Movielens Capstone Project Report

Pinar Asker

3/18/2021

Introduction

This project is one of the final projects in the Harvardx Professional Certificate in Data Science. The analyses in this project closely follow the Data Science textbook by Rafael Irizarry.

This project aims to create a movie recommendation system with the MovieLens dataset, which is publicly available in the dslabs package. Approximately 10 million movie ratings within the Movielens dataset were used for this project.

Executive Summary

Movielens dataset was analyzed to train a machine learning algorithm to predict user ratings on movies on the validation dataset.

The dataset consists of six variables which are "userId", "movieId", "rating" , "timestamp" "title" and "genres". Models were developed by considering "rating "as the dependent variable and remaining variables as the predictor variables.

As a first step, the dataset was analyzed to decide that the data is tidy, with every row represent only one observation and made sure that there is no null values in the dataset.

Before implementing further steps, the Movielens dataset was divided into validation and edx (train set) datasets. The validation set was not used during model development but used only in the final model. Therefore, to use a test dataset during model development,the edx data set (train data) was partitioned into train_edx dataset (80%) and test_edx dataset (20%).This is the percentage which is a frequently used threshold in data partitioning. Models were developed on the train_edx and tested on the test_edx during model development.

The project's ultimate aim is to develop the final model that minimizes residual mean squared error (RMSE), and the expected value of RMSE is to be below **0.86449**. The RMSE formula is:

$$RMSE = \sqrt(\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2)$$

Before developing the model, the effects of the variables (movie, user, time, and genres) were visualized with graphs to show how significant these effects are on the rating. Four models were developed with the effects, which seem visually important, to achieve the minimum RMSE. As the further step, penalized least squares were applied to restrain total variability of the effects' size, and the best tuning parameter was determined through cross-validation. The final model was developed with an RMSE value is **0.86445**.

Data Loading

First, the data set was examined to determine whether it was tidy or not. As shown below, each row represents each column, and the dataset looks tidy. Additionally, each column is appropriately classed.

```
## # A tibble: 10,000,054 x 6
##    userId movieId rating timestamp title                genres
##     <int>   <dbl>  <dbl>     <int> <chr>                <chr>
##  1      1     122      5 838985046 Boomerang (1992)     Comedy|Romance
##  2      1     185      5 838983525 Net, The (1995)      Action|Crime|Thriller
##  3      1     231      5 838983392 Dumb & Dumber (199~ Comedy
##  4      1     292      5 838983421 Outbreak (1995)      Action|Drama|Sci-Fi|Thri~
##  5      1     316      5 838983392 Stargate (1994)      Action|Adventure|Sci-Fi
##  6      1     329      5 838983392 Star Trek: Generat~ Action|Adventure|Drama|S~
##  7      1     355      5 838984474 Flintstones, The (~ Children|Comedy|Fantasy
##  8      1     356      5 838983653 Forrest Gump (1994) Comedy|Drama|Romance|War
##  9      1     362      5 838984885 Jungle Book, The (~ Adventure|Children|Roman~
## 10      1     364      5 838983707 Lion King, The (19~ Adventure|Animation|Chil~
## # ... with 10,000,044 more rows
```

Data Partitioning

Before moving on exploratory data analysis and visualization, the dataset was first split into the validation and edx dataset (training dataset).Validation set is 10% of movielens dataset.

After data partitioning, the null values in the dataset were checked and made sure that there are no null values in the edx dataset.

```
sum(is.na(edx))
```

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

```
sum(is.na(edx))
```

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

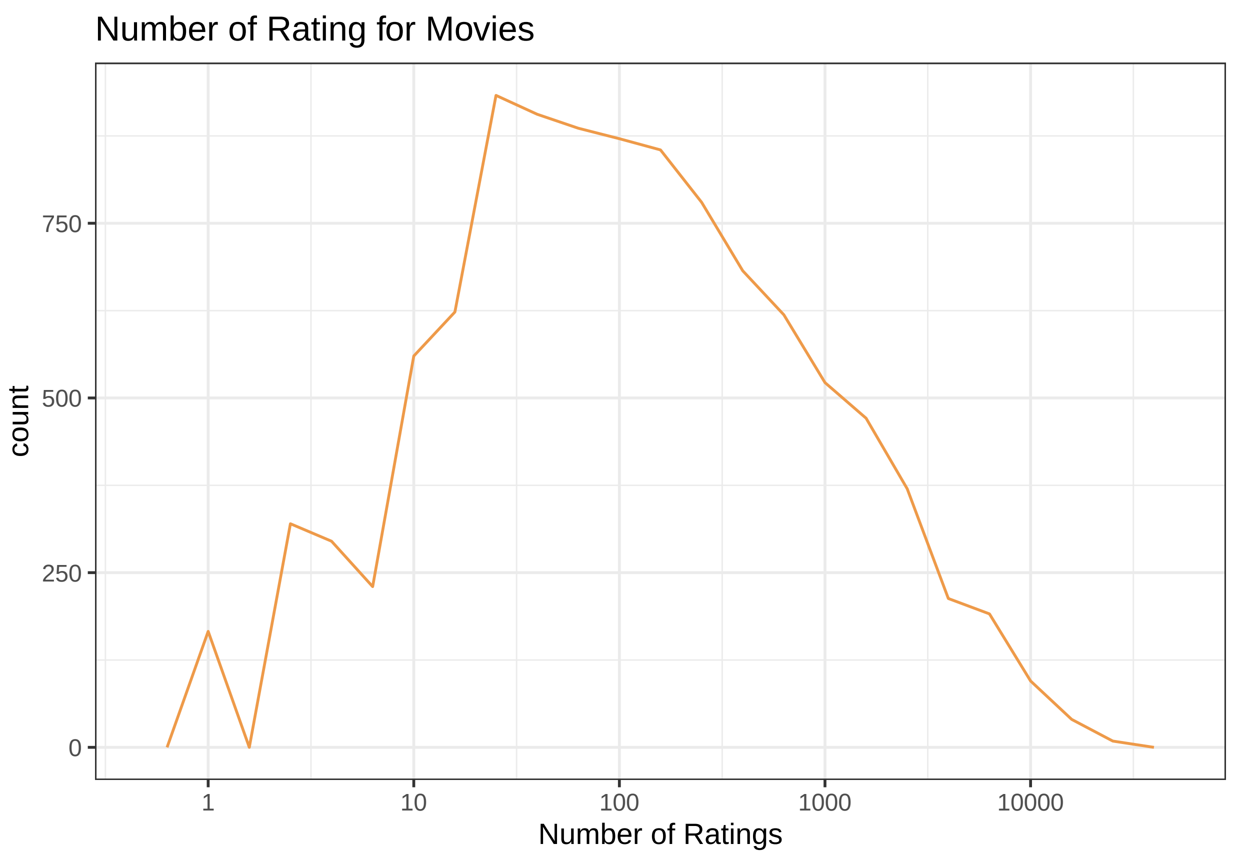
Next, the edx dataset was split into train_edx and test_edx by 0.8 and 0.2.

Visualization

Graphs based on different movies

Graph 1.MovieId and the Number of Rating

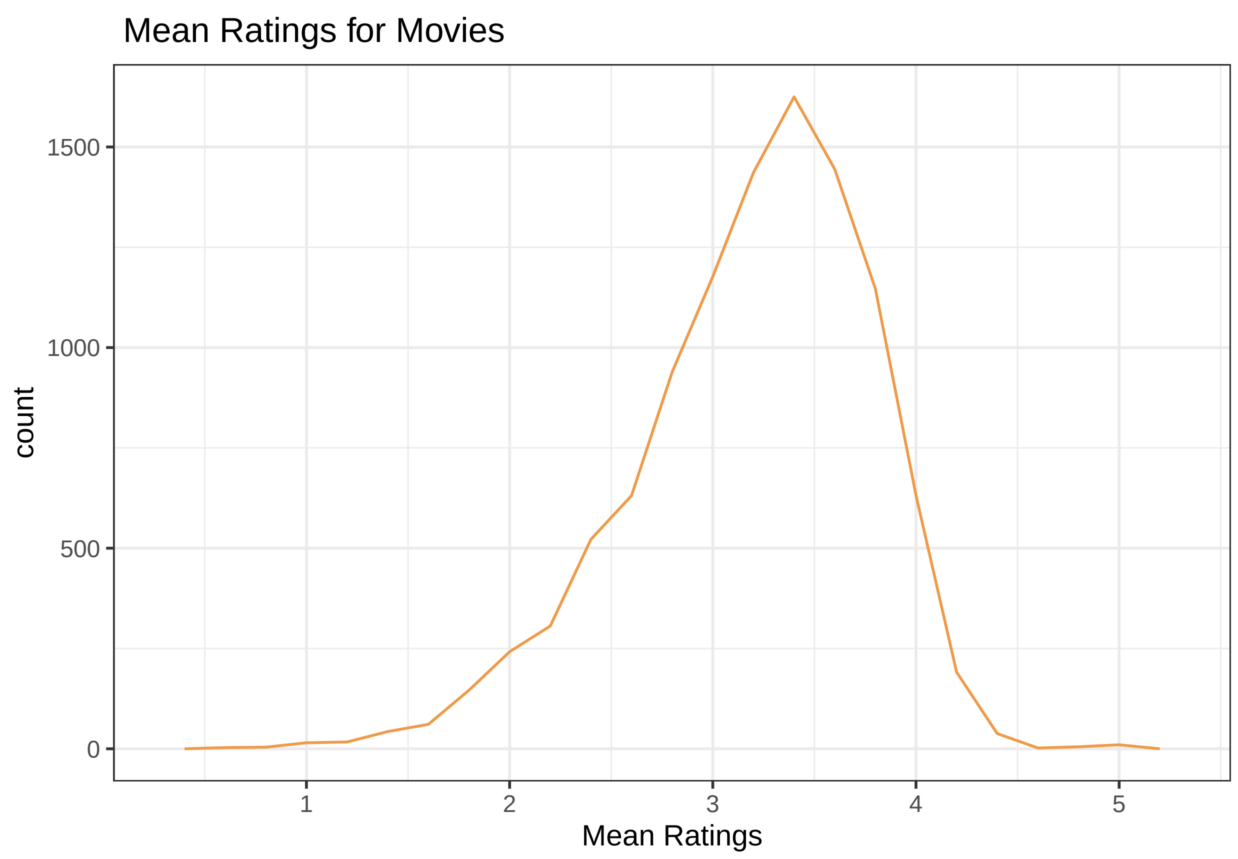
```
train_edx%>%group_by(movieId)%>%
  summarize(n=n())%>%ggplot(aes(n))+
  geom_freqpoly(color='tan2',bins = 30, binwidth = 0.2)+
  scale_x_log10()+
  xlab("Number of Ratings")+
  ggtitle("Number of Rating for Movies")+
  theme_bw()
```



As seen in the above, number of ratings substantially vary among different movies. Some movies have much greater number of rating than others.

Graph 2.MovieId and the Mean Rating

```
train_edx%>%group_by(movieId)%>%
  summarize(mean_rating=mean(rating))%>%ggplot(aes(mean_rating))+
  geom_freqpoly(color='tan2',bins = 30, binwidth = 0.2)+
  xlab("Mean Ratings")+
  ggtitle(" Mean Ratings for Movies")+
  theme_bw()
```

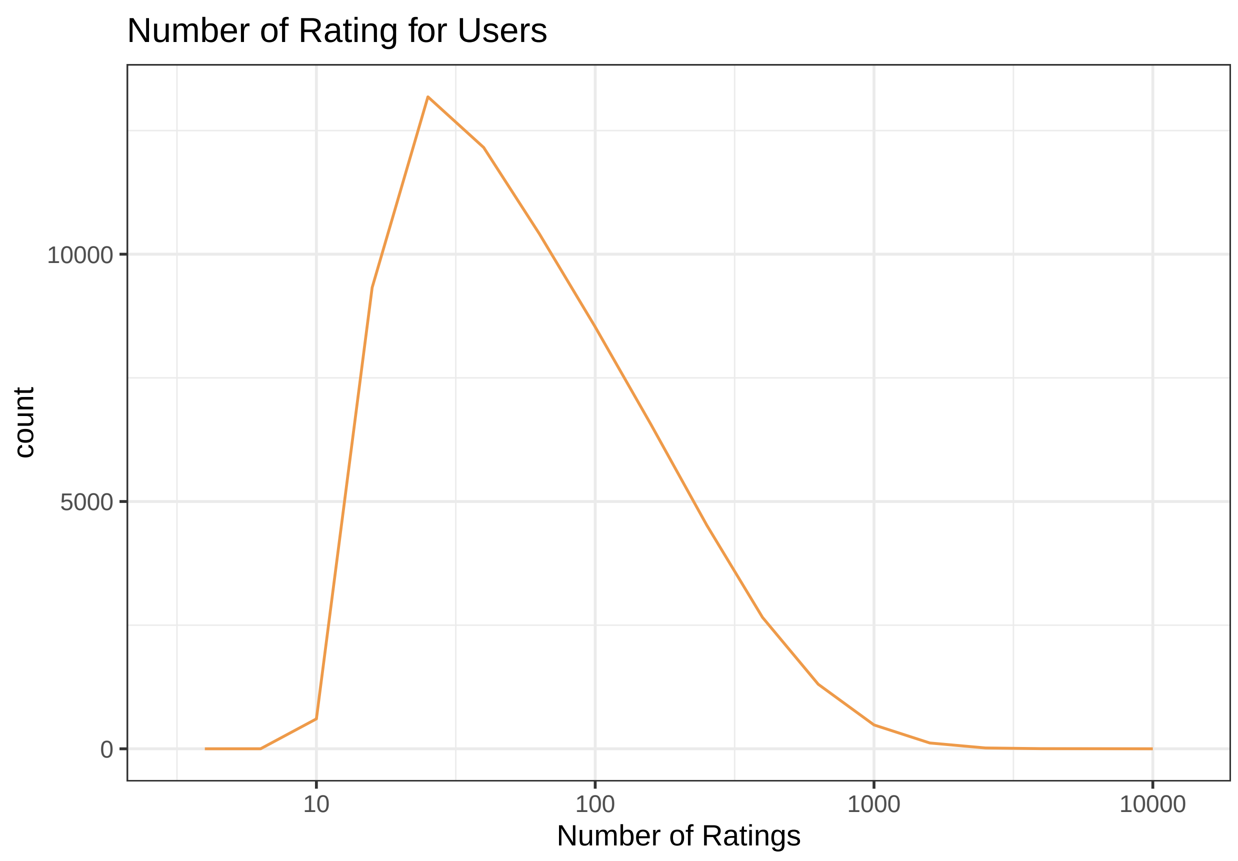


The chart above was created by taking the movies' ratings averages after grouped them with their id. In the graph above, it is shown that average ratings are changed significantly among movies. Therefore, the movie effect should be considered in the model development.

Graphs based on different users

Graph 3.UserId and the Number of Ratings

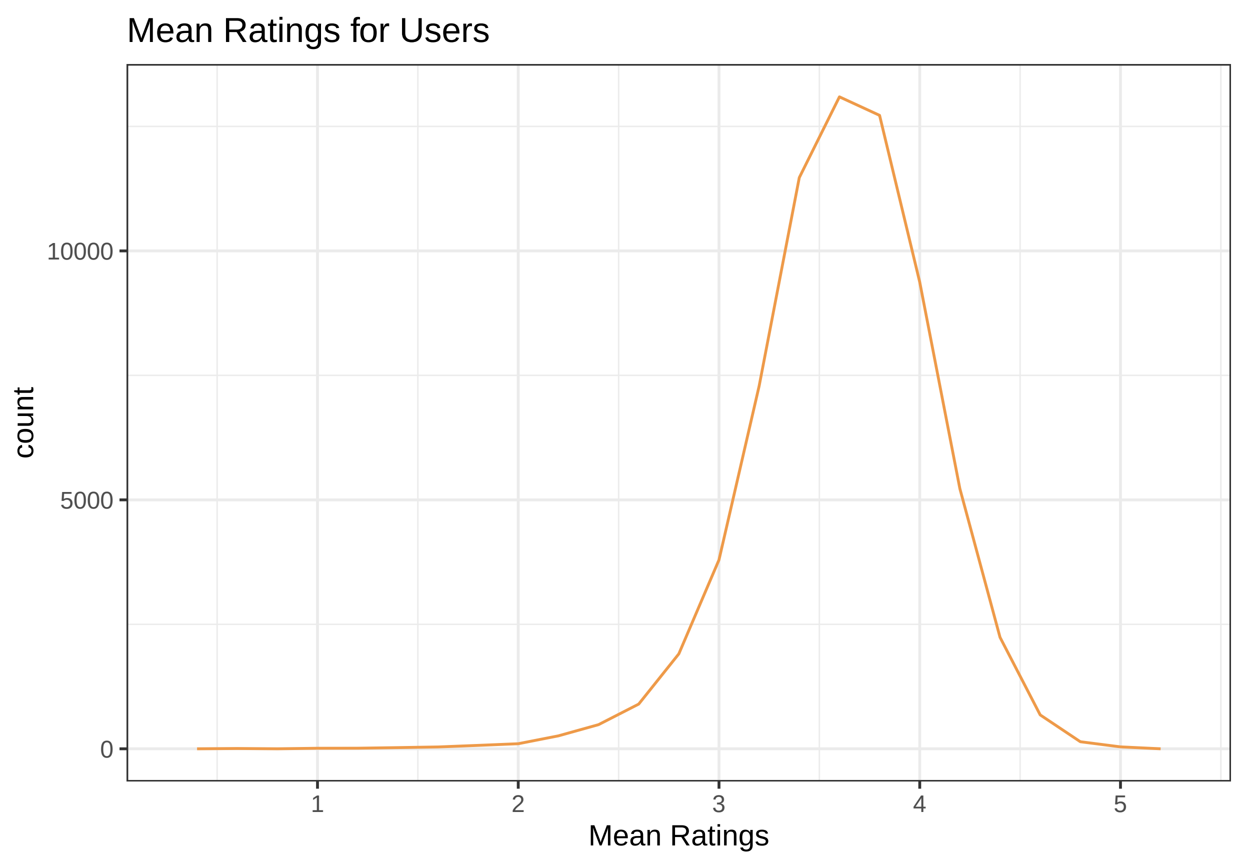
```
train_edx%>%group_by(userId)%>%
  summarize(n=n())%>%ggplot(aes(n))+
  geom_freqpoly(color='tan2',bins = 30, binwidth = 0.2)+
  scale_x_log10()+
  xlab("Number of Ratings")+
  ggtitle("Number of Rating for Users")+
  theme_bw()
```



As shown in the graph above, some users are more involved in rating than others. Therefore, there is a substantial difference between users in terms of the number of ratings.

Graph 4.UserId and the Mean Rating

```
train_edx%>%group_by(userId)%>%
  summarize(mean_rating=mean(rating))%>%ggplot(aes(mean_rating))+
  geom_freqpoly(color='tan2',bins = 30, binwidth = 0.2)+
  xlab("Mean Ratings")+
  ggtitle("Mean Ratings for Users")+
  theme_bw()
```



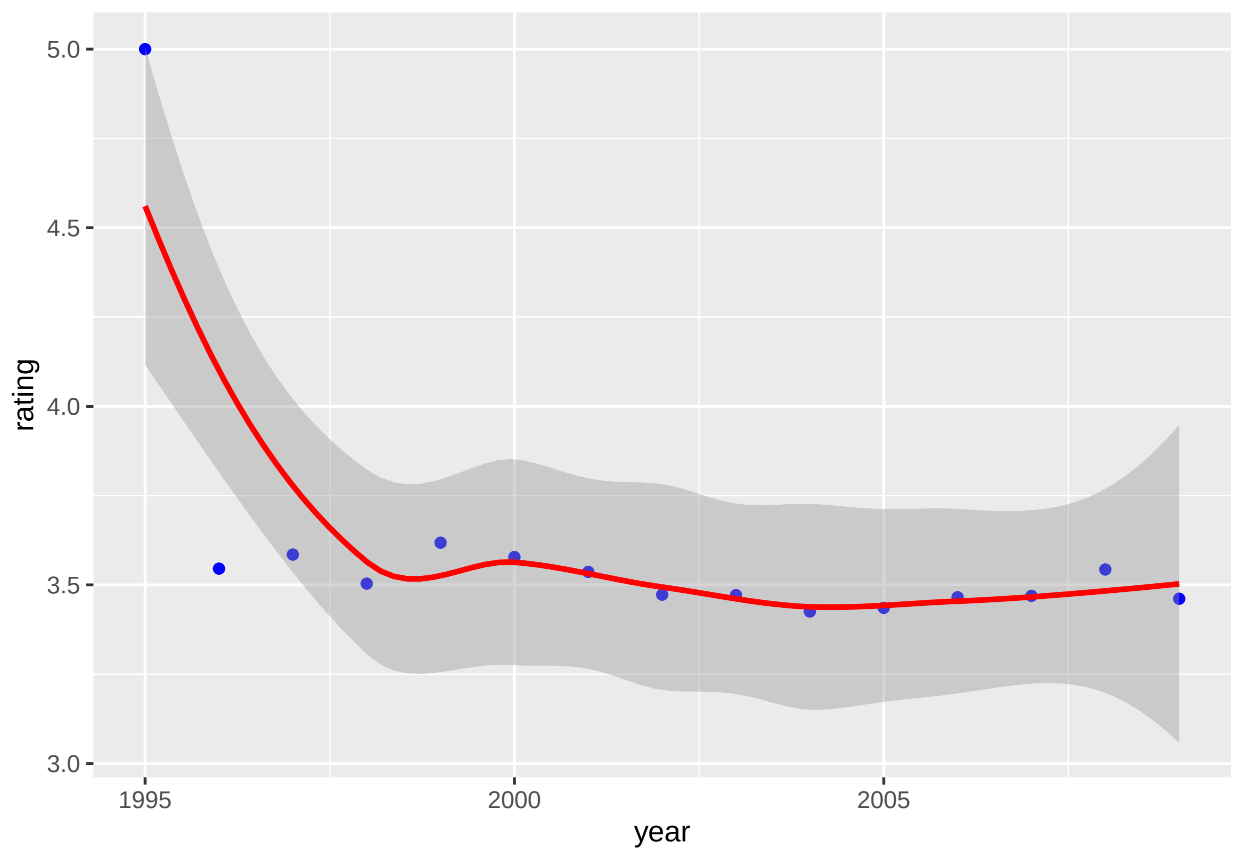
Therefore, there is also a substantial difference between users in terms of the average rating. The user's effect also should be included in the model development.

Graph based on time

Graph 5.Time(year and the average ratings)

```r
library(lubridate)
train_edx$timestamp=as_datetime(train_edx$timestamp)
train_edx %>% mutate(year=year(timestamp))%>%
  group_by(year) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(year, rating)) +
  geom_point(color="blue") +
  geom_smooth(color="red")
```

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

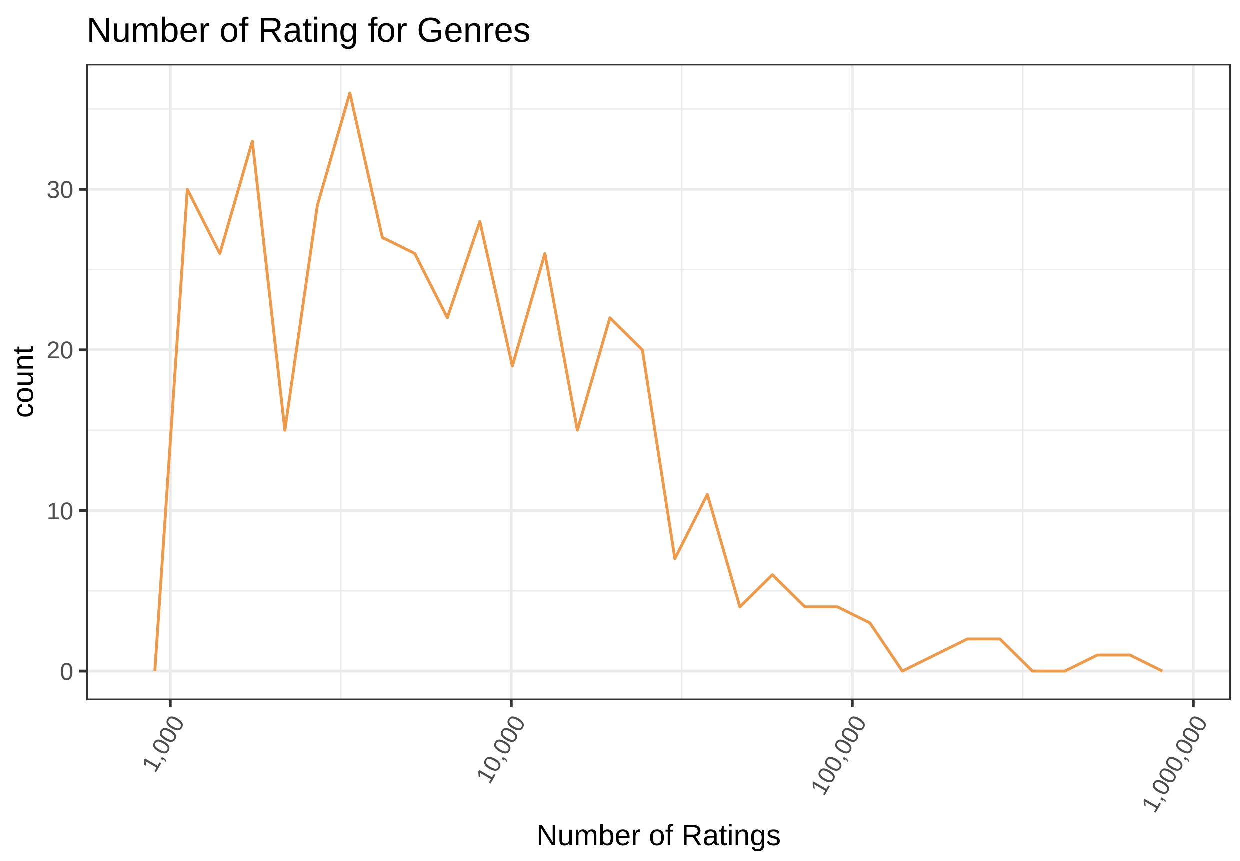


The chart above was created to show how the average rating changes yearly. As shown in the graph, the time(year) has no substantial effect on rating averages. Therefore, the time effect will not be added to the model.

Graphs based on genres

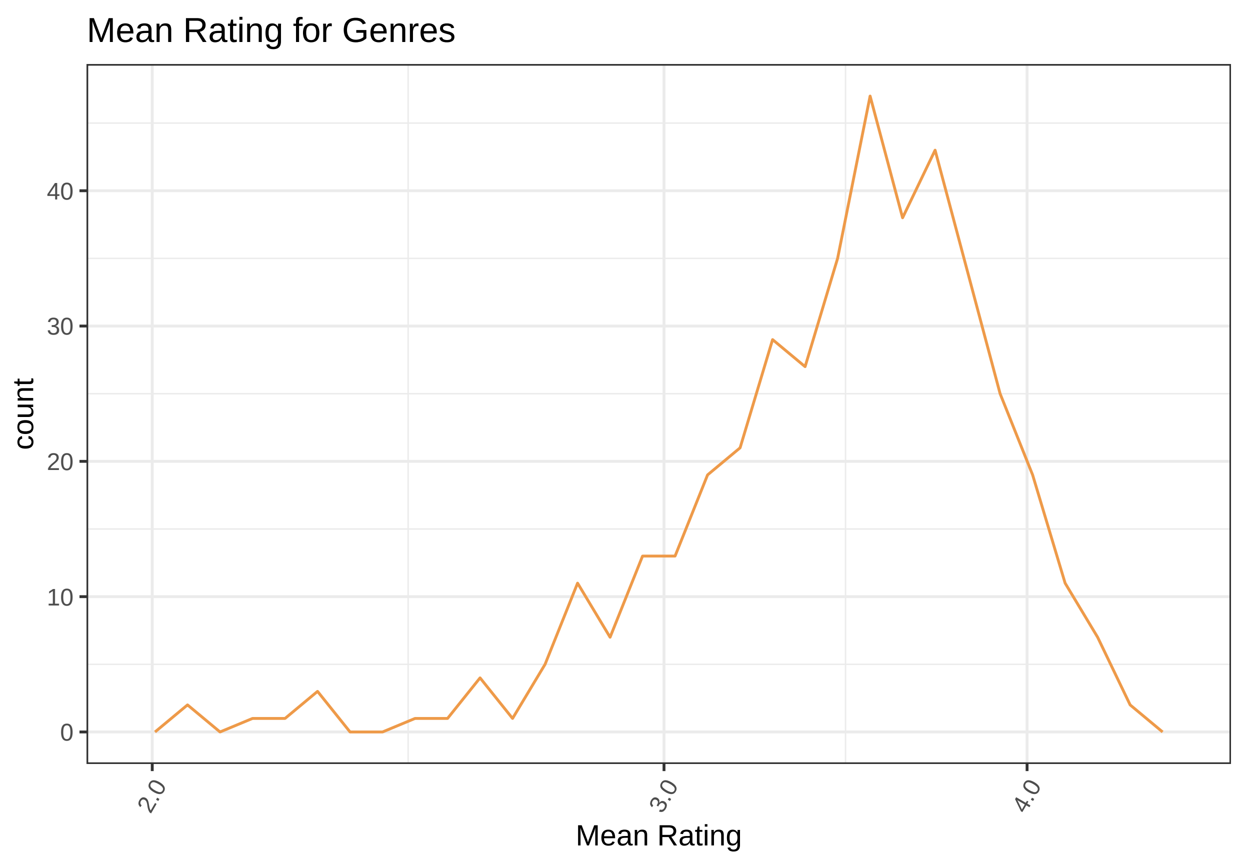
Graph 6.Genres and the Number of Ratings

```
train_edx%>%group_by(genres)%>%
  summarize(n=n())%>%
  filter(n>1000)%>%
  ggplot(aes(n))+
  geom_freqpoly(color='tan2',bins = 30)+
  scale_x_log10(labels=scales::comma)+
  xlab("Number of Ratings")+
  ggtitle("Number of Rating for Genres")+
  theme_bw()+
  theme(axis.text.x=element_text(angle=60,hjust=1))
```



Graph 7.Genres and the Mean Rating

```
train_edx%>%group_by(genres)%>%
  filter(n()>1000)%>%
  summarize(mean_rating=mean(rating))%>%
  ggplot(aes(mean_rating))+
  geom_freqpoly(color='tan2',bins = 30)+
  scale_x_log10(labels=scales::comma)+
  xlab("Mean Rating")+
  ggtitle("Mean Rating for Genres")+
  theme_bw()+
  theme(axis.text.x=element_text(angle=60,hjust=1))
```



Both graphs of genres show that genre also has an important effect on ratings' pattern (average and number of rating). Then, the genres' effect will be placed on the model.

Model Development

Model 1

Model 1 was built as formulated below:

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

$Y_{u,i}$ = Dependent variable (rating)

μ = Mean of the movie ratings

ϵ_u, i = Independent error centered at zero

This model predicts the rating by simply averaging movie ratings regardless of user.

```
## [1] 3.512482
```

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

```
RMSE1=RMSE(test_edx$rating,mu)
RMSE1
```

```
## [1] 1.059904
```

```
all_rmse=tibble(Model="Model_1", RMSE=RMSE1)
all_rmse%>%knitr::kable()
```

Model	RMSE
Model_1	1.059904

Since the RMSE is greater than 1, Model 1 predicts rating one star larger than actual rating values. Other effects should be considered to improve RMSE.

Model 2: Modeling Movie Effect

As shown in graph 1 and 2, different movie are rated differently. Therefore, the movie effect was added to the model. Model 2 formula becomes as shown below.

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

b_i = movie effect for movie i

b_i was calculated by the code below

```
b_i=train_edx%>%
  group_by(movieId)%>%
  summarize(b_i=mean(rating-mu))
```

RMSE for the model 2 is:

```
predicted_ratings2 <- test_edx %>%
left_join(b_i, by='movieId') %>%
mutate(pred=mu+b_i)%>%
.$pred
RMSE2= RMSE(predicted_ratings2, test_edx$rating)
RMSE2
```

```
## [1] 0.9437429
```

```
all_rmse=tibble(Model=c("Model_1","Model_2"), RMSE=c(RMSE1,RMSE2))
all_rmse%>%knitr::kable()
```

Model	RMSE
Model_1	1.0599043
Model_2	0.9437429

RMSE has been improved, although new models need to be developed to achieve desired RMSE value.

Model 3: User Effects

As shown in graph 3 and graph 4, some users involve in rating much more compared to some others. Therefore, the user effect may be added to the previous model to improve RMSE. The previous model was extended with users effect, and the new formula becomes:

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

b_u=user-specific effect, which was calculated as below:

```r
#adding b_u (user effects) to the previous model and creating model 3
b_u<-train_edx%>%
  left_join(b_i,by="movieId")%>%
  group_by(userId)%>%
  summarize(b_u=mean(rating-mu-b_i))
```

The previous model was augmented with the user-specific effect then RMSE was calculated with the new model (model 3).

RMSE for Model 3 is:

```r
#making prediction with model 3
predicted_ratings3<-test_edx%>%
  left_join(b_i,by="movieId")%>%
  left_join(b_u,by="userId")%>%
  mutate(pred=mu+b_i+b_u)%>%
  .$pred


#calculating rmse of model 3
RMSE3=RMSE(predicted_ratings3,test_edx$rating)
RMSE3 # 0.865932
```

```
## [1] 0.865932
```

There is an improvement in the RMSE.

```r
all_rmse<-data.frame(method=c("Model_1","Model_2","Model_3"),RMSE=c(RMSE1,RMSE2,RMSE3))
all_rmse%>%knitr::kable()
```

method	RMSE
Model_1	1.0599043
Model_2	0.9437429
Model_3	0.8659320

Model 4 Genres Effect

As shown in the graph 6 and 7, the genre also affects movie ratings. In other words, average rating and number of ratings vary among different genres. For that reason, the previous model enlarged with genres effect.

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

#making prediction with model 4
predicted_ratings4<-test_edx%>%
  left_join(b_i,by="movieId")%>%
  left_join(b_u,by="userId")%>%
  left_join(b_g,by="genres")%>%
  mutate(pred=mu+b_i+b_u+b_g)%>%
  .$pred
```

```
#calculating rmse of model 4
RMSE4=RMSE(predicted_ratings4,test_edx$rating)
RMSE4
```

```
## [1] 0.8655941
```

The genres effect improved the RMSE to 0.8655941.

```
all_rmse<-data.frame(method=c("Model_1","Model_2","Model_3","Model_4"),RMSE=c(RMSE1,RMSE2,RMSE3,RMSE4))
all_rmse%>%knitr::kable()
```

method	RMSE
Model_1	1.0599043
Model_2	0.9437429
Model_3	0.8659320
Model_4	0.8655941

Before proceeding, an important aspect of the data set should be considered. There may be high-rated movies with very few number of ratings in the data set, which may cause uncertainties due to the effect size. However, models developed so far do not take into account any uncertainty regarding the size. Therefore, a regularization approach should be applied to solve this problem.

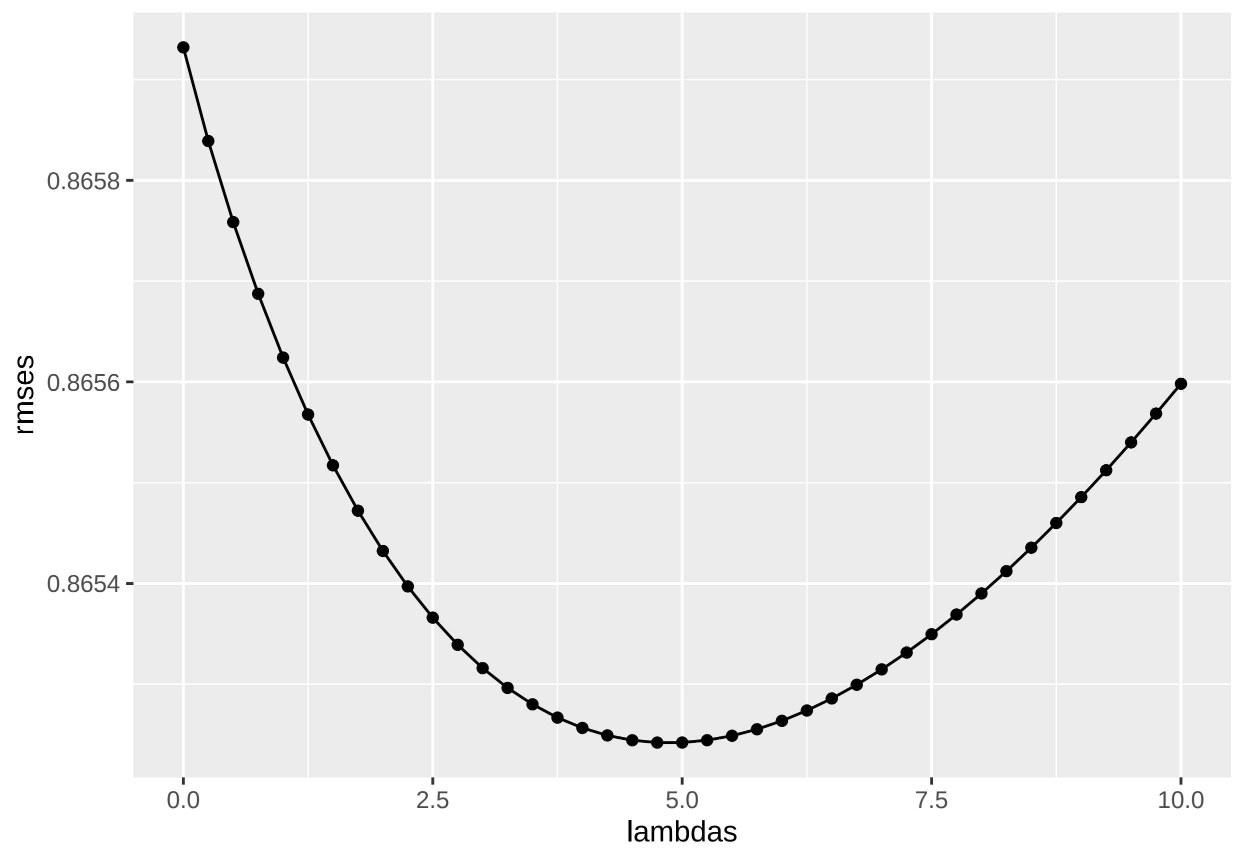
Regularization

Regularization allows to shrunk large estimates that are predicted with a small sample size. It penalizes the effects which has small number of ratings with the penalty term (λ). According to this approach, when sample size is small,the penalty term (λ) shrinks the estimates.

Penalized least squares were estimated with different tuning ( ) parameters to constraint uncertainties from effect size. Cross-validation was used to choose the best tuning parameter that minimizes RMSE.

Model 5- Regularized Movie,User and Genres Effect

```r
lambdas<-seq(0,10,0.25)
rmses<-sapply(lambdas,function(x){

  mu<-mean(train_edx$rating)

  b_i<-train_edx%>%
    group_by(movieId)%>%
    summarize(b_i=sum(rating-mu)/(n()+x))

  b_u<-train_edx%>%
    left_join(b_i,by="movieId")%>%
    group_by(userId)%>%
    summarize(b_u=sum(rating-b_i-mu)/(n()+x))

  predicted_rating_reg<-test_edx%>%
    left_join(b_i,by="movieId")%>%
    left_join(b_u,by="userId")%>%
    mutate(pred=mu+b_i+b_u)%>%
    pull(pred)

  return(RMSE(predicted_rating_reg,test_edx$rating))
})

qplot(lambdas,rmses,geom=c("point","line"))
```



The minimum RMSE generated with regularization is:

```
RMSE_final=min(rmses)
RMSE_final
```

```
## [1] 0.8652421
```

The RMSE improved to *0.8652421*

Results

The all RMSEs calculated so far are summarized in table below.

```
all_rmse<-data.frame(method=c("Model_1","Model_2","Model_3","Model_4","Model_Regularization"),RMSE=c(RMS
all_rmse%>%knitr::kable()
```

method	RMSE
Model_1	1.0599043
Model_2	0.9437429
Model_3	0.8659320
Model_4	0.8655941
Model_Regularization	0.8652421

The table above provides a summary of all models built so far. The first model built only with averages gives the RMSE greater than 1. RMSE decreases with the following models, and the Regularized Movie, User, and Genres Effect models calculated minimum RMSE. Model development was finalized with this model.

The lamda (λ) that produces the min RMSE is:

```
data.frame(lambdas,rmses)%>%filter(rmses==min(rmses))%>%.$lambdas
```

```
## [1] 4.75
```

The model will be finalized with the lambda (λ) 4.75, which yields the best RMSE.

Final Model and Conclusion

Finally, all edx datasets will be used to train the previous model, and the test RMSE will be calculated on the validation dataset.

```
mu<-mean(edx$rating)

b_i<-edx%>%
  group_by(movieId)%>%
  summarize(b_i=sum(rating-mu)/(n()+4.75))

b_u<-edx%>%
  left_join(b_i,by="movieId")%>%
  group_by(userId)%>%
  summarize(b_u=sum(rating-b_i-mu)/(n()+4.75))

b_g<-edx%>%
  left_join(b_i,by="movieId")%>%
  left_join(b_u,by="userId")%>%
  group_by(genres)%>%
  summarize(b_g=sum(rating-mu-b_i-b_u)/(n()+4.75))


predicted_rating_reg<-validation%>%
  left_join(b_i,by="movieId")%>%
  left_join(b_u,by="userId")%>%
  left_join(b_g,by="genres")%>%
  mutate(pred=mu+b_i+b_u+b_g)%>%
  pull(pred)

RMSE_final=RMSE(predicted_rating_reg,validation$rating)
RMSE_final
```

```
## [1] 0.8644514
```

The project's ultimate aim is to reduce RMSE to **0.86490** or below; the latest model calculated the RMSE as **0.8644514**, which exceeds the target RMSE and provides better prediction.

As the conclusion, Regularized Movie,User and Genres Effect (Model 5) linear model is suggested to predict movie rating since this model produces the minimum RMSE.