Movielens Report

vvorre

3/30/2020

Contents

1 Overview and Dataset

1.1 Introduction

Recommendation systems are extensively used in many commercial applications. Websites like Youtube, Amazon, Netflix use various recommender algorithms to suggest related videos, items and movies to users. A recommender system uses machine learning and statistical techniques to predict how a particular user would rate an item based on their preferences and ratings of similar users.

In this report, we study the 10M Movielens dataset provided by the GroupLens research lab to predict ratings of movies. We use some of the techniques developed during the Netflix Challenge. We observe that by using only the normalization of user and movie effects, a root-mean-square error (RMSE) of 0.8653488 was achieved. Normalization of time effect and genre effect did not contribute to any significant improvement in RMSE (0.8646861). Next, we calculated the residuals by subtracting the user, movie, time and genre effects from the ratings. The residuals are then trained using a matrix factorization method, specifically a Stochastic gradient descent algorithm (SGD), to further reduce the RMSE to .78. Furthermore, we train the ratings without any normalization using SGD and achieve a RMSE of .78. When contrasted with matrix factorization of residuals, the number of iterations for RMSE to converge has increased from 20 to 30.

1.2 Goal of the project

The 10M Movielens dataset consists of ratings of various movies by different users. In the dataset, each user has rated only a certain number of movies. We build a model to predict the ratings of the movies that the users have not rated. To accomplish this, we divide the dataset into edx and validation sets, in a ratio of 9:10. We make sure that the movies and user that are in validation set are also in the edx. A statistical and a machine learning model is trained with the edx dataset. The model predicts ratings for the validation set and is compared to the actual ratings. The objective is to minimize the error between predicted ratings and the actual ratings in the validation dataset.

1.3 Dataset

We use the code given in Appendix 5.1 to download the 10M movie lens data and split it dataset into edx and validation. The validation set contains 10% of the data whereas the edx contains 90% of the data.

Look at the data frames

```
glimpse(edx)
```

```
Observations: 9,000,055
Variables: 6
$ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
$ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 37...
$ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5...
$ timestamp <int> 838985046, 838983525, 838983421, 838983392, 83898339...
$ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (19...
$ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|D...
```

```
glimpse(validation)
```

```
Observations: 999,999
Variables: 6
$ userId    <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5...
$ movieId   <dbl> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 432, 4...
$ rating    <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5.0, 3.0, 3....
$ timestamp <int> 838983392, 838983653, 838984068, 868246450, 86824564...
$ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)", "Hom...
$ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "Child...
```

Each set consists of 6 variables.

userId - Unique ID given to a user u

movieId - Unique ID given to a movie i

rating - a rating between 0 and 5

timestamp - timestamp of when a rating was given

title - title of the movie along with the release year

genres - Genres that the movie is associated with

1.4 Preprocessing data

We extract the date from the timestamp variable and add it as *date_reviewed* to the edx and validation data frames. We also extract the *year_released* variable from the *title* variable and rewrite it, so it only contains the title of the movie.

The *genres* variable contains of various genres grouped together. We process them in two different ways.

1. Create a new data frame which adds columns equal to the unique number of genres available. The corresponding element of a movie in a genre column 1 if genre is present in *genres* variable or else 0. The corresponding data frames are edx_genre_split and validation_genre_split.
2. Create a new data frame where the generes column is processed to be in long format. The corresponding data frames are edx_genre_long and validation_genre_long.

The code to preprocess the data is given in Appendix 5.2

Glimpsing at all the data frames created

```
glimpse(edx)
```

```
Observations: 9,000,055
Variables: 8
$ userId        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ movieId       <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370...
$ rating        <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
$ timestamp     <int> 838985046, 838983525, 838983421, 838983392, 8389...
$ title         <chr> "Boomerang", "Net, The", "Outbreak", "Stargate",...
$ year_released <chr> "1992", "1995", "1995", "1994", "1994", "1994", ...
$ genres        <chr> "Comedy|Romance", "Action|Crime|Thriller", "Acti...
$ date_reviewed <date> 1996-08-02, 1996-08-02, 1996-08-02, 1996-08-02,...
```

```
glimpse(validation)
```

```
Observations: 999,999
Variables: 8
$ userId        <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, ...
$ movieId       <dbl> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 43...
$ rating        <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5.0, 3.0...
$ timestamp     <int> 838983392, 838983653, 838984068, 868246450, 8682...
$ title         <chr> "Dumb & Dumber", "Jurassic Park", "Home Alone", ...
$ year_released <chr> "1994", "1993", "1990", "1995", "1972", "1997", ...
$ genres        <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "C...
$ date_reviewed <date> 1996-08-02, 1996-08-02, 1996-08-02, 1997-07-07,...
```

```
glimpse(edx_genre_split)
```

```
Observations: 9,000,055
Variables: 28
$ userId               <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
$ movieId              <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 3...
$ rating               <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
$ timestamp            <int> 838985046, 838983525, 838983421, 83898339...
$ title                <chr> "Boomerang", "Net, The", "Outbreak", "Sta...
$ year_released        <chr> "1992", "1995", "1995", "1994", "1994", "...
$ genres               <chr> "Comedy|Romance", "Action|Crime|Thriller"...
$ date_reviewed        <date> 1996-08-02, 1996-08-02, 1996-08-02, 1996...
$ Comedy               <dbl> 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1,...
$ Romance              <dbl> 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,...
$ Action               <dbl> 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0,...
$ Crime                <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,...
$ Thriller             <dbl> 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,...
$ Drama                <dbl> 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,...
$ `Sci-Fi`             <dbl> 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ Adventure            <dbl> 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,...
$ Children             <dbl> 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,...
$ Fantasy              <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,...
$ War                  <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,...
$ Animation            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,...
$ Musical              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,...
$ Western              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ Mystery              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ `Film-Noir`          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ Horror               <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ Documentary          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ IMAX                 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ `(no genres listed)` <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

```
glimpse(validation_genre_split)
```

```
Observations: 999,999
Variables: 28
$ userId               <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5,...
$ movieId              <dbl> 231, 480, 586, 151, 858, 1544, 590, 4995,...
$ rating               <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5...
$ timestamp            <int> 838983392, 838983653, 838984068, 86824645...
$ title                <chr> "Dumb & Dumber", "Jurassic Park", "Home A...
$ year_released        <chr> "1994", "1993", "1990", "1995", "1972", "...
$ genres               <chr> "Comedy", "Action|Adventure|Sci-Fi|Thrill...
$ date_reviewed        <date> 1996-08-02, 1996-08-02, 1996-08-02, 1997...
$ Comedy               <dbl> 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,...
$ Romance              <dbl> 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,...
$ Action               <dbl> 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,...
$ Crime                <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ Thriller             <dbl> 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,...
$ Drama                <dbl> 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1,...
$ `Sci-Fi`             <dbl> 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,...
$ Adventure            <dbl> 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,...
$ Children             <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,...
$ Fantasy              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,...
$ War                  <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ Animation            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

```
$ Musical              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ Western              <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,...
$ Mystery              <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,...
$ `Film-Noir`          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ Horror               <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,...
$ Documentary          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ IMAX                 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ `(no genres listed)` <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

```
glimpse(edx_genre_long)
```

```
Observations: 23,371,423
Variables: 8
$ userId        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ movieId       <dbl> 122, 122, 185, 185, 185, 292, 292, 292, 292, 316...
$ rating        <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
$ timestamp     <int> 838985046, 838985046, 838983525, 838983525, 8389...
$ title         <chr> "Boomerang", "Boomerang", "Net, The", "Net, The"...
$ year_released <chr> "1992", "1992", "1995", "1995", "1995", "1995", ...
$ genres        <chr> "Comedy", "Romance", "Action", "Crime", "Thrille...
$ date_reviewed <date> 1996-08-02, 1996-08-02, 1996-08-02, 1996-08-02,...
```

```
glimpse(validation_genre_long)
```

```
Observations: 2,595,771
Variables: 8
$ userId        <int> 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...
$ movieId       <dbl> 231, 480, 480, 480, 480, 586, 586, 151, 151, 151...
$ rating        <dbl> 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 3.0, 3.0, 3.0...
$ timestamp     <int> 838983392, 838983653, 838983653, 838983653, 8389...
$ title         <chr> "Dumb & Dumber", "Jurassic Park", "Jurassic Park...
$ year_released <chr> "1994", "1993", "1993", "1993", "1993", "1990", ...
$ genres        <chr> "Comedy", "Action", "Adventure", "Sci-Fi", "Thri...
$ date_reviewed <date> 1996-08-02, 1996-08-02, 1996-08-02, 1996-08-02,...
```

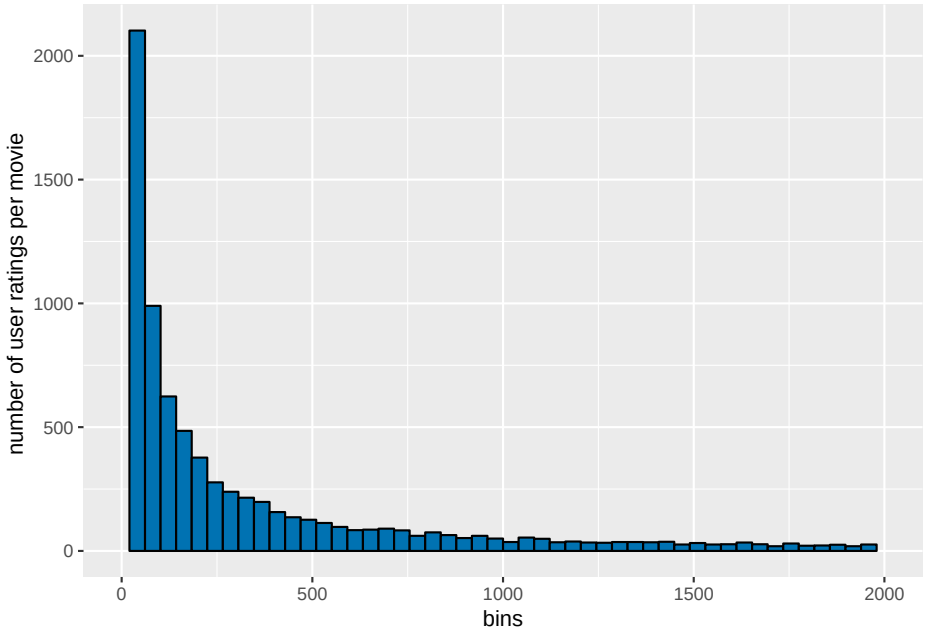

Figure 1: Distribution of User Ratings per movie

2 Methods and Analysis

2.1 Preliminary Data exploration

n_users	n_movies
69878	10677

The number of unique users are 69878 whereas unique movies are 10677.

Note that two movies can have the same title and the year. In fact, this particular dataset has a title War of the Worlds (2005) for two movieID's (64997 and 34048)

We look at the user distribution and movie distributions in Figure 1 and 2. We see two things, there are some movies which are rated by many users and some users are more active. Hence, there is strong bias from users as well as movies.

Next, we look at genre and time effects on the ratings in Figure 3 and 4. We observe that the average rating for the initial years was high and there is some trend in the mean rating across the years. So, we fit a smooth function to look at the mean trend. We also see the effects of genres on the mean ratings i.e. some genres are rated higher than others.

2.2 Models

We follow the same approach of developing models as given in this data science text book. We define a variable $Y_{u,i}$ to explain the ratings of a movie i given by a user u.

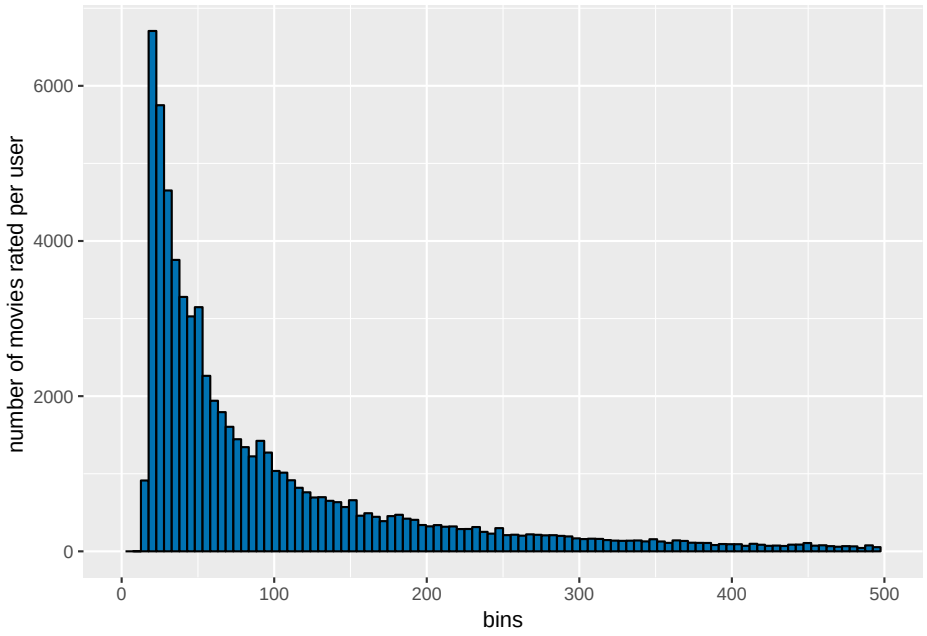

Figure 2: Distribution of Movies rated per user

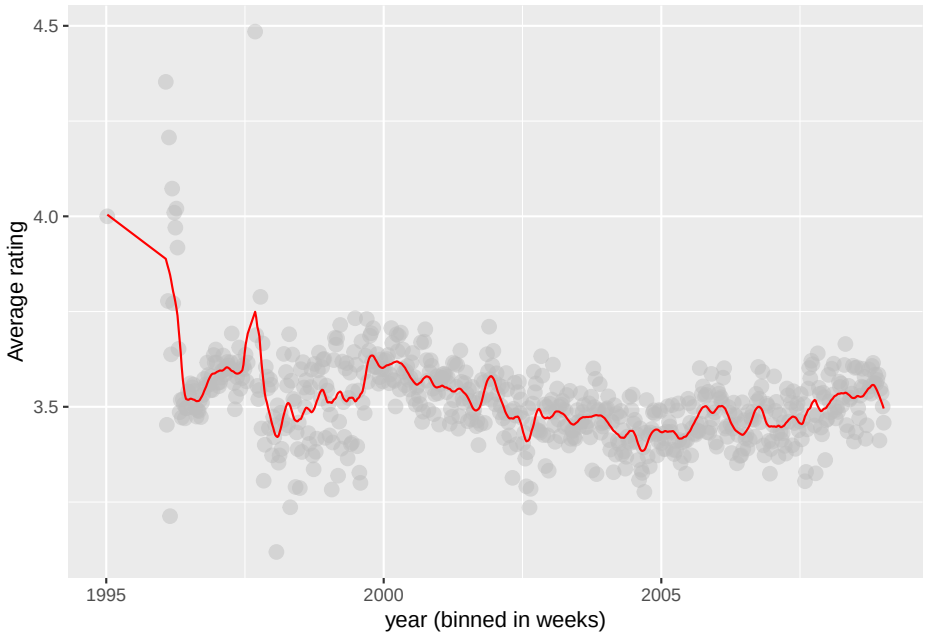

Figure 3: Average Rating per week

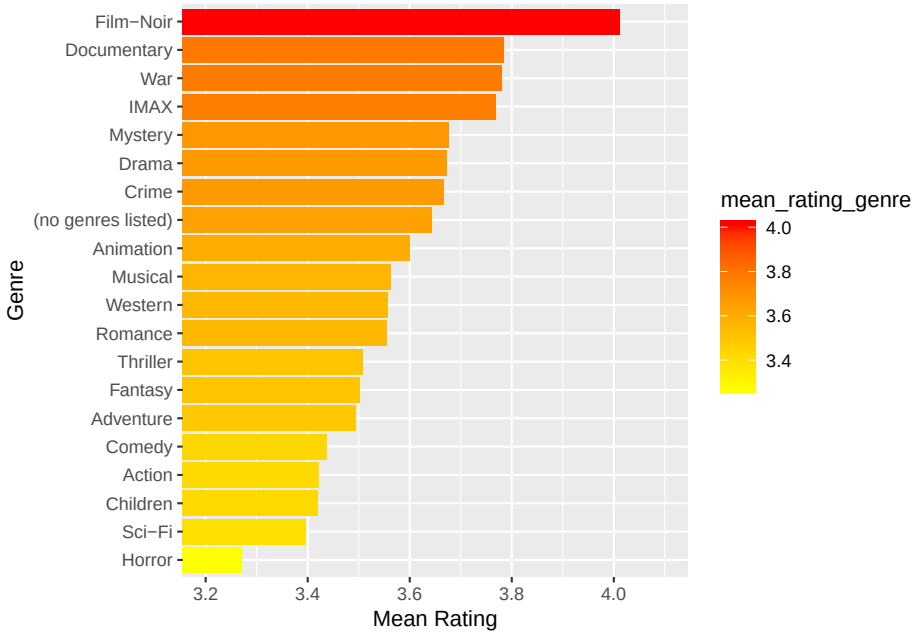

Figure 4: Average Rating per genre

2.2.1 Loss function (RMSE)

We evaulate the accuracy of our models using root-mean-square error (RMSE). If $\hat{y}_{u,i}$ is the rating predicted for a movie i by a user u and $y_{u,i}$ is the actual rating. RMSE is given by the equation,

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

The lower RMSE is, the better our model predicts the data. We report RMSE values of the validation set across different models and compare their performance.

2.2.2 Model 1: Simple model

We can explain all the ratings using a simple mean μ and a random variable $\varepsilon_{u,i}$. The model would be given by

$$Y_{u,i} = \mu + \varepsilon_{u,i} \tag{2}$$

Here, $\varepsilon_{u,i}$ are the independent errors sampled from the same distribution with mean 0.

The parameters are choosen as to minimize the loss/cost function,

$$Costfunction_{Model\ 1} = \frac{1}{N}\sum_{u,i}(y_{u,i} - \mu)^2 \tag{3}$$

We get the parameter μ that minimizes the cost function by taking the derivative and setting it to zero which gives

$$\hat{\mu} = \hat{Y}_{u,i} \tag{4}$$

Hence, the predicted rating for a user u and a movie i is given by

$$\hat{y}_{u,i} = \hat{\mu} \tag{5}$$

2.2.3 Model 2: Movie effects model

We have seen in Figure 1 that there is bias in distribution of movies which we include now in the model.

$$Y_{u,i} = \mu + b_i + \varepsilon_{u,i} \tag{6}$$

where, b_i is the average rating for a movie i and explains bias/effects from the movie distributions.

The cost function is given by

$$Costfunction_{Model\ 2} = \frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i)^2 \tag{7}$$

The parameters μ,b_i that minimizes the cost function are

$$\begin{aligned} \hat{\mu} &= \hat{Y}_{u,i} \\ \hat{b}_i &= \sum_{u=1}^{n_i}(Y_{u,i} - \hat{\mu}) \end{aligned} \tag{8}$$

where n_i are the number of total movies

Hence, the predicted rating for a user u and a movie i is given by

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i \tag{9}$$

2.2.4 Model 3: Movie and User effects

We have seen in Figure 2 that there is bias in the distribution of users, which we include now in the model. The model is given by

$$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i} \tag{10}$$

where, b_u is the average rating for a user u and explains bias/effects from the user distributions. The cost function is given by

$$Costfunction_{Model\ 3} = \frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u)^2 \tag{11}$$

The parameters μ,b_i,b_u that minimizes the cost function are

$$\begin{aligned} \hat{\mu} &= \hat{Y}_{u,i} \\ \hat{b}_i &= \sum_{u=1}^{n_i}(Y_{u,i} - \hat{\mu}) \\ \hat{b}_u &= \sum_{i=1}^{n_u}(Y_{u,i} - \hat{\mu} - \hat{b}_i) \end{aligned} \tag{12}$$

where n_u are the number of total users

Hence, the predicted rating for a user u and a movie i is given by

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i + \hat{b}_u \tag{13}$$

2.2.5 Model 4: Movie and User effects with Regularization

Regularization is a technique used to penalize the coefficients in the model to avoid overfitting the data. For example, if there are very small number ratings for a movie, we would want the movie effect b_i to be close to zero. This can be accomplished by modifying the cost function and penalizing the bias terms with a regularization coefficient λ

We use the same model as before,

$$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i} \tag{14}$$

but minimize a new cost function given by

$$Costfunction_{Model\ 4} = \frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u)^2 + \lambda\left(\sum_i b_i^2 + \sum_u b_u^2\right) \tag{15}$$

The parameters μ,b_i,b_u that minimizes the cost function are

$$\begin{aligned}
\hat{\mu} &= \hat{Y}_{u,i} \\
\hat{b}_i &= \frac{1}{\lambda + n_i}\sum_{u=1}^{n_i}(Y_{u,i} - \hat{\mu}) \\
\hat{b}_u &= \frac{1}{\lambda + n_u}\sum_{i=1}^{n_u}(Y_{u,i} - \hat{\mu} - \hat{b}_i)
\end{aligned} \tag{16}$$

We see that is if n_i and n_u are large, the estimated coefficients ignore the λ. On the other hand if n_i and n_u are small, the coefficients are shrunk by λ if the numbers are small.

The predicted rating for a user u and a movie i the same as the previous model

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i + \hat{b}_u \tag{17}$$

2.2.6 Model 5: Movie, User Effect with Regularization and Time Effect

We have seen in Figure 3 that there is bias in the distribution of date on which the movie is rated. We include this in our model with a smoothing function of the rounded date in weeks.

The model now transforms to

$$Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \varepsilon_{u,i} \tag{18}$$

where f is smooth function of $d_{u,i}$.

The cost function is given by

$$Costfunction_{Model\ 5} = \frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u - f(d_{u,i}))^2 + \lambda\left(\sum_i b_i^2 + \sum_u b_u^2\right) \tag{19}$$

The parameters μ,b_i,b_u,f(d_{u,i})$ that minimizes the cost function are

$$\begin{aligned}
\hat{\mu} &= \hat{Y}_{u,i} \\
\hat{b}_i &= \frac{1}{\lambda + n_i}\sum_{u=1}^{n_i}(Y_{u,i} - \hat{\mu}) \\
\hat{b}_u &= \frac{1}{\lambda + n_u}\sum_{i=1}^{n_u}(Y_{u,i} - \hat{\mu} - \hat{b}_i) \\
\hat{f}(d_{u,i}) &= \sum_{d=1}^{n_d}(Y_{u,i} - \hat{\mu} - \hat{b}_i - \hat{b}_u)
\end{aligned} \tag{20}$$

where n_d is the number of rounded dates available in the dataset.

The predicted rating for a user u and a movie i is given by

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i + \hat{b}_u + \hat{f}(d_{u,i}) \tag{21}$$

2.2.7 Model 6: Movie, User Effect with Regularization, Time and Genre Effect

We have seen in Figure 4 that there is bias of genre in the distribution of ratings. We model the effect by adding a coefficient for each genre. The model now transforms to

$$Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \sum_{k=1}^{n_k} x_{i,k} b_k + \varepsilon_{u,i} \tag{22}$$

where n_k is the total number of genres and b_k is coefficient for each genre. $x_{i,k} = 1$ if a movie i is in genre k or else $x_{i,k} = 0$

The cost function is given by

$$Costfunction_{Model\ 6} = \frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u - f(d_{u,i}) - \sum k = 1^{n_k} x_{i,k} b_k)^2 + \lambda \left(\sum_i b_i^2 + \sum_u b_u^2 \right) \tag{23}$$

The parameters μ,b_i,b_u,$f(d_{u,i})$,b_k that minimizes the cost function are

$$\begin{aligned}
\hat{\mu} &= \hat{Y}_{u,i} \\
\hat{b}_i &= \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu}) \\
\hat{b}_u &= \frac{1}{\lambda + n_u} \sum_{i=1}^{n_u} (Y_{u,i} - \hat{\mu} - \hat{b}_i) \\
\hat{f}(d_{u,i}) &= \sum_{d=1}^{n_d} (Y_{u,i} - \hat{\mu} - \hat{b}_i - \hat{b}_u) \\
\hat{b}_k &= \sum_{i=1}^{n_{k,i}} (Y_{u,i} - \hat{\mu} - \hat{b}_i - \hat{b}_u - \hat{f}(d_{u,i}))
\end{aligned} \tag{24}$$

where $n_{k,i}$ is number of movies with $x_{i,k} = 1$ for a genre k.

The predicted rating for a user u and a movie i is given by

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i + \hat{b}_u + \hat{f}(d_{u,i}) + \sum_{k=1}^{K} x_{i,k} \hat{b}_k \tag{25}$$

2.2.8 Model 7: Bias effects with Matrix Factorization of Residuals using SG

Thus far, we looked at effects of normalization from different features. The data also has groupings of similar users and similar movies which the model has not captured yet. Matrix factorization decomposes the users and movies into sets using feature latent factors. An example of latent factors for movies is the distribution of movie. If a movie is a big blockbuster, the latent factor would have a positive value whereas small independent movies would have a negative value. By using a good number of latent factors, one can predict the outcome to a good degree.

The model to include the matrix factorization is

$$Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \sum_{k=1}^{n_k} x_{i,k} b_k + rsi_{u,i} + \varepsilon_{u,i} \tag{26}$$

where $rsi_{u,i}$ are the residuals that we wish to factorize. $rsi_{u,i}$ can be converted to a matrix $R_{u\times i}$ where the rows represent the user and columns represent the movies. The matrix factorization decomposes the matrix $R \approx P^T Q$, where $P_{k\times u}$ and $Q_{k\times i}$ are the matrices with k latent factors for the users and movies. The individual elements of R can be obtained by $rsi_{u,i} \approx p_u^T q_i$ where, p_u is the uth column vector P and q_i is ith the column of Q.

The model can be also written as

$$Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \sum_{k=1}^{n_k} x_{i,k} b_k + p_u^T q_i + \varepsilon_{u,i} \tag{27}$$

We find the residuals after fitting the models from the previous bias models

$$rsi_{u,i} = Y_{u,i} - \hat{\mu} - \hat{b}_i - \hat{b}_u - \hat{f}(d_{u,i}) - \sum_{k=1}^{K} x_{i,k} \hat{b}_k \tag{28}$$

The cost function is given by

$$Costfunction_{Model\ 7} = \frac{1}{N} \sum_{u,i} (rsi_{u,i} - p_u^T q_i)^2 + \lambda_p ||p_u||^2 + \lambda_q ||q_i||^2 \tag{29}$$

where, λ_p and λ_q are the regularization parameters for p_u and q_i.

The recosystem package minimizes the cost function using stochastic gradient algorithm to find the matrices $\hat{P}$ and $\hat{Q}$. Note that the package also has regularization terms for absolute vaues of $||p_u||$ and $||q_u||$ which we set to zero in the simulation.

The predicted rating for a user u and a movie i is given by

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i + \hat{b}_u + \hat{f}(d_{u,i}) + \sum_{k=1}^{K} x_{i,k} \hat{b}_k + \hat{p}_u^T \hat{q}_i \tag{30}$$

2.2.9 Model 8: Matrix Factorization of Ratings using SGD

Here, we do not include any bias terms and use matrix factorization using the ratings directly. The model can be written as

$$Y_{u,i} = p_u^T q_i + \varepsilon_{u,i} \tag{31}$$

The cost function is given by

$$Costfunction_{Model\ 8} = \frac{1}{N} \sum_{u,i} (y_{u,i} - p_u^T q_i)^2 + \lambda_p ||p_u||^2 + \lambda_q ||q_i||^2 \tag{32}$$

As before, the SGD algorithm computes the $\hat{P}$ and $\hat{Q}$ matrices.

The predicted rating for a user u and a movie i is given by

$$\hat{y}_{u,i} = \hat{p}_u^T \hat{q}_i \tag{33}$$

3 Results

We present the results for each model and present the corresponding RMSE.

3.1 Model 1: Simple model

In the simple model, only the mean rating of the training set is used to predict the rating of the validation set. $Y_{u,i} = \mu + \varepsilon_{u,i}$

method	RMSE
Simple Model	1.061202

3.2 Model 2: Movie effects

In model 2, movie bias and the mean ratings are used.

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

method	RMSE
Simple Model	1.0612018
Movie Effects Model	0.9439087

3.3 Model 3: Movie and User effects

In model 3, movie bias, user bias and the mean ratings are used. $Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}$

method	RMSE
Simple Model	1.0612018
Movie Effects Model	0.9439087
Movie and User Effects Model	0.8653488

3.4 Model 4: Movie and User effects with Regularization

In model 4, movie bias and user bias with regularization are used. We divide the edx set into train (edx_train- 80%) and cross-validation (edx_cv - 20%) sets to train the lambda (λ) for regularization.

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

We plot the lambdas in Figure 5 and find the minimum lambda to be 5. We use it to predict the RMSE.

method	RMSE
Simple Model	1.0612018
Movie Effects Model	0.9439087
Movie and User Effects Model	0.8653488
Movie and User Effects with Regularization Model	0.8648201

3.5 Model 5: Movie, User Effect with Regularization and Time Effect

In model 5, time effect is added to regularized movie bias, user bias and the mean ratings. We fit a smooth function to the time effect $d_{u,i}$ using locally estimated scatterplot smoothing (LOESS). $Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \varepsilon_{u,i}$

method	RMSE
Simple Model	1.0612018
Movie Effects Model	0.9439087
Movie and User Effects Model	0.8653488
Movie and User Effects with Regularization Model	0.8648201
Movie, User Effect with Regularization and Time Effect	0.8647751

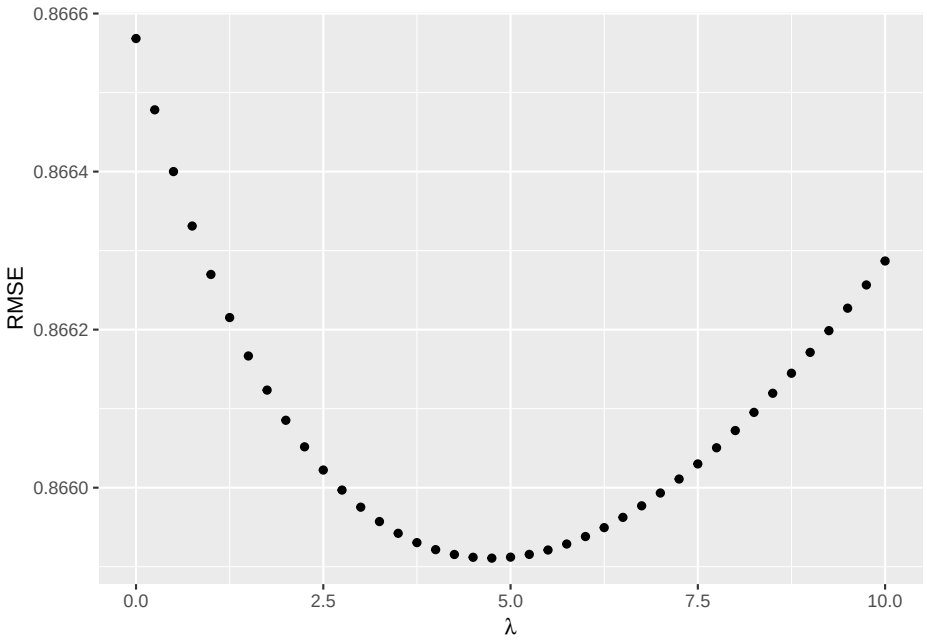

Figure 5: RMSE vs lambda

3.6 Model 6: Movie, User Effect with Regularization, Time and Genre Effect

In model 6, time effect and genre effect is added to regularized movie bias, user bias and the mean ratings. We use the edx_genre_split to implement the contributions of genre.

$Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \sum_{k=1}^{n_k} x_{i,k} b_k + \varepsilon_{u,i}$

method	RMSE
Simple Model	1.0612018
Movie Effects Model	0.9439087
Movie and User Effects Model	0.8653488
Movie and User Effects with Regularization Model	0.8648201
Movie, User Effect with Regularization and Time Effect	0.8647751
Movie, User Effect with Regularization, Time and Genre Effect	0.8646861

3.7 Model 7: Bias effects with Matrix Factorization of Residuals using SGD

In model 7, the residuals are calculated by subtracting the bias predictions from the ratings. Matrix factorization using SGD is applied on the residuals using the recosystem package. The recosystem package implements a Stochastic gradient method using the Reduced Per-coordinate Schedule as the learning rate.
$Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \sum_{k=1}^{n_k} x_{i,k} b_k + p_u^T q_i + \varepsilon_{u,i}$

We train the model using the various values of latent parameters and iterations. The tune function performs an nfold = 10 cross validation on the training set and returns the loss functions/RMSE on them. We store the values of iterations, latent parameters and their corresponding loss function in a data frame cross_val_rmse_res. We choose regularization terms costp_l2 = 0.01, costq_l2 = 0.1 respectively after initially tuning the model. We also choose learning rate to be .05 and L1 regularization terms (costp_l1,costq_l1) to be zero.

We observe that for iterations >= 20, the models exhibits more or less the same curve which implies that they have converged to the solution in 20 iterations. We also observe that RMSE's are almost converged for around 150-200 latent factors. This implies that approximately 150 latent factors are sufficient to explain

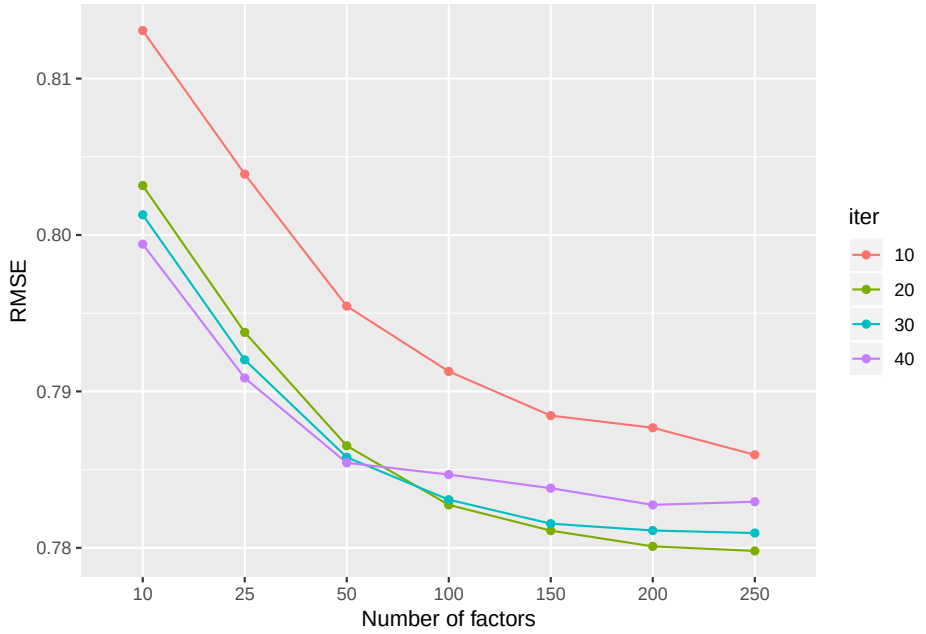

Figure 6: Cross-validation RMSE vs latent factor for different iteration numbers with residuals data

most of the variance using this model.

We choose 20 iterations and 200 latent factors for our model.

method	RMSE
Simple Model	1.0612018
Movie Effects Model	0.9439087
Movie and User Effects Model	0.8653488
Movie and User Effects with Regularization Model	0.8648201
Movie, User Effect with Regularization and Time Effect	0.8647751
Movie, User Effect with Regularization, Time and Genre Effect	0.8646861
Bias effects with Matrix Factorization of Residuals using SGD	0.7814388

3.8 Model 8: Matrix Factorization of Ratings using SGD

In model 8, matrix factorization using SGD is applied directly on the ratings of the edx without subtracting any bias effects. We perform a similar tuning as for model 7.

$Y_{u,i} = p_u^T q_i + \varepsilon_{u,i}$

We observe that the number of minimum iterations must be increased to 30 for the RMSE to converge whereas the latent factors almost remain the same. The increase in iterations is due to the added processing needed to remove the biases in the data.

We choose 30 iterations and 200 latent factors for our model.

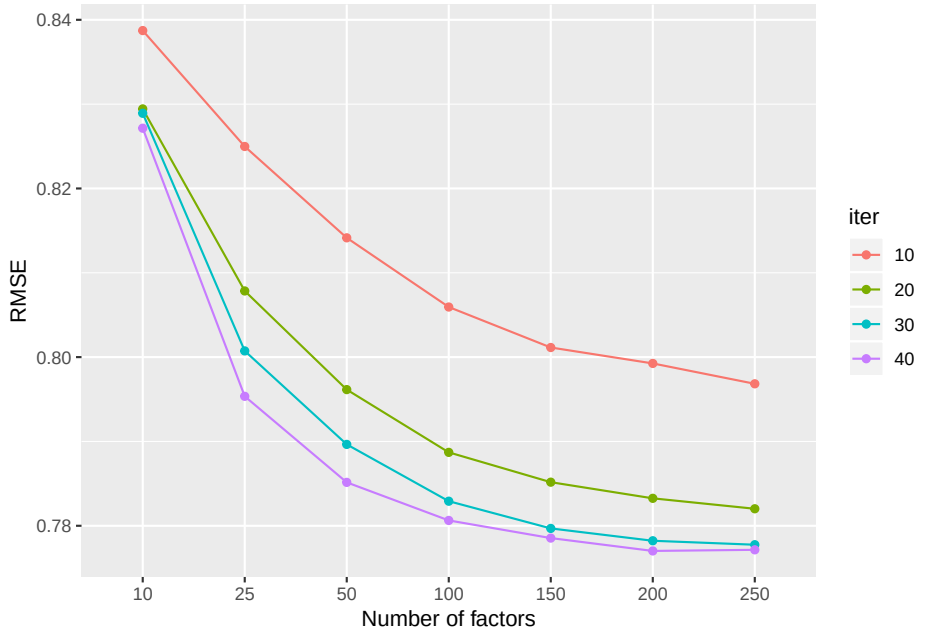

Figure 7: Cross-validation RMSE vs latent factor for different iteration numbers with ratings data

method	RMSE
Simple Model	1.0612018
Movie Effects Model	0.9439087
Movie and User Effects Model	0.8653488
Movie and User Effects with Regularization Model	0.8648201
Movie, User Effect with Regularization and Time Effect	0.8647751
Movie, User Effect with Regularization, Time and Genre Effect	0.8646861
Bias effects with Matrix Factorization of Residuals using SGD	0.7814388
Matrix Factorization of Ratings using SGD	0.7754950

4 Conclusion

In this report, we have trained several models to predict the ratings from 10M Movielens dataset. We saw that biases and normalization could only improve RMSE's to an extent. Specifically, in our case the normalization could only decrease the RMSE to 0.86468. On the other hand, matrix factorization is a fast and an efficient technique which resulted in RMSE's of ≈ 0.78. Matrix factorization was also able to work directly with the data without any removal of bias to produce a similar RMSE. We used only one approach of matrix factorization using stochastic gradient based on Reduced Per-coordinate Schedule learning rate in our analysis. However, there are several other methods such as alternating least squares, regularizing the weights of the factors, assymetric single value decompostions and ensemble methods as described here. Combining several methods could result in a lower RMSE.

5 Appendix

5.1 A: Code to download the movielens data

We use the following code to download the Movielens data from grouplens website.

```
################################
# Create edx set, validation set
################################

library(tidyverse)
library(caret)
library(data.table)
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

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")
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)
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)

# Save the file to data.Rda
save(edx,validation,file="./data/data.Rda")
```

5.2 B: Code to preprocess the data

We use the following code to preprocess the data.

```
load("./data/data.Rda")

#Convert the timestamp into date_reviewed and seperate released year and title
temp1 <- edx %>% mutate(date_reviewed = date(as_datetime(timestamp))) %>%
mutate(title = str_trim(title)) %>%
extract(title,c("title", "year_released"), regex = "^(.*) \\((\\d{4})\\)$", remove = TRUE)
#Validating that title and year_released have no null values
temp1%>% filter(is.na(temp1$year_released) | is.na(temp1$title))%>% knitr::kable(.)

#Assigning data frame to edx
edx <- temp1
# remove the temp1 variable
rm(temp1)

# Add date reviewed and seperate released year and title for validation
temp2 <- validation %>% mutate(date_reviewed = date(as_datetime(timestamp))) %>%
mutate(title = str_trim(title)) %>%
extract(title,c("title", "year_released"), regex = "^(.*) \\((\\d{4})\\)$", remove = TRUE)
temp2%>% filter(is.na(temp2$year_released) | is.na(temp2$title))%>% knitr::kable(.)
validation <- temp2
rm(temp2)

#Unique Genres in the whole list
genres_list <- unique((unlist(str_split(edx$genres, "\\|"))))

#Create columns for each genre. If a particular genre is present - 1 else 0
genres_columns <- sapply(seq(1,length(genres_list)),function(k){
  return(as.numeric(str_detect(genres_list[k],edx$genres)))})

#Rename the columns to the genres avaialable
colnames(genres_columns) = genres_list

#data frame with genres split
edx_genre_split <- cbind(edx,genres_columns)

#Create columns for each genre. If a particular genre is present - 1 else 0
genres_columns <- sapply(seq(1,length(genres_list)),function(k){
  return(as.numeric(str_detect(genres_list[k],validation$genres)))})

#Rename the columns to the genres avaialable
colnames(genres_columns) = genres_list

#data frame with genres split
validation_genre_split <- cbind(validation,genres_columns)

# Partition the edx into 4 parts to convert it to long format of genres. Partitioning due
#to small RAM
nelem_part <- round(nrow(edx)/4)
edx_partion_1 <- edx[1:nelem_part,]
edx_partion_2 <- edx[(nelem_part+1):(2*nelem_part),]
edx_partion_3 <- edx[(2*nelem_part+1):(3*nelem_part),]
edx_partion_4 <- edx[(3*nelem_part+1):nrow(edx),]
```

```
split_edx_1  <- edx_partion_1  %>% separate_rows(genres, sep = "\\|")
split_edx_2  <- edx_partion_2  %>% separate_rows(genres, sep = "\\|")
split_edx_3  <- edx_partion_3  %>% separate_rows(genres, sep = "\\|")
split_edx_4  <- edx_partion_4  %>% separate_rows(genres, sep = "\\|")

edx_genre_long <- rbind(split_edx_1,split_edx_2,split_edx_3,split_edx_4)
#Cleanup
rm(edx_partion_1,edx_partion_2,edx_partion_3,edx_partion_4,split_edx_1,split_edx_2,
   split_edx_3,split_edx_4)

#Convert Validation into long format
validation_genre_long <- validation%>% separate_rows(genres, sep = "\\|")

#Save the data into preprocessedData.Rda
save(edx,genres_list,validation,edx_genre_split,validation_genre_split,
     edx_genre_long,validation_genre_long,file="./data/preprocessedData.Rda")
```

References

- Irizzary,R., 2020,*Introduction to Data Science*, book, https://rafalab.github.io/dsbook/.
- Chen, E.,Winning the Netflix Prize: A Summary, blog, http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/
- Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
- Chin, W. S., Zhuang, Y., Juan, Y. C., & Lin, C. J. (2015). A fast parallel stochastic gradient method for matrix factorization in shared memory systems. ACM Transactions on Intelligent Systems and Technology (TIST), 6(1), 1-24.
- Chin, W. S., Zhuang, Y., Juan, Y. C., & Lin, C. J. (2015, May). A learning-rate schedule for stochastic gradient methods to matrix factorization. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 442-455). Springer, Cham.
- Recosystem package, github, https://github.com/yixuan/recosystem