**Introduction**

**an introduction/overview/executive summary section that describes the dataset and summarizes the goal of the project and key steps that were performed**

In this report, our goal is to predict the movie rating by using a machine learning algorithm. The data set is coming from Movielens with about 10M rows of userId, movieId, rating, timestamp, title and genres. Movies are released from early 20th century till 2008. Movie rating are made from 1995 to 2009.

Data cleaning is applied to the original data first following with data exploration. 4 major effects are identified. Our approach is using Normalization of these global effects on baseline rating and regularization (by tuning parameter on lambda) to penalize large estimates that come from small sample size.

1. Movie specific effect
2. User specific effect
3. Genre specific effect
4. Rate per Year specific effect

The evaluation of algorithm is based on root mean squared error (RMSE) of the predicted rating against actual rating. Algorithm is trained on train set and being test on test set. Final RMSE is presented basing the on the final hold-out validation set with result in the tier of “RMSE < 0.86490”.

**Method**

**a methods/analysis section that explains the process and techniques used, including data cleaning, data exploration and visualization, insights gained, and your modeling approach**

**1. Data cleaning**

edx data set contains 6 columns (userId, movieId, rating, timestamp, title and genres).

[head(edx)]

In order to facilitate movie rating prediction modelling, pre-process data cleaning is applied to edx data set prior to data partition creation. Title column is split to title and year. Timestamp which is number of second since 1-Jan-1970 00:00:00 is converted to date. Using createDataPatition function of caret package to create train set and test set with percentage of 80% and 20% correspondingly. Semi-join by movieId and userId is applied to test set to avoid #NA situation when joining is applied to test set in validation stage.

[head(train\_set)]

**2. Data exploration**

From train set, we can simplify find the average rating across all movies and all users is mu = 3.51 and the end year of all movies is 2008. To avoid dividing by zero in rate per year calculation, we are using 2009 (2008 + 1) as the end year.

[mu, maxyear]

1. Movie specific bias

From below chart, we can find that average movie rating adjusted by mu is at -0.32 with distribution skewed to the left side.

[code for average]



1. User specific bias

On the other hand, we can find that average user rating adjusted by mu is 0.10 with more user giving above average rating.

[code for average]



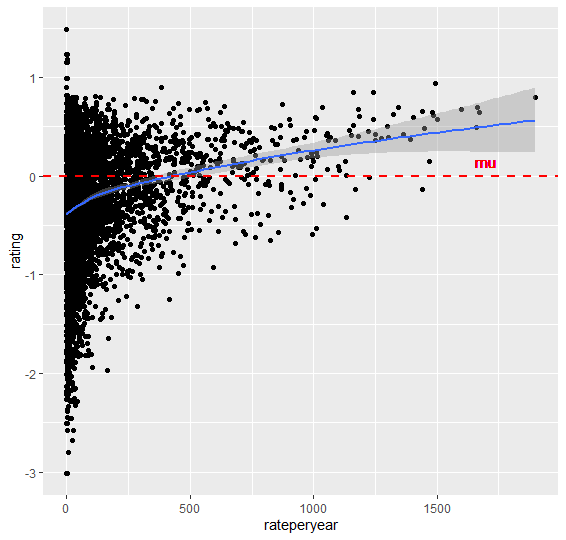
1. Genre specific bias

We are only showing genres with rating more than 20K times in below chart. We can see that there is clear relation between average rating adjusted by mu and genre. The lowest average rating is coming from “Comedy | Horror” while “Crime | Mystery | Thriller” has the highest average rating.



1. Rate per Year specific bias

From below chart, we can see that the more often a movie is rated per year, the higher its average rating adjusted by mu (the blue line). Basing on the observation, a lower value will be given if rate per year is lower the corresponding rating per year of mu. For details, please refer to section of modeling approach.



**3. Modeling approach**

**Normalization of Global Effects**

We are using the approach of normalization of global effects in this project. Basing on the above 4 findings, we decompose 4 global bias starting from assuming the same rating (μ, average rating) across all movies and all users. The differences is explained by specific bias from Movie (b\_i) / User (b\_u) / Genre (b\_g) / Rate per Year (br) and random variation (εu,i).

Yu,i = μ + b\_i + b\_u + b\_g + b\_r + εu,i

Movie / User / Genre bias tables are created by taking average of the rating minus μ and the other bias one by one. Codes are extracted below.

summarize(b\_i = sum(rating - mu) / (n() + l))

summarize(b\_u = sum(rating - mu - b\_i) / (n() + l))

summarize(b\_g = sum(rating - mu - b\_i - b\_u) / (n() + l))

For Rate per Year bias table, a linear regression model fit\_rateperyear is created. First, we adjust the rating with μ and above 3 bias and fit the rating with rate per year in a simple linear regression model. Using the linear regression model, if rate per year is less than the corresponding number of rating per year of average rating, we use model to predict new rating. If not, we keep the original rating. With this approach, we are trying to shrink the prediction of rating affected by rate per year.

rating = rating - mu - b\_i - b\_u - b\_g

[code]

**Regularization**

Movie / User / Genre bias are regularized with λ to penalize large estimates that come from small sample size with the shrunk prediction. λ is a tuning parameter using cross-validation to choose minimum RSME on train set only. This approach doesn’t apply to Rate per Year bias because this has already penalized the prediction with lower than average Rate per Year. λ of 2.9 is chosen.

summarize(b\_i = sum(rating - mu) / (n() + l))

summarize(b\_u = sum(rating - mu - b\_i) / (n() + l))

summarize(b\_g = sum(rating - mu - b\_i - b\_u) / (n() + l))

[code]



**Prediction Restraint and NA handling for Movie / User outside of train set**

Since our target is to predict the movie rating which is from 0 to 5. It is not meaningful to predict a rating less than 0 or greater than 5. As a result, prediction is restrained to 0 to 5 with below code.

mutate(pred = ifelse(pred <0, 0, ifelse(pred >5 , 5, pred)))

It is possible to encounter a movie or a user in validation set that does not appear in the train set. Under this scenario, μ is predicted.

[code]

**Result**

**a results section that presents the modeling results and discusses the model performance**

RMSE result for test set is 0.8643338.

RMSE result for validation set is 0.8647539.

Both results are in the tier of “RMSE < 0.86490”

The most time-wasting procedure is regularization since the process involves lambda section under selected values. Wider range and wider increment is applied first and then narrow the range to 1 and increment to 0.1.

lambdas <- seq(1, 10, 1)

lambdas <- seq(2, 5, 0.25)

lambdas <- seq(2.5, 3.5, 0.1)

The second time-wasting process is data cleaning specially on the split of title and year from the original title column. The main reason is separator uses Regex to detect the pattern for handling of different scenarios.

**Conclusion**

**a conclusion section that gives a brief summary of the report, its limitations and future work**

The report is only using the approach of Normalization of Global Effects and Regularization to capture the main effects in the data. The result RMSE is quite significant in the tier of “RMSE < 0.86490” with handling the 4 effects of Movie / User / Genre / Rate per Year. We can find that these baseline effects have clear impact on the rating distribution in data exploration.

In addition to the baseline effects, more sophisticated models can be applied, like Neighborhood Model and Matrix factorization. Neighborhood Model (Movie-Movie approach / User-User approach) can identify similar movie and user that are similar to each other. Their ratings are closed to each other. Matrix factorization (SVD / PCA) can identify the latent factors like coefficient of different genres or relevant impact on big name serial movies.