EdX Data Science Capstone Project

JWL

March 24, 2019

Executive Summary

This project's goal was to predict the ratings of movies using information about past ratings of movies by a large number of users. The movielens dataset contained only an identifier for the user, an identifier number for the movie, the movie title, the genre, the precise time of the rating, and the rating the user gave the movie. Each row was one user's rating of one movie. Since there were so few predictors minimal data wrangling and almost no dimension reduction was needed. Variables for the movie and user effect were created using the mean ratings for each movie and each user. The overall mean for every rating was then subtracted and then the movie effect was subtracted from the user effect. Then the results were regularized, using a lambda which was found to minimize root mean square error (RMSE) by penalizing means for small number of ratings by scaling them towards 0. Dummy variables were created from the genres variable for each unique genre. Five genre variables with few values, and thus low variance, were dropped.

The outcome of interest, the movie rating, is semi-continuous and ranged from 0.5 to 5. Since the outcome of interest, the dependent variable, was continuous it meant that certain algorithims, such as regression trees were better fits. Five models were tested, including linear regression, ridge regression, regression trees, random forest, and K nearest neighbors. For each model certain parameters were tuned using the train function of the caret package. Some models, such as KNN, took considerable computation time so to speed testing only 2 fold cross validation was used. Furthermore, to choose the best model quicker, a 100,000 record sample of the full movielense dataset was taken and divided into a training and test set. Ultimately the method which produced the lowest root mean square error was linear regression, which produced an RMSE of 0.68172 using the full dataset. This was a somewhat simple model, with only 16 predictors or independent variables.

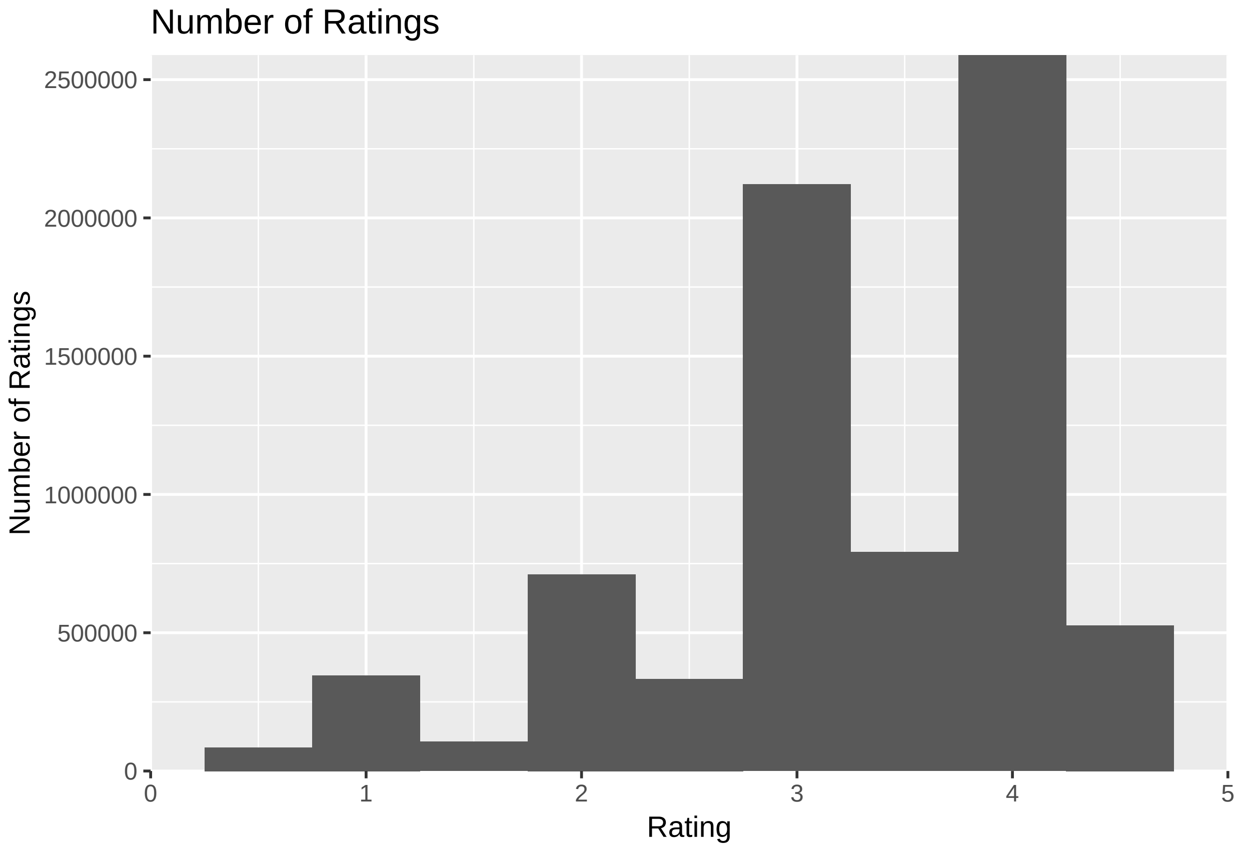
Methods

Then full training dataset contained approximately 9 million records. There were 10677 movies and 69878 users.

The first step was to prepare the dataset for analysis through data wrangling and preprocessing. A good first step is examine the outcome variable of interest. ##Table of the values of rating.

rating	Percent
0.5	0.9
1.0	3.8
1.5	1.2
2.0	7.9
2.5	3.7
3.0	23.6
3.5	8.8
4.0	28.8
4.5	5.9
5.0	15.4

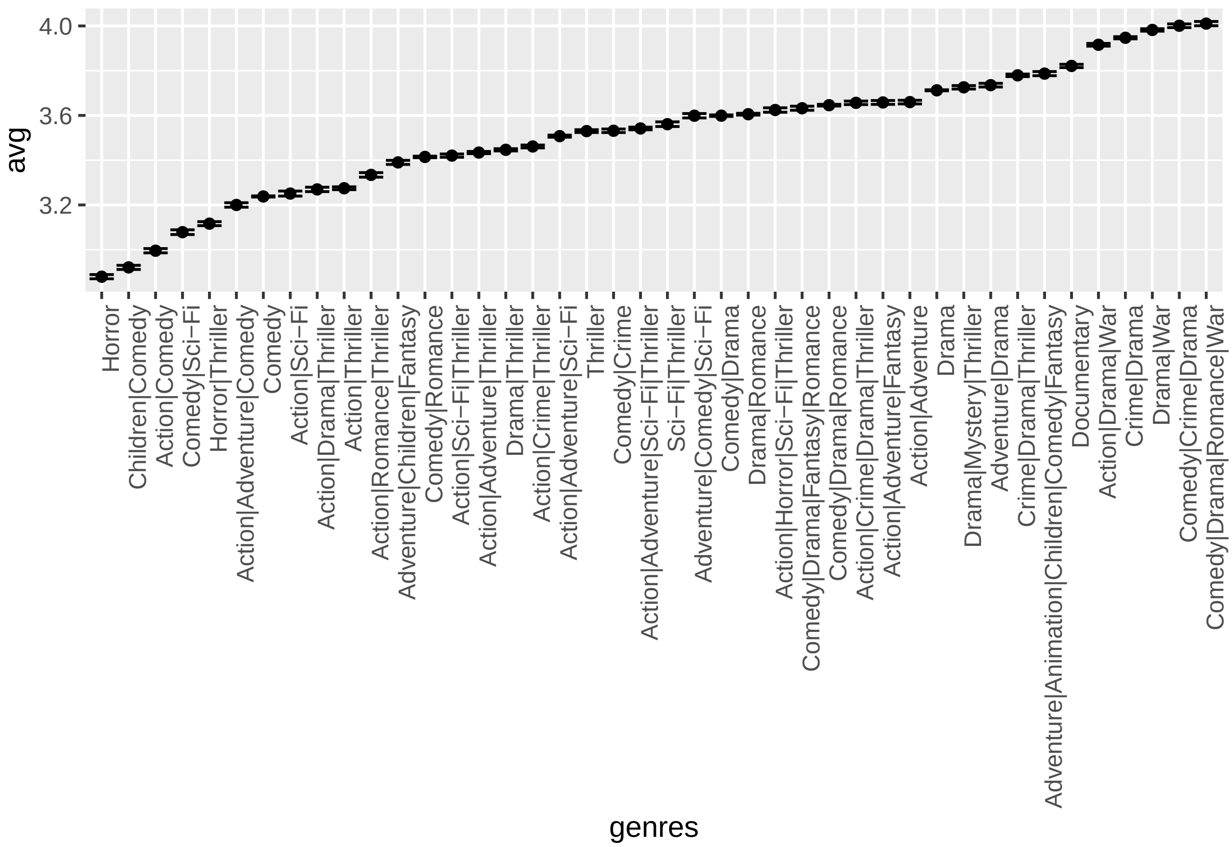
```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

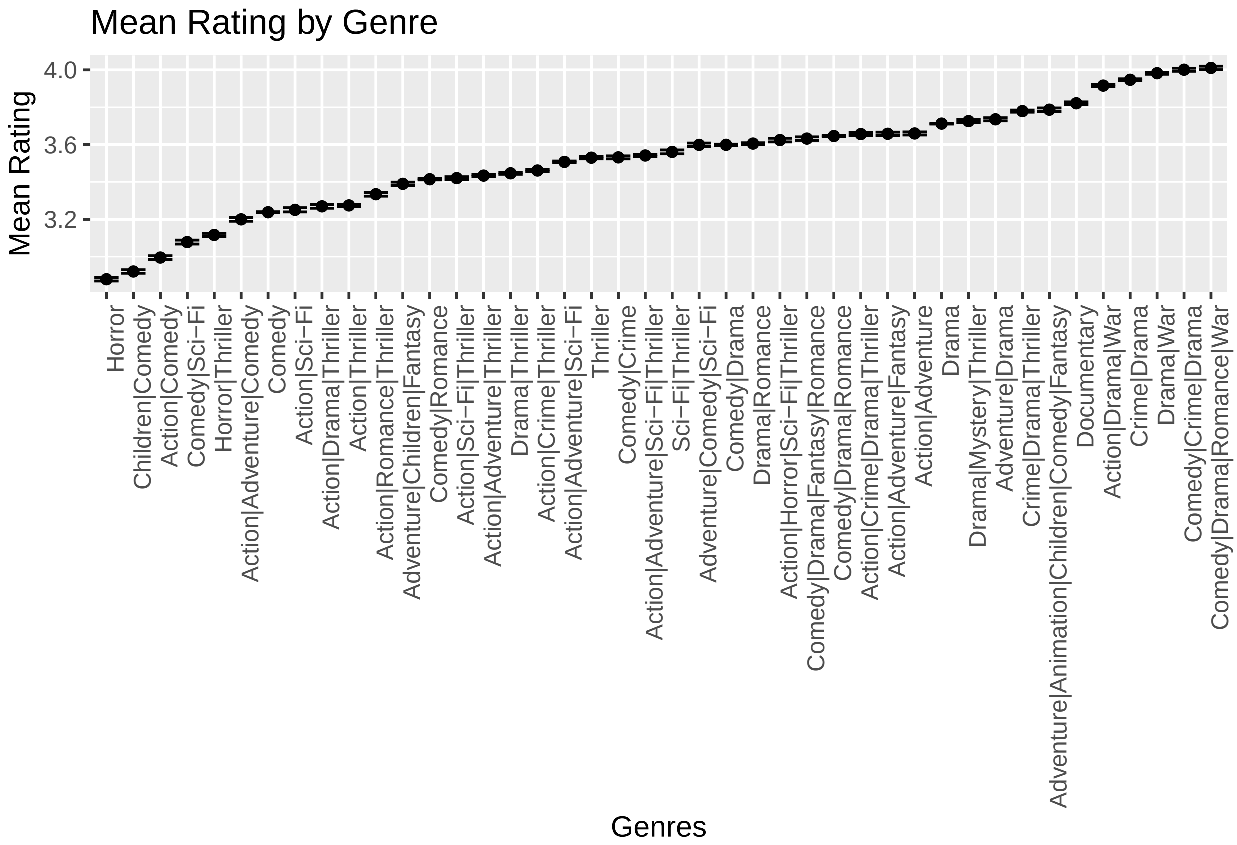


The outcom variable, rating, is semi-continuous and ordinal. The outcome variable does not appear to need processing. Examing the variables in the datast the only variables one would expect to be predictive of future movie ratings were past ratings, the movie's genre, and the time the rating was completed.

Since this dataset has fewer predictors it's not really necessary to engage in extensive dimension reduction. The data was examined for missing values amongst the timestamp, and genres variables and no missing values were found. There were no missing values for the rating variable, the dependent variable, either. I removed variables, such as title, movieId and userId, which have no predictive power on their own, after they were used to create other variables.

A good way to examine the relationship between different variables is to create simple graphs. I examined whether their might be a relationship between genre and rating using a graph. Since there are many different genres, to make the graph readable only genres with at leats 40,000 ratings were kept.

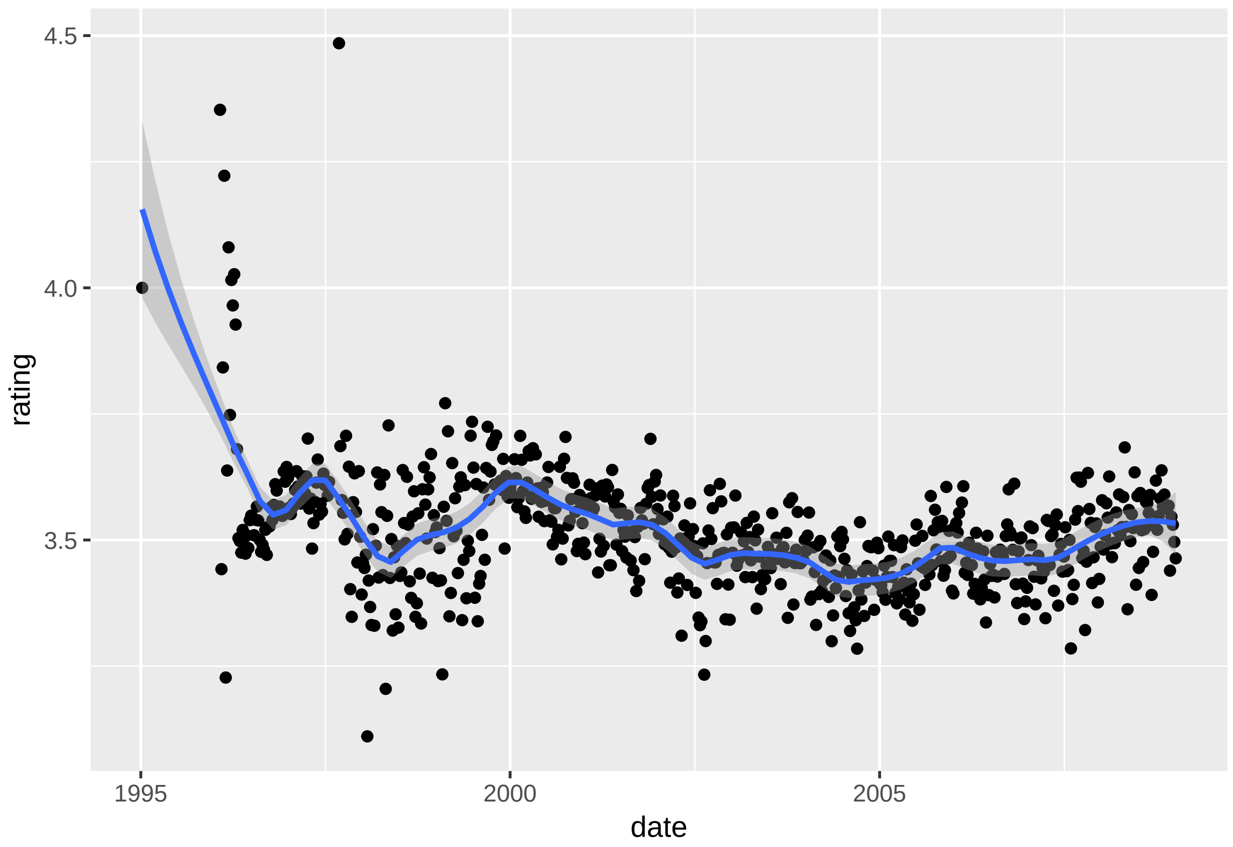

avg
4.0
3.6
3.2
genres
Horror
Children|Comedy
Action|Comedy
Comedy|Sci−Fi
Horror|Thriller
Action|Adventure|Comedy
Comedy
Action|Sci−Fi
Action|Drama|Thriller
Action|Thriller
Action|Romance|Thriller
Adventure|Children|Fantasy
Comedy|Romance
Action|Sci−Fi|Thriller
Action|Adventure|Thriller
Drama|Thriller
Action|Crime|Thriller
Action|Adventure|Sci−Fi
Thriller
Comedy|Crime
Action|Adventure|Sci−Fi|Thriller
Sci−Fi|Thriller
Adventure|Comedy|Sci−Fi
Comedy|Drama
Drama|Romance
Action|Horror|Sci−Fi|Thriller
Comedy|Drama|Fantasy|Romance
Comedy|Drama|Romance
Action|Crime|Drama|Thriller
Action|Adventure|Fantasy
Action|Adventure
Drama
Drama|Mystery|Thriller
Adventure|Drama
Crime|Drama|Thriller
Adventure|Animation|Children|Comedy|Fantasy
Documentary
Action|Drama|War
Crime|Drama
Drama|War
Comedy|Crime|Drama
Comedy|Drama|Romance|War



As can be seen in the graph there are clearly some genres with higher ratings. The genres variable contained many different combinations of the 19 different genres, combined by a "|". To make that information more usable 19 dummy variables were created so that if a genres value contained multiple genres then multiple dummy variables equaled 1. The original genres variable was deleted.

The timestamp variable is the time when the rating was made, in seconds. To make it more useful it was converted into a date time format. Then a simple plot of ratings over time was made and a loess smoothing line was included to better show the relationship between date and rating.

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

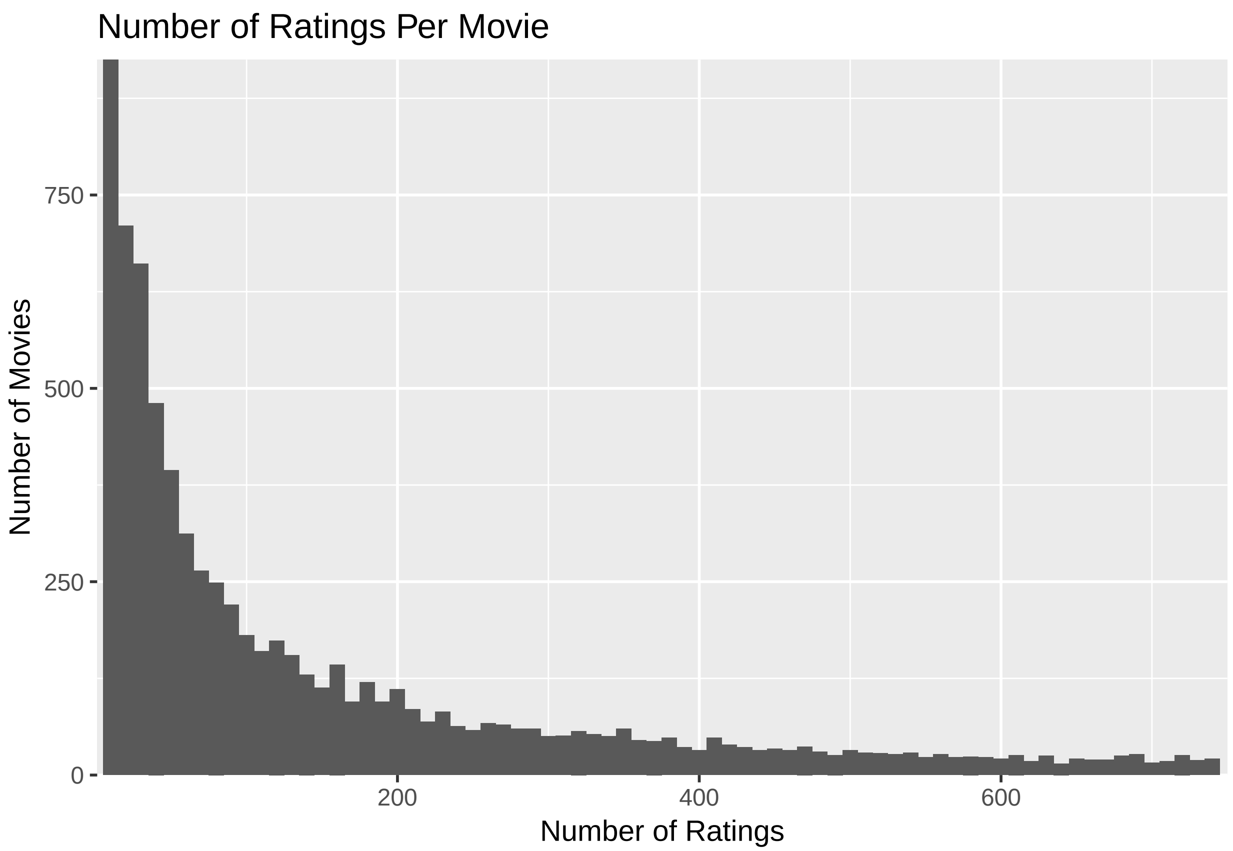


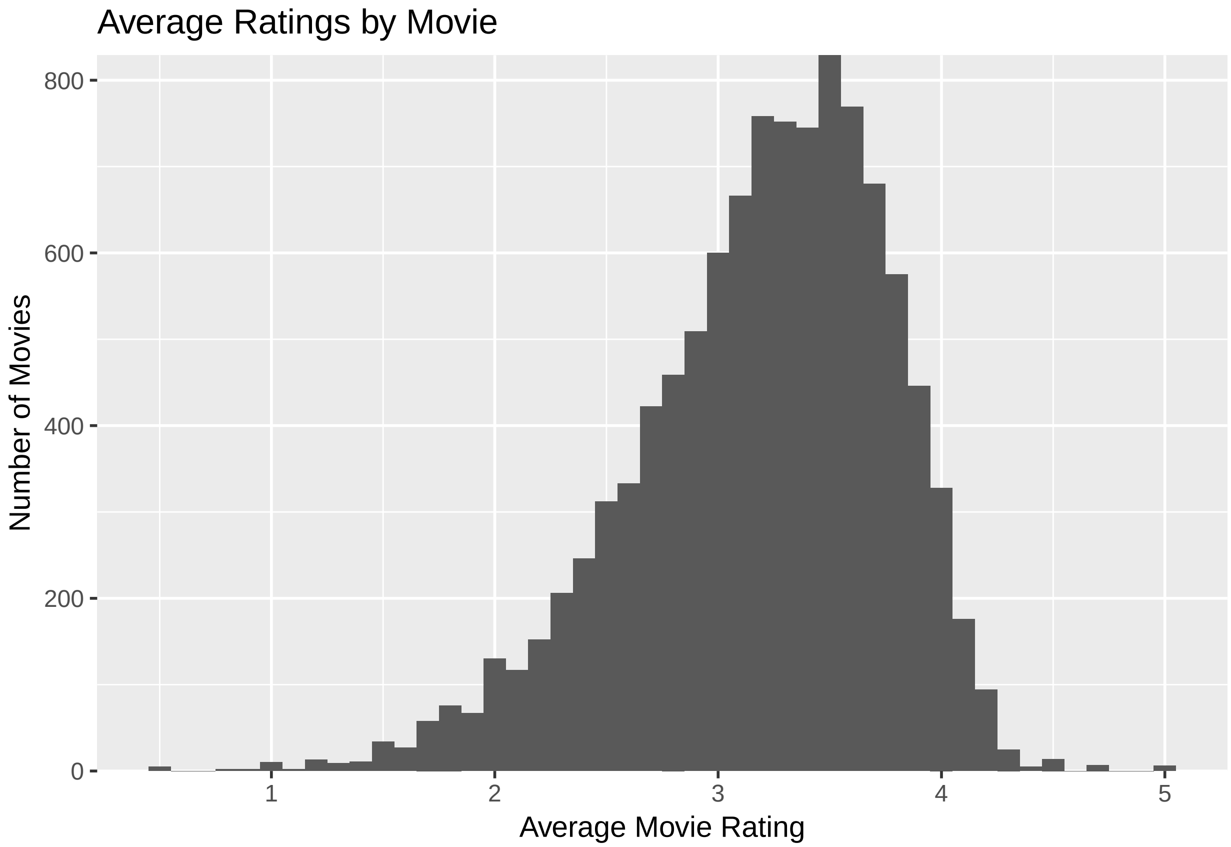
As can be seen in the above plot there appears to be very little relationship between rating and time, therefore it was best to drop the time variable otherwise it would add noise and computational time to the models.

The best predictors of how an individual will rate a movie are that individuals past ratings and that movie's past ratings. To examine this relationship I created graphs of the average rating by movie and user and the number of times movies were rated and the number of times users rated movies.

```
## Warning: Removed 2274 rows containing non-finite values (stat_bin).
```

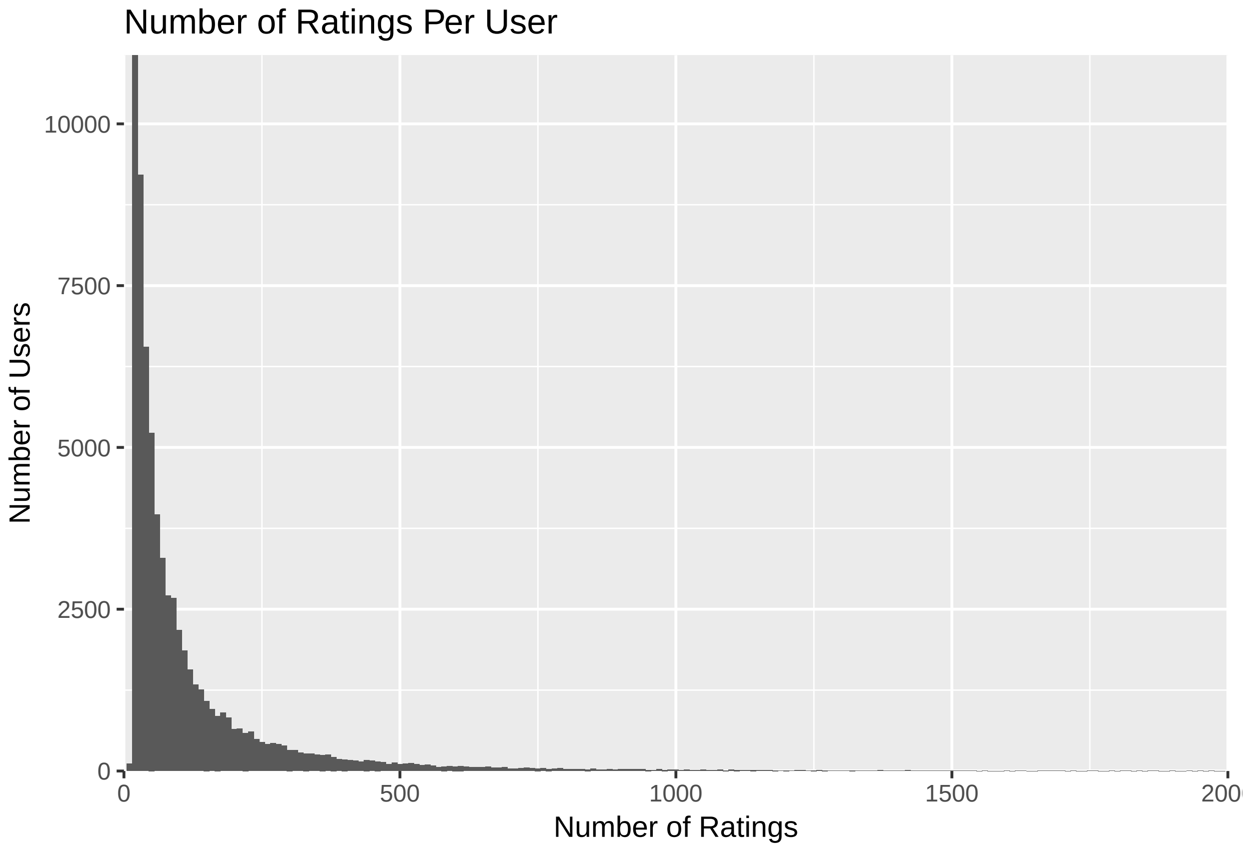
```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

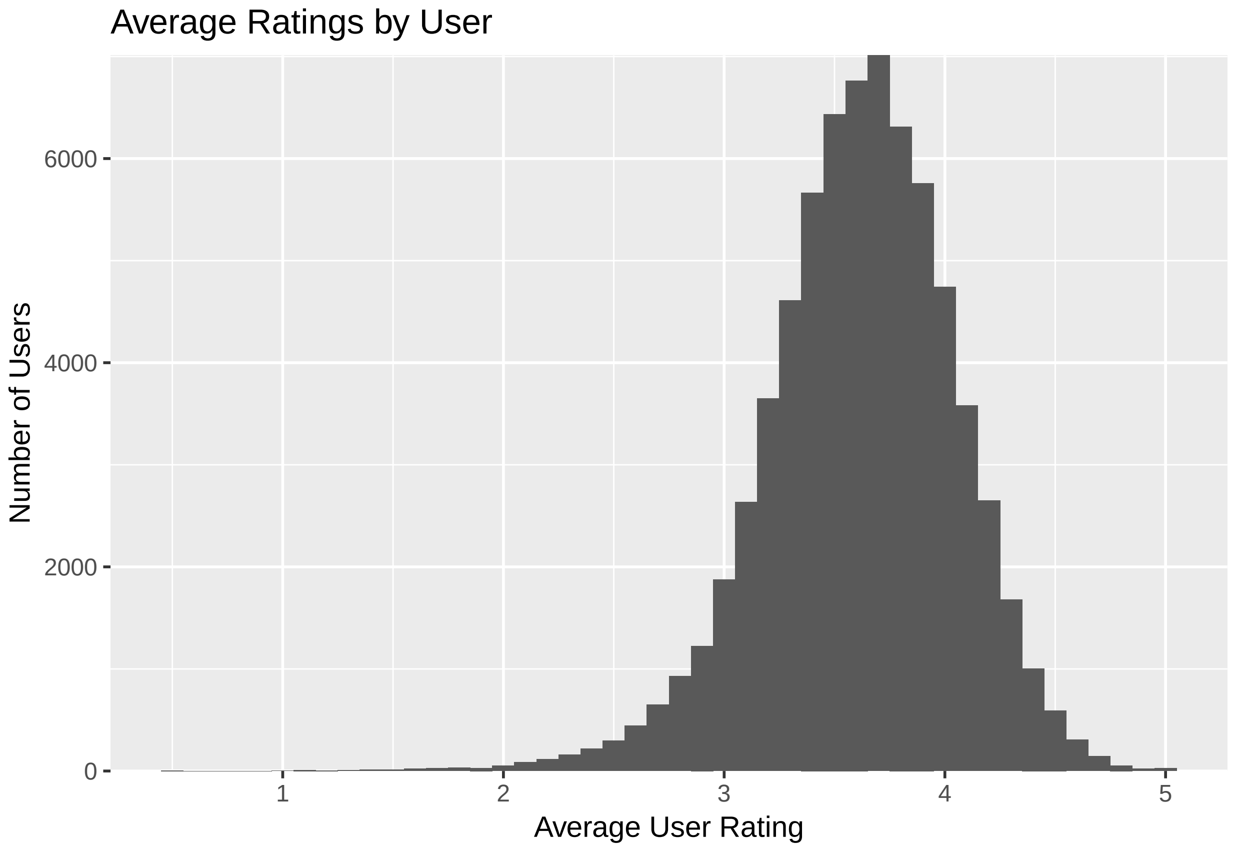

Number of Ratings Per Movie
Number of Movies
750
500
250
0
200
400
600
Number of Ratings



```
## Warning: Removed 58 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

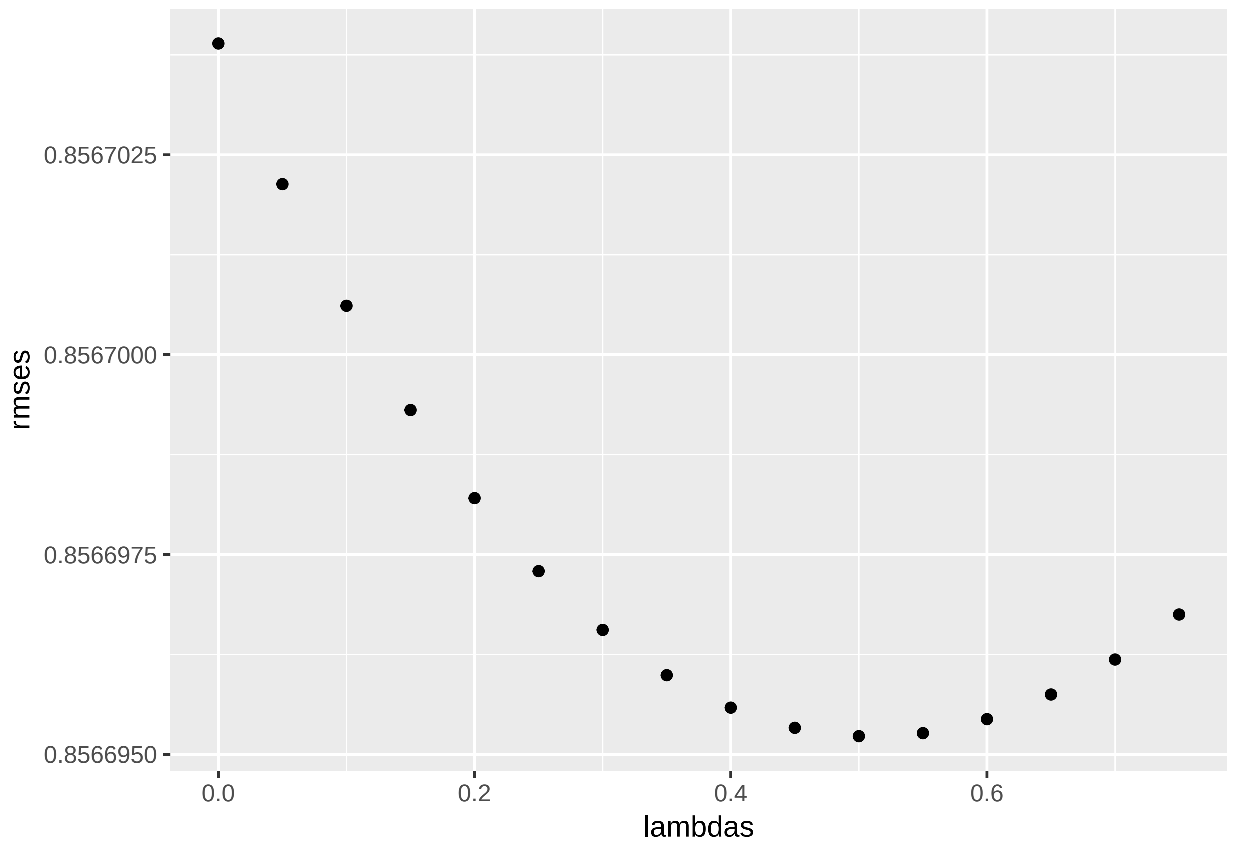

Number of Ratings Per User
10000
7500
5000
2500
0
Number of Users
0
500
1000
1500
200
Number of Ratings



These graphs show there is considerable variation in the average movie rating and the average rating by each user. Additonally you can see that some movies were rated by many more users than others. Since some movies were rated by very few users then those movie means are less reliable.

Since there is significant variation in the mean ratings and one would expect they would be predictive. Before dropping the userId and movieId variables two additional predictors were created, the average rating for each movie and the average rating by that user. Because a good predictor of any rating will be the average rating for all movies, 3.5124652, the averages for each movie and user will be transformed using this mu so that they are the difference between the overall mean and the mean for that movie or user. By subtracting by mu the averages become centered on 0.

However, because some movies are rated by few users and some users rate few movies these averages need to be transformed further. The fewer ratings per user or per movie the less reliable that average is as a predictor, especially compared to the overall mu. If a movie is rated by very few users then the overall mu is a better predictor than the mean for that individual movie. So after centering the movie and user averages on 0, by calculating their difference from the overall mean, they should be regularized to penalize averages based on few ratings. Regularization penalizes estimates that come from small sample sizes. The penalization term is lamda and since it is a model parameter tuning can be used to choose it. The lamda that produced the lowest RMSE was 0.5 so that was what was used to create b_i and b_u. After this the movieId and userId variables were dropped.



b_i is the variable representing the regularized difference between the mean rating for a movie and the overall mean. It was created by first calculating the overal mean for all ratings in the dataset, mu, then subtracting that for each rating for a movie, summing all of those values then dividing that sum by the number of ratings for that movie plus the lamba, 0.5. b_u is the variable representing the regularized difference between the mean user rating and the overall mean and was created in the same way as b_i, except the b_i value for that movie was also subtracted from that movie's rating in addition to overall mean before summing and dividing by the number of ratings for that user plus lamda. Thus b_u takes into account the movie's average rating before looking at the user's effect on that rating. The exact code used to create these two variables is below.

```
l <- 0.5
mu <- mean(edx$rating)
b_i <- edx %>%  group_by(movieId) %>%summarize(b_i = sum(rating - mu)/(n()+l))
b_u <- edx %>%  left_join(b_i, by="movieId") %>% group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+l))

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

Overall there was little need for dimension reduction since there are only a few predictors. However, the remaining predictors were checked for near zero variance using the caret packages nearZeroVar function. Removing predictors with near zero variance improves model speed and efficiency without sacrificing predictive power. The dummy variables for the following genres had near zero variance and were dropped: IMAX, Musical, Western, FilmNoir, Documentary. This left 16 predictors remaining, the b_i and b_u variables and 14 movie genre dummy variables. Now that the final training dataset was created, the model was developed.

Model development

Since the original EdX dataset was very large to test machine learning models a sample of 100,000 was created and that was divided into a train and set set. These were created in such a way to ensure all the movies in the train dataset were in the test dataset.

The outcome in question here, ratings, is semi-continuous and ordinal. Since it can be thought of as continuous good models to try are linear regressions and regression trees. A linear regression would seem to be a good fit for this data since the relationship between the predictors could be expected to be linear and there are few predictor variables. The outcome variable also has a small range, making the usefullness of more complex transformations, such as a quadratic, less useful.

A linear model was fitted on the train dataset and then estimates of the rating (y hat) were generated on the test data set. This produced a low RMSE of 0.8487.

Due to the effectiveness of the linear model with the regularized means, a ridge regression was tested as well. However, this ridge regresion was tested using a different training dataset where the value of b_i was just the average rating for that movie and the value for b_u was the average rating for that user. This was done using the caret package, so the lamda could be tuned and ten fold cross validation was used as well. The lamda value that produced the lowest RMSE was 0.0, different than found previously. Using this lamda produced an RMSE of 0.86753. This was low, but still higher than the linear model.

Other machine learning algorithims are rule based or decision tree models. These decision tree models create rules whereby certain values within cutoffs of the predictors result in different predictions. These rules are multilevel, so that different predictors align in a tree with different rules as to when to predict certain values. A row of data's predictor values will then fall down one branch resulting in a prediction, which for regression trees is the mean value of the outcome variable for all the observations that fit that branch's criteria. These rules are created to minimuze the residual sum of squares (RSS). When the outcome is continuous then you need to use regression tree models. The caret package was again used (the rpart method) to tune the model parameter, which in this case was cp or the complexity parameter. The caret package by default will use cross validation to decide the best tuning parameter value when the train function is used. The default is 25 bootstrap samples of 25 percent of the observations (Irizarry 2019). Several training iterations were run (with differen ranges of cp) until the final, best value of cp was found to be very low, 0.00024. However this model did not work well, it produced an RMSE of 0.95242.

To improve on this regression tree prediction a random forest model was tried. Random forest models average the predictions of a number of trees created from different bootstrap random samples. When they make a split for a new tree they don't look at all predictors but just a random sample of them (Irizarry 2019). There are multiple models for random forests in the caret package and they have different tuning parameters. First the Rborist method was used and the tuning parameter was the minNode, which is the number of nodes or values in a tree that are needed to be split. Predfixed was set to 2 and values for minNode between 1 and 10 were tried. The value with the lowest RMSE was when minNode was equal to 4. However this only produced an RMSE of 0.91647, higher than linear regression. To attempt to improve on this the rf method was used and the mtry paramter was tuned. mtry is the number of variables randomly sampled as candidates at each split and is equiavalent to predFixed in the Rborist method. This intially took too long to run, so the trainControl caret function was used to limit the number of cross validations to 2 of 90 percent of the observations. The best value of mtry was found to be 3 and this produced a low RMSE of 0.8605.

The last model tested was the KNN model, or K nearest neighbors model. This model looks for similar observations in the dataset, observations that are close in distance. For this model the parameter to tune is K, with larger K values leading to smoother estimates and smaller Ks resulting in more flexible estimates (Irizarry 2019). Again the caret package train function was used and the number of cross validation samples used was 2, each comprised of 90 percent of the observations to limit computation time, which was significant for the KNN model. The optimatal K value was found to be 40 and this produced an RMSE of 0.86899.

Results

While several of the machine learning algorithims produced RMSE below 0.9, the lowest RMSE was still from the linear regresion model. A table below shows the RMSE generated with the tuned models using the smaller training dataset with 79,999 observations and the test dataset with 15,287 observations. It's possible that with a larger training dataset the best model would have been different, however the computation time on the desktop computer used was significant even with the reduced size dataset.

Table of RMSEs and models

method	RMSE
Linear Regression	0.8486970
Penalized Ridge Regression	0.8675346
Regression Trees Model	0.9524245
Random Forest	0.8604956
KNN	0.8689867

RMSE

The final result RMSE using the linear regression model created using the full Edx dataset with approximately 9 million records and the almost 1 million validation records was 0.68172.

Linear Regression Model Coefficients

term	estimate	std.error	statistic	p.value
(Intercept)	3.5204884	0.0008190	4298.327533	0.00e+00
DramaTRUE	0.0100828	0.0007085	14.230560	0.00e+00
WarTRUE	-0.0052747	0.0012974	-4.065768	4.79e-05
CrimeTRUE	0.0063073	0.0008892	7.092978	0.00e+00
ActionTRUE	-0.0124705	0.0008061	-15.469722	0.00e+00
ComedyTRUE	-0.0039846	0.0007086	-5.623364	0.00e+00
HorrorTRUE	0.0083006	0.0011600	7.155418	0.00e+00
SciFiTRUE	-0.0057470	0.0008915	-6.446729	0.00e+00
FantasyTRUE	0.0058174	0.0010277	5.660666	0.00e+00
MysteryTRUE	0.0104922	0.0012461	8.420339	0.00e+00
RomanceTRUE	-0.0105030	0.0007698	-13.643945	0.00e+00
ChildrenTRUE	-0.0335446	0.0014750	-22.742005	0.00e+00
ThrillerTRUE	-0.0113751	0.0007827	-14.532741	0.00e+00
AdventureTRUE	-0.0073109	0.0008340	-8.766447	0.00e+00
AnimationTRUE	0.0098987	0.0017015	5.817665	0.00e+00
b_i	1.0059596	0.0006405	1570.507112	0.00e+00
b_u	1.0058817	0.0007310	1375.994805	0.00e+00

Conclusion

Perhaps it should not be suprising that the best model for RMSE was linear regression since the number of predictors was small. One would expect the relationship between the predictors and the outcome to be linear. For example, the higher others have rated a movie the higher we expect the current user to rate the movie and we'd generally expect someone to rate all movies higher or lower, taking into account their genres preferences. Regularizing the movie and user means into the effects also reduced some of the noise, making the predictions more accurate. It's possible that using the full movielens dataset or trying more models with different parameters could have resulted in a smaller RMSE. However, the time it took a desktop computer to run the models using the 80,000 record sample was still considerable and given the small number of predictors a much better model was not gauranteed.