HarvardX PH125.9x Movie Lens

Daniel Yinanc

1/23/2022

1. Introduction

Movielens dataset is a data set created by GroupLens, a research lab at the University of Minnesota. **DSLabs** package contains a 100K sized version of it as movielens data package which we used to make predictions on a recommendation engine.

Original Movilens dataset contains 27M entries, dataset we are going to use is 10M in size yet it will be demonstrated below that it is still too large for standard regression R packages. Using alternative techniques we will be developing a recommendation engine predicting rating from other predictors in the dataset.

First we will be creating a training (edx) and validation (validation) sets from Movielens 10M dataset. Afterwards we will conduct data profiling and exploratory data analysis to assess relationships between variables and potential characteristics of data that we can utilize to build our machine learning models.

Machine learning models we will progress from the baseline mean to models of greater sophistication utilizing different predictors from the dataset, culminating with utilization of regularization to develop higher predictive capability model as our final model. We will present all models in terms of their RMSE to demonstrate the principles of model development.

2. Data Analysis and Methods

2.1 Data Exploration

Below are variables transformed from original MovieLens dataset.

- userId: Anonymized userIds (numeric)
- movieId: Unique ID assigned for movies (numeric)
- rating: Ratings given by individual users, 0.5-5 in 0.5 increments. (numeric)
- timestamp: Epoch timestamp of rating delivery time (numeric)
- title: Title of the movie with year of production. (char)
- genres: Pipe separated genres from list below. (char)

2.1.1 List of Genres:

- Action
- Adventure
- Animation
- Children's
- Comedy
- Crime
- Documentary
- Drama
- Fantasy
- Film-Noir

- Horror
- Musical
- Mystery
- Romance
- Sci-Fi
- Thriller
- War
- Western

2.1.2 MovieLens 10M

Download locations can be found for result replication purposes

- https://grouplens.org/datasets/movielens/10m/
- http://files.grouplens.org/datasets/movielens/ml-10m.zip

2.1.3 Data Load

In order to predict the ratings as a recommendation engine, I need to load initial variables to dataframes and transform them to a format that can be used later on.

This part of the code is verbatim provided as part of delivery instructions.

```
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# if using R 3.6 or earlier:
#movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
#                                           title = as.character(title),
#                           genres = as.character(genres))
# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
```

```
removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)

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

2.1.4 Transform data for review years

In order to be able to use review years as a predictor variable, I am transforming edx and validation datasets in the following manner.

```
dates_edx <- as.Date(as.POSIXct(edx$timestamp, origin="1970-01-01"))
years_edx <- format(dates_edx, format="%Y")

edx_combined <- cbind(edx,years_edx)
edx_combined <- rename(edx_combined, years = years_edx)

dates_validation <- as.Date(as.POSIXct(validation$timestamp, origin="1970-01-01"))
years_validation <- format(dates_validation, format="%Y")

validation_combined <- cbind(validation,years_validation)
validation_combined <- rename(validation_combined, years = years_validation)
```

2.2 Basic Summary Statistics

2.2.0 First few entries

From first few entries, we can see that this dataset has a substantial number of char columns that will need to be converted to factors. Additionally a quick analysis of ratings indicate that perhaps it is not continious (which it actually is not as dealt with below), which can be created as a classification problem instead of a regression case here. However for the purposes of this assignment, we will ignore this finding and use a regression type of approach and RMSE as the loss function.

```
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                            genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

2.2.1 Summary Analysis

Summary data shows us that in the training (edx) and test (validation) sets :

- Ratings are between 0.5 and 5

- There are 65133 movies
- There are 71567 users

```
summary(edx)
```

```
##      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
##
##
##
```

2.3 Variables Analysis

2.3.1 Rating

We would like to see how ratings are distributed, to be able to have a better understanding of the dependent variable to be modeled here. Median is between 3.5 to 4, and results are roughly distributed in a skewed normal around it.

Average rating for a title is **3.198**

```
mean(edx %>% group_by(title) %>% summarize(m = mean(rating)) %>% .$m)
```

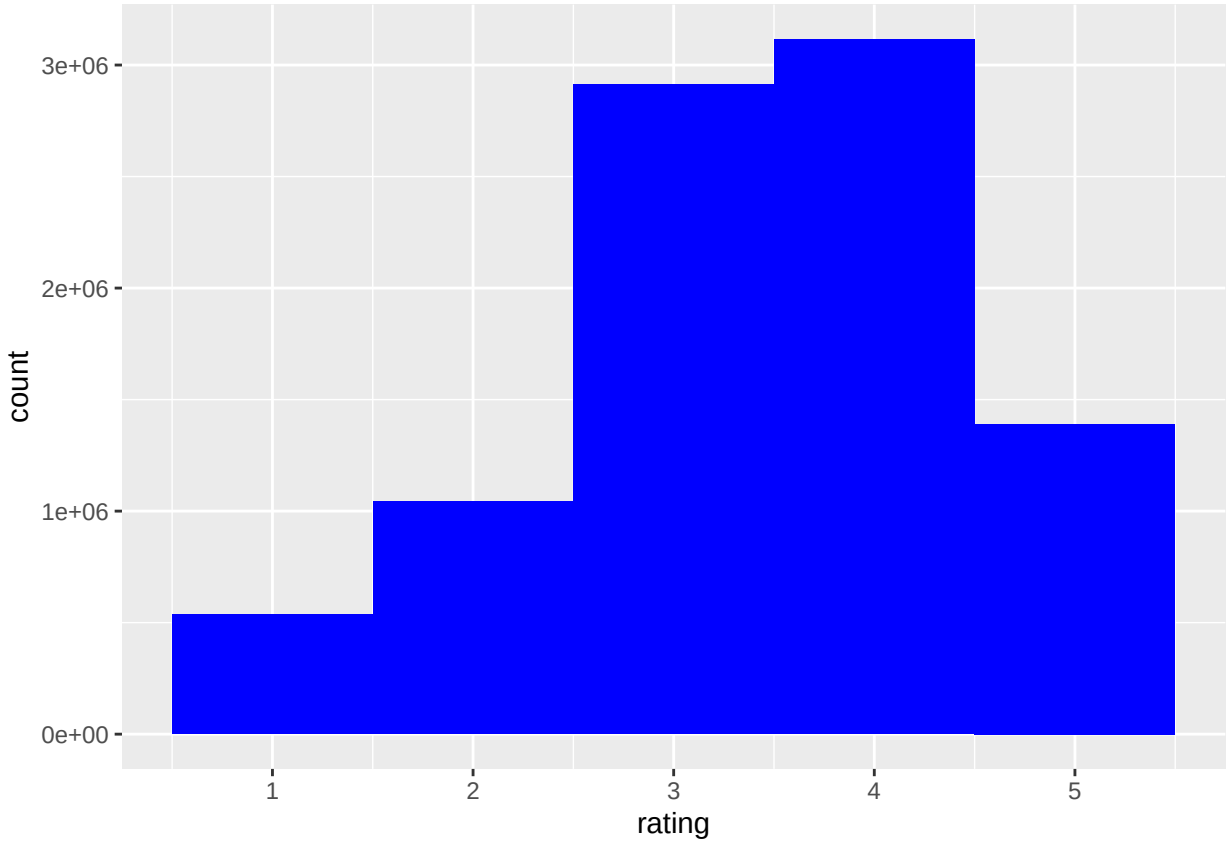
```
## [1] 3.191762
```

Standard deviation of ratings for titles is **0.571**

```
sd(edx %>% group_by(title) %>% summarize(m = mean(rating)) %>% .$m)
```

```
## [1] 0.5713468
```

How about distribution of ratings?



Ratings are not evenly distributed as expected. Top movies receive 10K+ ratings. While lesser known movies receive virtually any.

```
edx %>% group_by(title) %>% summarize(ratings = n()) %>% arrange(desc(ratings))
```

```
## # A tibble: 10,676 x 2
##    title                                                     ratings
##    <chr>                                                       <int>
##  1 Pulp Fiction (1994)                                         31362
##  2 Forrest Gump (1994)                                         31079
##  3 Silence of the Lambs, The (1991)                            30382
##  4 Jurassic Park (1993)                                        29360
##  5 Shawshank Redemption, The (1994)                            28015
##  6 Braveheart (1995)                                           26212
##  7 Fugitive, The (1993)                                        25998
##  8 Terminator 2: Judgment Day (1991)                           25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)   25672
## 10 Apollo 13 (1995)                                            24284
## # ... with 10,666 more rows
```

Very few reviews for the lesser known movies.

```
edx %>% group_by(title) %>% summarize(ratings = n()) %>% arrange(ratings)
```

```
## # A tibble: 10,676 x 2
##    title                                                     ratings
##    <chr>                                                       <int>
##  1 1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)                   1
```

```
##  2 100 Feet (2008)                                                          1
##  3 4 (2005)                                                                 1
##  4 Accused (Anklaget) (2005)                                                1
##  5 Ace of Hearts (2008)                                                     1
##  6 Ace of Hearts, The (1921)                                                1
##  7 Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di..~    1
##  8 Africa addio (1966)                                                      1
##  9 Aleksandra (2007)                                                        1
## 10 Bad Blood (Mauvais sang) (1986)                                          1
## # ... with 10,666 more rows
```

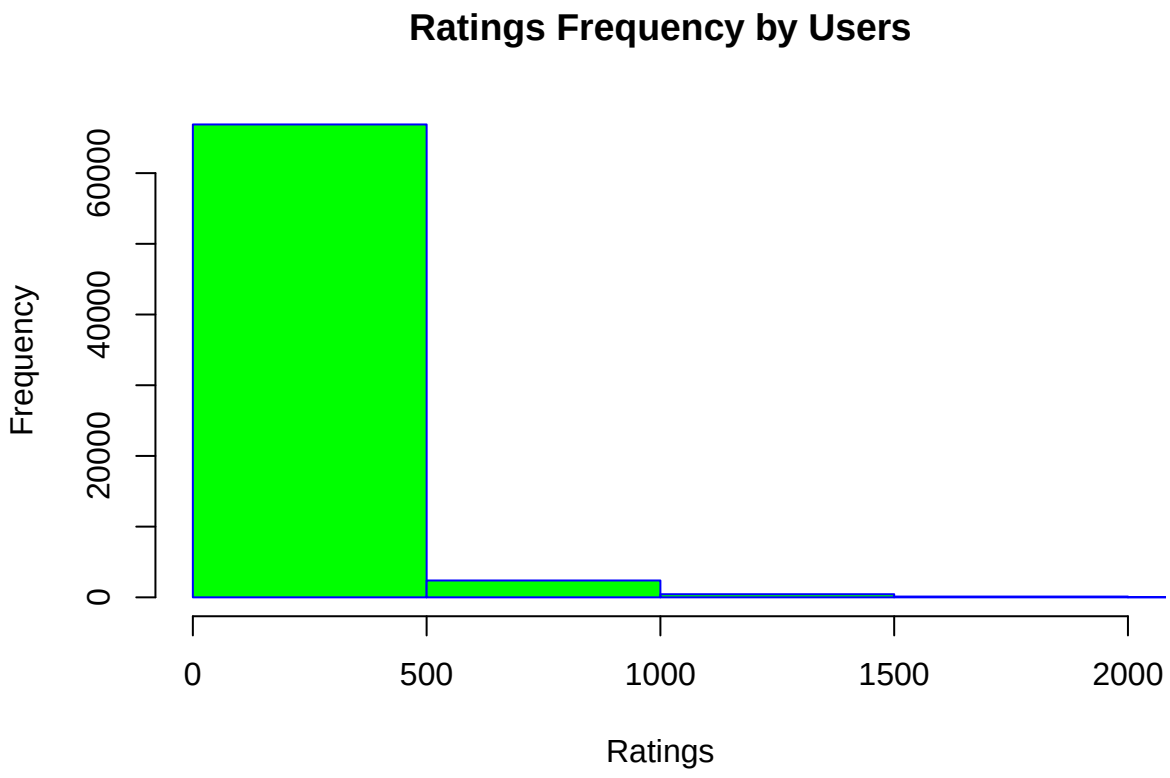
2.3.2 User Preferences Effect

In total there are 69878 distinct users in training set. However this does not mean users are unique, due to anonymization we have no way of knowing but sign up mechanism of MovieLens website can be easily circumvented by design or not, via providing different email addresses.

This can easily lead to multiple end userIds corresponding to same human person rating same movies in same or similar ways, introducing an error factor that is hard to quantify and address to our algorithm.

```
##   n_users
## 1   69878
```

Below graph shows us that most users provide very limited number of ratings, below 500 per person. Considering there is over 70K movies in combined 10M data set. It is a very sparsely populated rating system that can pose challenges for a lot of matrix algebra if we try to use traditional fitting techniques.



2.3.3 Movie Effect

In total, there are 10677 movies in training set.

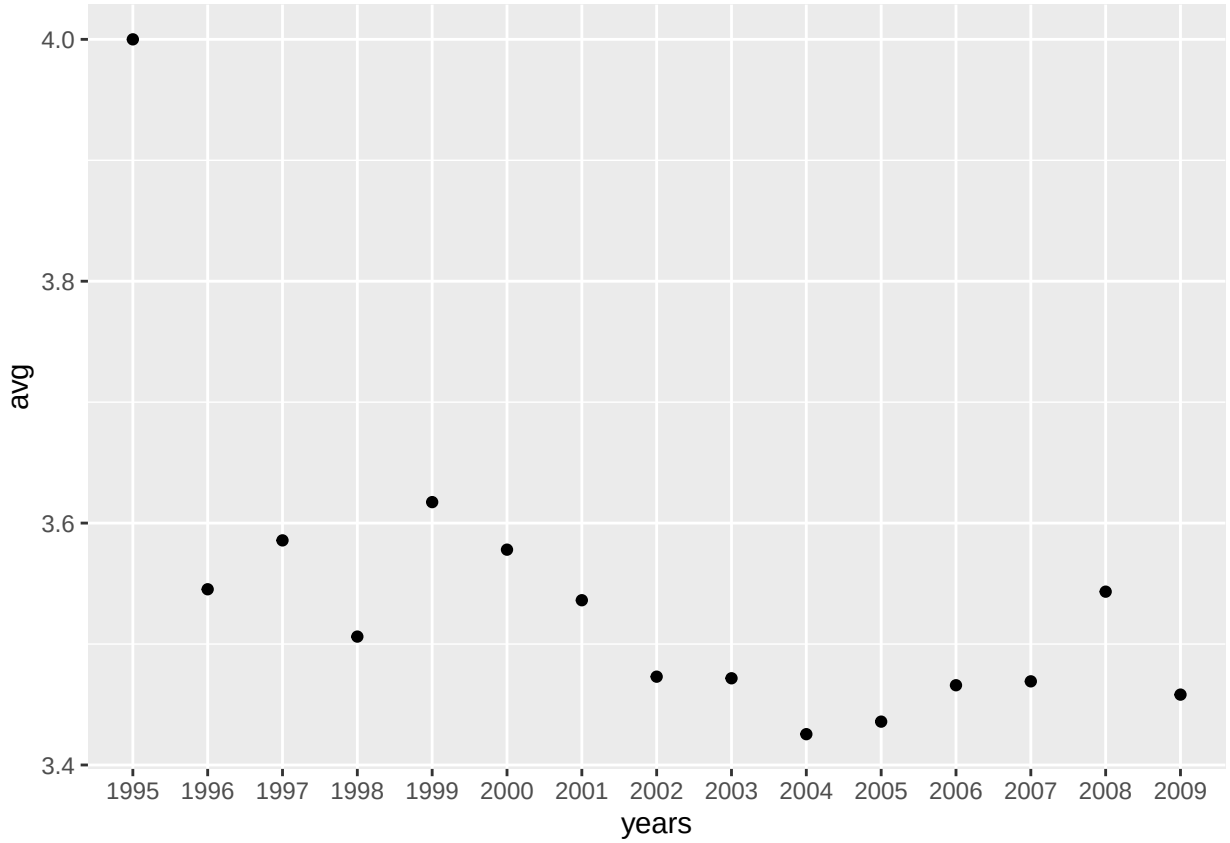
```
##   n_movies
## 1    10677
```

There is a real concern about data quality with this variable as we found at least one movie being classified with two different ids (War of the Worlds (2005).

```
## # A tibble: 10,676 x 2
##    title                                                        n_id
##    <chr>                                                       <int>
##  1 "War of the Worlds (2005)"                                      2
##  2 "...All the Marbles (a.k.a. The California Dolls) (1981)"       1
##  3 "...And God Created Woman (Et Dieu... créa la femme) (1956)"    1
##  4 "...And God Spoke (1993)"                                       1
##  5 "...And Justice for All (1979)"                                 1
##  6 "'burbs, The (1989)"                                            1
##  7 "'night Mother (1986)"                                          1
##  8 "'Round Midnight (1986)"                                        1
##  9 "'Til There Was You (1997)"                                     1
## 10 "\"Great Performances\" Cats (1998)"                            1
## # ... with 10,666 more rows
```

2.3.4 Review Time Effect

Timestamp variable corresponds to the time of review creation by the user. Further analysis of the year of review indicate an effect on ratings, as there seems to be a downward drift on average ratings since records began.



As we can see from the graph above, 1995 average reviews were 4, 1990s they are about 3.5, they went down to about 3.4s in 2000s. This indicates a possible effect of review year on provided reviews.

2.3.5 Genre Effect

Genres are a pipe separated list of adjectives such as "Action", "Animation" or "Comedy", for example "Action|Animation|Comedy|Horror" which happens to be the lowest rated of all genres.

In its relationship with Rating, we found a clear correlation between genres as their averages were substantially different from a measly 1.5 for "Action|Animation|Comedy|Horror" to 4.30 for "Action|Crime|Drama|IMAX". A lot of top rated movies happen to be in "Action|Crime" category perhaps going a long to explain proliferation of CSI type movies.

As below, we can see that all top rated genres are some form of Action combined with Crime, Adventure and Drama:

```
## # A tibble: 252 x 2
##    genres                                        avg
##    <chr>                                       <dbl>
##  1 Action|Crime|Drama|IMAX                      4.30
##  2 Action|Adventure|Comedy|Fantasy|Romance      4.20
##  3 Action|Crime|Drama|Film-Noir|Mystery         4.12
##  4 Action|Drama|Thriller|War                    4.11
##  5 Action|Adventure|Crime|Drama|Mystery|Thriller 4.08
##  6 Action|Drama|Mystery|Romance|Thriller        4.04
##  7 Action|Adventure|Animation|Comedy|Sci-Fi     4
##  8 Action|Drama|Thriller|Western                3.97
##  9 Action|Adventure|Animation|Children|Comedy   3.96
```

```
## 10 Action|Adventure|Sci-Fi|Western                3.95
## # ... with 242 more rows
```

As below, we can see that all bottom rated genres are some form of Action combined with Children, Sci-Fi and Horror:

```
## # A tibble: 252 x 2
##    genres                                          avg
##    <chr>                                         <dbl>
##  1 Action|Animation|Comedy|Horror                  1.5
##  2 Action|Horror|Mystery|Thriller                  1.61
##  3 Action|Drama|Horror|Sci-Fi                      1.75
##  4 Action|Adventure|Drama|Fantasy|Sci-Fi           1.90
##  5 Action|Children|Comedy                          1.91
##  6 Action|Adventure|Children                       1.92
##  7 Action|Horror|Mystery|Sci-Fi                    1.93
##  8 Action|Children                                 2.04
##  9 Action|Adventure|Children|Comedy|Fantasy|Sci-Fi  2.06
## 10 Action|Adventure|Children|Comedy|Mystery         2.15
## # ... with 242 more rows
```

2.3.6 Variables Analysis Conclusion

In our analysis, we arrived at certain key conclusions that will be instrumental in building our machine learning models.

- Ratings are not evenly distributed indicating impact of other variables in its distribution
- **Movie quality drive ratings** movieId and rating is somewhat linked
- **User preferences drive ratings** userId and rating is somewhat linked
- **Genres impact ratings** genres and rating is somewhat linked
- **Review time impact ratings** timestamp and rating is somewhat linked

2.4 Variable Interdependencies

Analyzing variable interdependencies is the next step in determining if relationships we started to suspect as a result of data and variables analysis above can be further elicidated to give us a clear mathematical basis to determine if what we observe is supported by data.

Correlation is a common technique to determine if rate of change between variables are related somehow indicating, increase or decrease in a variable somehow impacts increase or decrease in another variable. This is the first technique we are going to try.

Chi-square test is another technique that is used to determine if variables are independent of each other. Usually with categorical variables, correlation does not yield meaningful results as rate of change of categorical variables are meaningless especially when they are not hierarchical.

2.4.1 How are variables correlated with each other?

I would like to find out if variables are correlated with each other, this is a good indicator about the predictive power of a predictor variable as well as good for eliminating variables that are too correlated with another predictor variable, simplifying the model in process.

```
numericVars <- which(sapply(edx, is.numeric))
numericVarNames <- names(numericVars)
cat("There are", length(numericVarNames), "numeric variables")
```

```
## There are 4 numeric variables
```

```
all_numVar <- edx %>% select(numericVars)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(numericVars)` instead of `numericVars` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
cor_numVar <- cor(all_numVar, use = "pairwise.complete.obs")
#Sort on decreasing correlations with Rating
cor_sorted <- as.matrix(sort(cor_numVar[,"rating"], decreasing = TRUE))

#Selecting high correlations
Cor_High <- names(which(apply(cor_sorted, 1, function(x) abs(x) > 0.175)))
Cor_High
```

```
## [1] "rating"
```

```
#cor_numVar <- cor_numVar[Cor_High, Cor_High]
#library(corrplot)
#corrplot.mixed(cor_sorted, tl.col = "black", tl.pos = "lt")
```

Result however is as to be expected, numeric variables (UserId and MovieId) are actually categorical variables, each entry being unique and have no hierarchical relationships which means movieID 2 does not mean it is "greater" than movieId 1 or vice-versa for all numerical variables.

This indicates that correlation study is the wrong approach here as their changes have no meaning due to non-hierarchical nature. Hence investigating for rate of change that is linked with each other is not a viable option.

2.4.2 Variable Dependence: Chi Square Test

Another approach to determine relationship between categorical variables is the chi square test, if a p-value < 0.05 can be achieved in this test, we can safely reject the independent variables hypothesis. I will be using this method to filter key predictor variables to see if our response variable (Rating) is related to each of them.

Rating vs userId p-value < 0.05, we can reject the hypothesis that userId and rating are independent variables

```
chisq.test(edx$rating, edx$userId)
```

```
##
##  Pearson's Chi-squared test
##
## data:  edx$rating and edx$userId
## X-squared = 7284653, df = 628893, p-value < 2.2e-16
```

Rating vs movieId p-value < 0.05, we can reject the hypothesis that movieId and rating are independent variables

```
chisq.test(edx$rating, edx$movieId)
```

```
##
##  Pearson's Chi-squared test
##
## data:  edx$rating and edx$movieId
## X-squared = 3682460, df = 96084, p-value < 2.2e-16
```

Rating vs timestamp p-value < 0.05, we can reject the hypothesis that movieId and rating are independent variables

```
chisq.test(edx$rating, edx$timestamp)
```

```
##
##  Pearson's Chi-squared test
##
## data:  edx$rating and edx$timestamp
## X-squared = 69407155, df = 58676301, p-value < 2.2e-16
```

Rating vs genres p-value < 0.05, we can reject the hypothesis that movieId and rating are independent variables

```
chisq.test(edx$rating, edx$genres)
```

```
##
##  Pearson's Chi-squared test
##
## data:  edx$rating and edx$genres
## X-squared = 1106756, df = 7164, p-value < 2.2e-16
```

2.4.3 Variable Interdependencies Conclusion

Conclusion is clear, userId, movieId, timestamp as review date and genres all have **predictive power** over Rating. Chi-square test is indicative of dependency between these variables, supporting our empirical observations fully.

2.5 Model Development

We will be using a series of techniques to develop a linear model using variables and regularization techniques to ensure our linear model produces a reasonable model fit using our loss function.

2.5.1 Loss Function

By defining $y_{u,i}$ as the rating i by user u and denote our prediction as $\hat{y_{u,i}}$, we can define the RMSE (Root Mean Square Error), our core error metric as:

$$\sqrt{\frac{1}{N}\sum y_{u,i} - \hat{y_{u,i}}}$$

Using r, we would be defining our function in this form:

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

2.5.2 Introduction to Machine Learning Models

We will build models starting with very simple mean driven to more sophisticated by using different predictors and a regularization technique.

Let us get started with the simplest of all models, assigning average value of ratings as the estimate can serve

2.5.3 Naive Model

We can start with the simplest of all recommendation systems, a system based on providing the mean rating for all movies to any new movie to be rated. This can serve as a great baseline as any other static value but the mean will result in a higher RMSE.

This can serve as a great baseline as best guess-timate value for us to compare with other more sophisticated models.

Mathematically we can represent this model as:

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

Naive RMSE is **1.061202**.

```
mu_hat <- mean(edx_combined$rating)
mu_hat
```

```
## [1] 3.512465
```

```
naive_rmse <- RMSE(validation_combined$rating, mu_hat)
naive_rmse
```

```
## [1] 1.061202
```

```
results <- tibble(method="Just the average", rmse=naive_rmse)
```

2.5.4 Movie Effects Model

Movie's latent quality as an artistic product we found to play a role in determining ratings from exploratory data analysis above.

As we have seen on 2.3.3 Movie Effect section, movie averages are **not** evenly distributed, hence strongly indicating a strong movie effect on predicting individually provided ratings. We can augment the previous model by adding a b_i term representing average ranking for a movie i.

Mathematically we can model this interaction as:

$$Y_{u,i} = \mu + b_u + b_i + \epsilon_{u,i}$$

RMSE of the movie effect alone is **0.9439087**.

```
movie_avgs <- edx_combined %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_hat))

predicted_ratings <- mu_hat + validation_combined %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

movie_RMSE <- RMSE(predicted_ratings, validation_combined$rating)
results <- results %>% add_row(method="Movie Effect Model", rmse=movie_RMSE)
```

2.5.5 User Effects Model

User preferences play a role in ratings choice, that is what we were able to find out from our exploratory data analysis, we found that user's selection of providing ratings indicate a movie that user have seen and felt like providing a rating in the first place. This self-selection mechanism goes a long way in explaining user effect.

As we have seen on 2.3.2 User Preferences Effect section, most users provide ratings for a very small percentage of 70K movies available in our training set. We can augment the previous model by adding a b_u term representing average ranking for a user u.

Mathematically we can model this interaction as:

$$Y_{u,i} = \mu + b_u + b_i + \epsilon_{u,i}$$

RMSE of the movie and user effects is **0.865348811**.

```
  b_i <- edx_combined%>%
    group_by(movieId) %>%
    summarize(b_i = mean(rating - mu_hat))

  b_u <- edx_combined %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = mean(rating - b_i - mu_hat))

 predicted_ratings <-
    validation_combined %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu_hat + b_i + b_u) %>%
    pull(pred)

user_RMSE <- RMSE(predicted_ratings, validation_combined$rating)
results <- results %>% add_row(method="Movie + User Effects Model", rmse=user_RMSE)
```

2.5.6 Genre Effects Model

As we have seen in our exploratory data analysis, genres play an important role in determining ratings. Their means are radically divergent indicating some genres are consistently better received by audiences than others.

As we have seen on 2.3.5 Genre Effect section, Average ratings for some popular categories like Action with Crime can range very close to 4 while some others like Action combined with Horror and SciFi can range very close to 1.5. We can augment the previous model by adding a b_g term representing average ranking for a user g.

Mathematically we can model this interaction as:

$$Y_{u,i} = \mu + b_g + b_u + b_i + \epsilon_{u,i}$$

RMSE of the movie, user and genre effect is **0.8649469**.

```
  b_i <- edx_combined%>%
    group_by(movieId) %>%
    summarize(b_i = mean(rating - mu_hat))

  b_u <- edx_combined %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = mean(rating - b_i - mu_hat))

  b_g <- edx_combined %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = mean(rating - b_u - b_i - mu_hat))

 predicted_ratings <-
```

```
    validation_combined %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = mu_hat + b_g + b_i + b_u) %>%
    pull(pred)

genre_RMSE <- RMSE(predicted_ratings, validation_combined$rating)
results <- results %>% add_row(method="Movie + User + Genre Effects Model", rmse=genre_RMSE)
```

2.5.7 Time of Review Effects Model

Exploratory analysis of review times indicate a small but definite downwards bias about review time on ratings. Starting on 1995 as 4, average review ratings started to go down to about 3.4.

As we have seen on 2.3.4 Review Time Effect section, Average ratings for some popular categories like Action with Crime can range very close to 4 while some others like Action combined with Horror and SciFi can range very close to 1.5. We can augment the previous model by adding a b_t term representing average ranking for a review year t.

Mathematically we can model this interaction as:

$$Y_{u,i} = \mu + b_t + b_g + b_u + b_i + \epsilon_{u,i}$$

In order to incorporate review years into my prediction algorithms, I converted timestamp column to years using the following formula:

RMSE of the movie, user, genre and review time effect is **0.8649282**.

```
  b_i <- edx_combined%>%
    group_by(movieId) %>%
    summarize(b_i = mean(rating - mu_hat))

  b_u <- edx_combined %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = mean(rating - b_i - mu_hat))

  b_g <- edx_combined %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = mean(rating - b_u - b_i - mu_hat))

  b_t <- edx_combined %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    left_join(b_g, by="genres") %>%
    group_by(years) %>%
    summarize(b_t = mean(rating - b_g - b_u - b_i - mu_hat))

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

```
    left_join(b_g, by = "genres") %>%
    left_join(b_t, by = "years") %>%
    mutate(pred = mu_hat + b_t + b_g + b_i + b_u) %>%
    pull(pred)

time_RMSE <- RMSE(predicted_ratings, validation_combined$rating)
results <- results %>% add_row(method="Movie + User + Genre+ Time of Review Effects Model", rmse=time_RM
```

2.5.8 Lasso Regresion: Penalized Least Squares

As a regularization technique, penalizing least squares is a time honored idea. In industry terms, this is referred as the Lasso Regression. General concept is about constraining total variability of effect sizes. By introducing a penalty to the loss function, we can convert model from:

$$Y_{u,i} = \mu + b_t + b_g + b_u + b_i + \epsilon_{u,i}$$

To one that minimizes not the least squares as in:

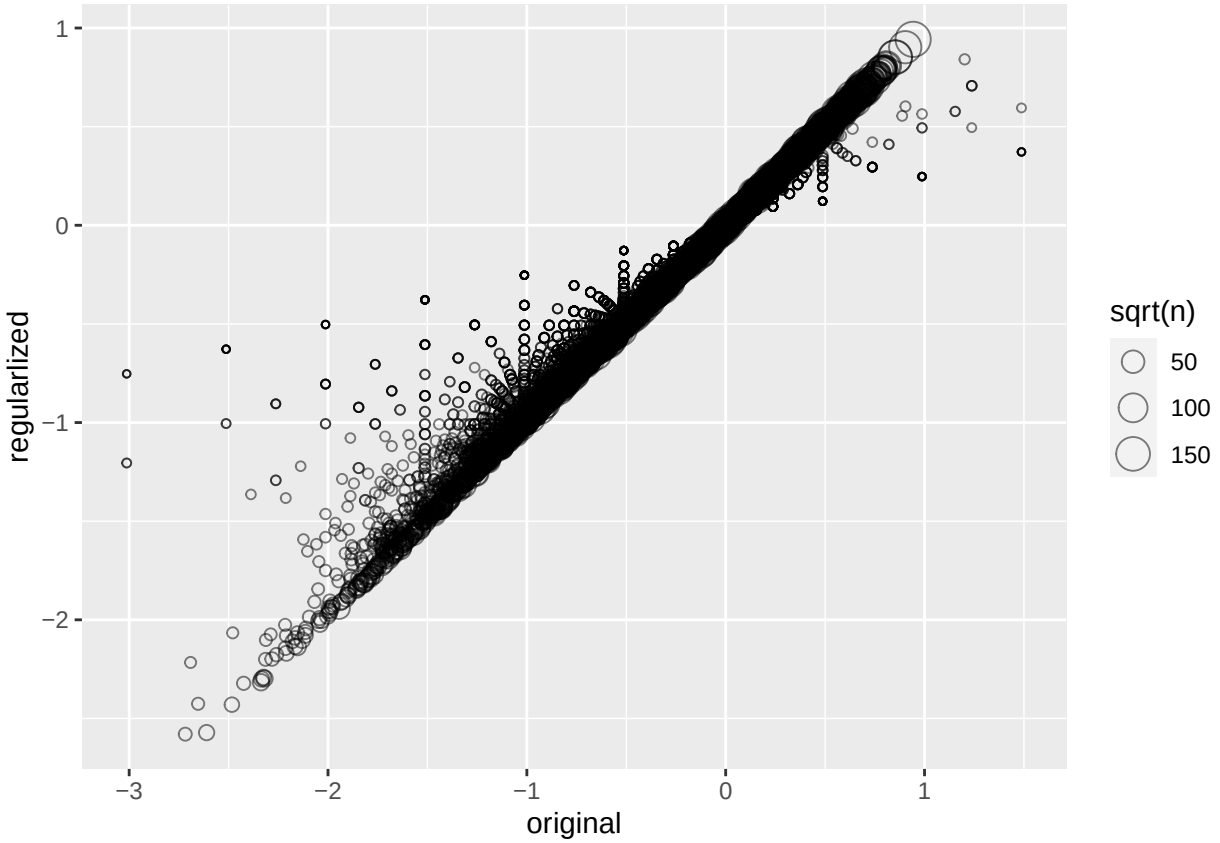
$$\sqrt{\frac{1}{N}\sum y_{u,i} - \hat{y_{u,i}}}$$

But to one that contains the penalty term λ for each mean used in our prediction model:

$$\sum (y_{u,i} - \mu - b_i)^2 + \lambda \sum (b_i^2)$$

To demonstrate lasso regression's effectiveness, we can see how least squares approaches to a mean faster than the least squares with below R code:

```
lambda <- 3
mu <- mean(edx_combined$rating)
movie_reg_avgs <- edx_combined %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

tibble(original = movie_avgs$b_i,
       regularlized = movie_reg_avgs$b_i,
       n = movie_reg_avgs$n_i) %>%
  ggplot(aes(original, regularlized, size=sqrt(n))) +
  geom_point(shape=1, alpha=0.5)
```



As above graph demonstrates how the estimates shrink, when we plot regularized estimates versus least squares estimates.

But does that translate into improved RMSE? We will demonstrate that in comparison to Movie effect alone, let's see how regularized Movie effect will perform.

RMSE of the regularized movie effect is **0.9438538** which is better than the RMSE for the unregularized **0.9439087**.

```
predicted_ratings <- validation_combined %>%
  left_join(movie_reg_avgs, by = "movieId") %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
RMSE(predicted_ratings, validation_combined$rating)
```

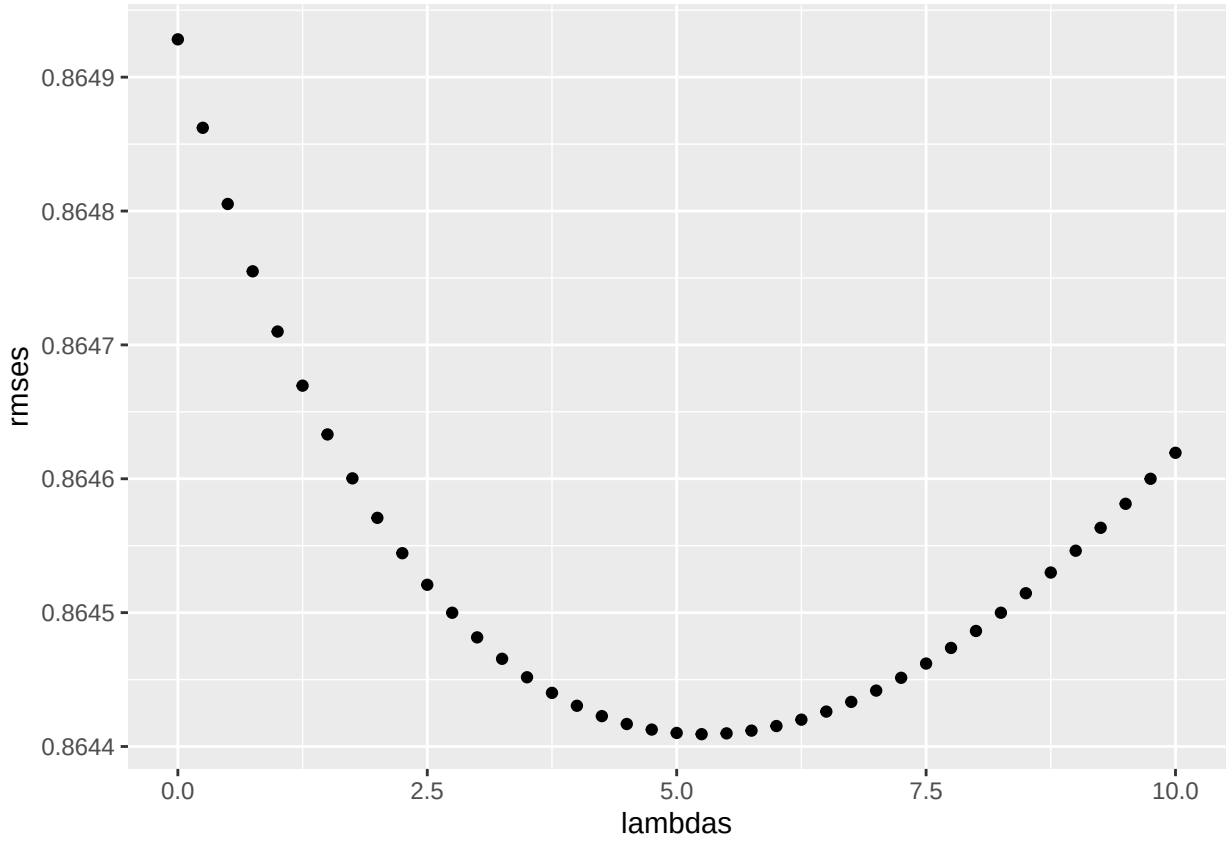
```
## [1] 0.9438538
```

3. Result

As we were able to demonstrate above, regularization improves accuracy of our predictions. Hence adding the concept to our combined model using predictors of Movie, User, Genre and Review time, our model mathematically becomes like this:

$$\sum (y_{u,i} - \mu - b_i - b_u - b_g - b_t)^2 + \lambda(\sum_i (b_i^2) + \sum_u (b_u^2) + \sum_g (b_g^2) + \sum_t (b_t^2))$$

```
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){
  b_i <- edx_combined%>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat)/(n()+l))

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

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

  b_t <- edx_combined %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    left_join(b_g, by="genres") %>%
    group_by(years) %>%
    summarize(b_t = sum(rating - b_g - b_u - b_i - mu_hat)/(n()+l))

 predicted_ratings <-
    validation_combined %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_t, by = "years") %>%
    mutate(pred = mu_hat + b_t + b_g + b_i + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings, validation_combined$rating))
})

qplot(lambdas, rmses)
```



Finding the lambda that optimizes that, we found our regularization constant for the final and most accurate prediction model:

```
# Smallest RMSE
min(rmses)
```

```
## [1] 0.8644092
```

```
# Which value minimizes the RMSE
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5.25
```

Yielding a final model's optimized RMSE score of **0.8644092** with a λ of **5.25**.

4. Conclusion

By using regularization and identifying predictor variables, we were able to predict ratings from userId, movieId, timestamp of request and genre. These variables will be available as part of a recommendation engine, allowing us to create reviews and only provide movies with a higher rating as recommended next movies to watch to end users.

Deploying this model will require frequent retraining as new ratings by end users need to be incorporated to the model to enhance future accuracy. New models, genres, users and ratings themselves will need to be incorporated from time to time for this model to reach production.

Following table summarizes all our machine learning models with RMSEs:

Table 1: All models and results

method	rmse
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Movie + User + Genre Effects Model	0.8649469
Movie + User + Genre+ Time of Review Effects Model	0.8649282
Regularized Movie + User + Genre+ Time of Review Effects Model	0.8644092