HarvardX PH125.9x Data Science Capstone

Gideon Vos

13 January 2019

1. Executive Summary

Background and Motivation

A recommender system or a recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. In this project the items are movies.

Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. There are also recommender systems for experts collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners (online dating), and Twitter page. Major companies such as Amazon, Netflix and Spotify utilize recommendation systems. A strong recommendation system was of such importance that in 2006, Netflix offered a million dollar prize to anyone who could improve the effectiveness of its recommendation system by 10%.

It should be noted that the winning Netflix model utilized an ensemble of very complex models, and the team spent several months perfecting the ensemble. While they won the first prize, no mention is made that can be publicly found as to the level of predictive accuracy, as their goal was not to predict ratings but merely recommend movies likely to be enjoyed by a user. Thus the Netflix problem and our own challenge is very much different in its goals.

DataSet

For this project a movie rating predictor is created using the 'MovieLens' dataset. This data set can be found and downloaded here:

- [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/
- [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip

Goal

The goal is to train a machine learning algorithm using the inputs of a provided training subset to predict movie ratings in a validation set.

The focus is on the predictive accuracy of the algorithm, which is in contrast with previous Kaggle competitions where RMSE or MAE were used as benchmarks. During analysis we will review RMSE as a guide, while using pure accuracy as the decisive factor on whether to proceed with an algorithm or not.

Data Loading and Setup

We will utilize and load several packages from CRAN to assist with our analysis. These will be automatically downloaded and installed during code execution. As per the project guidelines, the dataset will first be split into a training and validation set (10%), and the training set will then be further split into a train/test set with the test set being 10% of the training set.

2. Methods and Analysis

Exploratory Analysis

A review of the edx dataset shows 6 columns. The timestamp needs to be converted if used, and release year will need to be split from the title if to be used for prediction. Genres is a single pipe-delimited string containing the various genre categories a movie might be categorized under, and this will need to be split out if it affects rating outcome.

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1                 Comedy|Romance
## 2           Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5         Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7         Children|Comedy|Fantasy
```

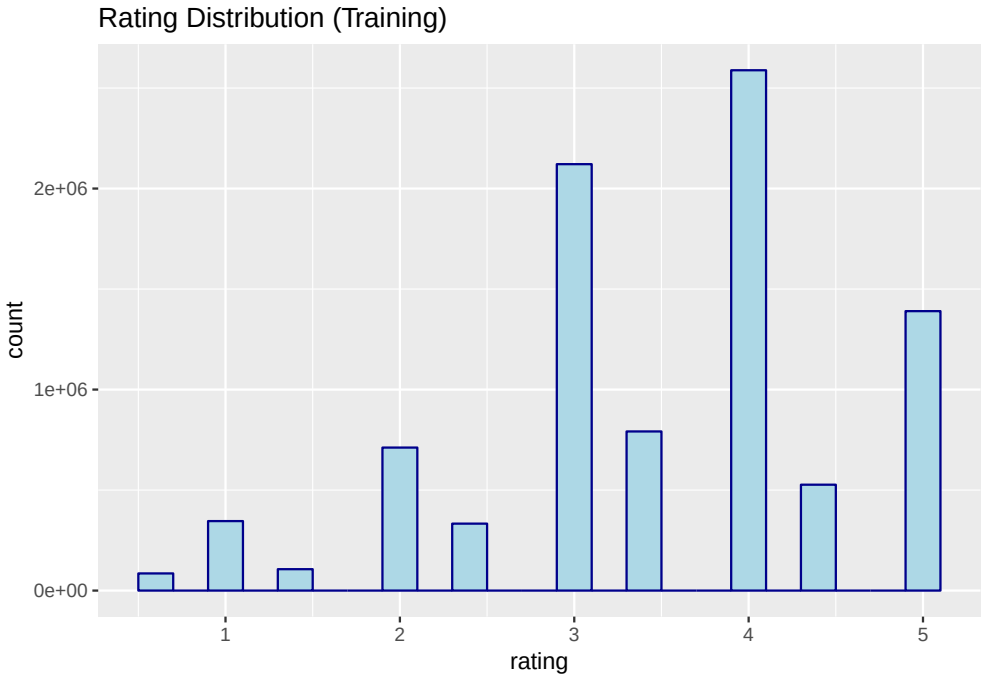
There are no missing values. Let's review a summary of the dataset.

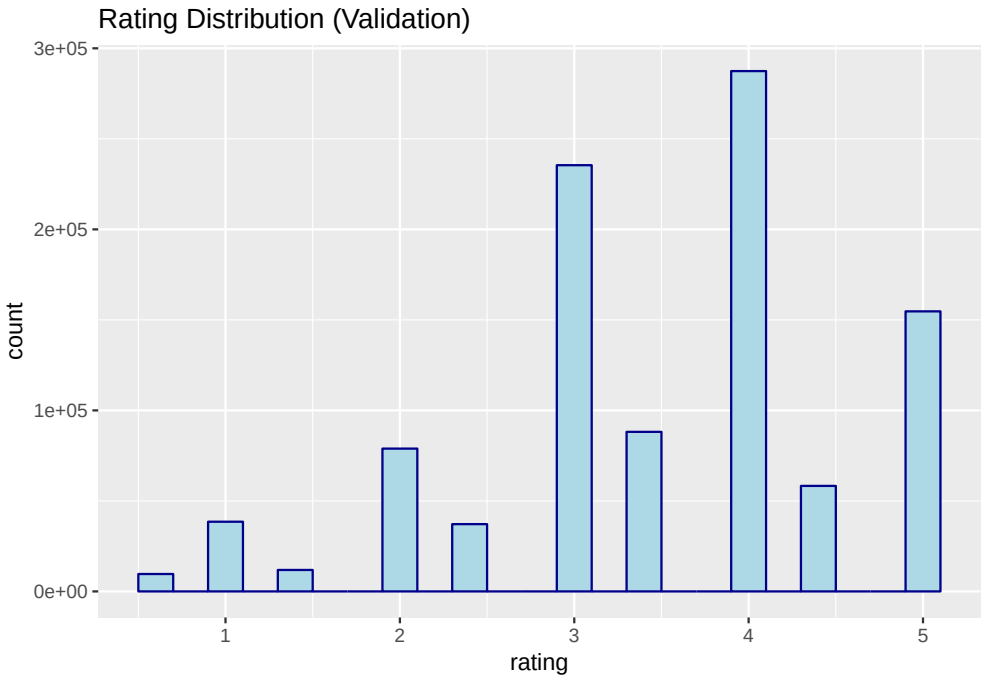
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Our dataset contains ~70000 unique users giving ratings to ~ 10700 different movies.

There are 10 different rating scores, lowest is 0.5 and highest is 5.

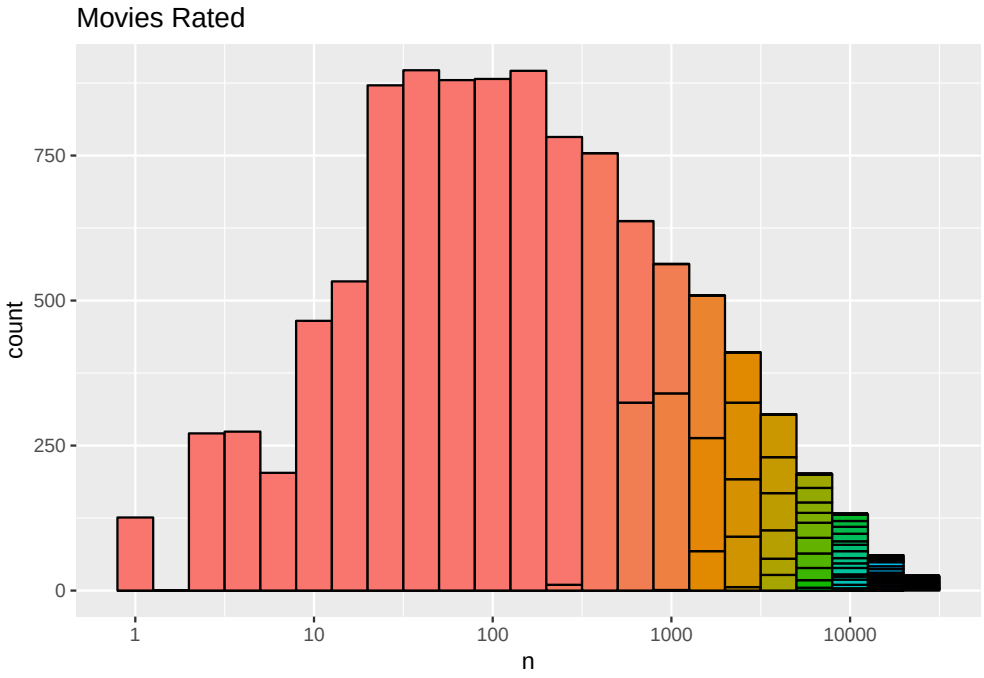
Let's take a look at the distribution of ratings between the training and validation set.



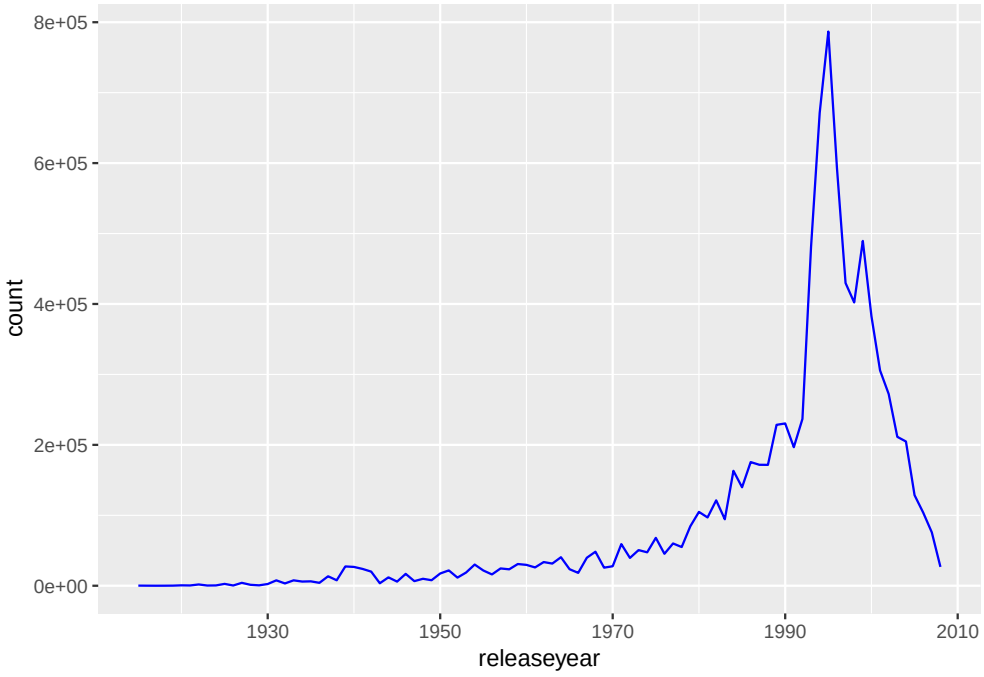


Both have very similar distributions.

We can plot the data and determine that some movies are rated more often than others.



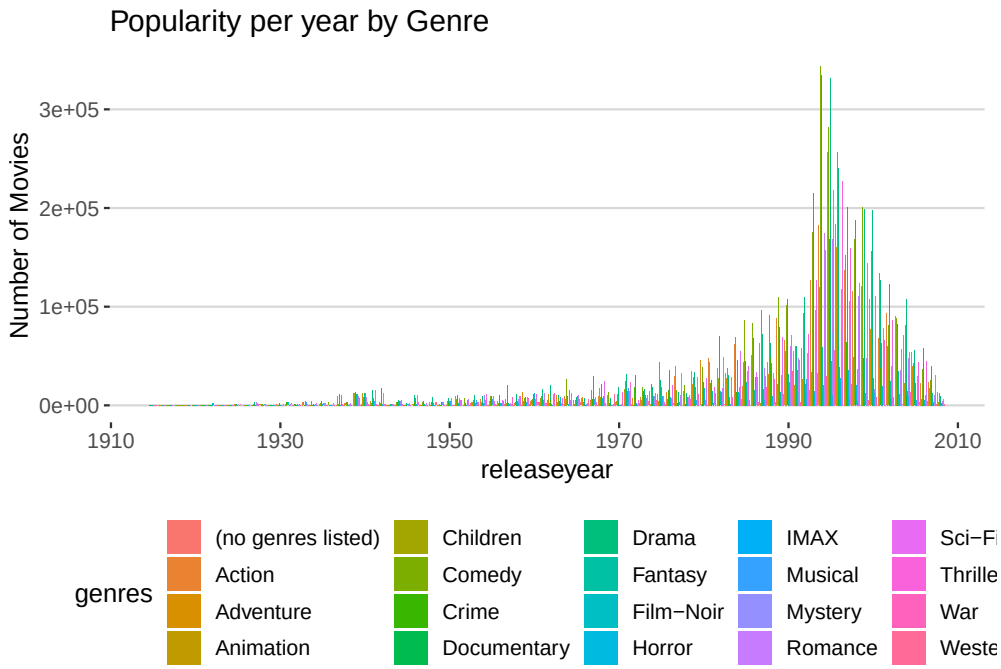
Let's review how many movies were produced over the last 80 years.



We can see an exponential growth of the movie business and a sudden drop in 2010.

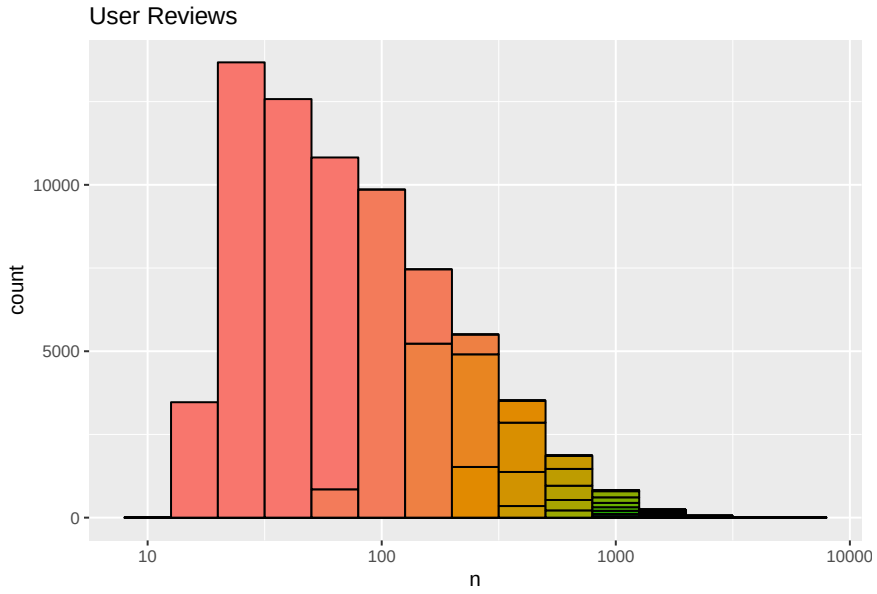
The latter is caused by the fact that the data is collected until October 2009 so we don't have the full data for that period.

We also note that different periods show certain genres being more popular during those periods.



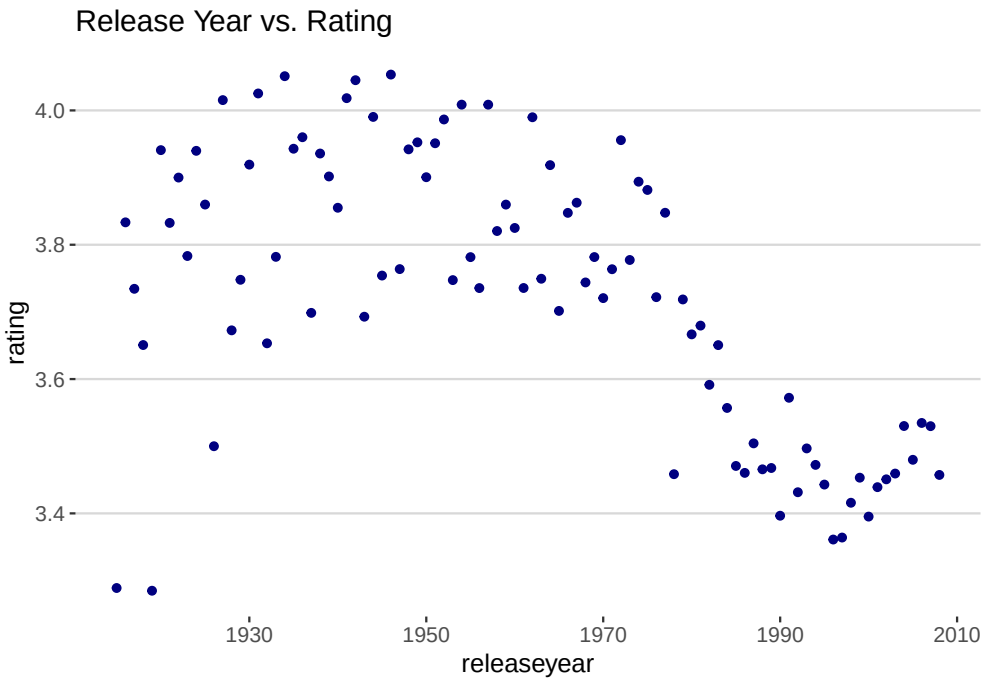
It will be very hard to incorporate genre into overall prediction given this fact.

Let's review the number of times each user has reviewed movies.



It seems most users have reviewed less than 200 movies.

Finally let's plot the release year vs rating.



Movies released prior to 1980 appear to get higher average ratings. This could allow us to penalize a movie based on release year by a calculated weight.

Model Building and Training

Naive Models

We start by writing a loss-function that computes the Residual Mean Squared Error ("typical error") as our measure of accuracy. The value is the typical error in star rating we would make.

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

We predict a new rating to be the average rating of all movies in our training dataset, which gives us a baseline RMSE. We observe that the mean movie rating is a pretty generous > 3.5.

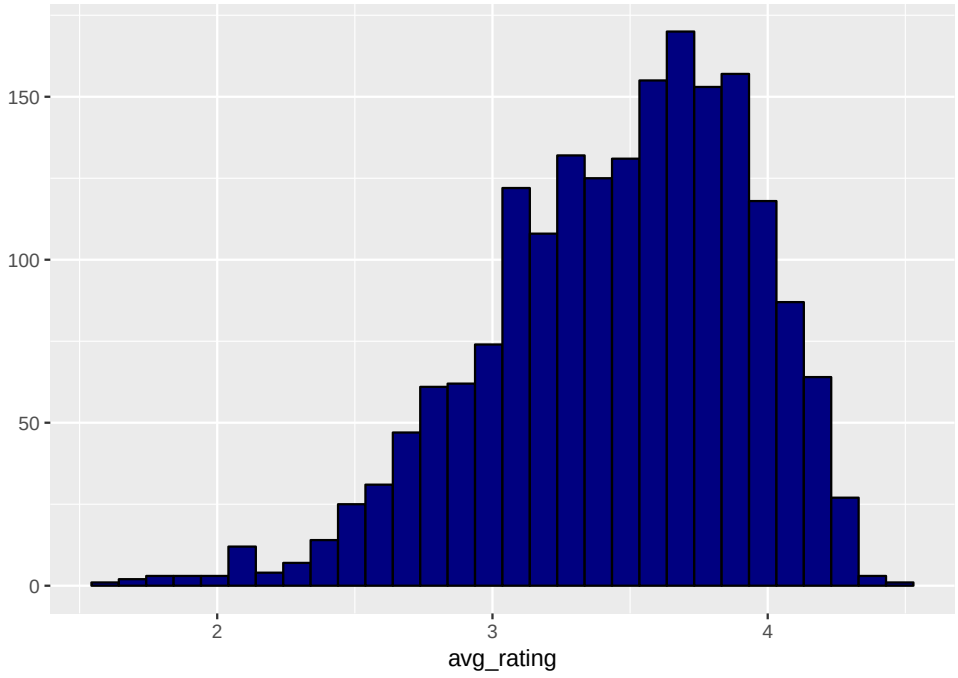
```
mu <- mean(edx$rating)
baseline_RMSE <- RMSE(edx$rating, mu)
```

We can now use this baseline model to predict against our test set and evaluate the resulting RMSE:

```
naive_rmse <- RMSE(temp$rating, mu)
naive_rmse
```

```
## [1] 1.061205
```

We know from experience that some movies are just generally rated higher than others. We can use data to confirm this. For example, if we consider movies with more than 1,000 ratings, the SE error for the average is at most 0.05. Yet plotting these averages we see much greater variability than 0.05:



So our intuition that different movies are rated differently is confirmed by data. We can augment our previous model by adding a term to represent average ranking for a movie.

Now we can form a prediction

```
model2_rmse <- RMSE(predicted_ratings, temp$rating)
model2_rmse
```

```
## [1] 0.9439049
```

We already see a big improvement. Can we make it better? Let's explore where we made mistakes.

```
##                              title prediction  residual
## 1             Pokémon Heroes (2003)   1.029197 -3.970803
## 2  Shawshank Redemption, The (1994)   4.455131  3.955131
## 3  Shawshank Redemption, The (1994)   4.455131  3.955131
## 4  Shawshank Redemption, The (1994)   4.455131  3.955131
## 5             Godfather, The (1972)   4.415366  3.915366
## 6             Godfather, The (1972)   4.415366  3.915366
## 7             Godfather, The (1972)   4.415366  3.915366
## 8        Usual Suspects, The (1995)   4.365854  3.865854
## 9        Usual Suspects, The (1995)   4.365854  3.865854
## 10       Usual Suspects, The (1995)   4.365854  3.865854
```

These all seem like obscure movies. Many of them have large predictions.

Let's look at the top 10 worst and best movies.

Here are the top ten movies:

```
## # A tibble: 10 x 2
##    title                                                    prediction
##    <chr>                                                         <dbl>
##  1 Hellhounds on My Trail (1999)                                     5
##  2 Satan's Tango (Sátántangó) (1994)                                 5
##  3 Shadows of Forgotten Ancestors (1964)                             5
##  4 Fighting Elegy (Kenka erejii) (1966)                              5
##  5 Sun Alley (Sonnenallee) (1999)                                    5
##  6 Blue Light, The (Das Blaue Licht) (1932)                          5
##  7 Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko ~      4.75
##  8 Human Condition II, The (Ningen no joken II) (1959)               4.75
##  9 Human Condition III, The (Ningen no joken III) (1961)             4.75
## 10 Constantine's Sword (2007)                                        4.75
```

Here are the bottom ten:

```
## # A tibble: 10 x 2
##    title                                  prediction
##    <chr>                                       <dbl>
##  1 Besotted (2001)                             0.5
##  2 Hi-Line, The (1999)                         0.5
##  3 Accused (Anklaget) (2005)                   0.5
##  4 Confessions of a Superhero (2007)           0.5
##  5 War of the Worlds 2: The Next Wave (2008)   0.5
##  6 SuperBabies: Baby Geniuses 2 (2004)         0.795
##  7 Hip Hop Witch, Da (2000)                    0.821
##  8 Disaster Movie (2008)                       0.859
##  9 From Justin to Kelly (2003)                 0.902
## 10 Criminals (1996)                            1
```

They all seem to be quite obscure.

Let's look at how often they are rated.

```
## # A tibble: 10 x 3
##    title                                                prediction     n
##    <chr>                                                     <dbl> <int>
##  1 Hellhounds on My Trail (1999)                              5        1
##  2 Satan's Tango (Sátántangó) (1994)                          5        2
##  3 Shadows of Forgotten Ancestors (1964)                      5        1
##  4 Fighting Elegy (Kenka erejii) (1966)                       5        1
##  5 Sun Alley (Sonnenallee) (1999)                             5        1
##  6 Blue Light, The (Das Blaue Licht) (1932)                   5        1
##  7 Who's Singin' Over There? (a.k.a. Who Sings Over There~     4.75     4
##  8 Human Condition II, The (Ningen no joken II) (1959)        4.75     4
##  9 Human Condition III, The (Ningen no joken III) (1961)      4.75     4
## 10 Constantine's Sword (2007)                                 4.75     2
```

```
## # A tibble: 10 x 3
##    title                                    prediction     n
##    <chr>                                         <dbl> <int>
##  1 Besotted (2001)                               0.5       2
##  2 Hi-Line, The (1999)                           0.5       1
##  3 Accused (Anklaget) (2005)                     0.5       1
##  4 Confessions of a Superhero (2007)             0.5       1
##  5 War of the Worlds 2: The Next Wave (2008)     0.5       2
##  6 SuperBabies: Baby Geniuses 2 (2004)           0.795    56
##  7 Hip Hop Witch, Da (2000)                      0.821    14
##  8 Disaster Movie (2008)                         0.859    32
##  9 From Justin to Kelly (2003)                   0.902   199
## 10 Criminals (1996)                              1         2
```

So the supposed "best" and "worst" movies were rated by very few users. These movies were mostly obscure ones. This is because with just a few users, we have more uncertainty. Therefore, larger estimates of b_i, negative or positive, are more likely. These are "noisy" estimates that we should not trust, especially when it comes to prediction. Large errors can increase our RMSE, so we would rather be conservative when not sure.

Regularization permits us to penalize large estimates that come from small sample sizes. It has commonalities with the Bayesian approach that "shrunk" predictions. The general idea is to minimize the sum of squares equation while penalizing for large values of b_i

Let's compute these regularized estimates of b_i using lambda=5. Then, look at the top 10 best and worst movies now.

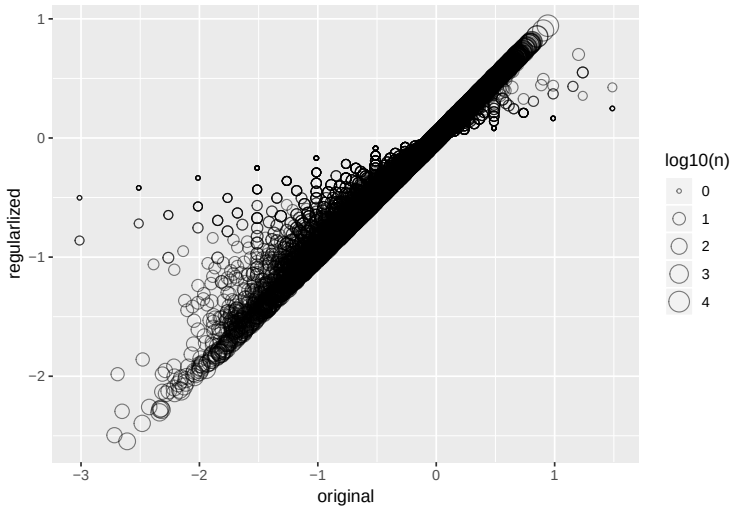
```
## # A tibble: 10 x 3
##    title                                              prediction     n
##    <chr>                                                   <dbl> <int>
##  1 Shawshank Redemption, The (1994)                         4.45 28015
##  2 Godfather, The (1972)                                    4.42 17747
##  3 Usual Suspects, The (1995)                               4.37 21648
##  4 Schindler's List (1993)                                  4.36 23193
##  5 Casablanca (1942)                                        4.32 11232
##  6 Rear Window (1954)                                       4.32  7935
##  7 Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)            4.31  2922
##  8 Third Man, The (1949)                                    4.31  2967
##  9 Double Indemnity (1944)                                  4.31  2154
## 10 Paths of Glory (1957)                                    4.31  1571
```

```
## # A tibble: 10 x 3
##    title                                              prediction     n
##    <chr>                                                   <dbl> <int>
##  1 Shawshank Redemption, The (1994)                         4.45 28015
##  2 Godfather, The (1972)                                    4.42 17747
##  3 Usual Suspects, The (1995)                               4.37 21648
##  4 Schindler's List (1993)                                  4.36 23193
##  5 Casablanca (1942)                                        4.32 11232
##  6 Rear Window (1954)                                       4.32  7935
##  7 Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)            4.31  2922
##  8 Third Man, The (1949)                                    4.31  2967
##  9 Double Indemnity (1944)                                  4.31  2154
## 10 Paths of Glory (1957)                                    4.31  1571
```

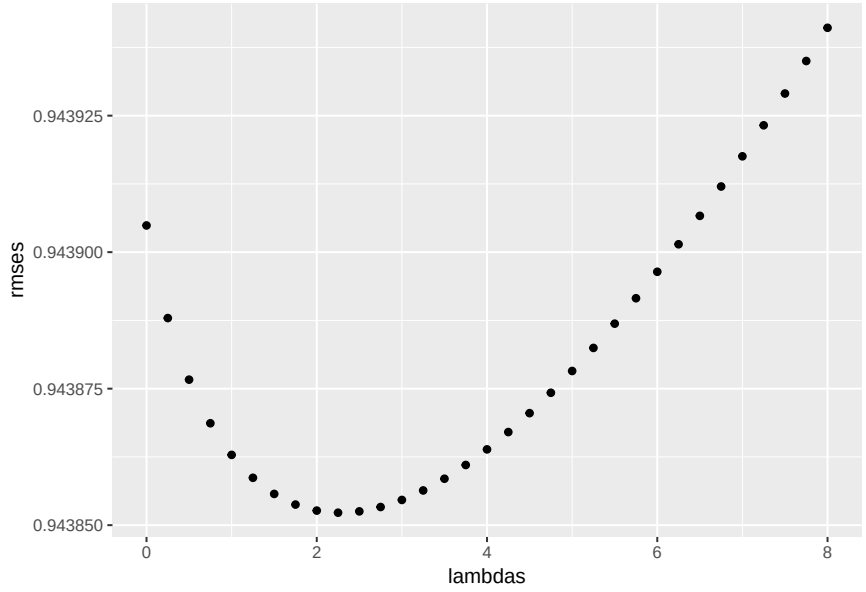
Did we improve our results?

```
## [1] 0.9438782
```

We improved our results slightly. We can visualize how the predictions with a small b_i are shrunken more towards 0.

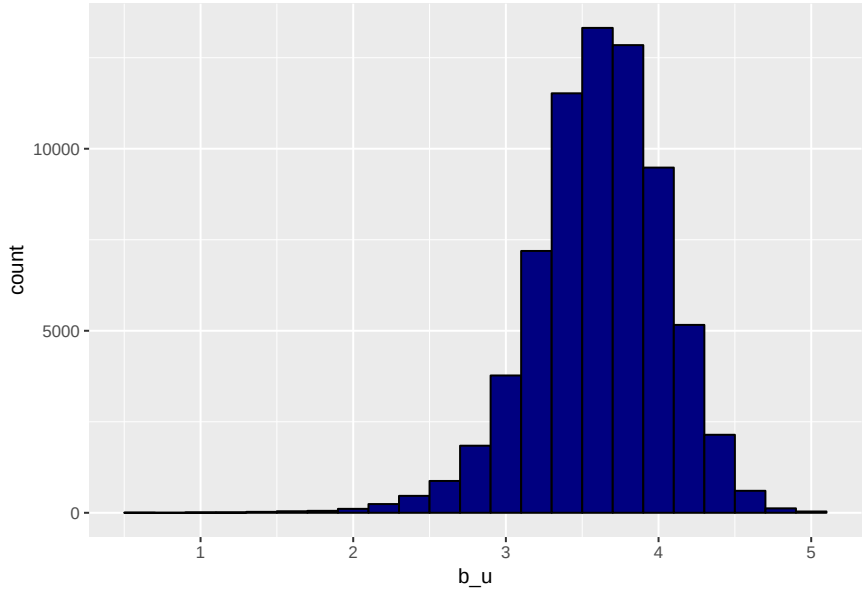


We can try other values of lambda as well:



We find that lambda at 2.3 produces the lowest RMSE.

We have improved the RMSE substantially from our initial naive guess. What else can we do to improve? Let's compute the average rating for user u, for those that have rated over 100 movies.



Note that there is substantial variability across users as well. This means some users are harsher than others and implies that a further improvement can be made to our model. Now it is possible that some users appear to be harsher than others only because they rate under-average movies. For this reason we prefer to estimate b_u taking into account the b_i. The least squares estimates will do this but, again we do not want to use `lm` here.

Instead we will take the average of the residuals. We will use lambda = 2.3:

Note that the RMSE remains the same:

```
## [1] 0.9438782
```

Let's measure the accuracy of our current model:

```
##  Accuracy
## 0.2263704
```

Our accuracy is currently at 22.6% which is much better than a coin-toss, but still far away from the 50% minimum required to score any points.

Other Techniques

Over the last 3 weeks I have built and tested numerous models using different algorithms, including Naive Bayes, SVD, Truncated SVD, Matrix Factorization, Neural Networks using Keras on Tensorflow, Recommenderlab and more. Random Forest on a small subset of 500,000 rows consistently scored above 80% accuracy, but due to resource constraints and limitations built into various R libraries this could not be extended to the full training set of 9 million items.

Various parallel libraries exist that can utilize machine clusters with multiple nodes to spread the training across machines, however for this project audience that would simply not be feasable. All testing was performed on an Azure instance with 20 vCPU's and 160GB RAM, with swap space extending that to over 220GB, yet this was totally consumed by algorithms capable of handling the dataset, while others would consume around 90GB before running into their own internal limitations. One promising library (recosys) uses disk space as virtual RAM and managed to process the full set without fail, yet the accuracy score was no better than the naive model, even after 15 hours of parameter tuning with 10-fold cross-validation.

Slope One Model

Slope One was introduced by Daniel Lemire and Anna Maclachlan in their paper 'Slope One Predictors for Online Rating-Based Collaborative Filtering'. This algorithm is one of the simplest ways to perform collaborative filtering based on items' similarity. This makes it very easy to implement and use, and accuracy of this algorithm equals the accuracy of more complicated and resource-intensive algorithms.

The Slope One method operates with average differences of ratings between each item and makes predictions based on their weighted value.

It is pretty fast, but for a very large dataset it needs a lot of RAM. For that reason we break the training set into 20 smaller sets for training, and merge that into a single model at the end. This process takes around 3 hours on a desktop and requires at least 32GB of RAM and 12GB of extra swap space.

Slope One was chosen as it is an algorithm that can support the splitting of the training set into smaller sets for processing, then re-combined prior to prediction without noticeable loss of accuracy.

Splitting was attempted with Random Forest and it technically worked, however the accuracy dropped below 40% when using the merged model, so was discarded as an option.

The full Slope One source code is included in the accompanying MovieLensProject.R file and not executed as part of this report due to resource constraints. The output of that is shown in the confusion matrix below:

```
Overall Statistics

               Accuracy : 0.8514
                 95% CI : (0.8507, 0.8521)
    No Information Rate : 0.2874
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8192
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0.5 Class: 1 Class: 1.5 Class: 2 Class: 2.5 Class: 3 Class: 3.5
Sensitivity            0.633139  0.63489    0.64546  0.66589    0.69474   0.7162    0.74944
Specificity            1.000000  0.99633    0.98577  0.99544    0.97263   0.9852    0.92672
Pos Pred Value         1.000000  0.87391    0.35242  0.92593    0.49488   0.9370    0.49711
Neg Pred Value         0.996451  0.98554    0.99570  0.97206    0.98803   0.9185    0.97453
Prevalence             0.009614  0.03850    0.01185  0.07888    0.03717   0.2354    0.08814
Detection Rate         0.006087  0.02444    0.00765  0.05253    0.02582   0.1686    0.06606
Detection Prevalence   0.006087  0.02797    0.02171  0.05673    0.05218   0.1800    0.13288
Balanced Accuracy      0.816570  0.81561    0.81562  0.83066    0.83368   0.8507    0.83808
                     Class: 4 Class: 4.5 Class: 5
Sensitivity            0.9992    0.99998   1.0000
Specificity            0.9690    0.99976   1.0000
Pos Pred Value         0.9286    0.99615   1.0000
Neg Pred Value         0.9997    1.00000   1.0000
Prevalence             0.2874    0.05829   0.1547
Detection Rate         0.2872    0.05829   0.1547
Detection Prevalence   0.3093    0.05851   0.1547
Balanced Accuracy      0.9841    0.99987   1.0000
```

3. Results

Accuracy for our naive model peaked at around 22%. Other students improved on that by including genre preferences as a one-hot encoded vector and using other techniques to introduce additional knowledge into the dataset, then applying Naive Bayes for prediction, reaching upwards of 60% in accuracy.

Slope One, using only 3 data elements managed to score an accuracy rate of 85.14% on the validation set, with an RMSE of 0.192 which is a substantial improvement over the naive model. Based on feedback from other Data Scientists using Slope One on the MovieLens 10M dataset, this is exactly as expected.

Accuracy: 85.14% RMSE: 0.192

While it requires more RAM and took several hours to train, the scoring of this challenge is on accuracy, not speed of training or RAM constraints of your machine.

We also avoided having to augment the dataset with additional facts.

4. Conclusion

As stated earlier, I tested Naive Bayes, Random Forest, Tensorflow Neural Networks, PCA, SVD, Recommenderlab, KNN, Kmeans, and various other models and algorithms. Some were fast but the accuracy poor. Others were accurate on smaller sets (Random Forest) but simply could not scale to this data set size, and offered no reliable means to split and combine as I did with Slope One.

While still requiring a lot of RAM, Slope One was the most repeatable. I re-ran this model using training subsets of 10 and 20 splits. The 10 split required 80GB of RAM while to 20 split managed to fit into 32GB + 12GB swap space. A typical machine used for machine learning is often equipped with more resources, especially RAM and multiple GPU's, so our requirements are not above the norm.

When tested, both the 10 and 20 split sets scored exactly the same - 85%, thus proving the splitting and combining approach works well on very large datasets using Slope One without accuracy loss.

In conclusion, utilizing only 3 elements of the data set (user, movie, rating) we were able to predict above 85% accuracy (RMSE: 0.192). Finding the right algorithm that suits the existing data set is often more useful than exploring augmentation of the data to reach your own conclusions, such as including or introducing an artificial bias or weights, as we did with the naive models.

The biggest challenge turned out to be the dataset size, not the prediction problem, and we found a suitable algorithm for our data that could be adapted to break the training set into manageable smaller subsets to be recombined into a single predictor model without noticeable loss in accuracy.

The final submission.csv file can be downloaded here (18MB): (https://github.com/gideonvos/MovieLens/blob/master/submission.csv)