Capstone Project: MovieLens

Dakshina Dhanasekaran

18/04/2021

Executive Summary

For this project, I have created a movie recommendation system using the MovieLens dataset. I have used the 10M version of the MovieLens dataset to reduce the computation time. The MovieLens dataset has ratings given by various users for various movies along with the timestamp of rating, genre of the movie and release year of the movie. Machine learning models were trained using the inputs in one subset to predict movie ratings in the validation set. The final model chosen is a linear model using regularization by k-fold cross-validation with predictors as movie, user, genre, release year and rating year. The RMSE of this model is 0.8644928

Analysis

Data Cleansing The 10M version of MovieLens dataset (zip file) is downloaded from grouplens website. Once its unzipped, two files are read (ratings.dat and movies.dat). The ratings dataset has userid, movieid, rating and timestamp. The movies dataset has movieid, title and genres. They are combined to a single dataset movielens by using movieid. This is split into edx (90%) and validation (10%) datasets. It is made sure that there is no user id/ movie id in validation dataset that is not in edx dataset. Since the release year is part of the movie title, it is extracted as a separate column in both datasets. Also, the year is extracted from the timestamp column to get the year it was rated. The timestamp and title columns are removed to save memory.

```r
#Pre-processing

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("DT", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(DT)

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

dl <- tempfile()
```

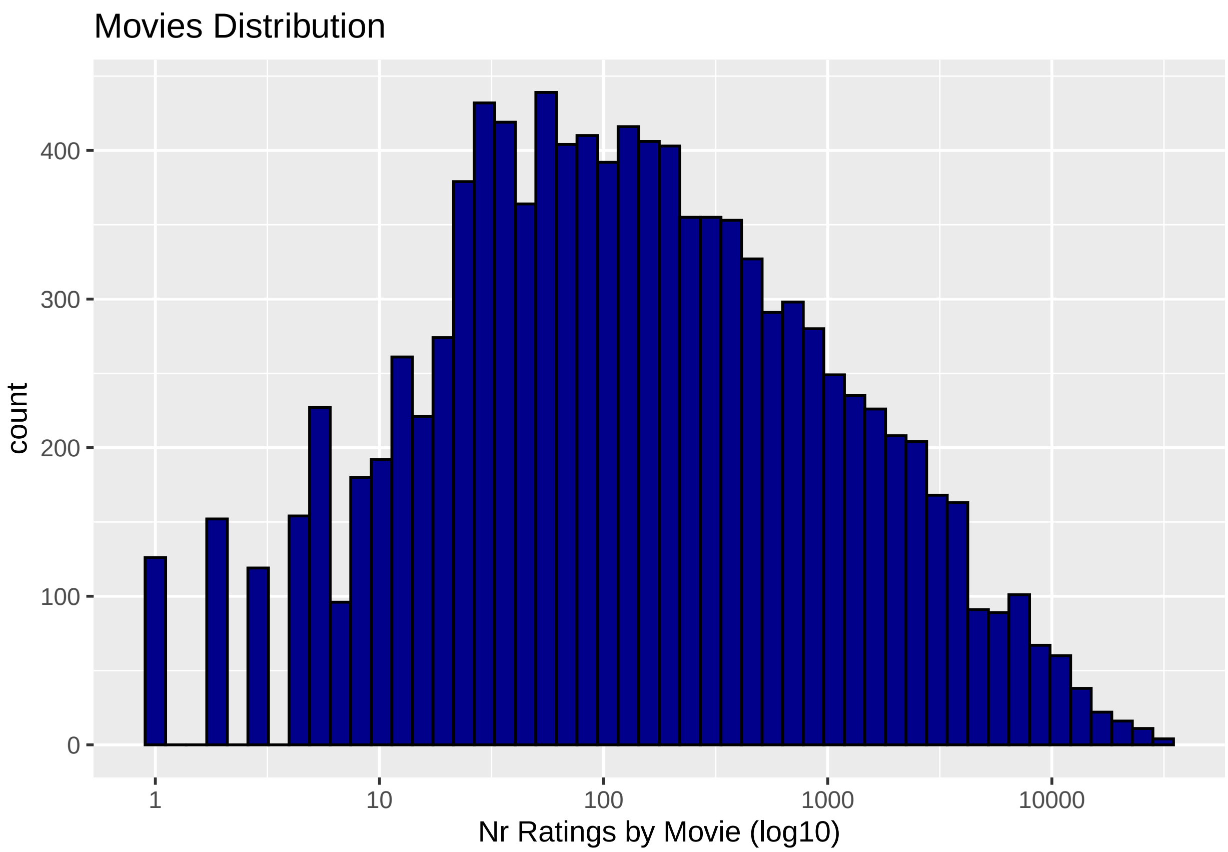
```r
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

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))


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)
edx <- rbind(edx, removed)

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

#Extract year of release and year of rating
edx <- edx%>%mutate(ReleaseYear =  as.numeric(str_sub(title,-5,-2)))
edx <- edx%>% mutate(RatingYear = year(as_datetime(timestamp)))
edx<-edx%>%select(-c(timestamp, title))

validation <- validation%>%mutate(ReleaseYear =  as.numeric(str_sub(title,-5,-2)))
validation <- validation%>% mutate(RatingYear = year(as_datetime(timestamp)))
validation<-validation%>%select(-c(timestamp, title))
```

```r
#Summary
edx %>%
  summarize(Nr_Users = n_distinct(userId),
            Nr_Movies = n_distinct(movieId),
            Nr_Ratings = n(),
            Nr_Genres = n_distinct(genres),
            Nr_ReleaseYears = n_distinct(ReleaseYear),
            Nr_RatingYears = n_distinct(RatingYear))
```

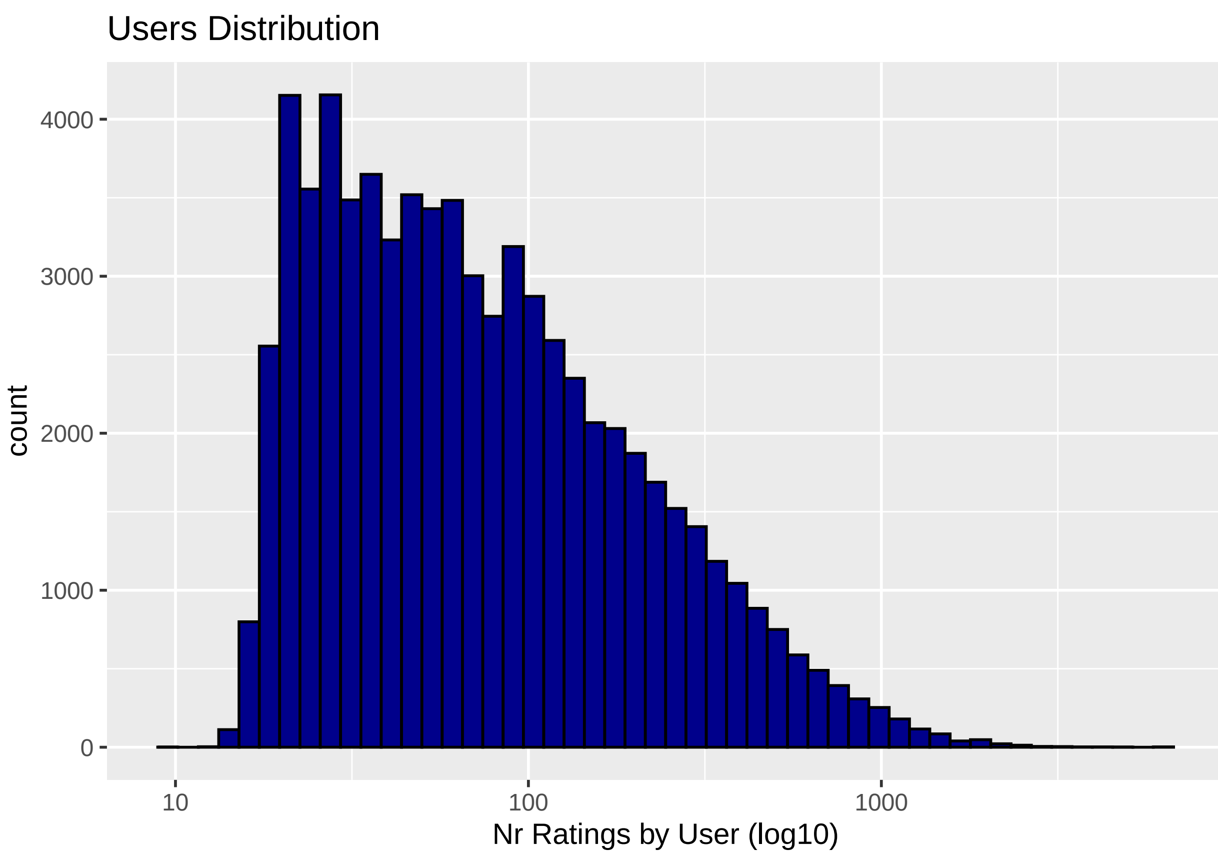
Exploratory Data Analysis

```
##   Nr_Users Nr_Movies Nr_Ratings Nr_Genres Nr_ReleaseYears Nr_RatingYears
## 1    69878     10677    9000055       797              94             15
```

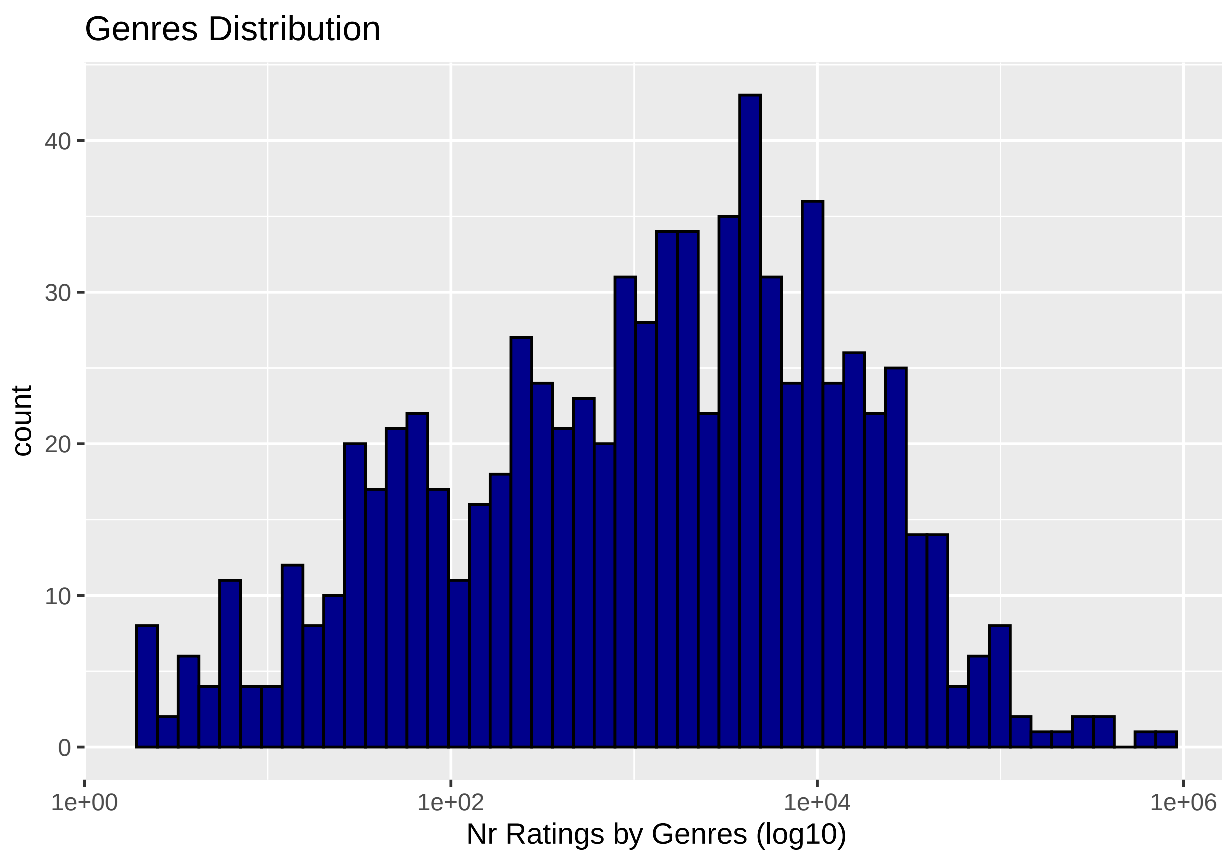
```
#Movies Distribution
edx %>%
  dplyr::count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 50, color = "black", fill="darkblue") +
  scale_x_log10() +
  labs(title="Movies Distribution",x="Nr Ratings by Movie (log10)", y = "count")
```



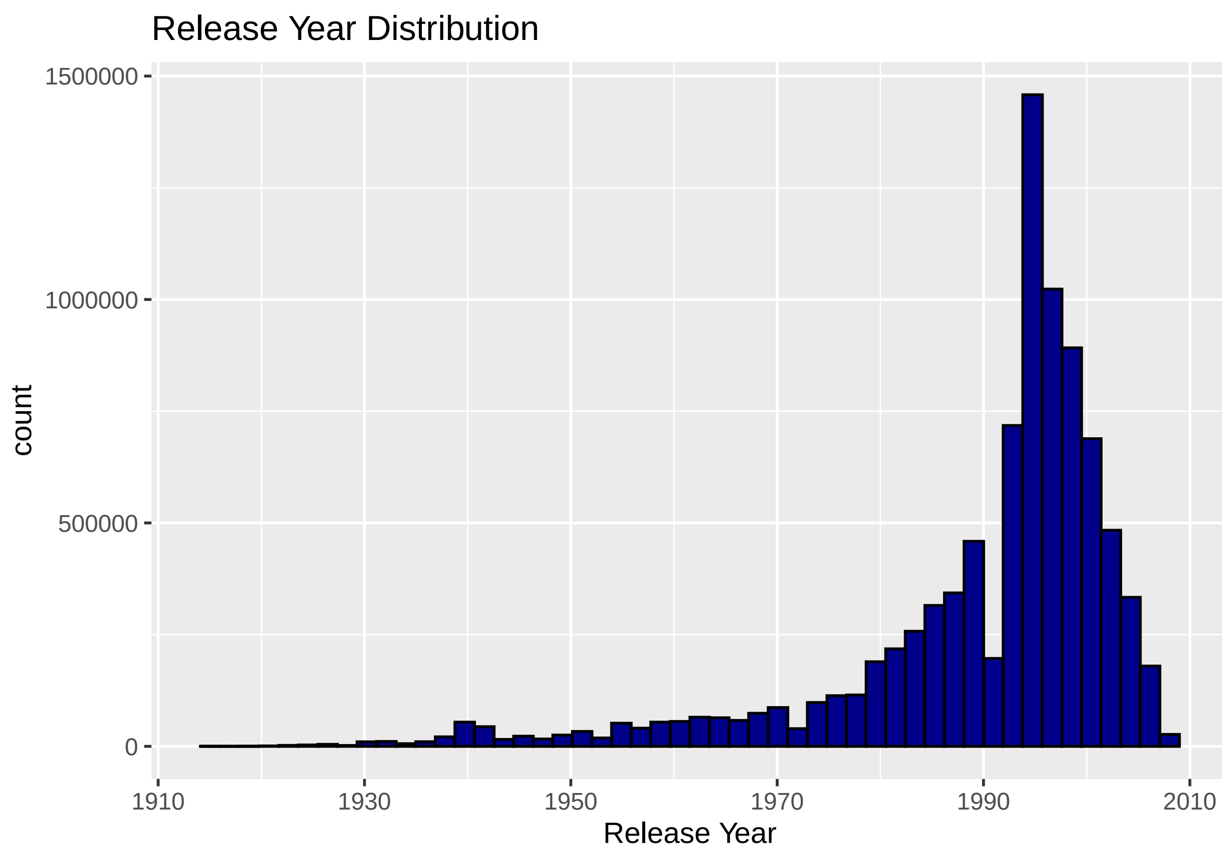

```
#Users Distribution
edx %>%
  dplyr::count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 50, color = "black", fill = "darkblue") +
  scale_x_log10() +
  labs(title="Users Distribution",x="Nr Ratings by User (log10)", y = "count")
```



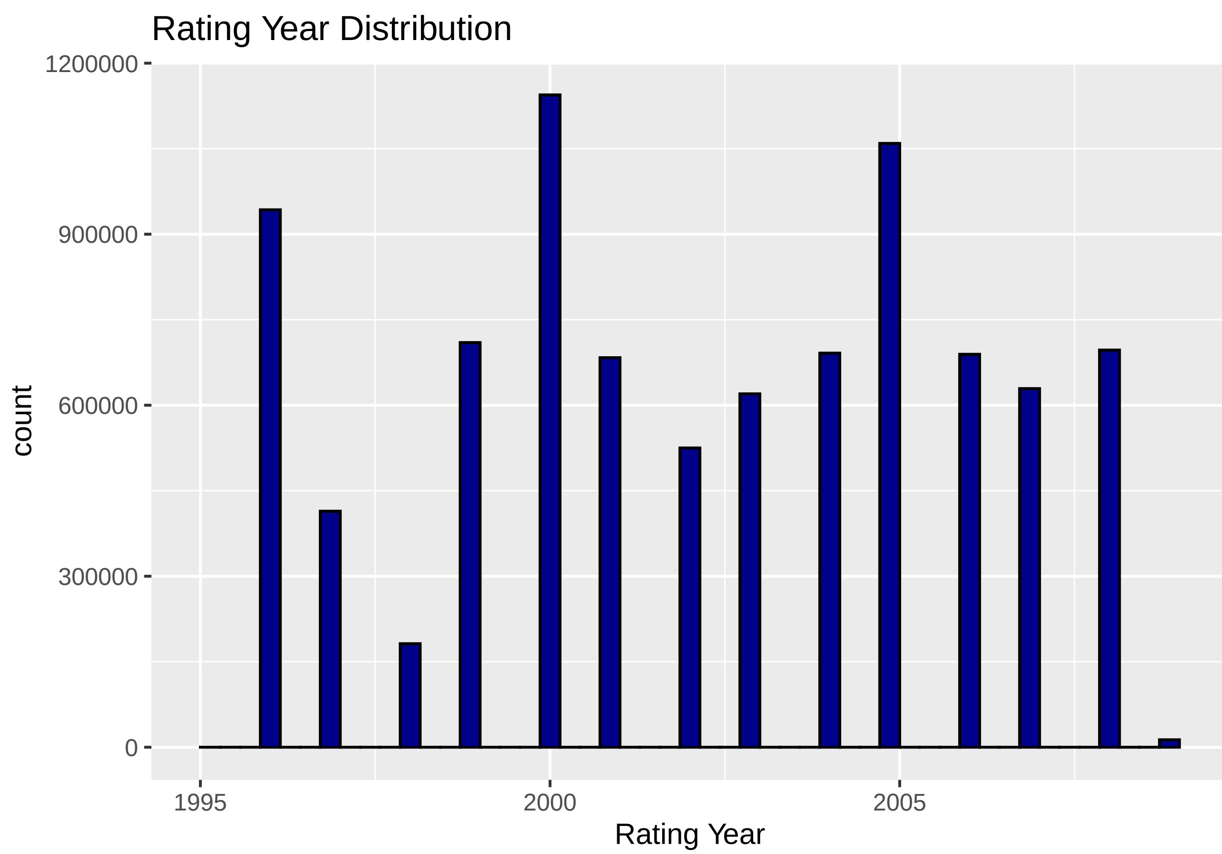

```r
#Genre Distribution
edx %>%
  dplyr::count(genres) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 50, color = "black", fill = "darkblue") +
  scale_x_log10() +
  labs(title="Genres Distribution",x="Nr Ratings by Genres (log10)", y = "count")
```



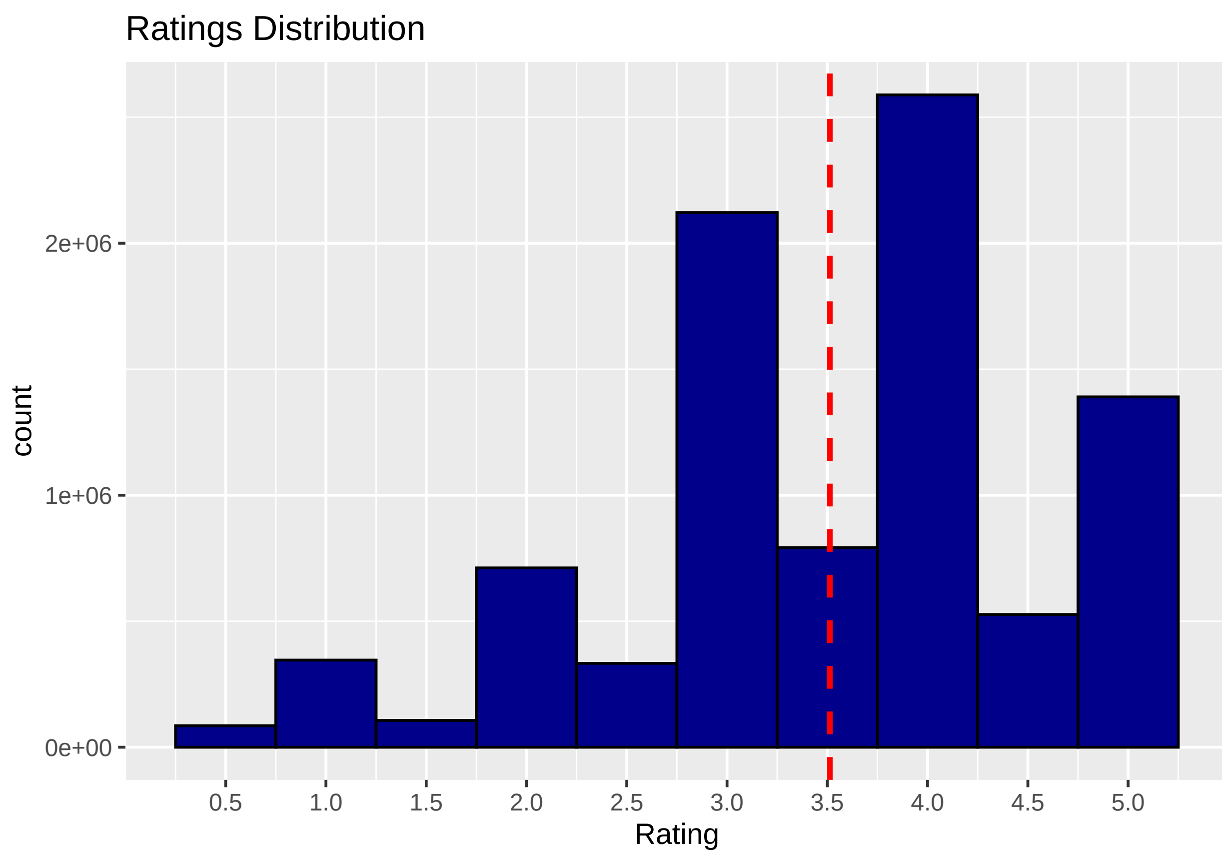

```
#Release Year Distribtion
edx%>%ggplot(aes(ReleaseYear))+
  geom_histogram(bins = 50, color = "black", fill = "darkblue") +
  labs(title="Release Year Distribution",x="Release Year", y = "count")
```




```
#Rating Year Distribtion
edx%>%ggplot(aes(RatingYear))+
  geom_histogram(bins = 50, color = "black", fill = "darkblue") +
  labs(title="Rating Year Distribution",x="Rating Year", y = "count")
```




```r
#Ratings Distribution
edx%>%ggplot(aes(rating))+
  geom_histogram(col='black',bins=10, fill = "darkblue")+
  scale_x_continuous(breaks = seq(0.5,5,0.5))+
  geom_vline(xintercept =mean(edx$rating), color="red", linetype="dashed", size=1)+
  labs(title="Ratings Distribution",x="Rating", y = "count")
```



Exploratory Data Analysis Findings The above distributions show that users, movies, genre, year of release and year of raing have an effect on rating. We see the following -

1) Some movies are more popular than others
2) Some users rate more movies than others
3) Some genres are more popular than others
4) Recently released movies are more popular compared to old moveies
5) Some years have more ratings than others

We also see that the average of the ratings is 3.5

Model Training: Partitioning Training and Test Sets The edx dataset is partitioned into training (90%) and test (10%) datasets

```
# Splitting edx dataset into training and test datasets
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.1, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]

test_set <- test_set %>%
```

```r
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

Model 1: Average Model 1 always predicts average as the rating.

```r
mu <- mean(train_set$rating)
Model1_RSME <- RMSE(test_set$rating, mu)
Models_RMSE <- tibble(Model = "Model 1: Average", RMSE = Model1_RSME)
datatable(Models_RMSE)
```

Show 10 entries Search:

	Model	RMSE
1	Model 1: Average	1.06113503425985

Showing 1 to 1 of 1 entries Previous 1 Next

Model 2: Movie Effect Model 2 takes into account the movie effect and adds +/- a value to the average depending on the movie. This is the same as using lm function (fitting linear models), but takes less computation time without crashing R.

```r
Movie_Bias <- train_set %>%  #Calculating movie bias
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu))

Model2_predicted_ratings <- mu + test_set %>%  #Testing the model
  left_join(Movie_Bias, by='movieId') %>%
  .$b_m

Model2_RMSE <- RMSE(test_set$rating, Model2_predicted_ratings) # Evaluating the RMSE of the model

Models_RMSE  <- rbind(Models_RMSE, c("Model 2: Movie Effect", Model2_RMSE)) #Storing the RMSE
datatable(Models_RMSE)
```

Show 10 entries Search:

	Model	RMSE
1	Model 1: Average	1.06113503425985
2	Model 2: Movie Effect	0.944156787180085

Showing 1 to 2 of 2 entries Previous 1 Next

Model 3: Movie + User Effect Model 3 is the same as model 2, but also takes into account the user effect. So, it adds another +/- value depending on the user.

```
User_Bias <- train_set %>%  #Calculating user bias
  left_join(Movie_Bias, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_m))

Model3_predicted_ratings <- test_set %>%  #Testing the model
  left_join(Movie_Bias, by='movieId') %>%
  left_join(User_Bias, by='userId') %>%
  mutate(predicted_ratings = mu + b_m + b_u) %>%
  .$predicted_ratings

Model3_RMSE <- RMSE(test_set$rating, Model3_predicted_ratings) # Evaluating the RMSE of the model

Models_RMSE  <- rbind(Models_RMSE, c("Model 3: Movie + User Effect", Model3_RMSE)) #Storing the RMSE
datatable(Models_RMSE)
```

Show 10 entries Search:

	Model	RMSE
1	Model 1: Average	1.06113503425985
2	Model 2: Movie Effect	0.944156787180085
3	Model 3: Movie + User Effect	0.865973629550804

Showing 1 to 3 of 3 entries Previous 1 Next

Model 4: Movie + User + Release Year Effect Model 4 is the same as model 3, but also takes into account the release year effect.

```
ReleaseYear_Bias <- train_set %>%  #Calculating release year bias
  left_join(Movie_Bias, by='movieId') %>%
  left_join(User_Bias, by='userId') %>%
  group_by(ReleaseYear) %>%
  summarize(b_y = mean(rating - mu - b_m - b_u))

Model4_predicted_ratings <- test_set %>% #Testing the model
  left_join(Movie_Bias, by='movieId') %>%
  left_join(User_Bias, by='userId') %>%
  left_join(ReleaseYear_Bias, by='ReleaseYear') %>%
  mutate(predicted_ratings = mu + b_m + b_u +b_y) %>%
  .$predicted_ratings

Model4_RMSE <- RMSE(test_set$rating, Model4_predicted_ratings) # Evaluating the RMSE of the model

Models_RMSE  <- rbind(Models_RMSE, c("Model 4: Movie + User + Release Year Effect", Model4_RMSE)) #Stor
datatable(Models_RMSE)
```

Show 10 entries

Search:

	Model	RMSE
1	Model 1: Average	1.06113503425985
2	Model 2: Movie Effect	0.944156787180085
3	Model 3: Movie + User Effect	0.865973629550804
4	Model 4: Movie + User + Release Year Effect	0.865602279838595

Showing 1 to 4 of 4 entries

Previous 1 Next

Model 5: Movie + User + Release Year + Genre Effect Model 5 is the same as model 4, but also takes into account the genre effect.

```
Genre_Bias <- train_set %>%  #Calculating genre bias
  left_join(Movie_Bias, by='movieId') %>%
  left_join(User_Bias, by='userId') %>%
  left_join(ReleaseYear_Bias, by='ReleaseYear') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_m - b_u - b_y))

Model5_predicted_ratings <- test_set %>% #Testing the model
  left_join(Movie_Bias, by='movieId') %>%
  left_join(User_Bias, by='userId') %>%
  left_join(ReleaseYear_Bias, by='ReleaseYear') %>%
  left_join(Genre_Bias, by='genres') %>%
  mutate(predicted_ratings = mu + b_m + b_u +b_y +b_g) %>%
  .$predicted_ratings

Model5_RMSE <- RMSE(test_set$rating, Model5_predicted_ratings) # Evaluating the RMSE of the model

Models_RMSE  <- rbind(Models_RMSE, c("Model 5: Movie + User + Release Year +Genre  Effect", Model5_RMSE)
datatable(Models_RMSE)
```

Show 10 entries | Search:

	Model	RMSE
1	Model 1: Average	1.06113503425985
2	Model 2: Movie Effect	0.944156787180085
3	Model 3: Movie + User Effect	0.865973629550804
4	Model 4: Movie + User + Release Year Effect	0.865602279838595
5	Model 5: Movie + User + Release Year +Genre Effect	0.865319173402735

Showing 1 to 5 of 5 entries | Previous 1 Next

Model 6: Movie + User + Release Year + Genre + Rating Year Effect Model 6 is the same as model 6, but also takes into account the rating year effect.

```
RatingYear_Bias <- train_set %>%  #Calculating rating year bias
  left_join(Movie_Bias, by='movieId') %>%
  left_join(User_Bias, by='userId') %>%
  left_join(ReleaseYear_Bias, by='ReleaseYear') %>%
  left_join(Genre_Bias, by='genres') %>%
  group_by(RatingYear) %>%
  summarize(b_r = mean(rating - mu - b_m - b_u - b_y - b_g))

Model6_predicted_ratings <- test_set %>% #Testing the model
  left_join(Movie_Bias, by='movieId') %>%
  left_join(User_Bias, by='userId') %>%
  left_join(ReleaseYear_Bias, by='ReleaseYear') %>%
  left_join(Genre_Bias, by='genres') %>%
  left_join(RatingYear_Bias, by='RatingYear') %>%
  mutate(predicted_ratings = mu + b_m + b_u +b_y +b_g + b_r) %>%
  .$predicted_ratings

Model6_RMSE <- RMSE(test_set$rating, Model6_predicted_ratings) # Evaluating the RMSE of the model

Models_RMSE  <- rbind(Models_RMSE, c("Model 6: Movie + User + Release Year + Genre + Rating Year Effect
datatable(Models_RMSE)
```

Show 10 entries Search:

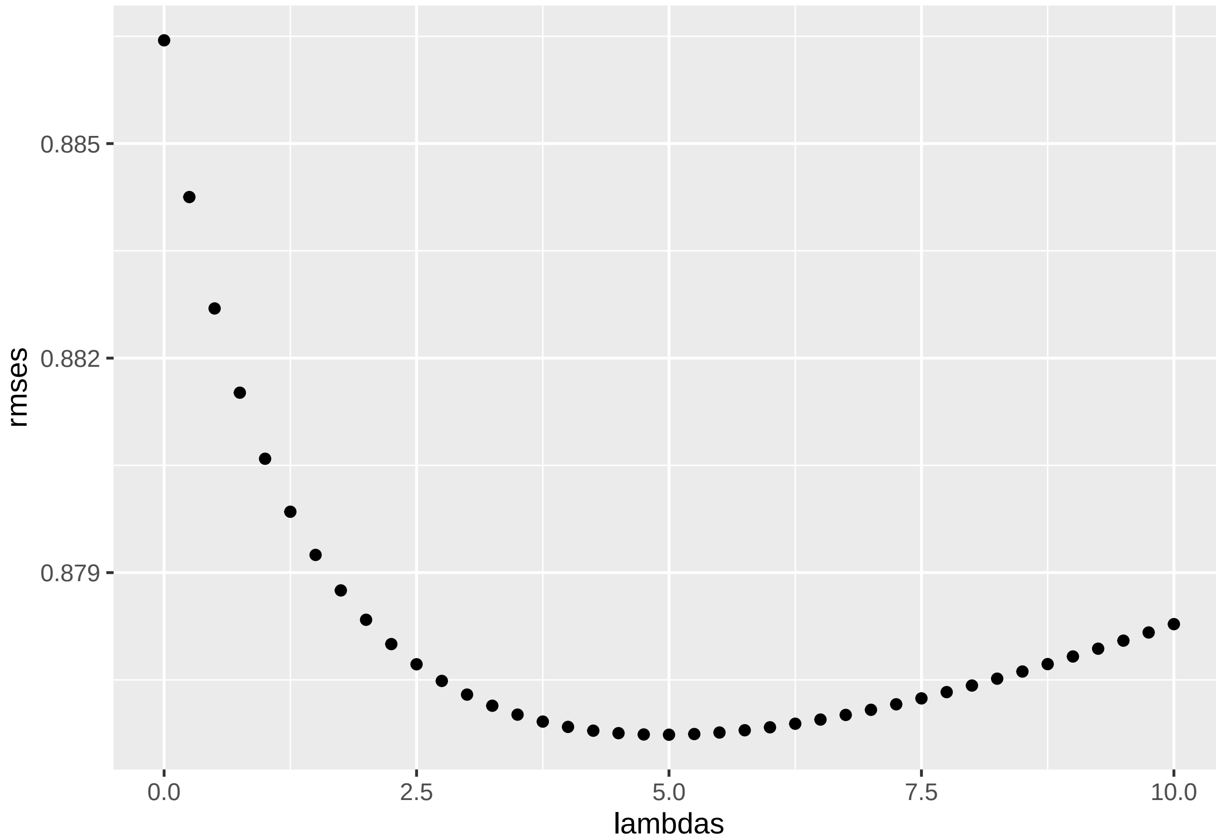
	Model	RMSE
1	Model 1: Average	1.06113503425985
2	Model 2: Movie Effect	0.944156787180085
3	Model 3: Movie + User Effect	0.865973629550804
4	Model 4: Movie + User + Release Year Effect	0.865602279838595
5	Model 5: Movie + User + Release Year +Genre Effect	0.865319173402735
6	Model 6: Movie + User + Release Year + Genre + Rating Year Effect	0.865188041154053

Showing 1 to 6 of 6 entries Previous 1 Next

Model 7: Regularization using k-fold Cross validation Model 7 is also a linear model taking into account the effects of movies, user, genres, release year and rating year. But, regularilization is also done. Regularilization is the process of adding a penalty term to the function to account for large estimates resulting from small sample sizes. Lamda is a tuning parameter and k-fold cross validation is done on the training dataset (k=5) to choose it. The training dataset is split into 5 sets and model training is done 5 times by combing various 4 sets each time and testing is done on the remaining set. The RMSE is the average of the RMSE from 5 times. This is repeated for each lamda value from 0 to 10 with interval of 0.25. So, in total 200 models are trained and tested to tune the parameter. Once the parameter value is chosen, the RMSE value is computed using the test dataset.

```
cv <- createFolds(train_set$rating, k=5,  list = TRUE, returnTrain = TRUE) #Splitting the training data
lambdas <- seq(0, 10, 0.25)
ks <- seq(1, 5, 1)

rmses <- sapply(lambdas, function(l){ #function for applying various lamda values

  rmses_kfold<- sapply(ks, function(k){ #function for applying various k values
    mu <- mean(train_set$rating[-cv[[k]]])
    Movie_Bias_Reg <- train_set[-cv[[k]],] %>% #Calculating movie bias with regularization
      group_by(movieId) %>%
      summarize(b_m_s = sum(rating - mu)/(n()+l))

    User_Bias_Reg <- train_set[-cv[[k]],] %>%  #Calculating user bias with regularization
      left_join(Movie_Bias_Reg, by='movieId') %>%
      group_by(userId) %>%
      summarize(b_u_s = sum(rating - mu - b_m_s)/(n()+l))

    ReleaseYear_Bias_Reg <- train_set[-cv[[k]],]%>% #Calculating release year bias with regularization
      left_join(Movie_Bias_Reg, by='movieId') %>%
```

```
      left_join(User_Bias_Reg, by='userId') %>%
      group_by(ReleaseYear) %>%
      summarize(b_y_s = sum(rating - mu - b_m_s - b_u_s)/(n()+l))

    Genre_Bias_Reg <- train_set[-cv[[k]],] %>%  #Calculating genre bias with regularization
      left_join(Movie_Bias_Reg, by='movieId') %>%
      left_join(User_Bias_Reg, by='userId') %>%
      left_join(ReleaseYear_Bias_Reg, by='ReleaseYear') %>%
      group_by(genres) %>%
      summarize(b_g_s = sum(rating - mu - b_m_s - b_u_s - b_y_s)/(n()+l))

    RatingYear_Bias_Reg <- train_set[-cv[[k]],] %>% #Calculating rating year bias with regularization
      left_join(Movie_Bias_Reg, by='movieId') %>%
      left_join(User_Bias_Reg, by='userId') %>%
      left_join(ReleaseYear_Bias_Reg, by='ReleaseYear') %>%
      left_join(Genre_Bias_Reg, by='genres') %>%
      group_by(RatingYear) %>%
      summarize(b_r_s = sum(rating - mu - b_m_s - b_u_s - b_y_s - b_g_s)/(n()+l))

    predicted_ratings <- train_set[cv[[k]],] %>% #Testing the model
      left_join(Movie_Bias_Reg, by='movieId') %>%
      left_join(User_Bias_Reg, by='userId') %>%
      left_join(ReleaseYear_Bias_Reg, by='ReleaseYear') %>%
      left_join(Genre_Bias_Reg, by='genres') %>%
      left_join(RatingYear_Bias_Reg, by='RatingYear') %>%
      mutate(predicted_ratings=  mu + b_m_s + b_u_s +b_y_s +b_g_s + b_r_s, predicted_ratings = coalesce
      .$predicted_ratings
    return(RMSE(predicted_ratings, train_set$rating[cv[[k]]])) #Evaluating the RMSE of the model
    })

  return(mean( rmses_kfold)) #Average of RMSE values
})

qplot(lambdas, rmses)
```




```r
l <- lambdas[which.min(rmses)] #Choosing lamda value with minimum rmse

mu <- mean(train_set$rating) #Calculating movie bias with regularization with chosen lamda value
Movie_Bias_Reg <- train_set%>%
  group_by(movieId) %>%
  summarize(b_m_s = sum(rating - mu)/(n()+l))

User_Bias_Reg <- train_set%>% #Calculating user bias with regularization with chosen lamda value
  left_join(Movie_Bias_Reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u_s = sum(rating - mu - b_m_s)/(n()+l))

ReleaseYear_Bias_Reg <- train_set%>% #Calculating release bias with regularization with chosen lamda va
  left_join(Movie_Bias_Reg, by='movieId') %>%
  left_join(User_Bias_Reg, by='userId') %>%
  group_by(ReleaseYear) %>%
  summarize(b_y_s = sum(rating - mu - b_m_s - b_u_s)/(n()+l))

Genre_Bias_Reg <- train_set%>% #Calculating genre bias with regularization with chosen lamda value
  left_join(Movie_Bias_Reg, by='movieId') %>%
  left_join(User_Bias_Reg, by='userId') %>%
  left_join(ReleaseYear_Bias_Reg, by='ReleaseYear') %>%
  group_by(genres) %>%
  summarize(b_g_s = sum(rating - mu - b_m_s - b_u_s - b_y_s)/(n()+l))

RatingYear_Bias_Reg <- train_set%>% #Calculating rating year bias with regularization with chosen lamda
```

```r
  left_join(Movie_Bias_Reg, by='movieId') %>%
  left_join(User_Bias_Reg, by='userId') %>%
  left_join(ReleaseYear_Bias_Reg, by='ReleaseYear') %>%
  left_join(Genre_Bias_Reg, by='genres') %>%
  group_by(RatingYear) %>%
  summarize(b_r_s = sum(rating - mu - b_m_s - b_u_s - b_y_s - b_g_s)/(n()+l))

Model7_predicted_ratings <- test_set%>% #Testing the model
  left_join(Movie_Bias_Reg, by='movieId') %>%
  left_join(User_Bias_Reg, by='userId') %>%
  left_join(ReleaseYear_Bias_Reg, by='ReleaseYear') %>%
  left_join(Genre_Bias_Reg, by='genres') %>%
  left_join(RatingYear_Bias_Reg, by='RatingYear') %>%
  mutate(predicted_ratings=  mu + b_m_s + b_u_s +b_y_s +b_g_s + b_r_s, predicted_ratings = coalesce(pre
  .$predicted_ratings


Model7_RMSE <- RMSE(test_set$rating, Model7_predicted_ratings) # Evaluating the RMSE of the model

Models_RMSE  <- rbind(Models_RMSE, c("Model 7: Regularized Movie + User + Release Year + Genre + Rating
datatable(Models_RMSE)
```

Show 10 entries

Search:

	Model	RMSE
1	Model 1: Average	1.06113503425985
2	Model 2: Movie Effect	0.944156787180085
3	Model 3: Movie + User Effect	0.865973629550804
4	Model 4: Movie + User + Release Year Effect	0.865602279838595
5	Model 5: Movie + User + Release Year +Genre Effect	0.865319173402735
6	Model 6: Movie + User + Release Year + Genre + Rating Year Effect	0.865188041154053
7	Model 7: Regularized Movie + User + Release Year + Genre + Rating Year Effect (using k-fold cross-validation)	0.864716319707901

Showing 1 to 7 of 7 entries

Previous 1 Next

Results

Since model 7 (Regularized Movie + User + Release Year + Genre + Rating Year Effect) gives the lowest RMSE, it has been chosen as the final model. The final model is evaluated by using the validation dataset, which hasn't been used previously.

```
Model7_predicted_ratings <- validation%>%
  left_join(Movie_Bias_Reg, by='movieId') %>%
  left_join(User_Bias_Reg, by='userId') %>%
  left_join(ReleaseYear_Bias_Reg, by='ReleaseYear') %>%
  left_join(Genre_Bias_Reg, by='genres') %>%
  left_join(RatingYear_Bias_Reg, by='RatingYear') %>%
  mutate(predicted_ratings=  mu + b_m_s + b_u_s +b_y_s +b_g_s + b_r_s, predicted_ratings = coalesce(pred
  .$predicted_ratings


Final_Model_RMSE<- RMSE(validation$rating, Model7_predicted_ratings)
Final_Model_RMSE
```

```
## [1] 0.8644928
```

Conclusion

The final model chosen is a linear model using regularization by k-fold cross-validation with predictors as movie, user, genre, release year and rating year. The RMSE of this model is 0.8644928

The limitations of this model is that it does not group similar movies and similar users. By grouping similar movies and similar users, the machine learning models will have more data to train and the accuracy will be improved. If there is demographic data of the users, it can be used as predictors to group different kinds of users and also better associate users with genres.