# Movie Recommender System

*Christopher Wheatley*

17 September 2022

# Contents

# 1 Executive Summary

This report satisfies the first capstone project of two; within the Data Science program provided by edX and HarvardX. The primary objective being to generate a movie rating recommendation system through data; analysis, visualization, hypothesis generation, optimization and testing. As such; we have proven that given a large enough data set of movie reviews with variables for movie rating, movie genre, user identification, movie identification. One is able to build the following machine learning algorithm which is able to predict a rating for movie i and user u; to a Root Mean Squared Error (RMSE) / accuracy of 0.8704178.

$$y_{hat} = \mu_{ratings} + (\frac{bias_{genres}}{(n() + \lambda)}) + (\frac{bias_{userId}}{(n() + \lambda)}) + (\frac{bias_{movieId}}{(n() + \lambda)})$$

# 2 Initial Setup | Verification

*Setup:*

Fortunately, the Harvard X Data Science capstone course has provided the necessary code to initialize both the training and validation data sets.

*Verification:*

Let's verify the initialization was a success; analyzing the following data frames:

- 'edx'; which will be our *training* data set.
- 'validation'; which will be our *test* data set.

The dimensions of the 'edx'/training set; details a data.table of over 9 million observations [m]; with 6 variables [n] associated with each observation. Let's assess the completeness of this data, looking for missing values in each column/variable.

| rows | variables |
|---|---|
| 9000055 | 6 |

| variable | class | first_values |
|---|---|---|
| userId | integer | 1, 1, 1 |
| movieId | double | 122, 185, 292 |
| rating | double | 5, 5, 5 |
| timestamp | integer | 838985046, 838983525, 838983421 |
| title | character | Boomerang (1992), Net, The (1995), Outbreak (1995) |
| genres | character | Comedy\|Romance, Action\|Crime\|Thriller, Action\|Drama\|Sci-Fi\|Thriller |

*Checking for missing Data:*

*Training data*

I like to utilize a for loop across each variable looking for missing data. See code below.

```
#example method to assess NAs with for loop.

for (i in 1:ncol(edx)) {
  na <- sum(is.na(edx[,..i]))
  print(na)
}
```

```
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
```

All 6 variables have zero missing values in the training set, lets evaluate our 'validation' (test) set the same way.

*Test / Validation Data*

The dimensions of the 'validation' / test set; details a data.table of just under 1 million observations [m]; with 6 variables [n] associated with each observation. The data.table has identified Nil missing values.

| rows | variables |
|---|---|
| 999999 | 6 |

| variable | class | first_values |
|---|---|---|
| userId | integer | 1, 1, 1 |
| movieId | double | 231, 480, 586 |
| rating | double | 5, 5, 5 |
| timestamp | integer | 838983392, 838983653, 838984068 |
| title | character | Dumb & Dumber (1994), Jurassic Park (1993), Home Alone (1990) |
| genres | character | Comedy, Action\|Adventure\|Sci-Fi\|Thriller, Children\|Comedy |

```
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
```

# 3 Objective | Strategy

*Objective:*

Develop and test a movie recommendation model able to predict ratings of movie [i] for user[u]. With accuracy measured through a RMSE $< 0.86490$.

*Strategy:*

First examine the relationship between the following variables: 'ratings' [dependent variable] and; 'userId', 'movieId' and 'genres' [independent variables]. The reasoning for choosing only these variables; is for brevity and based off an intuition that in a significant volume of reviews. Each user movie and genre should detail a generalized bias relative to the mean rating. Thus, given new observations with the same independent variables a model can be formed to compute [add the bias terms to a mean] and return a probabilistic estimate of a rating [y_hat].

# 4 Variable Analysis | Visualization

*Variable: rating*

Frequency distribution of ratings.

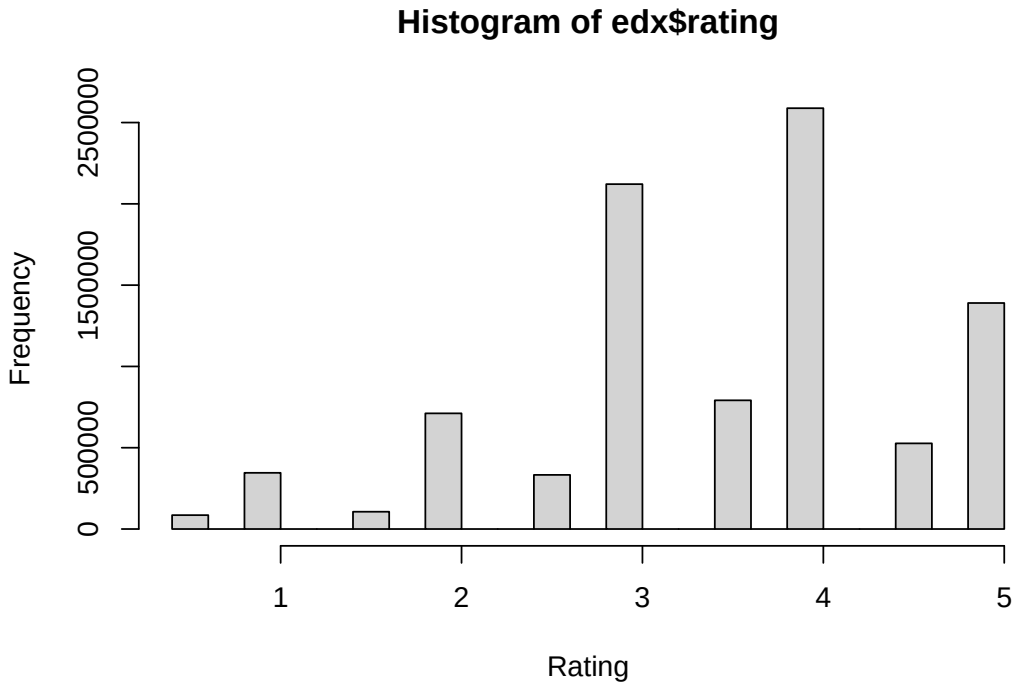


Figure 1: Histogram of Ratings

*Ratings histogram tabulated:*

| rating | count | proportion |
|---|---|---|
| 4.0 | 2588430 | 0.29 |
| 3.0 | 2121240 | 0.24 |
| 5.0 | 1390114 | 0.15 |
| 3.5 | 791624 | 0.09 |
| 2.0 | 711422 | 0.08 |
| 4.5 | 526736 | 0.06 |
| 1.0 | 345679 | 0.04 |
| 2.5 | 333010 | 0.04 |
| 1.5 | 106426 | 0.01 |
| 0.5 | 85374 | 0.01 |

To summarize what we have found with the 'rating' variable. The mode is 4 and mean is 3.5124652. Ratings between whole numbers are less frequent then their whole number equivalent. This variable can be utilized in a supervised learning model to provide real outcomes for training a hypothesis. As such, it may be necessary to convert this variable into a factor or category for optimization purposes.

---

*Variable: userId*

Magnitude of unique users.

| N_UserTrain | N_UserTest | Delta |
|---|---|---|
| 69878 | 68534 | 1344 |

The table above depicts the number of unique users in both the training data set [69978] and the test data set [68534]. Also highlighting the difference [1344] between each.

*Note:*
This apparent difference, may cause a problem later in system design/utilization, if we constrain the algorithm to only accept user data evident within the training set. If this is to be the case, our model will be limited to known parameters and could produce NAs. As such and if required, we will replace missing values with a mean bias value for each parameter. I.e.

*f(x) = ifelse(user bias missing, then replace with mean(all$userBias), else leave value)*

Figure 2 - depicts a histogram of reviews from users.

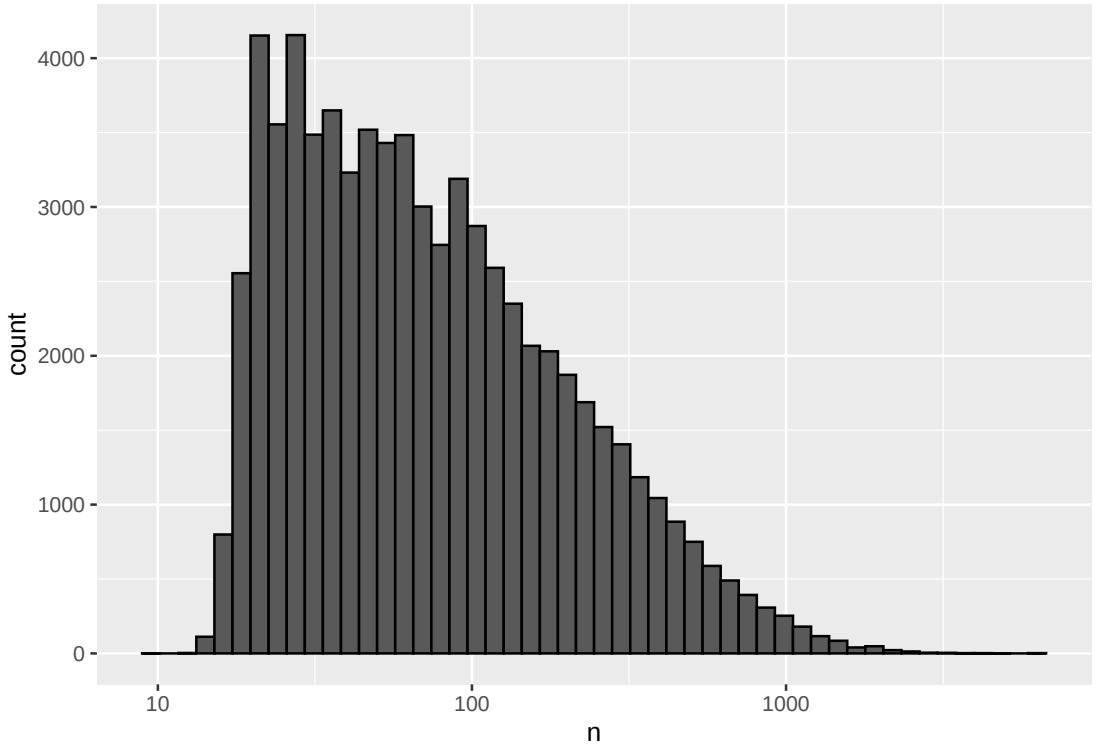


Figure 2: Histogram UserId

The above right skewed plot highlights 'outlier' users, at the higher and lower end of the number of reviews. Regularization may be useful to penalize predictions from users with the largest variance from the mean.

---

*Variable: movieId*

Magnitude of unique movies.

| N_MoviesTrain | N_MoviesTest | Delta |
|---|---|---|
| 10677 | 9809 | 868 |

The table above depicts the number of unique movies in both the training data set [10677] and the test data set [9809]. Also highlighting the difference [868] between each.

Figure 3 - depicts a histogram of the movieId variable:

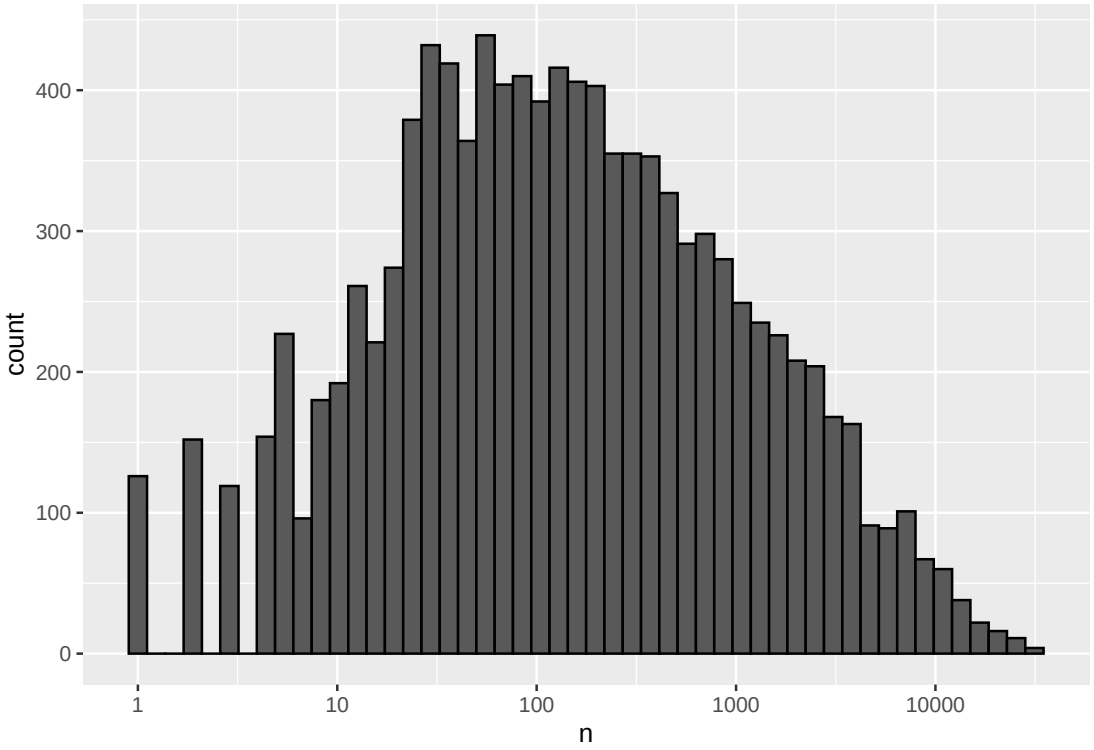


Figure 3: Histogram movieId

Similar to userId, movieId contains outliers which may also need to be regularized.

---

*Variable: genres*

Magnitude of unique genres.

| genres |
|---|
| 797 |

For curiosities sake, let's look at the genres with the most reviews.

| genres | count |
|---|---|
| Drama | 733296 |
| Comedy | 700889 |
| Comedy\|Romance | 365468 |
| Comedy\|Drama | 323637 |
| Comedy\|Drama\|Romance | 261425 |
| Drama\|Romance | 259355 |
| Action\|Adventure\|Sci-Fi | 219938 |
| Action\|Adventure\|Thriller | 149091 |
| Drama\|Thriller | 145373 |
| Crime\|Drama | 137387 |

Analyzing the 'genres' variable depicts 797 unique categories within the 'edx' data-set. People love Drama...

# 5 Hypothesis | Method

A Naive Bayes method will be utilized to form the model hypothesis. This approach starts with the mean rating for all training reviews and adds bias terms in an iterative fashion, assessing with each addition the accuracy of the model. As stated in the objective, accuracy will be measured through RMSE calculations between a training set and one cross validation set. Regularization will be applied where necessary.

$$y_{hat} = \mu_{ratings} + bias_{term1} + bias_{term2} + ..n$$

### 5.0.1 Stratify Data

As the validation data set is restricted to the final model accuracy assessment. I will split the training data into a training data-set and a cross-validation (CV) data-set. [.9 | .1 respectively]

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

index <- createDataPartition(y = edx$rating, times = 1, p = .1, list = FALSE)

train <- edx[-index,]

temp <- edx[index,]

#To avoid our model producing "NA"s;
#we must ensure the same categorical data is both in the training and cross-validation data-sets.

cv <- temp %>%
  semi_join(train, by = "movieId") %>%
  semi_join(train, by = "userId") %>%
  semi_join(train, by = "genres")

removed <- anti_join(temp, cv)

train <- rbind(train, removed)

rm(removed, temp, index)
```

### 5.0.2 Iteration 1 - Mode and Mean Prediction

```
#Initial model utilizing mode only.

modeOnly <- RMSE(cv$rating, modeTrain)

#Initial model utilizing mean only.

meanOnly <- RMSE(cv$rating, mean(train$rating))

results <- tibble(Model = c("Mode","Mean"),
       RMSE = c(modeOnly, meanOnly))

results
```

| Model | RMSE |
|---|---|
| Mode | 1.166756 |
| Mean | 1.060054 |

```
#Mean looks to be a more accurate constant for prediction.
```

### 5.0.3 Iteration 2 - Adding Genre.

Let's add a bias term for Genre as depicted below.

$$y_{hat} = \mu_{ratings} + bias_{genres}$$

```
#Mean + Genre

muTrain <- mean(train$rating)

genre_effect <- train %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - muTrain))

pred_1 <- cv %>%
  left_join(genre_effect, by = "genres") %>%
  mutate(y_hat = muTrain + b_g)

muPlusGenre <- RMSE(pred_1$y_hat,cv$rating)

results <- results %>% add_row(Model = "MeanPlusGenre",
                               RMSE = muPlusGenre)

results
```

| Model | RMSE |
|---|---|
| Mode | 1.166756 |
| Mean | 1.060054 |
| MeanPlusGenre | 1.017501 |

With the second iteration - our model has increased accuracy.

### 5.0.4 Iteration 3 - Adding userId.

Now let's add another bias term - userId.

$$y_{hat} = \mu_{ratings} + bias_{genres} + bias_{userId}$$

```
#Mean + Genre + userId

user_effect <- train %>%
  left_join(genre_effect, by = "genres") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - muTrain - b_g))

pred_2 <- cv %>%
  left_join(genre_effect, by = "genres") %>%
  left_join(user_effect, by = "userId") %>%
  mutate(y_hat = muTrain + b_g + b_u)

results <- results %>% add_row(Model = "MeanPlusGenre_PlusUser",
                               RMSE = RMSE(pred_2$y_hat, cv$rating))

results
```

| Model | RMSE |
|---|---|
| Mode | 1.1667562 |
| Mean | 1.0600537 |
| MeanPlusGenre | 1.0175012 |
| MeanPlusGenre_PlusUser | 0.9394804 |

### 5.0.5 Iteration 4 - Adding movieId.

Now let's add another bias term - movieId

$$y_{hat} = \mu_{ratings} + bias_{genres} + bias_{userId} + bias_{movieId}$$

```
#Mean + Genre + userId + movieId

movie_effect <- train %>%
  left_join(genre_effect, by = "genres") %>%
  left_join(user_effect, by = "userId") %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - muTrain - b_g - b_u))

pred_3 <- cv %>%
  left_join(genre_effect, by = "genres") %>%
  left_join(user_effect, by = "userId") %>%
  left_join(movie_effect, by = "movieId") %>%
  mutate(y_hat = muTrain + b_g + b_u + b_i)

results <- results %>% add_row(Model = "MeanPlusGenre_PlusUser_PlusMovie",
                               RMSE = RMSE(pred_3$y_hat, cv$rating))

results
```

| Model | RMSE |
|---|---|
| Mode | 1.1667562 |
| Mean | 1.0600537 |
| MeanPlusGenre | 1.0175012 |
| MeanPlusGenre_PlusUser | 0.9394804 |
| MeanPlusGenre_PlusUser_PlusMovie | 0.8705983 |

The model is becoming more accurate, however we are still a ways off from the performance objective. Let's add regularization into the mix.

#### 5.0.6 Iteration 5 - Adding regularization

As highlighted through the analysis and visualization phase. Certain parameter values have significantly less frequent observations, and since we are trying to generalize an average prediction for each dimension. These outlier values can add unwanted variability into our predictions. As such we will add the penalty term 'Lambda' ($\lambda$) to our hypothesis, reducing variability for each bias term.

$$y_{hat} = \mu_{ratings} + (\frac{bias_{genres}}{(n+\lambda)}) + (\frac{bias_{userId}}{(n+\lambda)}) + (\frac{bias_{movieId}}{(n+\lambda)})$$

To optimize our hypothesis and select the most accurate value for $\lambda$, we will iterate values from 4 to 10; in increments of .25. Plotting our results and selecting the $\lambda$ value which returns the minimum RMSE.

```
lambdaList <- seq(4, 10, .25)

rmseFinal <- sapply(lambdaList, function(l){

  mu <- mean(train)

  genre_effect <- train %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - muTrain)/(n() + l))

  user_effect <- train %>%
  left_join(genre_effect, by = "genres") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - muTrain - b_g)/(n() + l))

  movie_effect <- train %>%
  left_join(genre_effect, by = "genres") %>%
  left_join(user_effect, by = "userId") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - muTrain - b_g - b_u)/(n() + l))

  predFinal <- cv %>%
  left_join(genre_effect, by = "genres") %>%
  left_join(user_effect, by = "userId") %>%
  left_join(movie_effect, by = "movieId") %>%
  mutate(y_hat = muTrain + b_g + b_u + b_i) %>%
  pull(y_hat)

  return(RMSE(predFinal, cv$rating))

})
```

*Optimization Plot*

Looking at the below plot, Lambda set to 7.5, produces the highest performing model. With a RMSE score of 0.8697088.

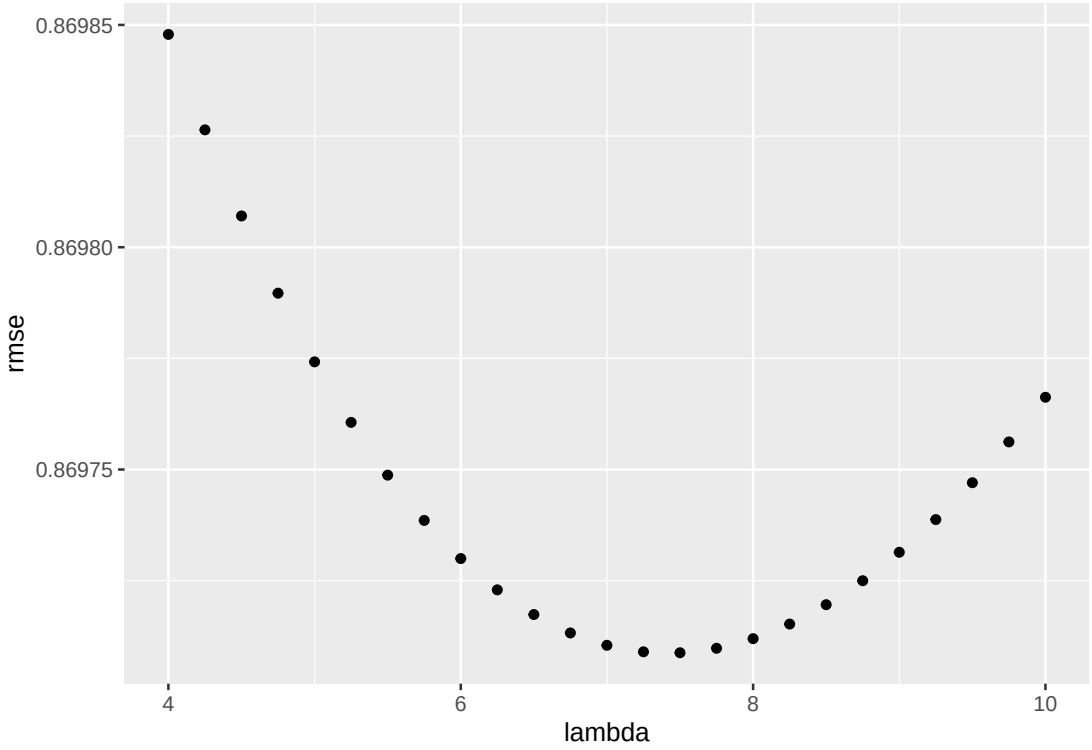


Figure 4: Optimization Plot - Lambda Values

| Model | RMSE |
|---|---|
| Mode | 1.1667562 |
| Mean | 1.0600537 |
| MeanPlusGenre | 1.0175012 |
| MeanPlusGenre_PlusUser | 0.9394804 |
| MeanPlusGenre_PlusUser_PlusMovie | 0.8705983 |
| MeanPlusGenre_PlusUser_PlusMovie_Regularized | 0.8697088 |

We haven't quite met the accuracy objective of $< 0.86490$. Delta $= 0.0048088$.

# 6 Hypothesis Testing

As the submission deadline is getting closer, I will implement our last iteration hypothesis on the validation data set and submit the project. Given more time; I would continue to add and evaluate new bias terms.

| Model | RMSE |
|---|---|
| Mode | 1.1667562 |
| Mean | 1.0600537 |
| MeanPlusGenre | 1.0175012 |
| MeanPlusGenre_PlusUser | 0.9394804 |
| MeanPlusGenre_PlusUser_PlusMovie | 0.8705983 |
| MeanPlusGenre_PlusUser_PlusMovie_Regularized | 0.8697088 |
| Validation | 0.8704178 |

# 7 Result

Our hypothesis has achieved a 'test' RMSE accuracy of: RMSE = 0.8704178.

# 8 Conclusion

It's nice to think back and appreciate how far I have come, this has been a very challenging yet rewarding program and project. Given more time, I would have liked to add additional bias terms or dimensions to the hypothesis. And given more computational power, a matrix factorization approach would be an interesting and rewarding alternative approach. Thankyou Edx and HarvardX.