)

# Create a numeric ID for Beer Styles
beer <- beer %>% mutate(beer_style_id = as.numeric(beer_style))

# CREATE TRAINING AND TESTING DATASETS
# The Testing Set will be 40% of the Beer Review Dataset

# NOTE: if using R version 3.6.0, use: set.seed(1, sample.kind = "Rounding") instead of set.seed(1)
# set.seed(1) # if using R 3.6.0: set.seed(1, sample.kind = "Rounding")
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(y = beer$rating, times = 1, p = 0.4, list = FALSE)
beer_train <- beer[-test_index,]
temp <- beer[test_index,]

# Make sure user_id and beer_id in the Testing set are also in Training set
beer_test <- temp %>%
  semi_join(beer_train, by = "beer_id") %>%
  semi_join(beer_train, by = "user_id")

# Add rows removed from beer_test set back into beer_train set
removed <- anti_join(temp, beer_test)
beer_train <- rbind(beer_train, removed)


#******************************************************************************
#******************************************************************************

# Create working copies of the Training (beer_train) and Test (beer_test) datasets
train_set <- beer_train
test_set <- beer_test

# Define the RMSE Evaluation Function
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Examining the Beer Data

```r
class(beer)
```

```
## [1] "data.frame"
```

```r
glimpse(beer)
```

```
## Observations: 1,586,614
## Variables: 10
## $ brewery_id    <int> 10325, 10325, 10325, 10325, 1075, 1075, 1075, 10...
## $ brewery_name  <fct> Vecchio Birraio, Vecchio Birraio, Vecchio Birrai...
## $ timestamp     <int> 1234817823, 1235915097, 1235916604, 1234725145, ...
## $ rating        <dbl> 1.5, 3.0, 3.0, 3.0, 4.0, 3.0, 3.5, 3.0, 4.0, 4.5...
## $ user_id       <dbl> 28584, 28584, 28584, 28584, 16283, 22916, 25211,...
## $ beer_style    <fct> Hefeweizen, English Strong Ale, Foreign / Export...
## $ beer_name     <fct> Sausa Weizen, Red Moon, Black Horse Black Beer, ...
## $ beer_abv      <dbl> 5.0, 6.2, 6.5, 5.0, 7.7, 4.7, 4.7, 4.7, 4.7, 4.7...
## $ beer_id       <int> 47986, 48213, 48215, 47969, 64883, 52159, 52159,...
## $ beer_style_id <dbl> 66, 52, 60, 62, 10, 67, 67, 67, 67, 67, 77, 67, ...
```

The **beer** dataset is a Data Frame with 1,586,614 observations of 10 variables:

- **rating:** User defined rating from 0 to 5 in 0.5 increments. This will be our target variable. There are several other attributes that are rated, but they were highly correlated to the Overall Rating, so those variables were excluded. Technically this is ordinal and could be used as categorical, but we will be treating it as a continuous variable since our evaluation method is RMSE
- **user_id** ID of the User giving the Rating observation
- **beer_id** ID of the Movie being Rated
- **timestamp:** POSIX datetime INT from 1 Jan 1970
- **brewery_id** ID of the Brewery
- **brewery_name** Name of the Brewery
- **beer_style** Style of Beer
- **beer_name** Name of the Beer
- **beer_abv** Beer Alcohol by Volume
- **beer_style_id** ID of the Beer Style

```r
#*****************************************************************************
# Create Additional Datasets
#*****************************************************************************

# USER RATINGS - Create dataframe with additional User rating information:
#       Average (u_avg), StDev (u_std), Number of Reviews (u_numrtg)
user_data <- train_set %>%
  group_by(user_id) %>%
  summarize(u_avg = mean(rating),
#            u_std = sd(rating),
            u_numrtg = as.numeric(n()))
```

```
# BEER RATINGS - Create dataframe with additional Beer rating information:
#       Average (b_avg), StDev (b_std), Number of Reviews (b_numrtg)
beer_data <- train_set %>%
  group_by(beer_id) %>%
  summarize(b_avg = mean(rating),
#            b_std = sd(rating),
            b_numrtg = as.numeric(n()))

# BREWERY RATINGS - Create dataframe with additional Brewery rating information:
#       Average (brw_avg), StDev (brw_std), Number of Reviews (brw_numrtg)
brewery_data <- train_set %>%
  group_by(brewery_id) %>%
  summarize(brw_avg = mean(rating),
#            brw_std = sd(rating),
            brw_numrtg = as.numeric(n()))

# BEER STYLE RATINGS - Create dataframe with additional Beer Style rating information:
#       Average (s_avg), StDev (s_std), Number of Reviews (s_numrtg)
beer_style_data <- train_set %>%
  group_by(beer_style_id) %>%
  summarize(s_avg = mean(rating),
#            s_std = sd(rating),
            s_numrtg = as.numeric(n()))

# Number of unique Users
nrow(user_data)
```

```
## [1] 33388
```

```
# Number of unique Beers
nrow(beer_data)
```

```
## [1] 66055
```

```
# Number of unique Breweries
nrow(brewery_data)
```

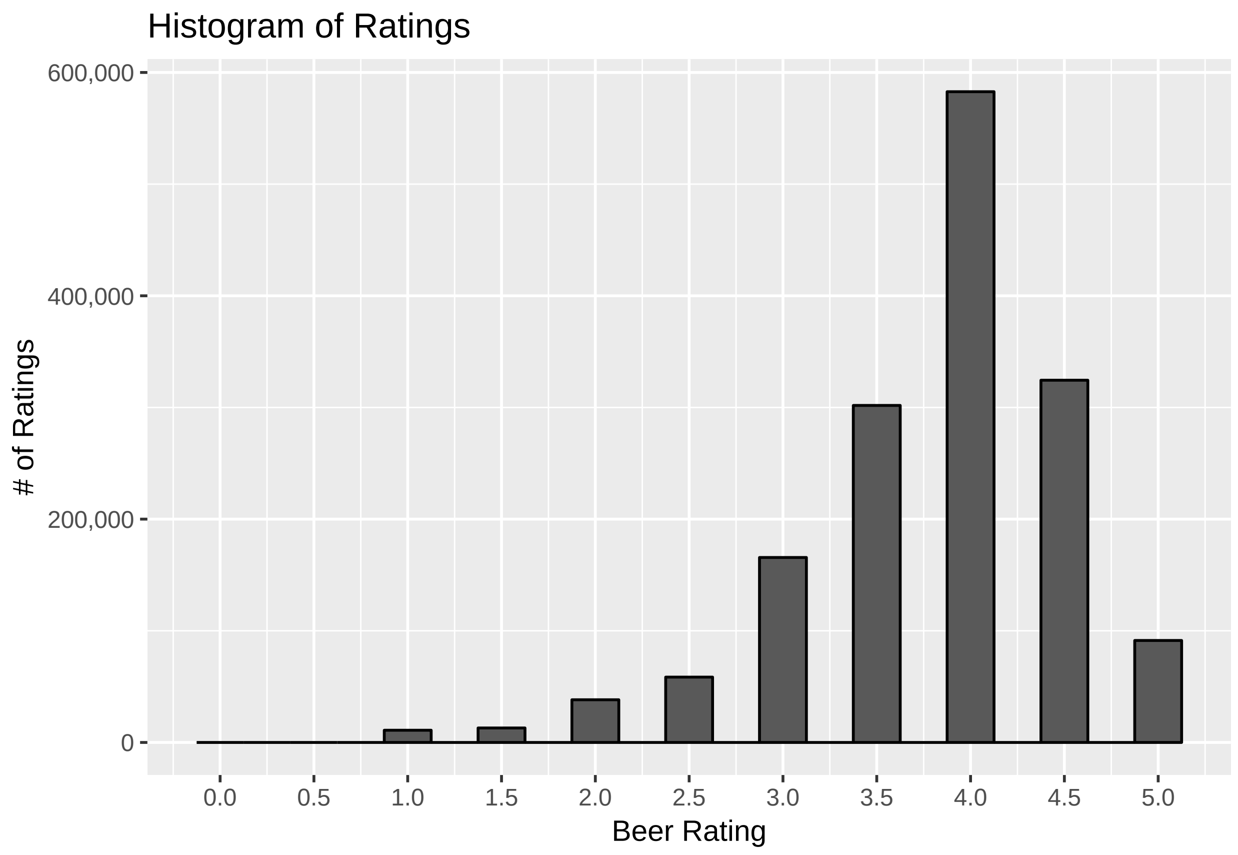
```
## [1] 5840
```

```
# Number of unique Beer Styles
nrow(beer_style_data)
```

```
## [1] 104
```

The dataset contains the ratings of 66,055 Beers by 33,388 Users. There are 5,840 Breweries represented and 104 Beer Styles.

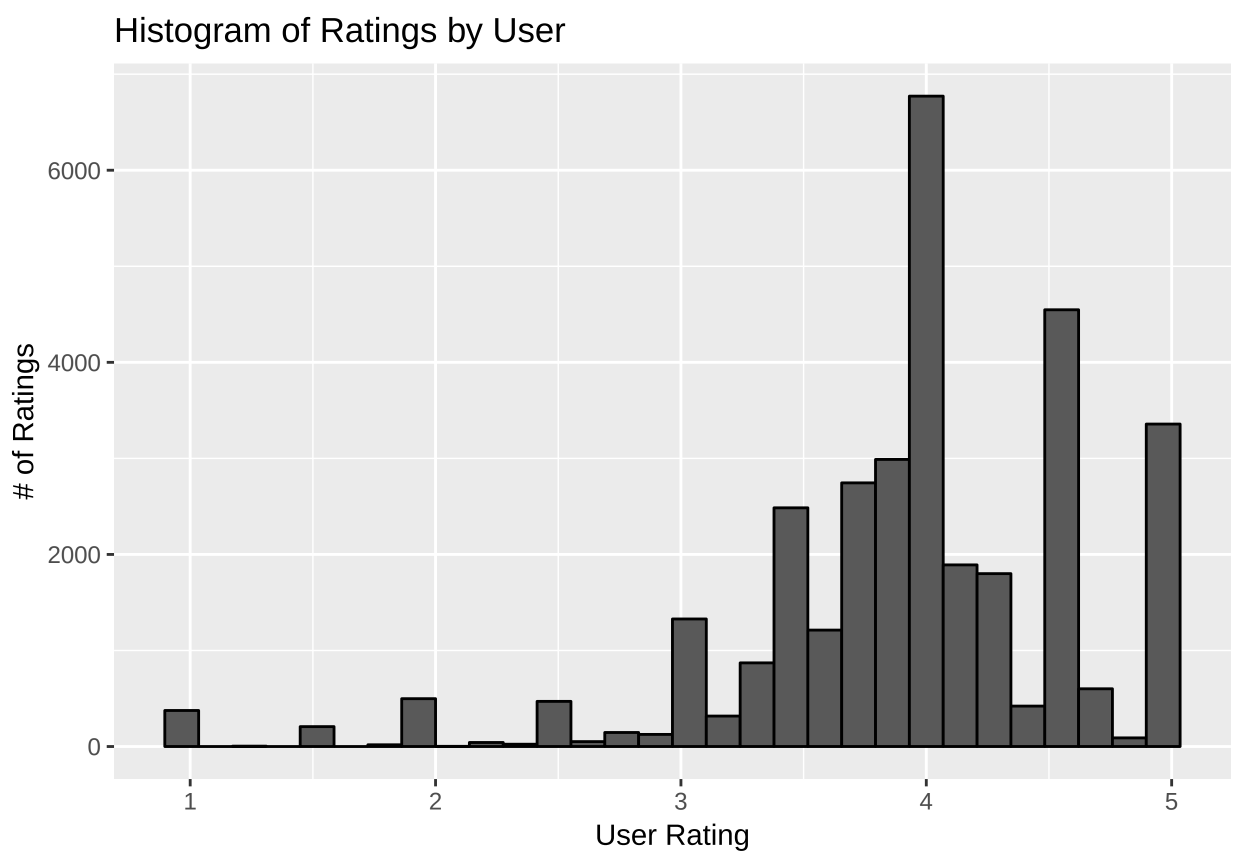
```
# Plot the distribution of Ratings in the Training set

beer %>%
  ggplot(aes(x= rating)) +
  geom_histogram(binwidth = 0.25, color = "black") +
  scale_x_continuous(breaks=seq(0, 5, 0.5)) +
  scale_y_continuous(labels=comma) +
  labs(x="Beer Rating", y="# of Ratings") +
  ggtitle("Histogram of Ratings")
```



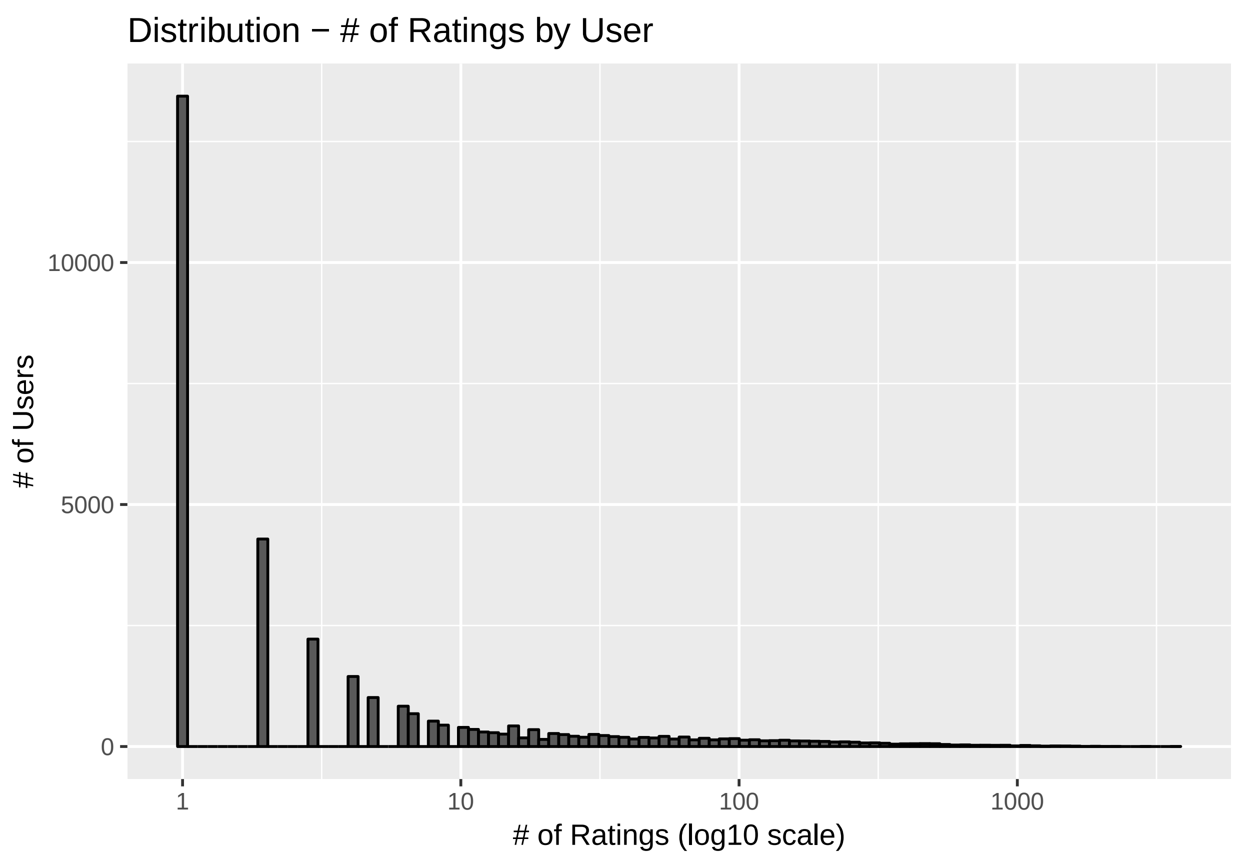
Unlike the MovieLens ratings, “half” ratings don’t seem to be an issue with people reviewing beers. It would be interesting to try and figure out why movie ratings stratify on whole numbers, but beer ratings don’t. Maybe it’s just because beer drinkers are having more fun and just don’t care. ;)

User Data - Here we can see that there are a lot of Users who have only submitted one review.

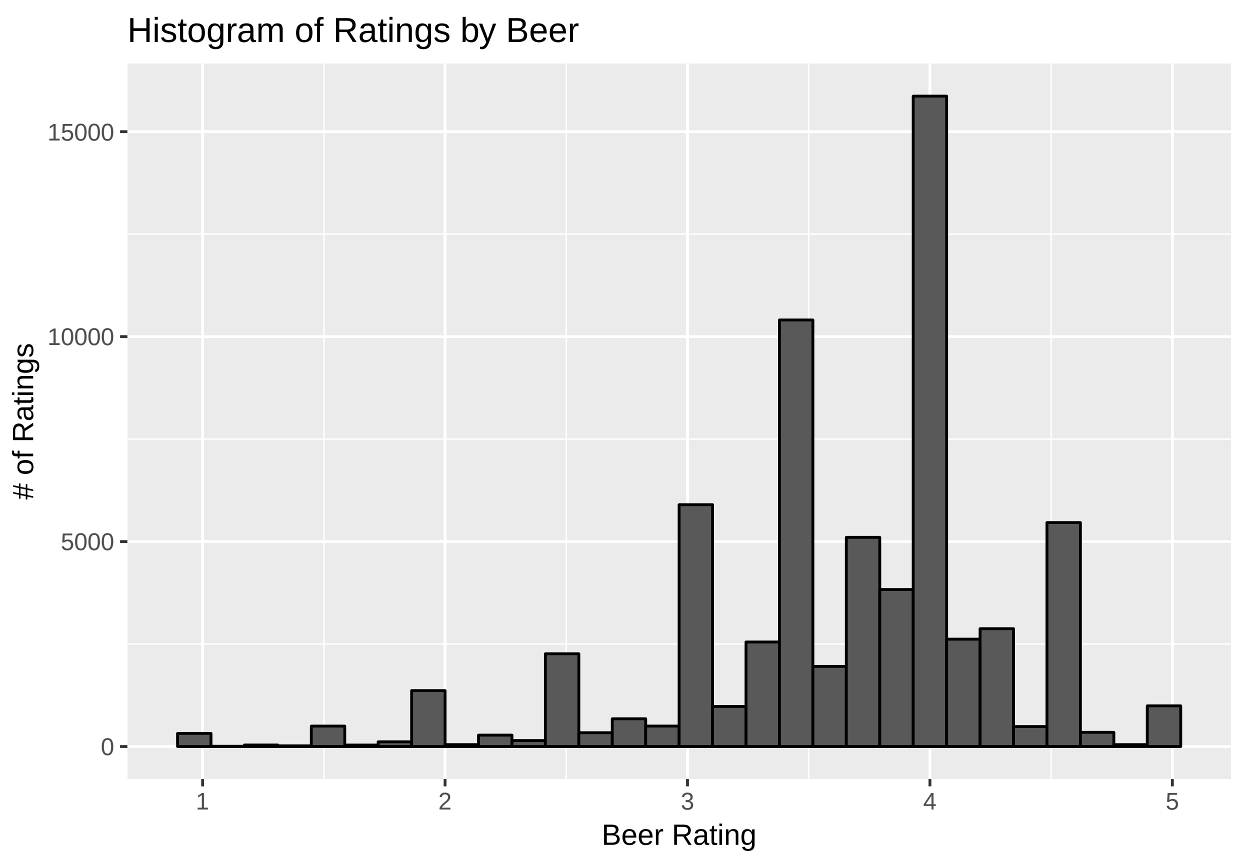
```
#******************************************************************************
# USER INFORMATION
#******************************************************************************

# Plot the distribution of Ratings by User ID in the Training set
train_set %>%
  group_by(user_id) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  labs(x="User Rating", y="# of Ratings") +
  ggtitle("Histogram of Ratings by User")
```



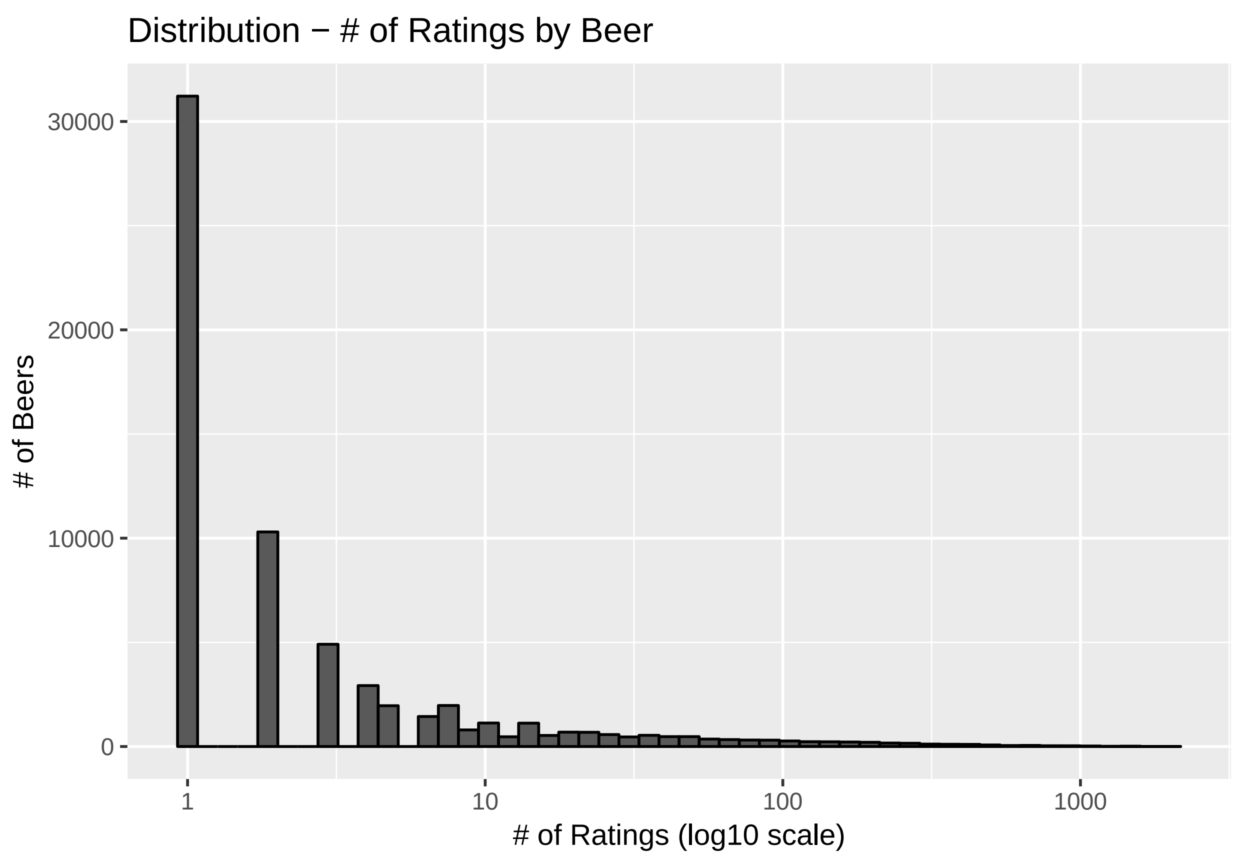
```
# Plot the log distribution of # of Ratings per User
user_data %>%
  ggplot(aes(x = u_numrtg)) +
  geom_histogram(bins = 100, color = "black") +
  scale_x_log10() +
  labs(x="# of Ratings (log10 scale)", y="# of Users") +
  ggtitle("Distribution - # of Ratings by User")
```



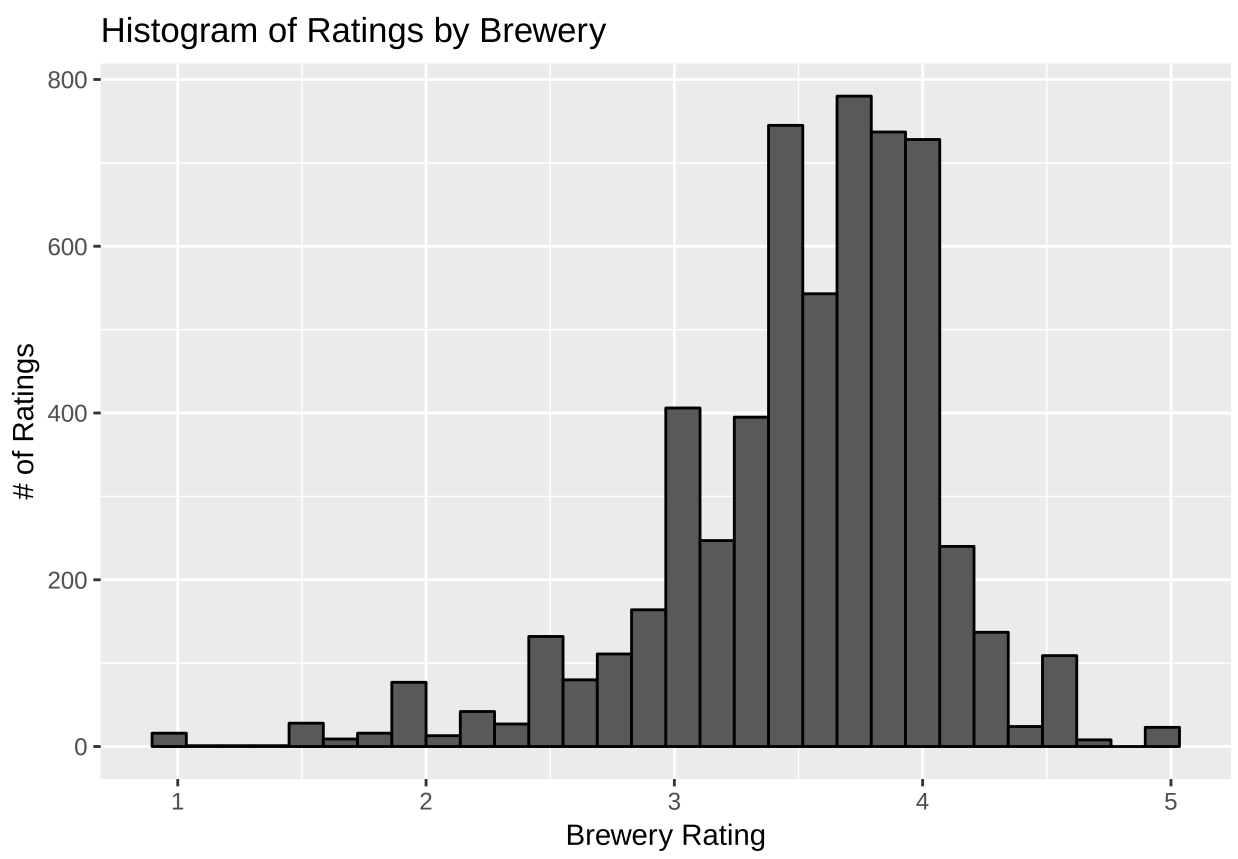
```
#****************************************************************************
# BEER INFORMATION
#****************************************************************************

# Plot the distribution of Ratings by Beer ID in the Training set
train_set %>%
  group_by(beer_id) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  labs(x="Beer Rating", y="# of Ratings") +
  ggtitle("Histogram of Ratings by Beer")
```



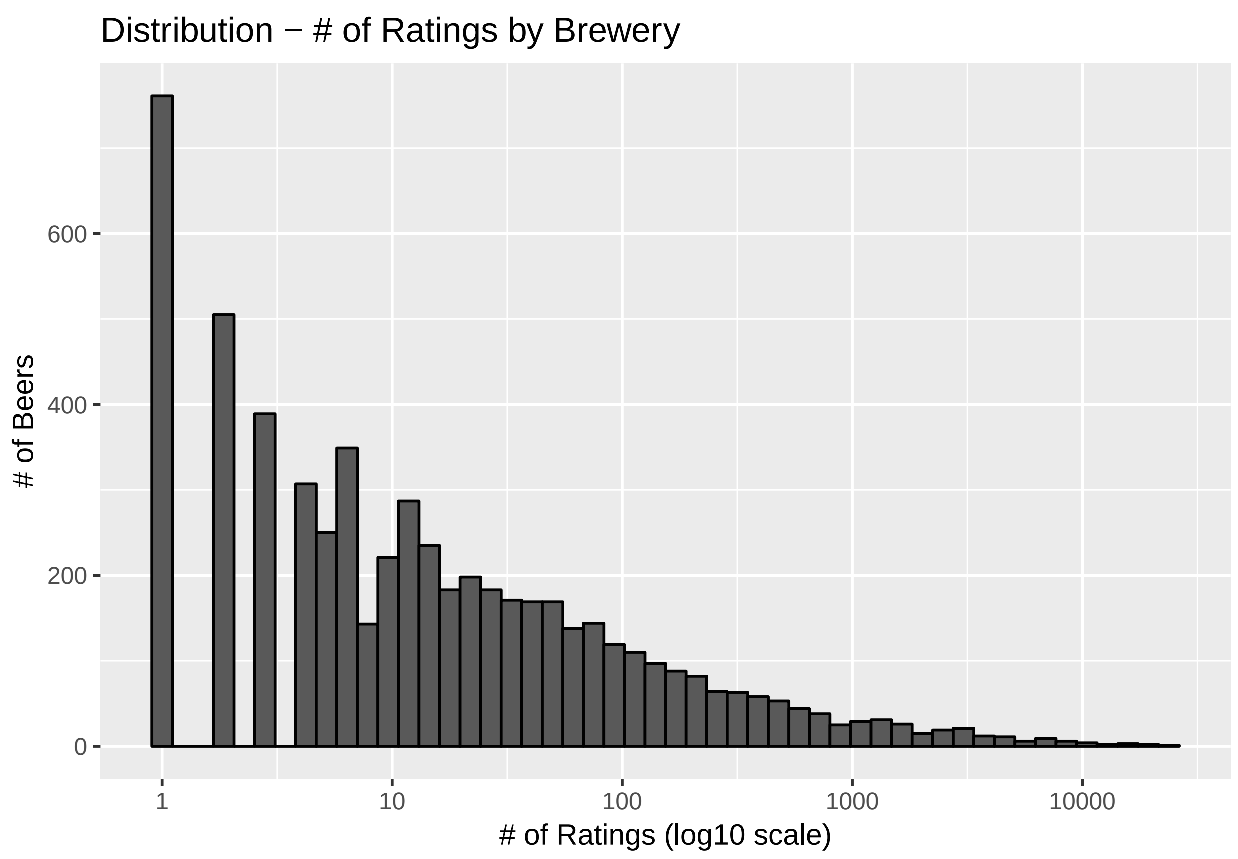
```
# Plot the log distribution of # of Ratings per Beer
beer_data %>%
  ggplot(aes(x = b_numrtg)) +
  geom_histogram(bins = 50, color = "black") +
  scale_x_log10() +
  labs(x="# of Ratings (log10 scale)", y="# of Beers") +
  ggtitle("Distribution - # of Ratings by Beer")
```



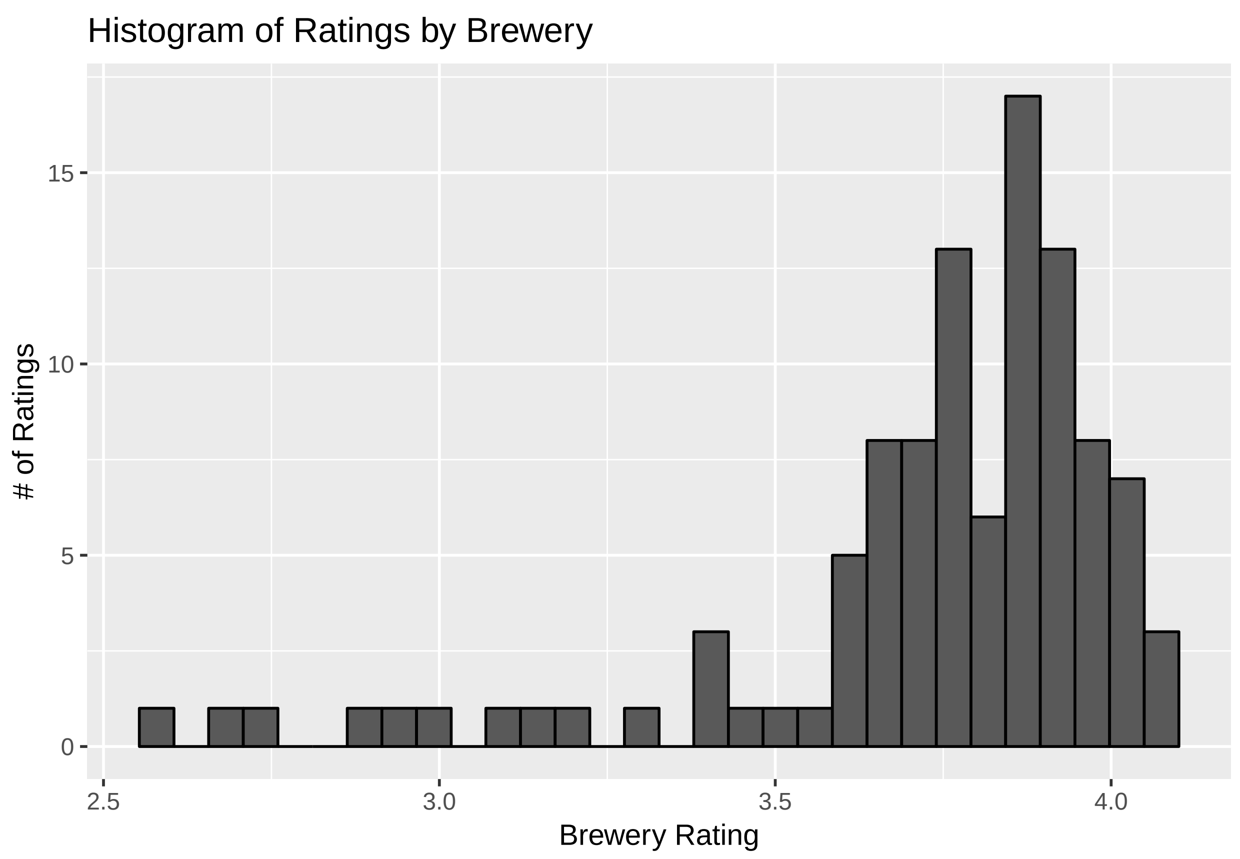
```
#*****************************************************************************
# BREWERY INFORMATION
#*****************************************************************************

# # of Breweries
# Top Breweries by Rating

# Plot the distribution of Ratings by Brewery ID in the Training set
train_set %>%
  group_by(brewery_id) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  labs(x="Brewery Rating", y="# of Ratings") +
  ggtitle("Histogram of Ratings by Brewery")
```



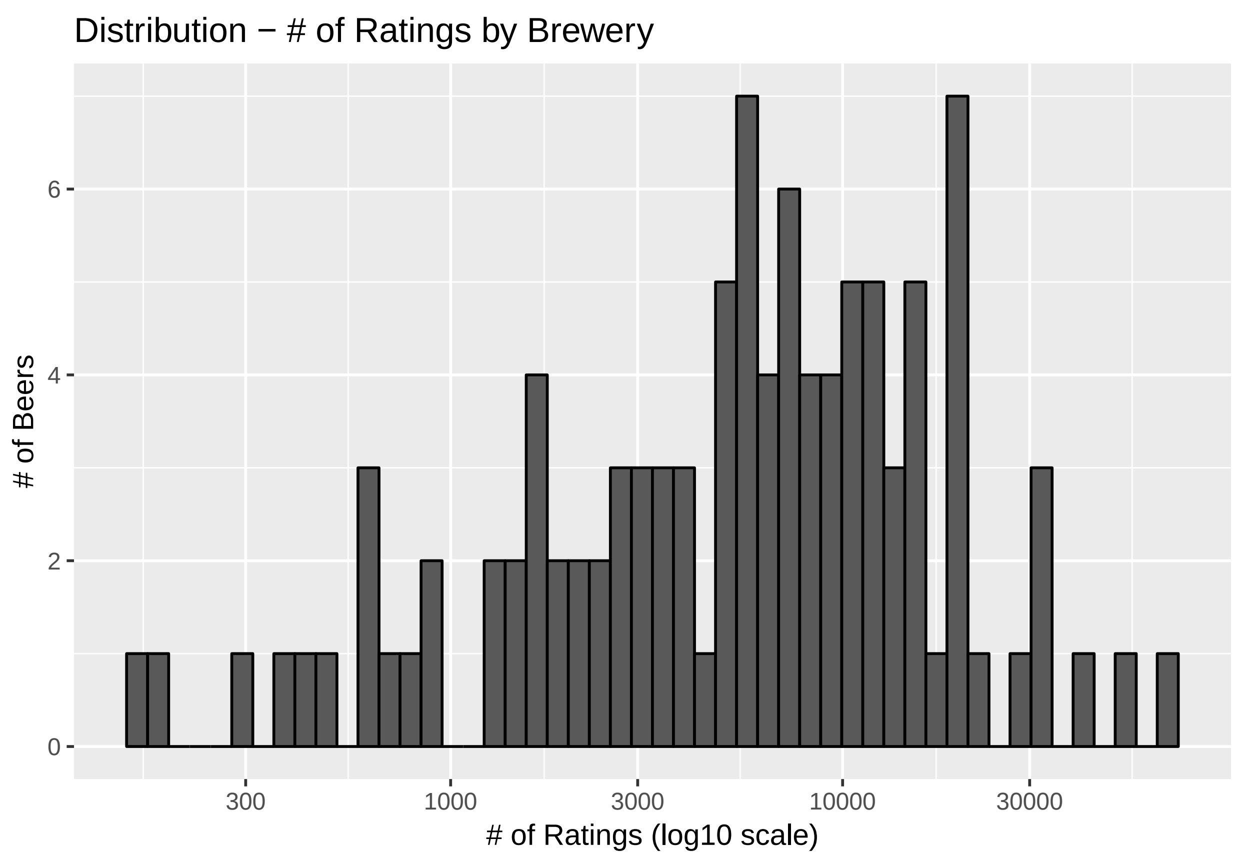
```
# Plot the log distribution of # of Ratings per Brewery
brewery_data %>%
  ggplot(aes(x = brw_numrtg)) +
  geom_histogram(bins = 50, color = "black") +
  scale_x_log10() +
  labs(x="# of Ratings (log10 scale)", y="# of Beers") +
  ggtitle("Distribution - # of Ratings by Brewery")
```



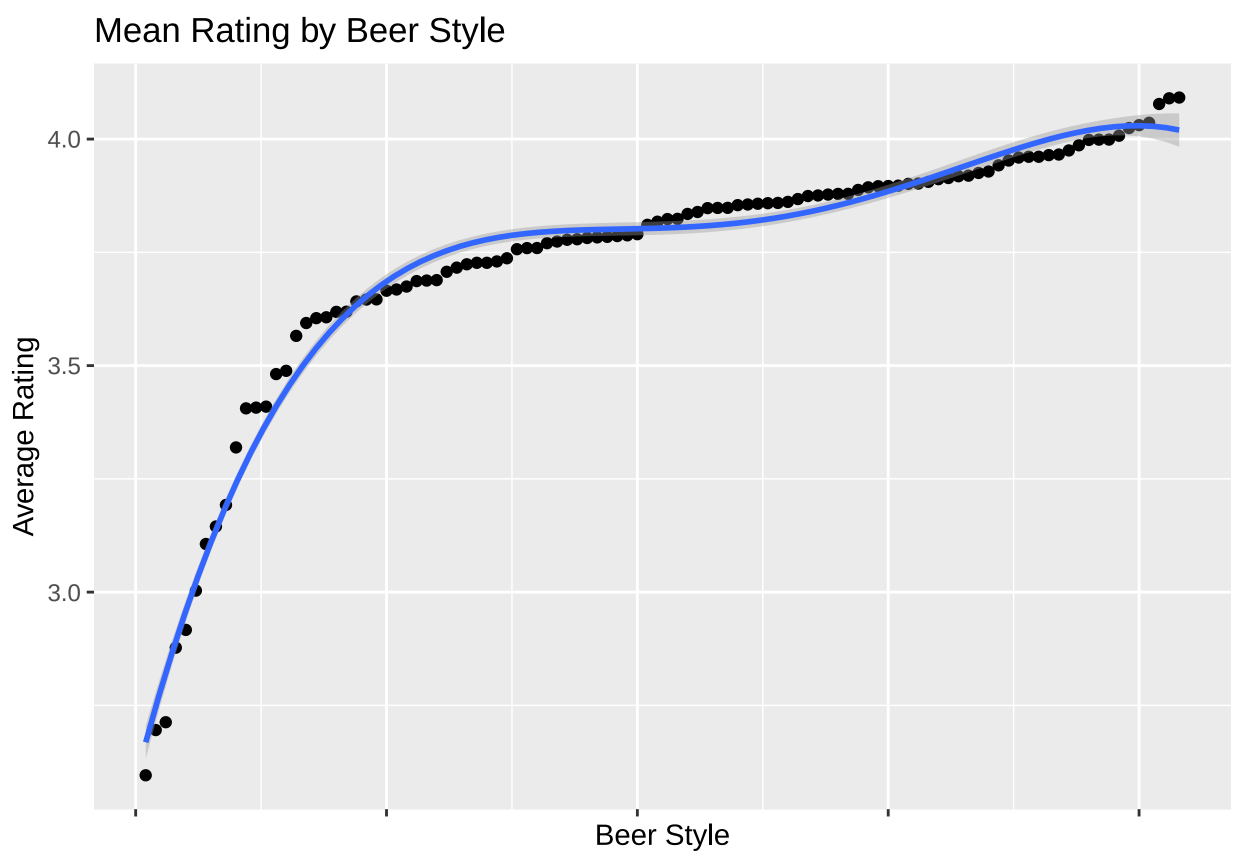
```
#******************************************************************************
# BEER STYLE INFORMATION
#******************************************************************************

# Plot the distribution of Ratings by Beer Style in the Training set
train_set %>%
  group_by(beer_style_id) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  labs(x="Brewery Rating", y="# of Ratings") +
  ggtitle("Histogram of Ratings by Brewery")
```



```r
# Plot the log distribution of # of Ratings per Beer Style
beer_style_data %>%
  ggplot(aes(x = s_numrtg)) +
  geom_histogram(bins = 50, color = "black") +
  scale_x_log10() +
  labs(x="# of Ratings (log10 scale)", y="# of Beers") +
  ggtitle("Distribution - # of Ratings by Brewery")
```



```r
# Display graph of Average Rating by all Beer Styles
train_set %>%
  group_by(beer_style_id) %>%
  summarize(avg = mean(rating)) %>%
  ggplot(aes(x = as.numeric(reorder(beer_style_id, avg)), y = avg)) +
  geom_point() +
  geom_smooth( aes(x = as.numeric(reorder(beer_style_id, avg)), y = avg),
               method = 'lm', formula = y ~ poly(x, 4), se = TRUE) +
  theme(axis.text.x=element_blank()) +
  labs(x="Beer Style", y="Average Rating") +
  ggtitle("Mean Rating by Beer Style")
```



Recommender System Methodology

As mentioned in the Introduction, this project will exapand on the previous MovieLens project and produce models from four different recommendation systems.

- Regularized Linear Regression
- XGB – Extreme Parallel Tree Boosting (xgBoost)
- RECO - Recommender w/ Matrix Factorization (recosystem)
- Salford Predictive Modeler v8.0

Standard & Regularized Linear Regression

As the first approach, we will build and test a Regularized Linear Regression model. As previously seen, Regularization improves the results in all models, so we will skip doing Standard Linear Regression and only build Regularized Linear Regression models. This will be done by successively adding mean distributions based upon each of the following variables in the dataset:

- Rating (mu) - Starting with an overall average of all ratings (mu)
- Beer (b_i) - Adding the Average rating of each individual Movie
- User (b_u) - Adding the Average rating of each individual User
- Style (s_u) - Adding the average for each unique Beer Style
- Brewery (brw_u) - Adding the average of each rating by Brewery

Linear Regression Model Creation

Simple Overall Average

Create and Display the RMSE results of using the overall Average (mu) of all ratings:

```
#*****************************************************************************
# We're only using this to get the absolute WORST RMSE that we're trying to beat
#*****************************************************************************
# Linear Regression - Using just the overall average
#   Yu,i = mu
#*****************************************************************************

# Calculate the overall average rating
mu <- mean(train_set$rating)
mu
```

```
## [1] 3.813921
```

```
# Evaluate the performance of simply guessing the overall average
rmse <- RMSE(test_set$rating, mu)

# Save the RMSE result to display later
LR_rmse_results <- tibble(Method = "LR: Base Mean Model",
                          RMSE = rmse)
kable(LR_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE
LR: Base Mean Model	0.7159839

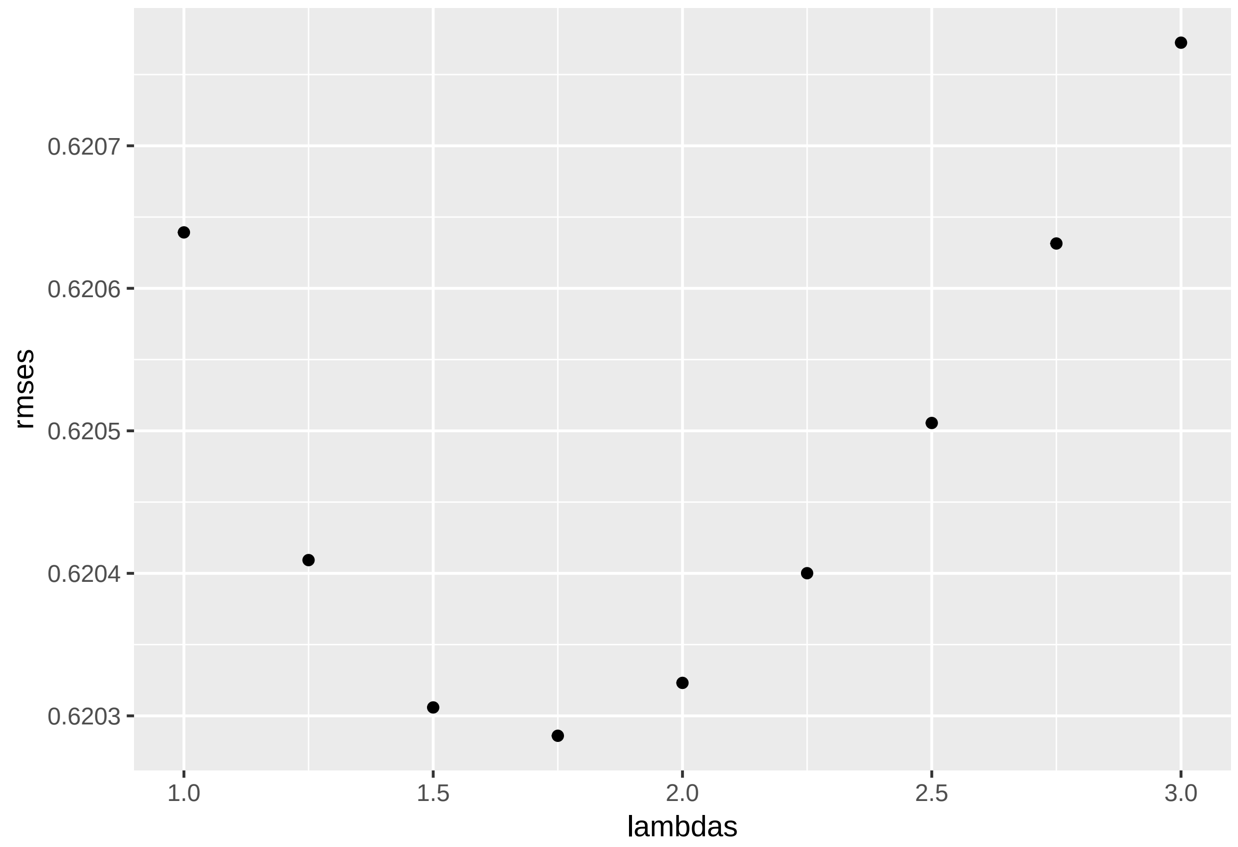
Using just the overall mean (mu), we do surprisingly well, but the goal now is to do better than **0.7159839**

Beer Effect

To enhance the LR model, we will take a look at the rating for each Beer. Plotting the distribution of Beer ratings (below) shows that ratings are not uniformly distributed and that the average of each beer should add predictive capabilities.

We will add a term to include the average Movie rating to the general simple mean, thereby adding the "Beer Effect:"

```
#*****************************************************************************
#*****************************************************************************
#***********    Regularized Linear Regression - Model Building    **************
#*****************************************************************************
#*****************************************************************************

# Since we've previously seen that Regularized Linear Regression almost always
# outperformes Linear Regression, we are only going to use Penalized Least Squares

#*****************************************************************************
# Regularized Linear Regression - Base Average Model + Beer Effect:
#   Yu,i = mu + Beer_reg_avgs$b_i
#*****************************************************************************

# Find the best RMSE across range of Lambdas
lambdas <- seq(1, 3, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(beer_id) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "beer_id") %>%
    mutate(pred = mu + b_i) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_set$rating))
})
qplot(lambdas, rmses)
```



```
# Lambda resulting in best fit / lowest RMSE
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 1.75
```

```
LR_rmse_results <- bind_rows(LR_rmse_results,
                              tibble(Method="Reg LR: Mean + Beer Effect Model",
                                     RMSE = min(rmses)))
kable(LR_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

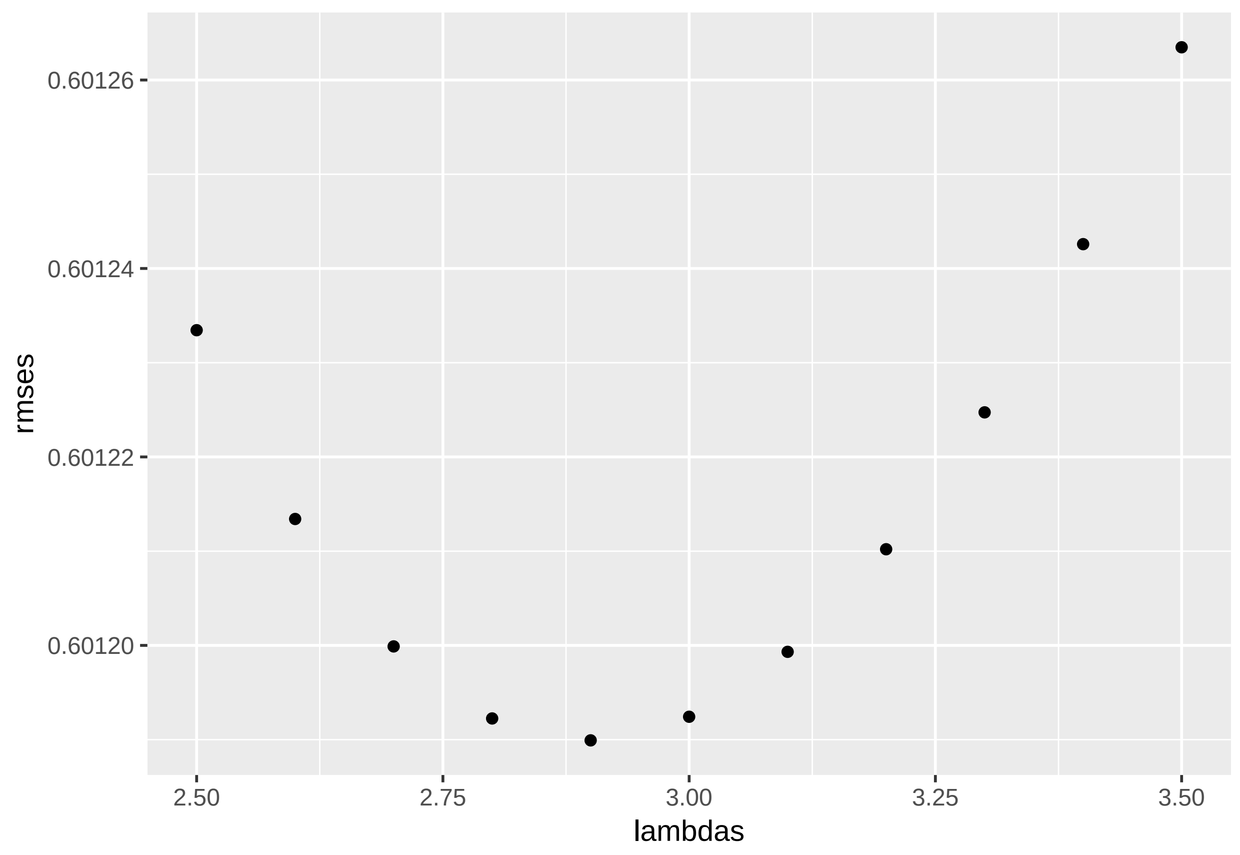
Method	RMSE
LR: Base Mean Model	0.7159839
Reg LR: Mean + Beer Effect Model	0.6202860

Adding the Beer average shows improvement, but we have to be able to do better.

User Effect

Next, we will add the "User Effect" by taking the average rating of each user and combining it with the current Mean + Beer model:

```
#******************************************************************************
# Regularized Linear Regression - Mean + Beer Effect + User Effect:
#   Yu,i = mu + b_i + b_u
#******************************************************************************

# Find the best RMSE across range of Lambdas
lambdas <- seq(2.5, 3.5, 0.1)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(beer_id) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i, by="beer_id") %>%
    group_by(user_id) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "beer_id") %>%
    left_join(b_u, by = "user_id") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_set$rating))
})
qplot(lambdas, rmses)
```



```
# Lambda resulting in best fit / lowest RMSE
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 2.9
```

```
LR_rmse_results <- bind_rows(LR_rmse_results,
                             tibble(Method="Reg LR: Mean + Beer + User Effect Model",
                                    RMSE = min(rmses)))
kable(LR_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE
LR: Base Mean Model	0.7159839
Reg LR: Mean + Beer Effect Model	0.6202860
Reg LR: Mean + Beer + User Effect Model	0.6011899

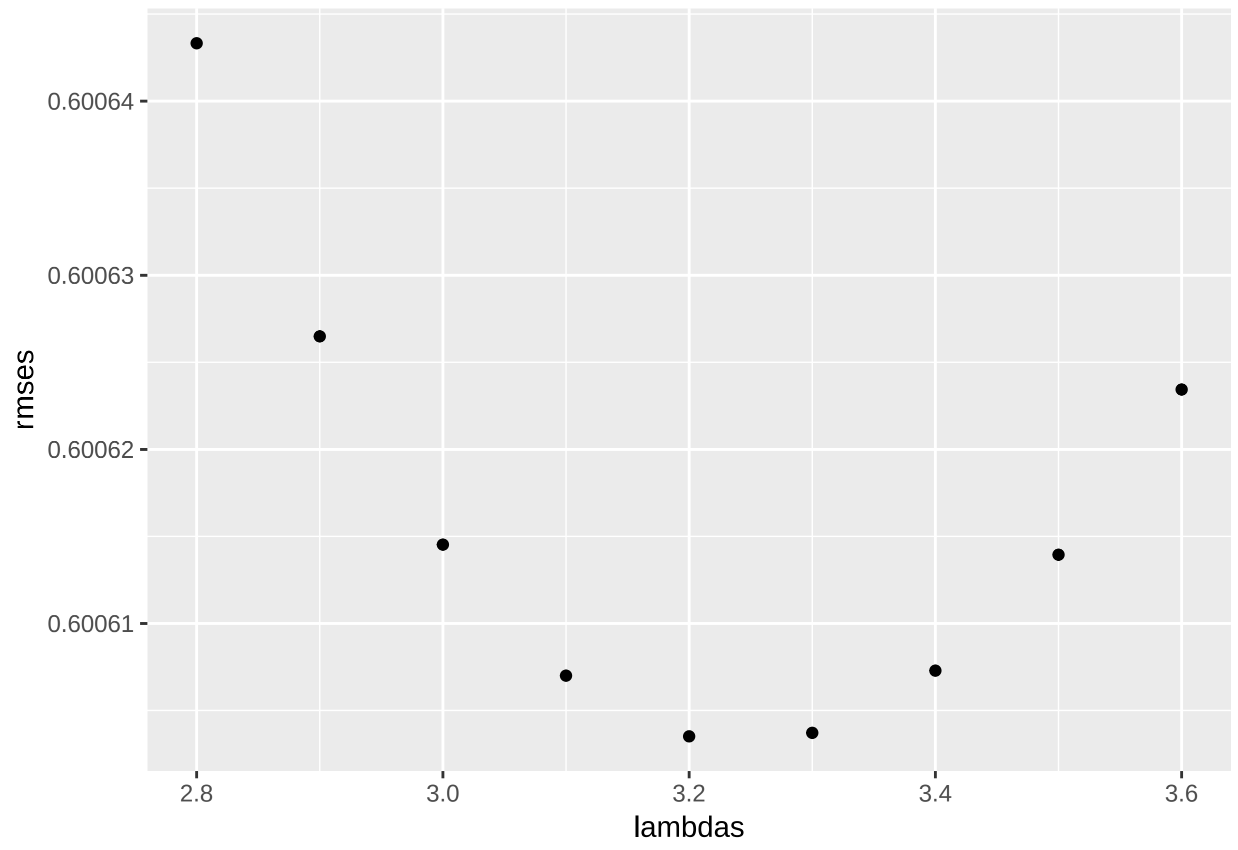
Beer Style Effect

As prviously seen, the beer styles get appreciably different ratings.

Adding in the Style Effect:

```
#********************************************************************************
# Regularized Linear Regression - Mean + Beer + User + Style Effect:
#   Yu,i = mu + b_i + b_u + s_u
#********************************************************************************

# Find best Lambda to minimize RMSE
```

```r
lambdas <- seq(2.8, 3.6, 0.1)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(beer_id) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i, by="beer_id") %>%
    group_by(user_id) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  s_u <- train_set %>%
    left_join(b_i, by="beer_id") %>%
    left_join(b_u, by="user_id") %>%
    group_by(beer_style_id) %>%
    summarize(s_u = sum(rating - b_i - b_u - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "beer_id") %>%
    left_join(b_u, by = "user_id") %>%
    left_join(s_u, by = "beer_style_id") %>%
    mutate(pred = mu + b_i + b_u + s_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_set$rating))
})
qplot(lambdas, rmses)
```



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

```
## [1] 3.2
```

```r
LR_rmse_results <- bind_rows(LR_rmse_results,
                             tibble(Method="Reg LR: Mean + Beer + User + Style Effect Model",
                                    RMSE = min(rmses)))
kable(LR_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

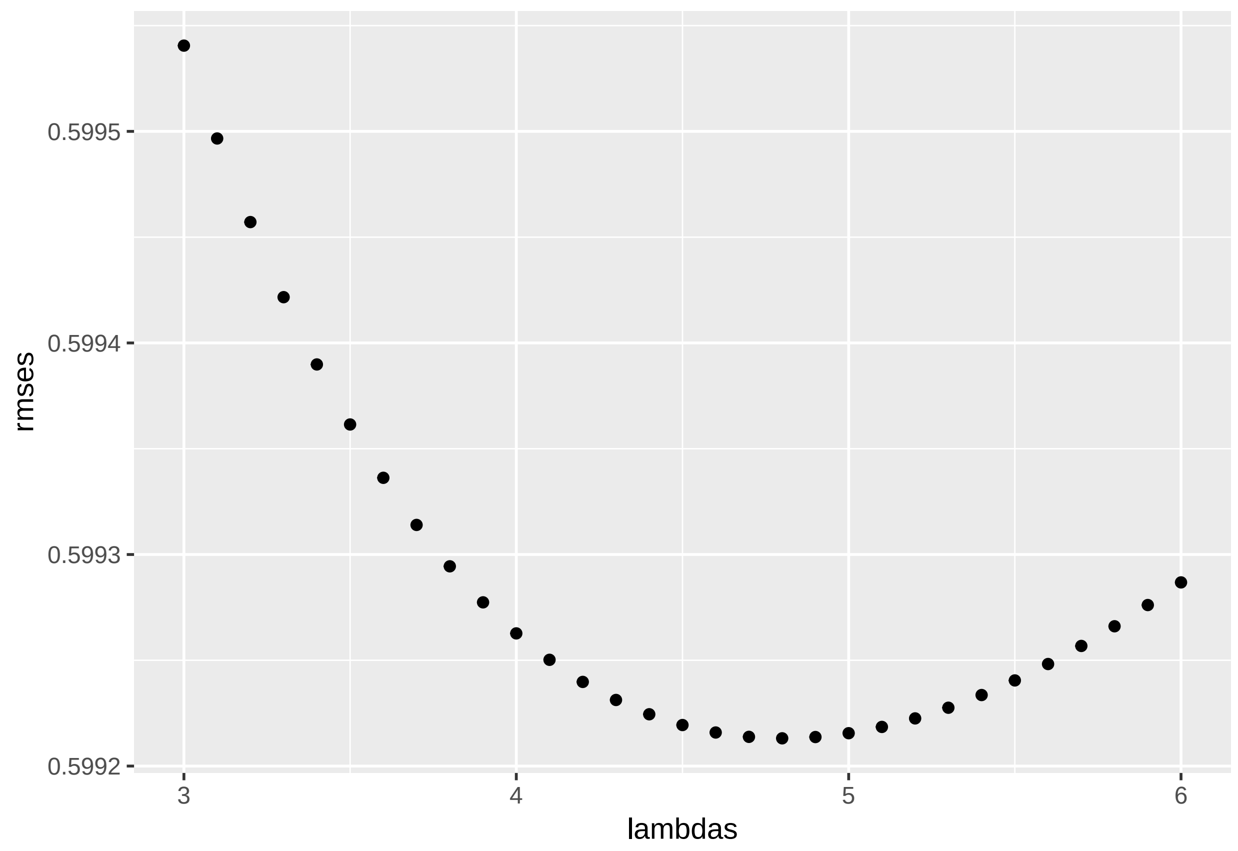
Method	RMSE
LR: Base Mean Model	0.7159839
Reg LR: Mean + Beer Effect Model	0.6202860
Reg LR: Mean + Beer + User Effect Model	0.6011899
Reg LR: Mean + Beer + User + Style Effect Model	0.6006035

Adding the Style Effect further improves the RMSE, but not by much.

Brewery Effect

Finally, we will add in the effect for each Brewery:

```r
#******************************************************************************
# Regularized Linear Regression - Mean + Beer + User + Style + Brewery Effect:
#   Yu,i = mu + b_i + b_u + s_u + brw_u
#******************************************************************************

# Find best Lambda to minimize RMSE
lambdas <- seq(3, 6, 0.1)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(beer_id) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i, by="beer_id") %>%
    group_by(user_id) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  s_u <- train_set %>%
    left_join(b_i, by="beer_id") %>%
    left_join(b_u, by="user_id") %>%
    group_by(beer_style_id) %>%
    summarize(s_u = sum(rating - b_i - b_u - mu)/(n()+l))
  brw_u <- train_set %>%
    left_join(b_i, by="beer_id") %>%
    left_join(b_u, by="user_id") %>%
    left_join(s_u, by="beer_style_id") %>%
    group_by(brewery_id) %>%
    summarize(brw_u = sum(rating - b_i - b_u -s_u - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "beer_id") %>%
    left_join(b_u, by = "user_id") %>%
    left_join(s_u, by = "beer_style_id") %>%
    left_join(brw_u, by = "brewery_id") %>%
    mutate(pred = mu + b_i + b_u + s_u + brw_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_set$rating))
})
qplot(lambdas, rmses)
```



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

```
## [1] 4.8
```

```r
min(rmses)
```

```
## [1] 0.5992131
```

```r
LR_rmse_results <- bind_rows(LR_rmse_results,
                             tibble(Method="Reg LR: Mean + Beer + User + Style + Brewery Effect Model",
                                    RMSE = min(rmses)))
kable(LR_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE
LR: Base Mean Model	0.7159839
Reg LR: Mean + Beer Effect Model	0.6202860
Reg LR: Mean + Beer + User Effect Model	0.6011899
Reg LR: Mean + Beer + User + Style Effect Model	0.6006035
Reg LR: Mean + Beer + User + Style + Brewery Effect Model	0.5992131

Adding the final Brewery Effect further lowers the RMSE value.

As seen from the above results, Regularizing using Penalized Least Squares improves every version of the model and gives the final "Best RMSE" of **0.5992131**

```
# Clean up Environment
rm(lambda, lambdas, rmses, mu, beer)
```

XGB – Extreme Parallel Tree Boosting (xgBoost)

Since its introduction in 2015, xgBoost, along with Deep Neural Networks, have largely dominated the winning solutions in Kaggle competitions.

xgBoost (Extreme Gradient Boosting) is an R library optimized for tree boosting. Its utilization of multi-threading and regularization deliver very fast (when compared to previous Random Forest methods) and accurate predictions. xgBoost is similar to other gradient boosting frameworks but focused on parallel computation on a single machine along with efficient memory usage.

xgBoost is also very flexible, implementing both linear models and tree learning algorithms. It also supports many different objective functions, including regression, classification and ranking.

And unlike the poor results on the MovieLens data.. xgb proves its worth on the Beer data!

xgBoost Data Preparation

xgBoost only works with numeric vectors.

Create Training and Test Datasets

```
#*****************************************************************************
#*****************************************************************************
#***********    XGB - Extreme Parallel Tree Boosting (xgBoost)    **************
#*****************************************************************************
#*****************************************************************************

# Load required libraries
if (!require(xgboost)) install.packages('xgboost')
library(xgboost)

if (!require(data.table)) install.packages('data.table')
library(data.table)

# Create new Training and Test datasets
# xgBoost requires that all values to be numeric

train_set <- beer_train %>% select(-one_of("brewery_name", "timestamp", "beer_style",
                                           "beer_name"))
test_set  <- beer_test  %>% select(-one_of("brewery_name", "timestamp", "beer_style",
                                           "beer_name"))
```

Additional Beer & User Derived Variables

```
# Merge User, Beer, Brewery, and Beer Style derived fields into the Training and Test datasets
train_set <- train_set %>% left_join(user_data,       by = "user_id")
train_set <- train_set %>% left_join(beer_data,       by = "beer_id")
train_set <- train_set %>% left_join(brewery_data,    by = "brewery_id")
train_set <- train_set %>% left_join(beer_style_data, by = "beer_style_id")
```

```
test_set <- test_set %>% left_join(user_data,       by = "user_id")
test_set <- test_set %>% left_join(beer_data,       by = "beer_id")
test_set <- test_set %>% left_join(brewery_data,    by = "brewery_id")
test_set <- test_set %>% left_join(beer_style_data, by = "beer_style_id")
```

Now that the Training & Test datasets have all of the additional derived variables included, the sets must be split to provide the data and the target "label" data tables. xgBoost uses its own Data Matrix format called xgb.DMatrix:

```
# xgBoost requires that the target variable, referred to as the "label" (rating) be in a separate data
# Here we create a Label and Data for the Training and Testing sets
# We are also dropping the Genres variable since it isn't numeric and no longer needed due to the One-H

train_data  <- train_set %>% select(-one_of("rating")) %>% as.data.table()
train_label <- train_set %>% select("rating")          %>% as.data.table()
test_data   <- test_set  %>% select(-one_of("rating")) %>% as.data.table()
test_label  <- test_set  %>% select("rating")          %>% as.data.table()

# Create the xgBoost Training & Testing matricies
train_matrix = xgb.DMatrix(as.matrix(train_data), label=as.matrix(train_label))
test_matrix = xgb.DMatrix(as.matrix(test_data), label=as.matrix(test_label))
```

xgBoost - XGB Model Creation

xgBoost allows the creation of both Linear, Tree, and Mixed Linear & Tree based models. xgBoost can also create Classification trees (binary and multi-class) by changing the objective function. While xgBoost is very fast when compared to other methods, the training times can be excessive.

Because the training times are acceptable for this relatively small dataset, we will be creating four small models:

- XGB Linear 50 Boost Rnds
- XGB Linear 1000 Boost Rnds
- XGB Mixed Tree 5 Boost Rnds

```
#*************************************************************************************************
# Model: XGB Linear 50 Boost Rnds
#*************************************************************************************************
# Set paramaters for most basic Linear tree
# These settings should train in less than a minute with 50 Boosting Rounds
xgb_params <- list(booster = "gblinear",      # Linear boosting alg
                   objective = "reg:linear",  # Linear regression (default)
                   eval_metric = "rmse",      # rmse as objective function
                   verbosity = 3,             # Highest verbosr level - shows all debug info
                   silent = 0)                # Not silent

# Train the XGB tree using the currently set xgb_params
start_time <- Sys.time()                     # Record start time

set.seed(1, sample.kind = "Rounding")
xgb_model <- xgboost(params = xgb_params,
                     data = train_matrix,
                     nrounds = 50,           # Maximum number of Boosting Rounds
                     nthread = 1,            # Must be set to 1 to get reproducible results
                     verbose = 2)            # Display pogress during Training
```

```
## [1]  train-rmse:0.868872
## [2]  train-rmse:0.799965
## [3]  train-rmse:0.767774
## [4]  train-rmse:0.748693
## [5]  train-rmse:0.735231
## [6]  train-rmse:0.724833
## [7]  train-rmse:0.716421
## [8]  train-rmse:0.709427
## [9]  train-rmse:0.703491
## [10] train-rmse:0.698385
## [11] train-rmse:0.693915
## [12] train-rmse:0.689967
## [13] train-rmse:0.686451
## [14] train-rmse:0.683280
## [15] train-rmse:0.680412
## [16] train-rmse:0.677784
## [17] train-rmse:0.675379
## [18] train-rmse:0.673172
## [19] train-rmse:0.671143
## [20] train-rmse:0.669249
## [21] train-rmse:0.667502
## [22] train-rmse:0.665871
## [23] train-rmse:0.664342
## [24] train-rmse:0.662928
## [25] train-rmse:0.661607
## [26] train-rmse:0.660363
## [27] train-rmse:0.659199
## [28] train-rmse:0.658099
## [29] train-rmse:0.657063
## [30] train-rmse:0.656093
## [31] train-rmse:0.655165
## [32] train-rmse:0.654293
## [33] train-rmse:0.653455
## [34] train-rmse:0.652669
## [35] train-rmse:0.651907
## [36] train-rmse:0.651190
## [37] train-rmse:0.650499
## [38] train-rmse:0.649836
## [39] train-rmse:0.649200
## [40] train-rmse:0.648591
## [41] train-rmse:0.647992
## [42] train-rmse:0.647425
## [43] train-rmse:0.646862
## [44] train-rmse:0.646321
## [45] train-rmse:0.645796
## [46] train-rmse:0.645286
## [47] train-rmse:0.644789
## [48] train-rmse:0.644307
## [49] train-rmse:0.643830
## [50] train-rmse:0.643361
```

```r
finish_time <- Sys.time()                   # Record finish time
finish_time - start_time                    # Display total training time
```

```
## Time difference of 4.52239 secs
```

```
# Use the trained model to predict the Test dataset
test_pred <- as.data.frame(predict(xgb_model , newdata = test_matrix))

# Calculate the RMSE of the Predictions
rmse <- RMSE(test_label$rating, test_pred$`predict(xgb_model, newdata = test_matrix)`)

XGB_rmse_results <- tibble(Method="XGB Linear 50 Boost Rnds",
                           RMSE = rmse,
                           Train_Time = paste(as.character(round(as.numeric(finish_time - start_time, u
XGB_rmse_results %>% knitr::kable()
```

Method	RMSE	Train_Time
XGB Linear 50 Boost Rnds	0.6547372	4.52 secs

```
#*******************************************************************************************
# Model: XGB Linear 1000 Boost Rnds
#*******************************************************************************************

xgb_params <- list(booster = "gblinear",      # Linear boosting alg
                   objective = "reg:linear",  # Linear regression (default)
                   eval_metric = "rmse",      # rmse as objective function
                   verbosity = 3,             # Highest verbosr level - shows all debug info
                   silent = 0)                # Not silent

# Train the XGB tree using the currently set xgb_params
start_time <- Sys.time()                     # Record start time

set.seed(1, sample.kind = "Rounding")
xgb_model <- xgboost(params = xgb_params,
                     data = train_matrix,
                     nrounds = 1000,         # Maximum number of Boosting Rounds
                     nthread = 1,            # Must be set to 1 to get reproducible results
                     verbose = 2)            # Display pogress during Training
```

```
## [1]  train-rmse:0.868872
## [2]  train-rmse:0.799965
## [3]  train-rmse:0.767774
## [4]  train-rmse:0.748693
## [5]  train-rmse:0.735231
## [6]  train-rmse:0.724833
## [7]  train-rmse:0.716421
## [8]  train-rmse:0.709427
## [9]  train-rmse:0.703491
## [10] train-rmse:0.698385
## [11] train-rmse:0.693915
## [12] train-rmse:0.689967
## [13] train-rmse:0.686451
## [14] train-rmse:0.683280
## [15] train-rmse:0.680412
## [16] train-rmse:0.677784
## [17] train-rmse:0.675379
## [18] train-rmse:0.673172
## [19] train-rmse:0.671143
## [20] train-rmse:0.669249
## [21] train-rmse:0.667502
## [22] train-rmse:0.665871
## [23] train-rmse:0.664342
## [24] train-rmse:0.662928
## [25] train-rmse:0.661607
## [26] train-rmse:0.660363
## [27] train-rmse:0.659199
## [28] train-rmse:0.658099
## [29] train-rmse:0.657063
## [30] train-rmse:0.656093
## [31] train-rmse:0.655165
## [32] train-rmse:0.654293
## [33] train-rmse:0.653455
## [34] train-rmse:0.652669
## [35] train-rmse:0.651907
## [36] train-rmse:0.651190
## [37] train-rmse:0.650499
## [38] train-rmse:0.649836
## [39] train-rmse:0.649200
## [40] train-rmse:0.648591
## [41] train-rmse:0.647992
## [42] train-rmse:0.647425
## [43] train-rmse:0.646862
## [44] train-rmse:0.646321
## [45] train-rmse:0.645796
## [46] train-rmse:0.645286
## [47] train-rmse:0.644789
## [48] train-rmse:0.644307
## [49] train-rmse:0.643830
## [50] train-rmse:0.643361
## [51] train-rmse:0.642906
## [52] train-rmse:0.642462
## [53] train-rmse:0.642020
## [54] train-rmse:0.641587
## [55] train-rmse:0.641166
## [56] train-rmse:0.640739
## [57] train-rmse:0.640328
## [58] train-rmse:0.639919
## [59] train-rmse:0.639515
## [60] train-rmse:0.639118
## [61] train-rmse:0.638727
## [62] train-rmse:0.638341
## [63] train-rmse:0.637959
## [64] train-rmse:0.637577
## [65] train-rmse:0.637200
## [66] train-rmse:0.636830
## [67] train-rmse:0.636461
## [68] train-rmse:0.636099
## [69] train-rmse:0.635739
## [70] train-rmse:0.635378
## [71] train-rmse:0.635024
## [72] train-rmse:0.634668
## [73] train-rmse:0.634322
## [74] train-rmse:0.633977
## [75] train-rmse:0.633633
## [76] train-rmse:0.633291
## [77] train-rmse:0.632953
## [78] train-rmse:0.632618
## [79] train-rmse:0.632286
## [80] train-rmse:0.631956
## [81] train-rmse:0.631627
## [82] train-rmse:0.631304
## [83] train-rmse:0.630983
## [84] train-rmse:0.630662
## [85] train-rmse:0.630340
## [86] train-rmse:0.630022
## [87] train-rmse:0.629705
## [88] train-rmse:0.629391
## [89] train-rmse:0.629077
## [90] train-rmse:0.628764
## [91] train-rmse:0.628457
## [92] train-rmse:0.628153
## [93] train-rmse:0.627846
## [94] train-rmse:0.627545
## [95] train-rmse:0.627245
## [96] train-rmse:0.626945
## [97] train-rmse:0.626645
## [98] train-rmse:0.626351
## [99] train-rmse:0.626059
## [100]    train-rmse:0.625771
## [101]    train-rmse:0.625478
## [102]    train-rmse:0.625191
## [103]    train-rmse:0.624907
## [104]    train-rmse:0.624624
## [105]    train-rmse:0.624342
## [106]    train-rmse:0.624063
## [107]    train-rmse:0.623779
## [108]    train-rmse:0.623502
## [109]    train-rmse:0.623227
## [110]    train-rmse:0.622948
## [111]    train-rmse:0.622675
## [112]    train-rmse:0.622400
## [113]    train-rmse:0.622126
## [114]    train-rmse:0.621858
## [115]    train-rmse:0.621595
## [116]    train-rmse:0.621329
## [117]    train-rmse:0.621066
## [118]    train-rmse:0.620803
## [119]    train-rmse:0.620542
## [120]    train-rmse:0.620282
## [121]    train-rmse:0.620026
## [122]    train-rmse:0.619771
## [123]    train-rmse:0.619512
## [124]    train-rmse:0.619256
## [125]    train-rmse:0.619001
## [126]    train-rmse:0.618749
## [127]    train-rmse:0.618500
## [128]    train-rmse:0.618257
## [129]    train-rmse:0.618010
## [130]    train-rmse:0.617764
## [131]    train-rmse:0.617519
## [132]    train-rmse:0.617277
## [133]    train-rmse:0.617033
## [134]    train-rmse:0.616796
## [135]    train-rmse:0.616558
## [136]    train-rmse:0.616321
## [137]    train-rmse:0.616083
## [138]    train-rmse:0.615851
## [139]    train-rmse:0.615619
## [140]    train-rmse:0.615385
## [141]    train-rmse:0.615155
## [142]    train-rmse:0.614925
## [143]    train-rmse:0.614698
## [144]    train-rmse:0.614469
## [145]    train-rmse:0.614242
## [146]    train-rmse:0.614014
## [147]    train-rmse:0.613791
## [148]    train-rmse:0.613564
## [149]    train-rmse:0.613344
## [150]    train-rmse:0.613126
## [151]    train-rmse:0.612907
## [152]    train-rmse:0.612686
## [153]    train-rmse:0.612472
## [154]    train-rmse:0.612263
## [155]    train-rmse:0.612048
## [156]    train-rmse:0.611832
## [157]    train-rmse:0.611620
## [158]    train-rmse:0.611412
## [159]    train-rmse:0.611205
## [160]    train-rmse:0.610996
## [161]    train-rmse:0.610789
## [162]    train-rmse:0.610580
## [163]    train-rmse:0.610376
## [164]    train-rmse:0.610172
## [165]    train-rmse:0.609971
## [166]    train-rmse:0.609768
## [167]    train-rmse:0.609569
## [168]    train-rmse:0.609370
## [169]    train-rmse:0.609171
## [170]    train-rmse:0.608972
## [171]    train-rmse:0.608781
## [172]    train-rmse:0.608584
## [173]    train-rmse:0.608389
## [174]    train-rmse:0.608197
## [175]    train-rmse:0.608007
## [176]    train-rmse:0.607815
## [177]    train-rmse:0.607627
## [178]    train-rmse:0.607442
## [179]    train-rmse:0.607253
## [180]    train-rmse:0.607069
## [181]    train-rmse:0.606884
## [182]    train-rmse:0.606699
## [183]    train-rmse:0.606513
## [184]	train-rmse:0.606330
## [185]	train-rmse:0.606145
## [186]	train-rmse:0.605963
## [187]	train-rmse:0.605783
## [188]	train-rmse:0.605604
## [189]	train-rmse:0.605428
## [190]	train-rmse:0.605254
## [191]	train-rmse:0.605080
## [192]	train-rmse:0.604905
## [193]	train-rmse:0.604731
## [194]	train-rmse:0.604559
## [195]	train-rmse:0.604385
## [196]	train-rmse:0.604215
## [197]	train-rmse:0.604048
## [198]	train-rmse:0.603882
## [199]	train-rmse:0.603717
## [200]	train-rmse:0.603550
## [201]	train-rmse:0.603386
## [202]	train-rmse:0.603215
## [203]	train-rmse:0.603049
## [204]	train-rmse:0.602886
## [205]	train-rmse:0.602723
## [206]	train-rmse:0.602562
## [207]	train-rmse:0.602400
## [208]	train-rmse:0.602243
## [209]	train-rmse:0.602083
## [210]	train-rmse:0.601926
## [211]	train-rmse:0.601767
## [212]	train-rmse:0.601609
## [213]	train-rmse:0.601456
## [214]	train-rmse:0.601304
## [215]	train-rmse:0.601149
## [216]	train-rmse:0.600994
## [217]	train-rmse:0.600842
## [218]	train-rmse:0.600691
## [219]	train-rmse:0.600538
## [220]	train-rmse:0.600389
## [221]	train-rmse:0.600241
## [222]	train-rmse:0.600094
## [223]	train-rmse:0.599947
## [224]	train-rmse:0.599799
## [225]	train-rmse:0.599652
## [226]	train-rmse:0.599509
## [227]	train-rmse:0.599363
## [228]	train-rmse:0.599219
## [229]	train-rmse:0.599074
## [230]	train-rmse:0.598930
## [231]	train-rmse:0.598790
## [232]	train-rmse:0.598649
## [233]	train-rmse:0.598511
## [234]	train-rmse:0.598373
## [235]	train-rmse:0.598232
## [236]	train-rmse:0.598099
## [237]	train-rmse:0.597962
## [238]	train-rmse:0.597824
## [239]	train-rmse:0.597692
## [240]	train-rmse:0.597553
## [241]	train-rmse:0.597423
## [242]	train-rmse:0.597287
## [243]	train-rmse:0.597153
## [244]	train-rmse:0.597024
## [245]	train-rmse:0.596891
## [246]	train-rmse:0.596759
## [247]	train-rmse:0.596633
## [248]	train-rmse:0.596506
## [249]	train-rmse:0.596373
## [250]	train-rmse:0.596248
## [251]	train-rmse:0.596121
## [252]	train-rmse:0.595990
## [253]	train-rmse:0.595867
## [254]	train-rmse:0.595743
## [255]	train-rmse:0.595619
## [256]	train-rmse:0.595499
## [257]	train-rmse:0.595377
## [258]	train-rmse:0.595255
## [259]	train-rmse:0.595136
## [260]	train-rmse:0.595015
## [261]	train-rmse:0.594894
## [262]	train-rmse:0.594778
## [263]	train-rmse:0.594657
## [264]	train-rmse:0.594539
## [265]	train-rmse:0.594423
## [266]	train-rmse:0.594306
## [267]	train-rmse:0.594190
## [268]	train-rmse:0.594074
## [269]	train-rmse:0.593959
## [270]	train-rmse:0.593844
## [271]	train-rmse:0.593732
## [272]	train-rmse:0.593617
## [273]	train-rmse:0.593504
## [274]	train-rmse:0.593391
## [275]	train-rmse:0.593281
## [276]	train-rmse:0.593168
## [277]	train-rmse:0.593056
## [278]	train-rmse:0.592946
## [279]	train-rmse:0.592838
## [280]	train-rmse:0.592731
## [281]	train-rmse:0.592624
## [282]	train-rmse:0.592517
## [283]	train-rmse:0.592408
## [284]	train-rmse:0.592304
## [285]	train-rmse:0.592199
## [286]	train-rmse:0.592095
## [287]	train-rmse:0.591988
## [288]	train-rmse:0.591883
## [289]	train-rmse:0.591779
## [290]	train-rmse:0.591677
## [291]	train-rmse:0.591574
## [292]	train-rmse:0.591476
## [293]	train-rmse:0.591377
## [294]	train-rmse:0.591276
## [295]	train-rmse:0.591174
## [296]	train-rmse:0.591076
## [297]	train-rmse:0.590980
## [298]	train-rmse:0.590880
## [299]	train-rmse:0.590785
## [300]	train-rmse:0.590688
## [301]	train-rmse:0.590592
## [302]	train-rmse:0.590497
## [303]	train-rmse:0.590400
## [304]	train-rmse:0.590305
## [305]	train-rmse:0.590208
## [306]	train-rmse:0.590114
## [307]	train-rmse:0.590022
## [308]	train-rmse:0.589930
## [309]	train-rmse:0.589838
## [310]	train-rmse:0.589745
## [311]	train-rmse:0.589652
## [312]	train-rmse:0.589560
## [313]	train-rmse:0.589469
## [314]	train-rmse:0.589377
## [315]	train-rmse:0.589288
## [316]	train-rmse:0.589197
## [317]	train-rmse:0.589110
## [318]	train-rmse:0.589020
## [319]	train-rmse:0.588931
## [320]	train-rmse:0.588843
## [321]	train-rmse:0.588755
## [322]	train-rmse:0.588668
## [323]	train-rmse:0.588582
## [324]	train-rmse:0.588497
## [325]	train-rmse:0.588414
## [326]	train-rmse:0.588327
## [327]	train-rmse:0.588239
## [328]	train-rmse:0.588154
## [329]	train-rmse:0.588067
## [330]	train-rmse:0.587985
## [331]	train-rmse:0.587904
## [332]	train-rmse:0.587823
## [333]	train-rmse:0.587742
## [334]	train-rmse:0.587661
## [335]	train-rmse:0.587579
## [336]	train-rmse:0.587502
## [337]	train-rmse:0.587422
## [338]	train-rmse:0.587342
## [339]	train-rmse:0.587264
## [340]	train-rmse:0.587187
## [341]	train-rmse:0.587109
## [342]	train-rmse:0.587032
## [343]	train-rmse:0.586953
## [344]	train-rmse:0.586874
## [345]	train-rmse:0.586797
## [346]	train-rmse:0.586722
## [347]	train-rmse:0.586644
## [348]	train-rmse:0.586569
## [349]	train-rmse:0.586493
## [350]	train-rmse:0.586420
## [351]	train-rmse:0.586345
## [352]	train-rmse:0.586270
## [353]	train-rmse:0.586196
## [354]	train-rmse:0.586119
## [355]	train-rmse:0.586047
## [356]	train-rmse:0.585976
## [357]	train-rmse:0.585905
## [358]	train-rmse:0.585833
## [359]	train-rmse:0.585765
## [360]	train-rmse:0.585697
## [361]	train-rmse:0.585627
## [362]	train-rmse:0.585556
## [363]	train-rmse:0.585487
## [364]	train-rmse:0.585418
## [365]	train-rmse:0.585348
## [366]	train-rmse:0.585275
## [367]	train-rmse:0.585205
## [368]	train-rmse:0.585138
## [369]	train-rmse:0.585069
## [370]	train-rmse:0.585000
## [371]	train-rmse:0.584933
## [372]	train-rmse:0.584866
## [373]	train-rmse:0.584801
## [374]	train-rmse:0.584737
## [375]	train-rmse:0.584671
## [376]	train-rmse:0.584606
## [377]	train-rmse:0.584544
## [378]	train-rmse:0.584482
## [379]	train-rmse:0.584417
## [380]	train-rmse:0.584351
## [381]	train-rmse:0.584286
## [382]	train-rmse:0.584222
## [383]	train-rmse:0.584156
## [384]	train-rmse:0.584094
## [385]	train-rmse:0.584035
## [386]	train-rmse:0.583974
## [387]	train-rmse:0.583913
## [388]	train-rmse:0.583852
## [389]	train-rmse:0.583790
## [390]	train-rmse:0.583727
## [391]	train-rmse:0.583666
## [392]	train-rmse:0.583606
## [393]	train-rmse:0.583546
## [394]	train-rmse:0.583484
## [395]	train-rmse:0.583425
## [396]	train-rmse:0.583363
## [397]	train-rmse:0.583305
## [398]	train-rmse:0.583246
## [399]	train-rmse:0.583191
## [400]	train-rmse:0.583135
## [401]	train-rmse:0.583074
## [402]	train-rmse:0.583017
## [403]	train-rmse:0.582958
## [404]	train-rmse:0.582903
## [405]	train-rmse:0.582848
## [406]	train-rmse:0.582791
## [407]	train-rmse:0.582737
## [408]	train-rmse:0.582681
## [409]	train-rmse:0.582628
## [410]	train-rmse:0.582573
## [411]	train-rmse:0.582519
## [412]	train-rmse:0.582466
## [413]	train-rmse:0.582410
## [414]	train-rmse:0.582354
## [415]	train-rmse:0.582300
## [416]	train-rmse:0.582246
## [417]	train-rmse:0.582192
## [418]	train-rmse:0.582139
## [419]	train-rmse:0.582083
## [420]	train-rmse:0.582032
## [421]	train-rmse:0.581980
## [422]	train-rmse:0.581928
## [423]	train-rmse:0.581876
## [424]	train-rmse:0.581825
## [425]	train-rmse:0.581774
## [426]	train-rmse:0.581723
## [427]	train-rmse:0.581674
## [428]	train-rmse:0.581627
## [429]	train-rmse:0.581577
## [430]	train-rmse:0.581528
## [431]	train-rmse:0.581476
## [432]	train-rmse:0.581426
## [433]	train-rmse:0.581379
## [434]	train-rmse:0.581331
## [435]	train-rmse:0.581284
## [436]	train-rmse:0.581237
## [437]	train-rmse:0.581188
## [438]	train-rmse:0.581141
## [439]	train-rmse:0.581094
## [440]	train-rmse:0.581048
## [441]	train-rmse:0.580999
## [442]	train-rmse:0.580950
## [443]	train-rmse:0.580901
## [444]	train-rmse:0.580855
## [445]	train-rmse:0.580810
## [446]	train-rmse:0.580763
## [447]	train-rmse:0.580715
## [448]	train-rmse:0.580668
## [449]	train-rmse:0.580623
## [450]	train-rmse:0.580577
## [451]	train-rmse:0.580532
## [452]	train-rmse:0.580490
## [453]	train-rmse:0.580448
## [454]    train-rmse:0.580404
## [455]    train-rmse:0.580362
## [456]    train-rmse:0.580321
## [457]    train-rmse:0.580280
## [458]    train-rmse:0.580237
## [459]    train-rmse:0.580193
## [460]    train-rmse:0.580148
## [461]    train-rmse:0.580105
## [462]    train-rmse:0.580062
## [463]    train-rmse:0.580019
## [464]    train-rmse:0.579975
## [465]    train-rmse:0.579933
## [466]    train-rmse:0.579891
## [467]    train-rmse:0.579849
## [468]    train-rmse:0.579808
## [469]    train-rmse:0.579768
## [470]    train-rmse:0.579727
## [471]    train-rmse:0.579686
## [472]    train-rmse:0.579647
## [473]    train-rmse:0.579608
## [474]    train-rmse:0.579567
## [475]    train-rmse:0.579526
## [476]    train-rmse:0.579486
## [477]    train-rmse:0.579446
## [478]    train-rmse:0.579408
## [479]    train-rmse:0.579368
## [480]    train-rmse:0.579327
## [481]    train-rmse:0.579289
## [482]    train-rmse:0.579250
## [483]    train-rmse:0.579211
## [484]    train-rmse:0.579171
## [485]    train-rmse:0.579134
## [486]    train-rmse:0.579095
## [487]    train-rmse:0.579057
## [488]    train-rmse:0.579018
## [489]    train-rmse:0.578980
## [490]    train-rmse:0.578943
## [491]    train-rmse:0.578909
## [492]    train-rmse:0.578872
## [493]    train-rmse:0.578835
## [494]    train-rmse:0.578800
## [495]    train-rmse:0.578763
## [496]    train-rmse:0.578726
## [497]    train-rmse:0.578691
## [498]    train-rmse:0.578655
## [499]    train-rmse:0.578620
## [500]    train-rmse:0.578585
## [501]    train-rmse:0.578552
## [502]    train-rmse:0.578518
## [503]    train-rmse:0.578484
## [504]    train-rmse:0.578449
## [505]    train-rmse:0.578416
## [506]    train-rmse:0.578380
## [507]    train-rmse:0.578345
## [508]	train-rmse:0.578312
## [509]	train-rmse:0.578279
## [510]	train-rmse:0.578246
## [511]	train-rmse:0.578211
## [512]	train-rmse:0.578177
## [513]	train-rmse:0.578144
## [514]	train-rmse:0.578111
## [515]	train-rmse:0.578076
## [516]	train-rmse:0.578043
## [517]	train-rmse:0.578011
## [518]	train-rmse:0.577980
## [519]	train-rmse:0.577947
## [520]	train-rmse:0.577914
## [521]	train-rmse:0.577882
## [522]	train-rmse:0.577850
## [523]	train-rmse:0.577817
## [524]	train-rmse:0.577784
## [525]	train-rmse:0.577752
## [526]	train-rmse:0.577722
## [527]	train-rmse:0.577689
## [528]	train-rmse:0.577660
## [529]	train-rmse:0.577629
## [530]	train-rmse:0.577598
## [531]	train-rmse:0.577567
## [532]	train-rmse:0.577536
## [533]	train-rmse:0.577508
## [534]	train-rmse:0.577477
## [535]	train-rmse:0.577447
## [536]	train-rmse:0.577416
## [537]	train-rmse:0.577387
## [538]	train-rmse:0.577357
## [539]	train-rmse:0.577328
## [540]	train-rmse:0.577301
## [541]	train-rmse:0.577271
## [542]	train-rmse:0.577240
## [543]	train-rmse:0.577212
## [544]	train-rmse:0.577185
## [545]	train-rmse:0.577155
## [546]	train-rmse:0.577126
## [547]	train-rmse:0.577100
## [548]	train-rmse:0.577071
## [549]	train-rmse:0.577043
## [550]	train-rmse:0.577013
## [551]	train-rmse:0.576985
## [552]	train-rmse:0.576957
## [553]	train-rmse:0.576930
## [554]	train-rmse:0.576903
## [555]	train-rmse:0.576876
## [556]	train-rmse:0.576847
## [557]	train-rmse:0.576820
## [558]	train-rmse:0.576792
## [559]	train-rmse:0.576763
## [560]	train-rmse:0.576735
## [561]	train-rmse:0.576709
## [562]	train-rmse:0.576682
## [563]	train-rmse:0.576656
## [564]	train-rmse:0.576630
## [565]	train-rmse:0.576604
## [566]	train-rmse:0.576578
## [567]	train-rmse:0.576551
## [568]	train-rmse:0.576525
## [569]	train-rmse:0.576500
## [570]	train-rmse:0.576476
## [571]	train-rmse:0.576452
## [572]	train-rmse:0.576428
## [573]	train-rmse:0.576403
## [574]	train-rmse:0.576378
## [575]	train-rmse:0.576352
## [576]	train-rmse:0.576328
## [577]	train-rmse:0.576303
## [578]	train-rmse:0.576278
## [579]	train-rmse:0.576254
## [580]	train-rmse:0.576233
## [581]	train-rmse:0.576207
## [582]	train-rmse:0.576181
## [583]	train-rmse:0.576155
## [584]	train-rmse:0.576132
## [585]	train-rmse:0.576109
## [586]	train-rmse:0.576084
## [587]	train-rmse:0.576059
## [588]	train-rmse:0.576034
## [589]	train-rmse:0.576012
## [590]	train-rmse:0.575988
## [591]	train-rmse:0.575967
## [592]	train-rmse:0.575945
## [593]	train-rmse:0.575919
## [594]	train-rmse:0.575897
## [595]	train-rmse:0.575875
## [596]	train-rmse:0.575853
## [597]	train-rmse:0.575831
## [598]	train-rmse:0.575809
## [599]	train-rmse:0.575786
## [600]	train-rmse:0.575763
## [601]	train-rmse:0.575740
## [602]	train-rmse:0.575718
## [603]	train-rmse:0.575696
## [604]	train-rmse:0.575672
## [605]	train-rmse:0.575649
## [606]	train-rmse:0.575626
## [607]	train-rmse:0.575604
## [608]	train-rmse:0.575582
## [609]	train-rmse:0.575560
## [610]	train-rmse:0.575540
## [611]	train-rmse:0.575519
## [612]	train-rmse:0.575498
## [613]	train-rmse:0.575476
## [614]	train-rmse:0.575454
## [615]	train-rmse:0.575433
## [616]	train-rmse:0.575415
## [617]	train-rmse:0.575394
## [618]	train-rmse:0.575371
## [619]	train-rmse:0.575349
## [620]	train-rmse:0.575328
## [621]	train-rmse:0.575309
## [622]	train-rmse:0.575288
## [623]	train-rmse:0.575269
## [624]	train-rmse:0.575250
## [625]	train-rmse:0.575231
## [626]	train-rmse:0.575210
## [627]	train-rmse:0.575189
## [628]	train-rmse:0.575168
## [629]	train-rmse:0.575147
## [630]	train-rmse:0.575128
## [631]	train-rmse:0.575106
## [632]	train-rmse:0.575084
## [633]	train-rmse:0.575064
## [634]	train-rmse:0.575045
## [635]	train-rmse:0.575028
## [636]	train-rmse:0.575009
## [637]	train-rmse:0.574990
## [638]	train-rmse:0.574970
## [639]	train-rmse:0.574950
## [640]	train-rmse:0.574931
## [641]	train-rmse:0.574914
## [642]	train-rmse:0.574893
## [643]	train-rmse:0.574874
## [644]	train-rmse:0.574857
## [645]	train-rmse:0.574839
## [646]	train-rmse:0.574822
## [647]	train-rmse:0.574804
## [648]	train-rmse:0.574786
## [649]	train-rmse:0.574768
## [650]	train-rmse:0.574750
## [651]	train-rmse:0.574732
## [652]	train-rmse:0.574716
## [653]	train-rmse:0.574697
## [654]	train-rmse:0.574678
## [655]	train-rmse:0.574660
## [656]	train-rmse:0.574644
## [657]	train-rmse:0.574625
## [658]	train-rmse:0.574608
## [659]	train-rmse:0.574590
## [660]	train-rmse:0.574573
## [661]	train-rmse:0.574556
## [662]	train-rmse:0.574542
## [663]	train-rmse:0.574523
## [664]	train-rmse:0.574506
## [665]	train-rmse:0.574488
## [666]	train-rmse:0.574470
## [667]	train-rmse:0.574453
## [668]	train-rmse:0.574436
## [669]	train-rmse:0.574419
## [670]	train-rmse:0.574403
## [671]	train-rmse:0.574386
## [672]	train-rmse:0.574369
## [673]	train-rmse:0.574352
## [674]	train-rmse:0.574336
## [675]	train-rmse:0.574319
## [676]	train-rmse:0.574302
## [677]	train-rmse:0.574284
## [678]	train-rmse:0.574268
## [679]	train-rmse:0.574253
## [680]	train-rmse:0.574237
## [681]	train-rmse:0.574220
## [682]	train-rmse:0.574204
## [683]	train-rmse:0.574187
## [684]	train-rmse:0.574170
## [685]	train-rmse:0.574154
## [686]	train-rmse:0.574140
## [687]	train-rmse:0.574123
## [688]	train-rmse:0.574107
## [689]	train-rmse:0.574091
## [690]	train-rmse:0.574075
## [691]	train-rmse:0.574061
## [692]	train-rmse:0.574045
## [693]	train-rmse:0.574030
## [694]	train-rmse:0.574016
## [695]	train-rmse:0.574000
## [696]	train-rmse:0.573985
## [697]	train-rmse:0.573970
## [698]	train-rmse:0.573954
## [699]	train-rmse:0.573940
## [700]	train-rmse:0.573925
## [701]	train-rmse:0.573910
## [702]	train-rmse:0.573895
## [703]	train-rmse:0.573882
## [704]	train-rmse:0.573866
## [705]	train-rmse:0.573853
## [706]	train-rmse:0.573839
## [707]	train-rmse:0.573824
## [708]	train-rmse:0.573809
## [709]	train-rmse:0.573796
## [710]	train-rmse:0.573782
## [711]	train-rmse:0.573768
## [712]	train-rmse:0.573755
## [713]	train-rmse:0.573741
## [714]	train-rmse:0.573727
## [715]	train-rmse:0.573712
## [716]	train-rmse:0.573698
## [717]	train-rmse:0.573683
## [718]	train-rmse:0.573669
## [719]	train-rmse:0.573655
## [720]	train-rmse:0.573640
## [721]	train-rmse:0.573626
## [722]	train-rmse:0.573613
## [723]	train-rmse:0.573600
## [724]	train-rmse:0.573585
## [725]	train-rmse:0.573572
## [726]	train-rmse:0.573559
## [727]	train-rmse:0.573544
## [728]	train-rmse:0.573530
## [729]	train-rmse:0.573515
## [730]	train-rmse:0.573502
## [731]	train-rmse:0.573486
## [732]	train-rmse:0.573473
## [733]	train-rmse:0.573459
## [734]	train-rmse:0.573445
## [735]	train-rmse:0.573432
## [736]	train-rmse:0.573419
## [737]	train-rmse:0.573406
## [738]	train-rmse:0.573393
## [739]	train-rmse:0.573381
## [740]	train-rmse:0.573366
## [741]	train-rmse:0.573354
## [742]	train-rmse:0.573342
## [743]	train-rmse:0.573328
## [744]	train-rmse:0.573316
## [745]	train-rmse:0.573304
## [746]	train-rmse:0.573292
## [747]	train-rmse:0.573279
## [748]	train-rmse:0.573265
## [749]	train-rmse:0.573253
## [750]	train-rmse:0.573240
## [751]	train-rmse:0.573227
## [752]	train-rmse:0.573215
## [753]	train-rmse:0.573204
## [754]	train-rmse:0.573192
## [755]	train-rmse:0.573180
## [756]	train-rmse:0.573168
## [757]	train-rmse:0.573156
## [758]	train-rmse:0.573142
## [759]	train-rmse:0.573130
## [760]	train-rmse:0.573117
## [761]	train-rmse:0.573104
## [762]	train-rmse:0.573090
## [763]	train-rmse:0.573079
## [764]	train-rmse:0.573067
## [765]	train-rmse:0.573055
## [766]	train-rmse:0.573045
## [767]	train-rmse:0.573034
## [768]	train-rmse:0.573023
## [769]	train-rmse:0.573011
## [770]	train-rmse:0.573000
## [771]	train-rmse:0.572989
## [772]	train-rmse:0.572977
## [773]	train-rmse:0.572965
## [774]	train-rmse:0.572953
## [775]	train-rmse:0.572940
## [776]	train-rmse:0.572930
## [777]	train-rmse:0.572918
## [778]	train-rmse:0.572907
## [779]	train-rmse:0.572897
## [780]	train-rmse:0.572887
## [781]	train-rmse:0.572877
## [782]	train-rmse:0.572866
## [783]	train-rmse:0.572854
## [784]	train-rmse:0.572844
## [785]	train-rmse:0.572832
## [786]	train-rmse:0.572820
## [787]	train-rmse:0.572809
## [788]	train-rmse:0.572798
## [789]	train-rmse:0.572787
## [790]	train-rmse:0.572777
## [791]	train-rmse:0.572765
## [792]	train-rmse:0.572756
## [793]	train-rmse:0.572746
## [794]	train-rmse:0.572735
## [795]	train-rmse:0.572725
## [796]	train-rmse:0.572714
## [797]	train-rmse:0.572703
## [798]	train-rmse:0.572694
## [799]	train-rmse:0.572684
## [800]	train-rmse:0.572673
## [801]	train-rmse:0.572662
## [802]	train-rmse:0.572652
## [803]	train-rmse:0.572643
## [804]	train-rmse:0.572632
## [805]	train-rmse:0.572621
## [806]	train-rmse:0.572611
## [807]	train-rmse:0.572601
## [808]	train-rmse:0.572591
## [809]	train-rmse:0.572580
## [810]	train-rmse:0.572570
## [811]	train-rmse:0.572561
## [812]	train-rmse:0.572552
## [813]	train-rmse:0.572541
## [814]	train-rmse:0.572531
## [815]	train-rmse:0.572522
## [816]	train-rmse:0.572512
## [817]	train-rmse:0.572500
## [818]	train-rmse:0.572491
## [819]	train-rmse:0.572481
## [820]	train-rmse:0.572470
## [821]	train-rmse:0.572461
## [822]	train-rmse:0.572453
## [823]	train-rmse:0.572444
## [824]	train-rmse:0.572434
## [825]	train-rmse:0.572425
## [826]	train-rmse:0.572415
## [827]	train-rmse:0.572406
## [828]	train-rmse:0.572396
## [829]	train-rmse:0.572388
## [830]	train-rmse:0.572378
## [831]	train-rmse:0.572368
## [832]    train-rmse:0.572358
## [833]    train-rmse:0.572349
## [834]    train-rmse:0.572341
## [835]    train-rmse:0.572329
## [836]    train-rmse:0.572320
## [837]    train-rmse:0.572310
## [838]    train-rmse:0.572301
## [839]    train-rmse:0.572293
## [840]    train-rmse:0.572284
## [841]    train-rmse:0.572276
## [842]    train-rmse:0.572268
## [843]    train-rmse:0.572259
## [844]    train-rmse:0.572249
## [845]    train-rmse:0.572240
## [846]    train-rmse:0.572231
## [847]    train-rmse:0.572220
## [848]    train-rmse:0.572211
## [849]    train-rmse:0.572201
## [850]    train-rmse:0.572194
## [851]    train-rmse:0.572185
## [852]    train-rmse:0.572176
## [853]    train-rmse:0.572168
## [854]    train-rmse:0.572159
## [855]    train-rmse:0.572150
## [856]    train-rmse:0.572142
## [857]    train-rmse:0.572133
## [858]    train-rmse:0.572126
## [859]    train-rmse:0.572117
## [860]    train-rmse:0.572108
## [861]    train-rmse:0.572100
## [862]    train-rmse:0.572091
## [863]    train-rmse:0.572082
## [864]    train-rmse:0.572074
## [865]    train-rmse:0.572065
## [866]    train-rmse:0.572057
## [867]    train-rmse:0.572048
## [868]    train-rmse:0.572040
## [869]    train-rmse:0.572032
## [870]    train-rmse:0.572023
## [871]    train-rmse:0.572016
## [872]    train-rmse:0.572008
## [873]    train-rmse:0.572001
## [874]    train-rmse:0.571993
## [875]    train-rmse:0.571985
## [876]    train-rmse:0.571977
## [877]    train-rmse:0.571969
## [878]    train-rmse:0.571960
## [879]    train-rmse:0.571953
## [880]    train-rmse:0.571945
## [881]    train-rmse:0.571936
## [882]    train-rmse:0.571928
## [883]    train-rmse:0.571920
## [884]    train-rmse:0.571912
## [885]    train-rmse:0.571904
## [886]    train-rmse:0.571898
## [887]    train-rmse:0.571890
## [888]    train-rmse:0.571882
## [889]    train-rmse:0.571873
## [890]    train-rmse:0.571866
## [891]    train-rmse:0.571859
## [892]    train-rmse:0.571853
## [893]    train-rmse:0.571845
## [894]    train-rmse:0.571837
## [895]    train-rmse:0.571829
## [896]    train-rmse:0.571820
## [897]    train-rmse:0.571811
## [898]    train-rmse:0.571803
## [899]    train-rmse:0.571795
## [900]    train-rmse:0.571787
## [901]    train-rmse:0.571779
## [902]    train-rmse:0.571772
## [903]    train-rmse:0.571765
## [904]    train-rmse:0.571757
## [905]    train-rmse:0.571750
## [906]    train-rmse:0.571743
## [907]    train-rmse:0.571736
## [908]    train-rmse:0.571729
## [909]    train-rmse:0.571722
## [910]    train-rmse:0.571714
## [911]    train-rmse:0.571707
## [912]    train-rmse:0.571700
## [913]    train-rmse:0.571693
## [914]    train-rmse:0.571685
## [915]    train-rmse:0.571677
## [916]    train-rmse:0.571671
## [917]    train-rmse:0.571666
## [918]    train-rmse:0.571658
## [919]    train-rmse:0.571651
## [920]    train-rmse:0.571644
## [921]    train-rmse:0.571637
## [922]    train-rmse:0.571631
## [923]    train-rmse:0.571623
## [924]    train-rmse:0.571617
## [925]    train-rmse:0.571610
## [926]    train-rmse:0.571602
## [927]    train-rmse:0.571594
## [928]    train-rmse:0.571586
## [929]    train-rmse:0.571580
## [930]    train-rmse:0.571574
## [931]    train-rmse:0.571568
## [932]    train-rmse:0.571560
## [933]    train-rmse:0.571553
## [934]    train-rmse:0.571546
## [935]    train-rmse:0.571539
## [936]    train-rmse:0.571532
## [937]    train-rmse:0.571526
## [938]    train-rmse:0.571519
## [939]    train-rmse:0.571511
## [940]    train-rmse:0.571505
## [941]    train-rmse:0.571498
## [942]    train-rmse:0.571490
## [943]    train-rmse:0.571483
## [944]    train-rmse:0.571475
## [945]    train-rmse:0.571468
## [946]    train-rmse:0.571462
## [947]    train-rmse:0.571456
## [948]    train-rmse:0.571449
## [949]    train-rmse:0.571443
## [950]    train-rmse:0.571436
## [951]    train-rmse:0.571428
## [952]    train-rmse:0.571422
## [953]    train-rmse:0.571416
## [954]    train-rmse:0.571409
## [955]    train-rmse:0.571402
## [956]    train-rmse:0.571396
## [957]    train-rmse:0.571388
## [958]    train-rmse:0.571382
## [959]    train-rmse:0.571375
## [960]    train-rmse:0.571369
## [961]    train-rmse:0.571364
## [962]    train-rmse:0.571358
## [963]    train-rmse:0.571351
## [964]    train-rmse:0.571346
## [965]    train-rmse:0.571340
## [966]    train-rmse:0.571332
## [967]    train-rmse:0.571325
## [968]    train-rmse:0.571317
## [969]    train-rmse:0.571312
## [970]    train-rmse:0.571306
## [971]    train-rmse:0.571300
## [972]    train-rmse:0.571293
## [973]    train-rmse:0.571286
## [974]    train-rmse:0.571280
## [975]    train-rmse:0.571273
## [976]    train-rmse:0.571266
## [977]    train-rmse:0.571260
## [978]    train-rmse:0.571254
## [979]    train-rmse:0.571248
## [980]    train-rmse:0.571241
## [981]    train-rmse:0.571234
## [982]    train-rmse:0.571229
## [983]    train-rmse:0.571222
## [984]    train-rmse:0.571216
## [985]    train-rmse:0.571211
## [986]    train-rmse:0.571206
## [987]    train-rmse:0.571200
## [988]    train-rmse:0.571194
## [989]    train-rmse:0.571188
## [990]    train-rmse:0.571181
## [991]    train-rmse:0.571175
## [992]    train-rmse:0.571168
## [993]    train-rmse:0.571162
## [994]    train-rmse:0.571156
## [995]    train-rmse:0.571152
## [996]    train-rmse:0.571146
## [997]    train-rmse:0.571140
## [998]    train-rmse:0.571134
## [999]    train-rmse:0.571128
## [1000]   train-rmse:0.571121
```

```
finish_time <- Sys.time()                    # Record finish time
finish_time - start_time                     # Display total training time
```

```
## Time difference of 1.357058 mins
```

```
# Use the trained model to predict the Test dataset
test_pred <- as.data.frame(predict(xgb_model , newdata = test_matrix))

# Calculate the RMSE of the Predictions
rmse <- RMSE(test_label$rating, test_pred$`predict(xgb_model, newdata = test_matrix)`)

XGB_rmse_results <- bind_rows(XGB_rmse_results,
                              tibble(Method="XGB Linear 1000 Boost Rnds",
                                     RMSE = rmse,
                                     Train_Time = paste(as.character(round(as.numeric(finish_time - sta

kable(XGB_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE	Train_Time
XGB Linear 50 Boost Rnds	0.6547372	4.52 secs
XGB Linear 1000 Boost Rnds	0.6178744	81.42 secs

```
#*********************************************************************************************
# Model: XGB Mixed Tree 5 Boost Rnds
#*********************************************************************************************

xgb_params <- list(booster = "gbtree",
                   objective = "reg:linear",
                   colsample_bynode = 0.8,
                   learning_rate = 1,
                   max_depth = 10,
                   num_parallel_tree = 25,
                   subsample = 0.8,
                   verbosity = 3,
                   silent = 0)

# Train the XGB tree using the currently set xgb_params
start_time <- Sys.time()                     # Record start time

set.seed(1, sample.kind = "Rounding")
xgb_model <- xgboost(params = xgb_params,
                     data = train_matrix,
                     nrounds = 5,            # Maximum number of Boosting Rounds
                     nthread = 1,            # Must be set to 1 to get reproducible results
                     verbose = 2)            # Display pogress during Training
```

```
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 726 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 724 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 700 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 748 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 612 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 712 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 784 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 862 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 762 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 918 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 644 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 686 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 672 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 784 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 758 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 802 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 698 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 688 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 930 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 672 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 688 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 778 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 780 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 980 extra r
## [10:40:49] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 762 extra r
## [1]  train-rmse:0.561407
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1284 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1730 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1830 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1542 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1022 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1558 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1696 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 984 extra r
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1544 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 998 extra r
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1512 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1576 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1252 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1496 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1632 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1700 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1582 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1320 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1050 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1478 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1264 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1078 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1052 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1634 extra
## [10:41:33] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1582 extra
## [2]  train-rmse:0.555592
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 952 extra r
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 892 extra r
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1152 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 928 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 920 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 888 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 810 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1574 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 930 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 960 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 918 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 960 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 812 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 768 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 814 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 922 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 938 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 920 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 942 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1632 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1016 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1608 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 932 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1446 extra
## [10:42:16] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 978 extra
## [3]  train-rmse:0.553008
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1222 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1330 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1414 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1204 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1192 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1592 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 706 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1160 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 988 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1176 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1182 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1066 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1230 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1294 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1208 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1012 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 870 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1452 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1224 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1492 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1096 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1558 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1374 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1150 extra
## [10:42:58] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1140 extra
## [4]  train-rmse:0.550817
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1230 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 954 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1384 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1446 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1270 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1186 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1584 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1580 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1132 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1332 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 990 extra n
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1612 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1198 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 944 extra n
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1108 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1274 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1606 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1428 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1282 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1620 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1284 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 832 extra n
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 872 extra n
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1508 extra
## [10:43:40] INFO: amalgamation/../src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 994 extra n
## [5]  train-rmse:0.548436
```

```
finish_time <- Sys.time()                    # Record finish time
finish_time - start_time                     # Display total training time
```

```
## Time difference of 3.541863 mins
```

```
# Use the trained model to predict the Test dataset
test_pred <- as.data.frame(predict(xgb_model , newdata = test_matrix))

# Calculate the RMSE of the Predictions
rmse <- RMSE(test_label$rating, test_pred$`predict(xgb_model, newdata = test_matrix)`)

XGB_rmse_results <- bind_rows(XGB_rmse_results,
                              tibble(Method="XGB Mixed Tree 5 Boost Rnds",
                                     RMSE = rmse,
                                     Train_Time = paste(as.character(round(as.numeric(finish_time - sta

kable(XGB_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE	Train_Time
XGB Linear 50 Boost Rnds	0.6547372	4.52 secs
XGB Linear 1000 Boost Rnds	0.6178744	81.42 secs
XGB Mixed Tree 5 Boost Rnds	0.6152551	212.51 secs

xgBoost - Observations and Additional Comments

Despite the ability to use a full Training set including many additional derived parameters, the xgBoost models were only able to outperform the Linear Regression Models when using 50 Boosting

Clean up the Environment before moving to the next model.

```
rm(finish_time, importance_matrix, movie_data, rmse, start_time, test_data, test_label,
   test_matrix, test_pred, test_set, train_data, train_label, train_matrix, train_set,
   user_data, xgb_model, xgb_params)
```

Recosystem - Matrix Factorization w/ Parallel Stochastic Gradient Descent

While we had success with xgBoost, there are several "Recommender" packages available in the R ecosystem focused specifically on Collaborative Filtering. Within the Recommender system, the main task is to predict unknown entries in a rating matrix composed of Users vs Items.

One technique used to solve the recommender problem is matrix factorization where the rating matrix is successively approximated by the product of two matrices of lower dimensions.

Recosystem is a wrapper for the open-source LibFM library that implements optimized and parallelized matrix factorization. The recosystem library also provides easy to use functions to perform parameter tuning across a grid of parameter options, training against an in-memory or on-disk dataset using the tuned parameters, and prediction of results against a test set.

RECO - Data Preparation

Recosystem only uses three variables:

- **Rating (rating):** Rating score provided by User for Movie
- **User (user_id):** User ID associated with the Rating
- **Beer (beer_id):** Beer ID associated with the Rating

Create Training and Test Datasets

Given the few required variables, the datasets are quite small and should easily fit into system memory. We will only be keeping the **rating**, **user_id**, and **beer_id** columns:

```
#*****************************************************************************
#*****************************************************************************
#** Recosystem - Matrix Factorization w/ Parallel Stochastic Gradient Descent **
#*****************************************************************************
#*****************************************************************************

# Load required libraries
if (!require(recosystem)) install.packages('recosystem')
library(recosystem)

#**********************************************************************************
# Recosystem - Matrix Factorization with Parallel Stochastic Gradient Descent
#**********************************************************************************

# Create new Traing and Test datasets. Recosystem only uses three columns: Rating, user_id, and beer_id
train_set <- beer_train %>% select("user_id","beer_id", "rating") %>% as.matrix()
test_set <-  beer_test %>% select("user_id","beer_id", "rating") %>% as.matrix()
```

Create the Recosystem Object

```
# Create the Reco Recommender object
r = Reco()
```

Create the Tuning Parameter Grid

One very nice feature of Recosystem, besides it's speed, is that tuning the training parameters is very easy and the overall best performing tuned parameters are stored directly in the Reco object making calling them against the Training set simple.

Below we will create a small Tuning grid as an example. This tuning grid only tunes between 2 values of **dim** (10 & 20).

NOTE: We can use nthread = X in the parameter list to define the number of threads to use. If omitted, Recosystem will try to use the maximum available. I used nthread = 8

```
# One very nice feature of Reco, besides it's speed, is that tuning training set is very easy and the
# overall best performing tuned paramaters are stored directly in the Reco object.

# Here we will create some Tuning Grids

#**********************************************************************************
# First Tuning Round - Takes around 10 minutes with 8 Threads
#**********************************************************************************
opts_list = list(dim      = c(10, 20),         # Number of Latent Features
                 costp_l1 = c(0, 0.1),         # L1 regularization cost for User factors
                 costp_l2 = c(0.01, 0.1),      # L2 regularization cost for User factors
                 costq_l1 = c(0, 0.1),         # L1 regularization cost for Beer factors
                 costq_l2 = c(0.01, 0.1),      # L2 regularization cost for Beer factors
                 lrate    = c(0.01, 0.1),      # Learning Rate - Aprox step size in Gradient Descent
                 niter    = 10,                # Number of Iterations for Training (Not used in Tuning)
                 nfolds   = 5,                 # Number of Folds for CV in Tuning
                 verbose  = FALSE,             # Don't Show Progress
                 nthread  = 8)        #!!! Can be set to higher values for Tuning, but MUST be set to 1
                                      #    for Training or the results are not reproducible
# RESULTS
# dim      20
# costp_l1 0
# costp_l2 0.01
# costq_l1 0.1
# costq_l2 0.1
# lrate    0.1

#**********************************************************************************
```

Tune the Model using the Parameter Grid

As noted, if you want, you can use the **nthread** paramater, but it can't be used as a Training parameter. It requires additional investigation, but **nthread** must be set to 1 during Training or the results cannot be reproduced. There must be some randomization in the parallelization that does not rely on setting the Seed value.

Next, we will tune the model given the parameter grid above:

```
#**********************************************************************************
# TUNE the Mode
#**********************************************************************************

start <- Sys.time()

set.seed(1, sample.kind = "Rounding")
opts_tune = r$tune(data_memory(train_set[, 1], train_set[, 2], train_set[, 3]),
                   opts = opts_list)

finish <- Sys.time()
tune_time <- finish - start
tune_time
```

```
## Time difference of 2.632348 mins
```

```
opts_tune$min
```

```
## $dim
## [1] 20
##
## $costp_l1
## [1] 0
##
## $costp_l2
## [1] 0.01
##
## $costq_l1
## [1] 0.1
##
## $costq_l2
## [1] 0.1
##
## $lrate
## [1] 0.1
##
## $loss_fun
## [1] 0.9160692
```

We see that the Tuned options chose 20 for the **dim** variable. Many iterations were performed to optimize the other model parameters. For the sake of time, we won't perform additional optimizations, but we will look later at the results of using Tuning and 5-fold Cross-Validation in order to find a semi-optimal number of Factors (**dim**)

Train the Model using the Best Parameters

Once the parameters have been optimized, we Train the model:

```
#**********************************************************************************
# TRAIN the Model over 50 Iterations
#**********************************************************************************
start <- Sys.time()

set.seed(1, sample.kind = "Rounding")
r$train(data_memory(train_set[, 1], train_set[, 2], train_set[, 3]),
        opts = c(opts_tune$min,
                 niter    = 50,      # Train over 50 Iterations
                 nthread = 1))       # Must be set to 1 for training
```

```
## iter      tr_rmse          obj
##    0       1.2728  2.3655e+006
##    1       0.7944  1.3629e+006
##    2       0.7418  1.2546e+006
##    3       0.7146  1.1941e+006
##    4       0.6966  1.1529e+006
##    5       0.6827  1.1216e+006
##    6       0.6714  1.0977e+006
##    7       0.6615  1.0765e+006
##    8       0.6526  1.0593e+006
##    9       0.6444  1.0444e+006
##   10       0.6365  1.0302e+006
##   11       0.6293  1.0175e+006
##   12       0.6222  1.0058e+006
##   13       0.6156  9.9562e+005
##   14       0.6090  9.8505e+005
##   15       0.6028  9.7503e+005
##   16       0.5973  9.6713e+005
##   17       0.5918  9.5928e+005
##   18       0.5865  9.5149e+005
##   19       0.5814  9.4377e+005
##   20       0.5768  9.3739e+005
##   21       0.5724  9.3093e+005
##   22       0.5681  9.2481e+005
##   23       0.5643  9.1915e+005
##   24       0.5606  9.1348e+005
##   25       0.5572  9.0877e+005
##   26       0.5537  9.0334e+005
##   27       0.5507  8.9962e+005
##   28       0.5476  8.9527e+005
##   29       0.5447  8.9023e+005
##   30       0.5424  8.8703e+005
##   31       0.5396  8.8282e+005
##   32       0.5372  8.7912e+005
##   33       0.5352  8.7619e+005
##   34       0.5329  8.7256e+005
##   35       0.5309  8.6943e+005
##   36       0.5291  8.6647e+005
##   37       0.5272  8.6313e+005
##   38       0.5256  8.6046e+005
##   39       0.5239  8.5749e+005
##   40       0.5224  8.5526e+005
##   41       0.5209  8.5240e+005
##   42       0.5195  8.5005e+005
##   43       0.5182  8.4778e+005
##   44       0.5170  8.4554e+005
##   45       0.5157  8.4300e+005
##   46       0.5145  8.4077e+005
##   47       0.5135  8.3911e+005
##   48       0.5123  8.3672e+005
##   49       0.5114  8.3479e+005
```

```
finish <- Sys.time()
train_time <- finish - start
train_time
```

```
## Time difference of 21.86424 secs
```

Predict the Test Set

After training, we use the trained model to predict the Ratings in the Test Set and calculate the RMSE:

```
#************************************************************************************
# PREDICT the Test Ratings
#************************************************************************************
start <- Sys.time()

pred <- r$predict(data_memory(test_set[, 1], test_set[, 2], test_set[, 3]), out_memory())
```

```
finish <- Sys.time()
pred_time <- finish - start
pred_time
```

```
## Time difference of 1.037113 mins
```

Calculate the RMSE

```
#*********************************************************************************
# Calculate the RMSE
#*********************************************************************************

rmse <- RMSE(test_set[,3],pred)

RECO_rmse_results <- tibble(Method="V1 - 20 x 50",
                            RMSE = rmse,
                            Train_Time = round(as.numeric(train_time, units="mins"), 2))
kable(RECO_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE	Train_Time
V1 - 20 x 50	0.6454194	0.36

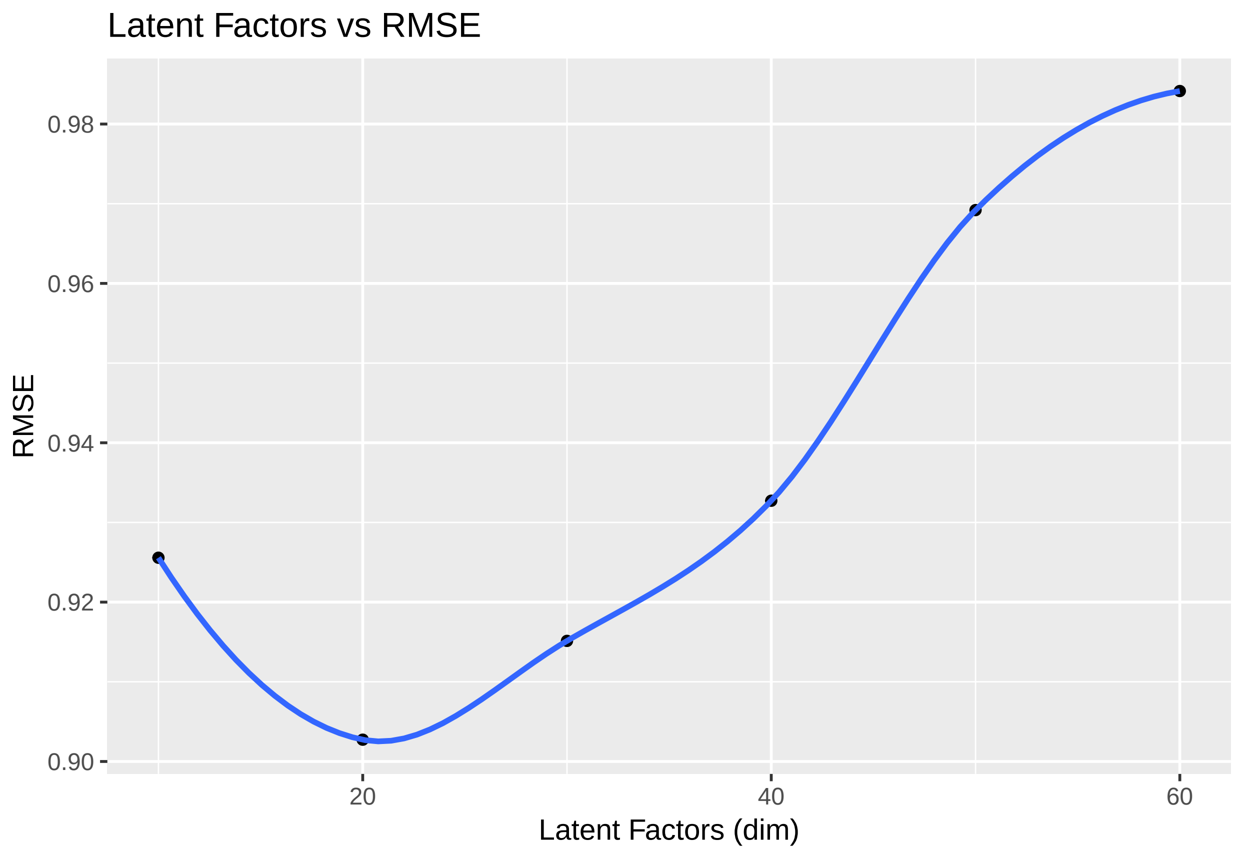
As mentioned previously, we will now look at the 5-Fold Cross Validation results using the Tuning function to predict the RMSE across values of the **dim** parameter between 10 and 100:

```
#*********************************************************************************
# Tune the DIM parameter - FOR 10-100 and then 10-20 and Graph
# !!!!  NOTE !!!!  THESE WILL TAKE A LONG TIME TO RUN
#*********************************************************************************
# Tune & Graph Latent Factors (dim) from 10 to 60 by 10
#*********************************************************************************

opts_list = list(dim      = c(seq(10, 60, 10)),     # Number of Latent Features
                 costp_l1 = c(0),        # L1 regularization cost for User factors
                 costp_l2 = c(0.01),     # L2 regularization cost for User factors
                 costq_l1 = c(0.1),      # L1 regularization cost for Beer factors
                 costq_l2 = c(0.1),      # L2 regularization cost for Beer factors
                 lrate    = c(0.1),      # Learning Rate - Aprox step size in Gradient Descent
                 nfold    = 5,           # Number of folds for Cross Validation
                 nthread  = 8,           # Set Number of Threads to 1 *** Required to get Reproducible R
                 niter    = 50,          # Number of Iterations
                 verbose  = FALSE)       # Don't Show Fold Details

start <- Sys.time()
set.seed(1, sample.kind = "Rounding")
opts_tune <- r$tune(data_memory(train_set[, 1], train_set[, 2], train_set[, 3]),
                    opts = opts_list)
finish <- Sys.time()
tune_time <- finish - start

opts_tune$min
```

```
## $dim
## [1] 20
```

```
##
## $costp_l1
## [1] 0
##
## $costp_l2
## [1] 0.01
##
## $costq_l1
## [1] 0.1
##
## $costq_l2
## [1] 0.1
##
## $lrate
## [1] 0.1
##
## $loss_fun
## [1] 0.9027375
```

```
opts_tune$res %>%
  ggplot(aes(dim, loss_fun)) +
  geom_point() +
  geom_smooth(method="loess") +
  labs(x="Latent Factors (dim)", y="RMSE") +
  ggtitle("Latent Factors vs RMSE")
```



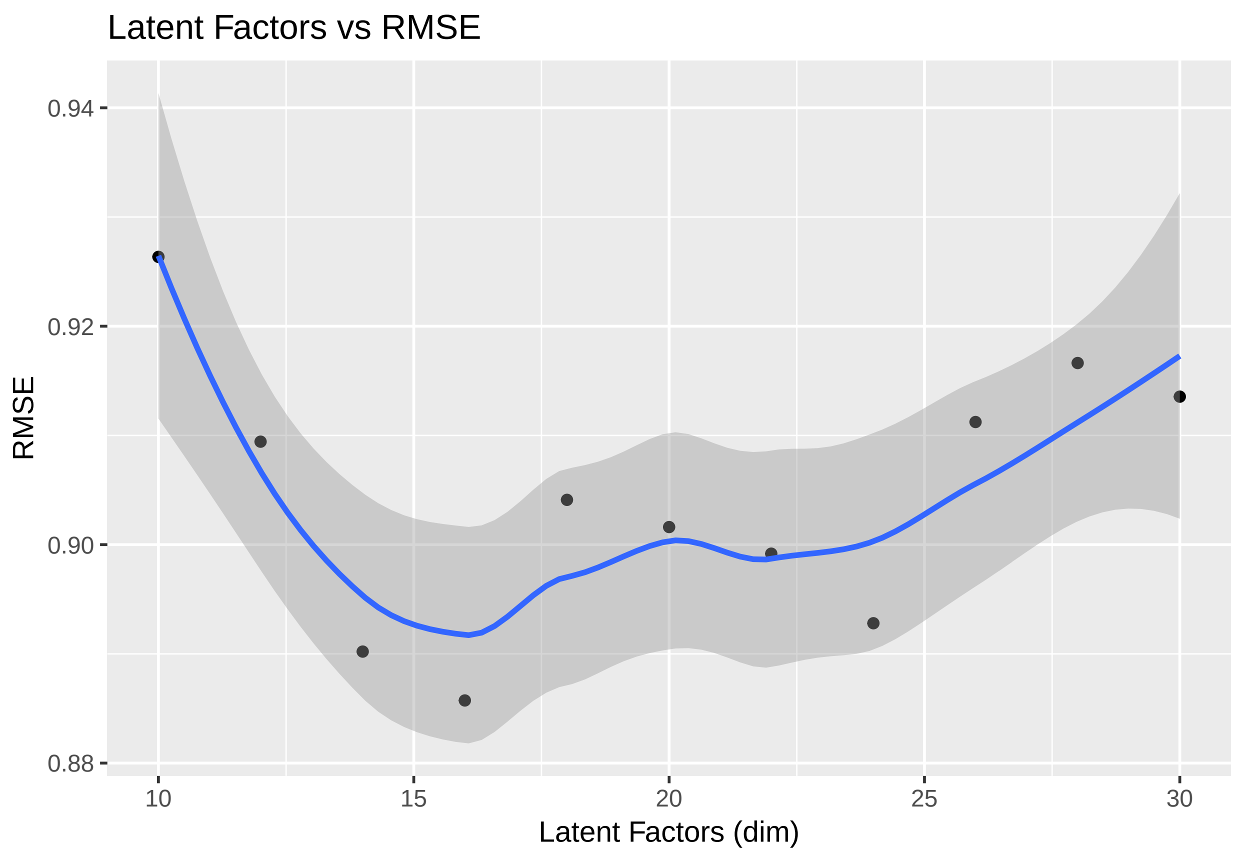
```r
RECO_CV_10_100 <- opts_tune
```

We see that the predicted minimum is somewhere between 10 & 30, so we'll re-run the Tuning between 10 and 40 to get a better look:

```r
#*************************************************************************************
# Tune & Graph Latent Factors (dim) from 10 to 30
#*************************************************************************************

opts_list = list(dim      = c(seq(10, 30, 2)),     # Number of Latent Features
                 costp_l1 = c(0),         # L1 regularization cost for User factors
                 costp_l2 = c(0.01),      # L2 regularization cost for User factors
                 costq_l1 = c(0.1),       # L1 regularization cost for Beer factors
                 costq_l2 = c(0.1),       # L2 regularization cost for Beer factors
                 lrate    = c(0.1),       # Learning Rate - Aprox step size in Gradient Descent
                 nfold    = 5,            # Number of folds for Cross Validation
                 nthread  = 8,            # Set Number of Threads to 1 *** Required to get Reproducible R
                 niter    = 50,           # Number of Iterations
                 verbose  = FALSE)        # Don't Show Fold Details

start <- Sys.time()
set.seed(1, sample.kind = "Rounding")
opts_tune <- r$tune(data_memory(train_set[, 1], train_set[, 2], train_set[, 3]),
                    opts = opts_list)
finish <- Sys.time()
tune_time <- finish - start

opts_tune$min
```

```
## $dim
## [1] 16
##
## $costp_l1
## [1] 0
##
## $costp_l2
## [1] 0.01
##
## $costq_l1
## [1] 0.1
##
## $costq_l2
## [1] 0.1
##
## $lrate
## [1] 0.1
##
## $loss_fun
## [1] 0.8857336
```

```r
opts_tune$res %>%
  ggplot(aes(dim, loss_fun)) +
  geom_point() +
  geom_smooth(method="loess") +
  labs(x="Latent Factors (dim)", y="RMSE") +
  ggtitle("Latent Factors vs RMSE")
```



```
RECO_CV_15_30 <- opts_tune
```

From this, **dim = 16** gives the smallest RMSE, so we will re-Train the model at **dim = 16** and see how it compares to the current RMSE result:

```
#**********************************************************************************
# Train "Optimal" RMSE - (dim = 16)  V3 - 16 x 50
#**********************************************************************************
opts_list = list(dim      = c(16),      # Number of Latent Features
                 costp_l1 = c(0),       # L1 regularization cost for User factors
                 costp_l2 = c(0.01),    # L2 regularization cost for User factors
                 costq_l1 = c(0.1),     # L1 regularization cost for Beer factors
                 costq_l2 = c(0.1),     # L2 regularization cost for Beer factors
                 lrate    = c(0.1),     # Learning Rate - Aprox step size in Gradient Descent
                 nfold    = 5,          # Number of folds for Cross Validation
                 nthread  = 1,          # Set Number of Threads to 1 *** Required to get Reproducible R
                 niter    = 50,         # Number of Iterations
                 verbose  = FALSE)      # Don't Show Fold Details

start <- Sys.time()

set.seed(1, sample.kind = "Rounding")
r$train(data_memory(train_set[, 1], train_set[, 2], train_set[, 3]),
        opts = c(opts_list,
                 nthread = 1))
```

```r
finish <- Sys.time()
train_time <- finish - start

start <- Sys.time()

pred <- r$predict(data_memory(test_set[, 1], test_set[, 2], test_set[, 3]), out_memory())
finish <- Sys.time()

pred_time <- finish - start
pred_time
```

```
## Time difference of 44.89091 secs
```

```r
rmse <- RMSE(test_set[,3],pred)

RECO_rmse_results <- bind_rows(RECO_rmse_results,
                               tibble(Method="V2 - 16 x 50 (Optimal DIM)",
                                      RMSE = rmse,
                                      Train_Time = round(as.numeric(train_time, units="mins"), 2)))
kable(RECO_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE	Train_Time
V1 - 20 x 50	0.6454194	0.36
V2 - 16 x 50 (Optimal DIM)	0.6440126	0.27

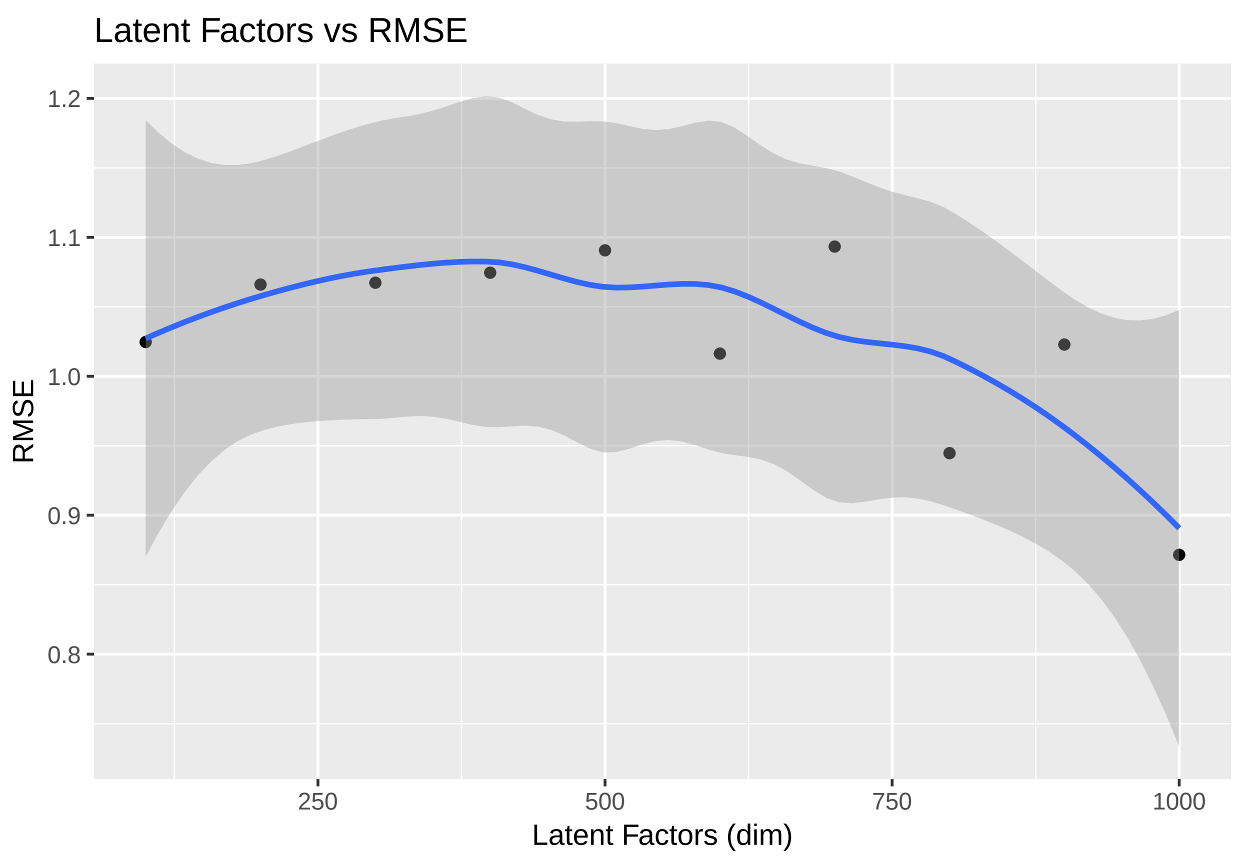
```r
#*************************************************************************************
```

Here the model with **dim = 16** produced the minimal RMSE in the Cross-Validation Tuning. As a test, we will create a very Large model with **dim = 1000** and train it over 100 iterations to see what the actual RMSE is on the Test set. The code is provided below but will not be run because it takes almost 1 Hour to complete. The resulting RMSE will simply be added to the current results table.

```r
#*************************************************************************************
# Tune & Graph Latent Factors (dim) from 100 to 1000 by 100
#*************************************************************************************

opts_list = list(dim      = c(seq(100, 1000, 100)),    # Number of Latent Features
                 costp_l1 = c(0),        # L1 regularization cost for User factors
                 costp_l2 = c(0.01),     # L2 regularization cost for User factors
                 costq_l1 = c(0.1),      # L1 regularization cost for Beer factors
                 costq_l2 = c(0.1),      # L2 regularization cost for Beer factors
                 lrate    = c(0.1),      # Learning Rate - Aprox step size in Gradient Descent
                 nfold    = 5,           # Number of folds for Cross Validation
                 nthread  = 8,           # Set Number of Threads to 1 *** Required to get Reproducible R
                 niter    = 50,          # Number of Iterations
                 verbose  = FALSE)       # Don't Show Fold Details

start <- Sys.time()
set.seed(1, sample.kind = "Rounding")
opts_tune <- r$tune(data_memory(train_set[, 1], train_set[, 2], train_set[, 3]),
                    opts = opts_list)
finish <- Sys.time()
tune_time <- finish - start

opts_tune$min
```

```
## $dim
## [1] 1000
##
## $costp_l1
## [1] 0
##
## $costp_l2
## [1] 0.01
##
## $costq_l1
## [1] 0.1
##
## $costq_l2
## [1] 0.1
##
## $lrate
## [1] 0.1
##
## $loss_fun
## [1] 0.8714909
```

```r
opts_tune$res %>%
  ggplot(aes(dim, loss_fun)) +
  geom_point() +
  geom_smooth(method="loess") +
  labs(x="Latent Factors (dim)", y="RMSE") +
  ggtitle("Latent Factors vs RMSE")
```



```
RECO_CV_100_1000 <- opts_tune


#**********************************************************************************
# Train "Large" - V4 - 1000 x 100
#**********************************************************************************
opts_list = list(dim      = c(1000),    # Number of Latent Features
                 costp_l1 = c(0),       # L1 regularization cost for User factors
                 costp_l2 = c(0.01),    # L2 regularization cost for User factors
                 costq_l1 = c(0.1),     # L1 regularization cost for Beer factors
                 costq_l2 = c(0.1),     # L2 regularization cost for Beer factors
                 lrate    = c(0.1),     # Learning Rate - Aprox step size in Gradient Descent
                 nfold    = 5,          # Number of folds for Cross Validation
                 nthread  = 1,          # Set Number of Threads to 1 *** Required to get Reproducible R
                 niter    = 100,        # Number of Iterations
                 verbose  = FALSE)      # Don't Show Fold Details

start <- Sys.time()

set.seed(1, sample.kind = "Rounding")
r$train(data_memory(train_set[, 1], train_set[, 2], train_set[, 3]),
        opts = c(opts_list,
                 nthread = 1))

finish <- Sys.time()
train_time <- finish - start
```

```
start <- Sys.time()

pred <- r$predict(data_memory(test_set[, 1], test_set[, 2], test_set[, 3]), out_memory())
finish <- Sys.time()

pred_time <- finish - start
pred_time
```

```
## Time difference of 47.0428 mins
```

```
rmse <- RMSE(test_set[,3],pred)

RECO_rmse_results <- bind_rows(RECO_rmse_results,
                               tibble(Method="V3 - 1000 x 100",
                                      RMSE = rmse,
                                      Train_Time = round(as.numeric(train_time, units="mins"), 2)))

kable(RECO_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE	Train_Time
V1 - 20 x 50	0.6454194	0.36
V2 - 16 x 50 (Optimal DIM)	0.6440126	0.27
V3 - 1000 x 100	0.6429790	19.00

```
#*********************************************************************************
```

These results show that a very large **dim** value for Latent Factors still delivers an increase in performance. It is unknown what the limit is for the number of Latent Factors.

Salford Predictive Modeler v8.0 - CART, MARS, and a Whole Lot of Other Models

Salford Systems, recently acquired by MiniTab, has produced the Predictive Modeling program and enjoyed very impressive results in many Data Science and Machine Learning comptetitions. It isn't R, but it also doesn't have any problem modeling large data sets. Out of curiosity, I decided to train some models and compare the results.

You can download a trial version of the software from their website… they want to charge $13K for the software, but given the results, it doesn't appear to be worth the money these days.

```
SPM_rmse_results <- tibble(Method="Random Forests:",
                           RMSE = 0.62616)

SPM_rmse_results <- bind_rows(SPM_rmse_results,
                              tibble(Method="CART - Ensemble:",
                                     RMSE = 0.62202))

SPM_rmse_results <- bind_rows(SPM_rmse_results,
                              tibble(Method="CART:",
                                     RMSE = 0.62089))

SPM_rmse_results <- bind_rows(SPM_rmse_results,
                              tibble(Method="MARS:",
                                     RMSE = 0.61482))
```

```r
SPM_rmse_results <- bind_rows(SPM_rmse_results,
                              tibble(Method="Regression:",
                                     RMSE = 0.61239))

SPM_rmse_results <- bind_rows(SPM_rmse_results,
                              tibble(Method="GPS - Generalized Lasso:",
                                     RMSE = 0.59585))

kable(SPM_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE
Random Forests:	0.62616
CART - Ensemble:	0.62202
CART:	0.62089
MARS:	0.61482
Regression:	0.61239
GPS - Generalized Lasso:	0.59585

Results

Below are the tabulated results of all methods.

- Standard & Regularized Linear Regression

```r
kable(LR_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE
LR: Base Mean Model	0.7159839
Reg LR: Mean + Beer Effect Model	0.6202860
Reg LR: Mean + Beer + User Effect Model	0.6011899
Reg LR: Mean + Beer + User + Style Effect Model	0.6006035
Reg LR: Mean + Beer + User + Style + Brewery Effect Model	0.5992131

- XGB – Extreme Parallel Tree Boosting (xgBoost)

```r
kable(XGB_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE	Train_Time
XGB Linear 50 Boost Rnds	0.6547372	4.52 secs
XGB Linear 1000 Boost Rnds	0.6178744	81.42 secs
XGB Mixed Tree 5 Boost Rnds	0.6152551	212.51 secs

- RECO - Recommender w/ Matrix Factorization (recosystem)

```r
kable(RECO_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE	Train_Time
V1 - 20 x 50	0.6454194	0.36
V2 - 16 x 50 (Optimal DIM)	0.6440126	0.27
V3 - 1000 x 100	0.6429790	19.00

- Salford Predictive Modeler v8.0

```r
kable(SPM_rmse_results) %>%
  kable_styling(full_width = FALSE, position = "left")
```

Method	RMSE
Random Forests:	0.62616
CART - Ensemble:	0.62202
CART:	0.62089
MARS:	0.61482
Regression:	0.61239
GPS - Generalized Lasso:	0.59585

Conclusion

This Capstone Individual project has applied four different techniques toward the development of a Beer Recommendation algorithm.

It is interesting to see that the Regularized Linear Regression performed better than both xgBoost and recosystem. At the end of the day, Salford won out, but it was close.