

HarvardX PH125.9 Movie Lens Project

Alexander Braune

19.02.2022

Abstract

This project shows basis steps of exploratory data analysis and the building of a basic linear regression machine learning model. As a use case the MovieLens data set is used to construct a forecast method that predicts movie ratings. The final model builds on average ratings and includes a user bias, movie impact and release year bias. A RMSE of 0.8248 on 5 star rating scale is achieved with a validation data set.

Contents

1	Introduction and Overview	2
2	Methods and Analysis	2
2.1	Data Analysis	2
2.2	Modelling Approach	7
3	Results	11
4	Conclusion	12

1 Introduction and Overview

Movie ratings are a well known field where Data Science techniques were further developed. The real world applicability is a strong arguments why students have an good access of understanding the concepts of Data Science within this field. Business interests also furthered the development and popularity of movie recommendation systems. Also, movie ratings from users that are tracked digitally are a vast source of heavily filled data set collections.

Therefore the development of a Machine Learning algorithm that predicts the movie rating of users will be the aim of this final project for the edX course PH125.9x form HarvardX.

The data set used is offered by the GroupLens research lab in the Department of Computer Science and Engineering at the University of Minnesota. For performance reasons the MovieLens 10M Dataset version is used which has 10 million ratings. It is labeled as a stable benchmark set and was released in 1/2009.

Before applying any data investigation the whole data set is split into a 90% training set and a 10% validation set. A model that is parametrized with the whole data set is prone to overfitting and one can not determine a true forecast accuracy as all data is already known to the model (bias). With randomly splitting the data into separate sets the training of the model can be done without overfitting it. After all parametrization of the model the true accuracy can be evaluated with a previously unconsidered validation data set.

First, the steps of performed exploratory data analysis in chapter 2.1 offer an in depth understanding of the data set. The data structure, unique values of variables are inspected and insights from variable constellations are generated. The insights generated are then used in chapter 2.2 to build the Machine Learning model from scratch. The models are compared by evaluating the overall root mean square error (RMSE) within the training set. Hyperparameters are found for a final model. All models are then evaluated with the validation data set in chapter 3 from which are conclusions drawn in chapter 4.

2 Methods and Analysis

In these sections the data is explored and the steps of model building and parametrization are described.

2.1 Data Analysis

The data set is downloaded and split as described in the Introduction. To investigate the data set we first evaluate the dimensions of both data tables:

```
## TRAIN SET
## count rows: 9000055
## count fields: 6
```

```

## VALIDATION SET
## count rows: 999999
## count fields: 6

```

The supposed 10m records were split 9:1 and both sets have the same width of 6 fields.

The completeness is also investigated:

```
anyNA(train)
```

```
## [1] FALSE
```

```
anyNA(validation)
```

```
## [1] FALSE
```

Both sets are not missing values.

With head() a preview of the train set is derived:

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

A line in the data set represents a rating which was done by userId and is linked to a movieId. The movie titles contain the release year in a systematic way (in between brackets). To make the release year accessible as a variable, a column with the extracted year is added to both data sets.

```

#add column with extracted release year to BOTH data sets
train <- mutate(train, year= train$title %>%
  str_extract("(?=<=\\"().\\d{4}(?=\\))")
  %>% as.integer())
validation <- mutate(validation, year= validation$title %>%
  str_extract("(?=<=\\"().\\d{4}(?=\\))")
  %>% as.integer())

```

The train set now has 7 columns:

	userId	movieId	rating	timestamp	title	genres	year
X	Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:9000055	Length:9000055	Min. :1915
X.1	1st Qu.:18124	1st Qu.: 648	1st Qu.:3.000	1st Qu.:9.468e+08	Class :character	Class :character	1st Qu.:1987
X.2	Median :35738	Median : 1834	Median :4.000	Median :1.035e+09	Mode :character	Mode :character	Median :1994
X.3	Mean :35870	Mean : 4122	Mean :3.512	Mean :1.033e+09			Mean :1990
X.4	3rd Qu.:53607	3rd Qu.: 3626	3rd Qu.:4.000	3rd Qu.:1.127e+09			3rd Qu.:1998
X.5	Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09			Max. :2008

The release years start at 1915 and goes up to 2008, but the median is 1994. This indicates there are more younger movies than older movies.

The rating starts at 0.5 goes and up to 5. Mean over all ratings is 3.5.

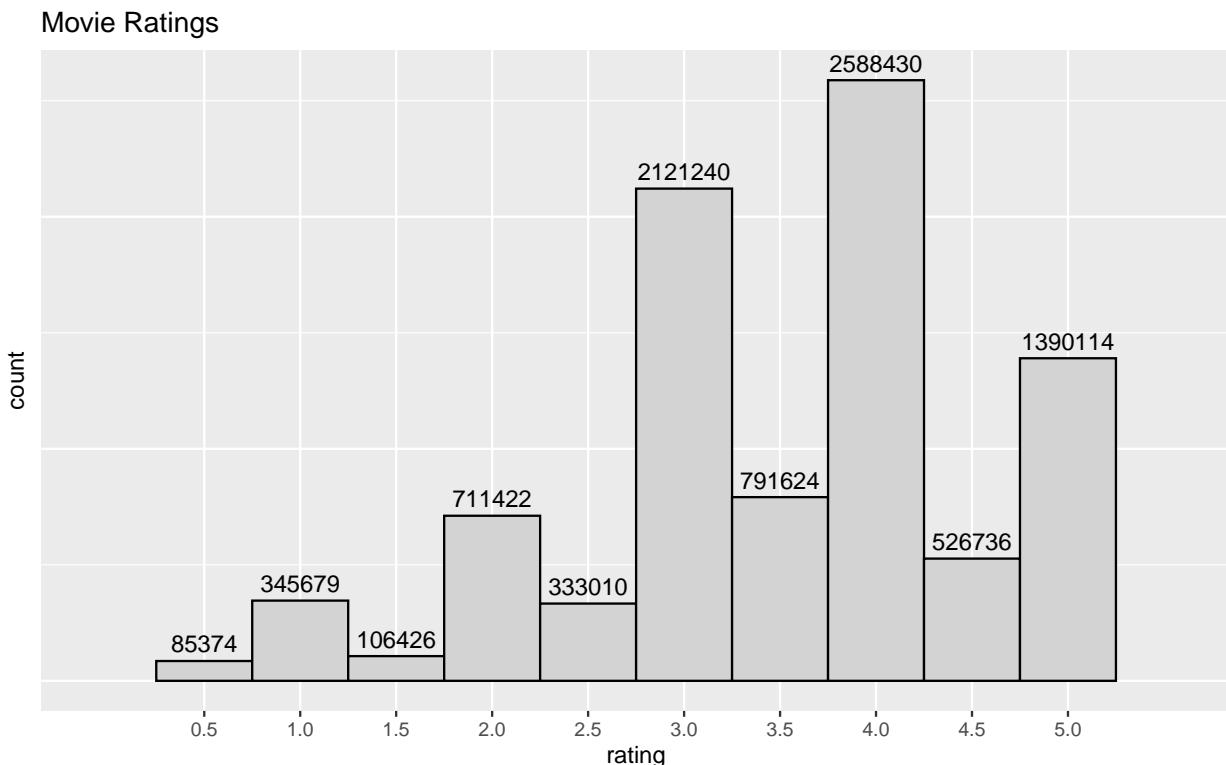
Further investigation of unique values:

```
##      userId    movieId     rating timestamp      title    genres   year
##      69878       10677       10  6519590      10676      797      94
```

The train data set contains 9m ratings from nearly 70,000 users on over 10,000 movies. Unique rating entries indicate a .5 rating scale.

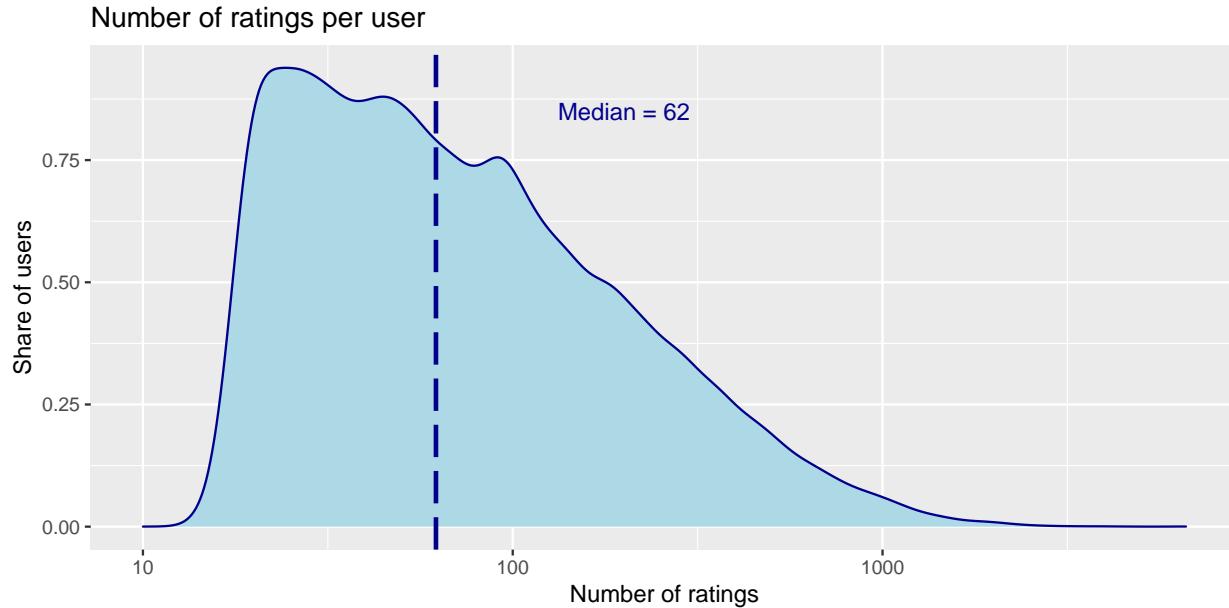
```
## Ratings: 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
```

Further describing the rating scale the following graph shows the count of different ratings:



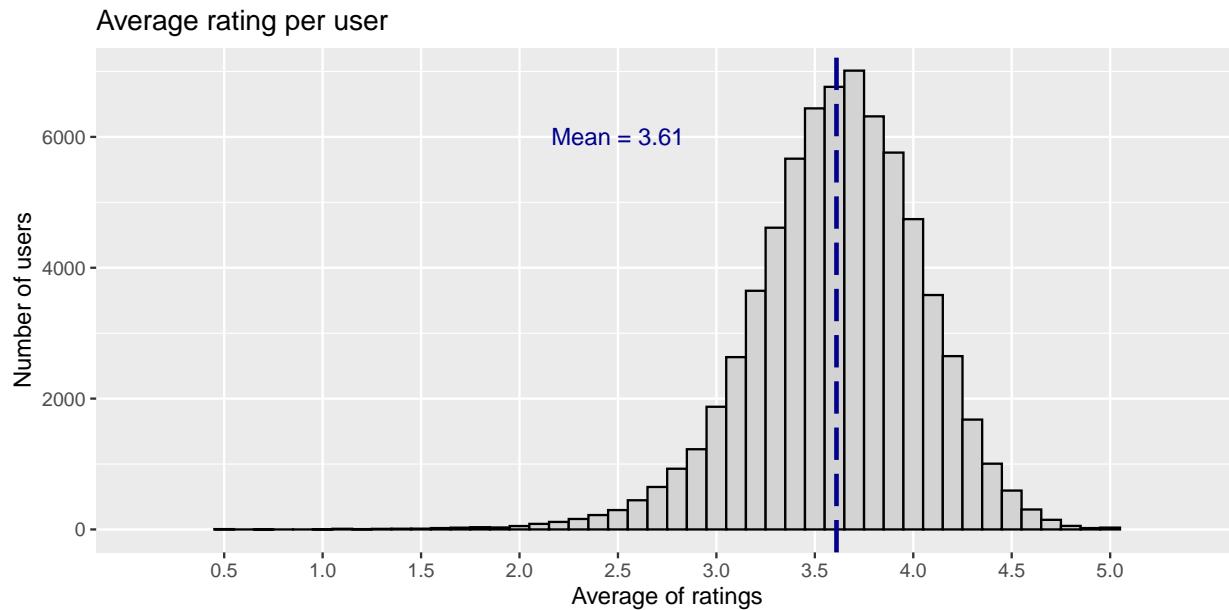
The whole number ratings are more likely and the whole rating distribution shows a left skewness.

Describing the ratings ratio the following graph shows the count of ratings per user:



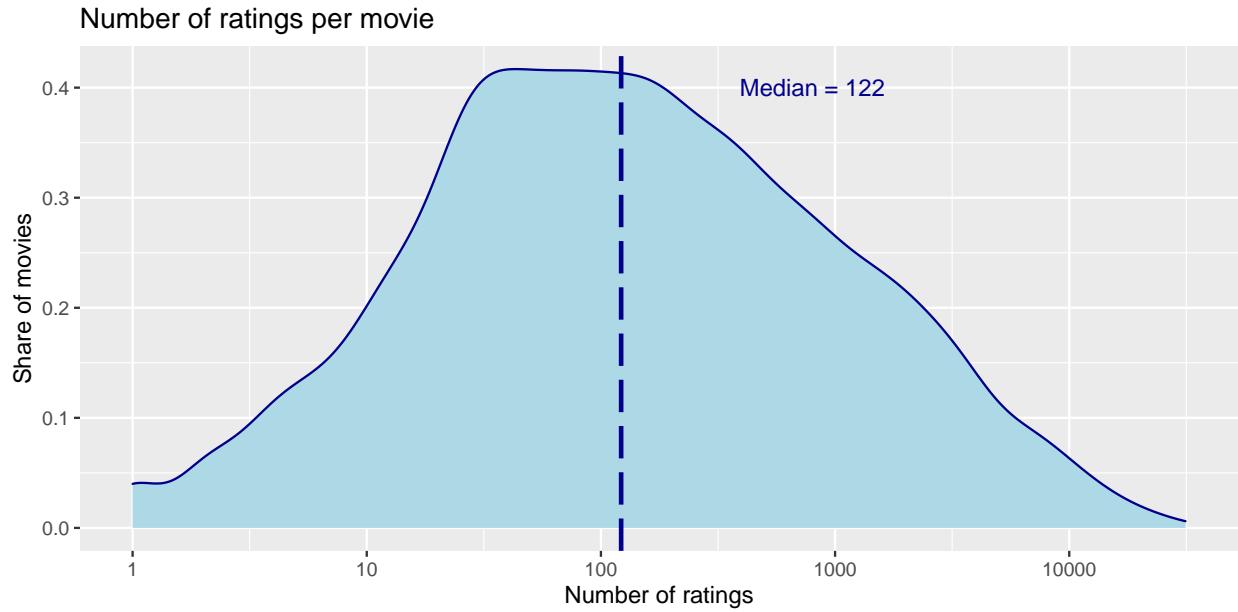
The median user gives 62 ratings. The average user with the highest share gives less, but a small proportion of users contribute $>1,000$ ratings.

If users give different amount of ratings they also may contribute with a user specific bias. To investigate the average rating per user is plotted:

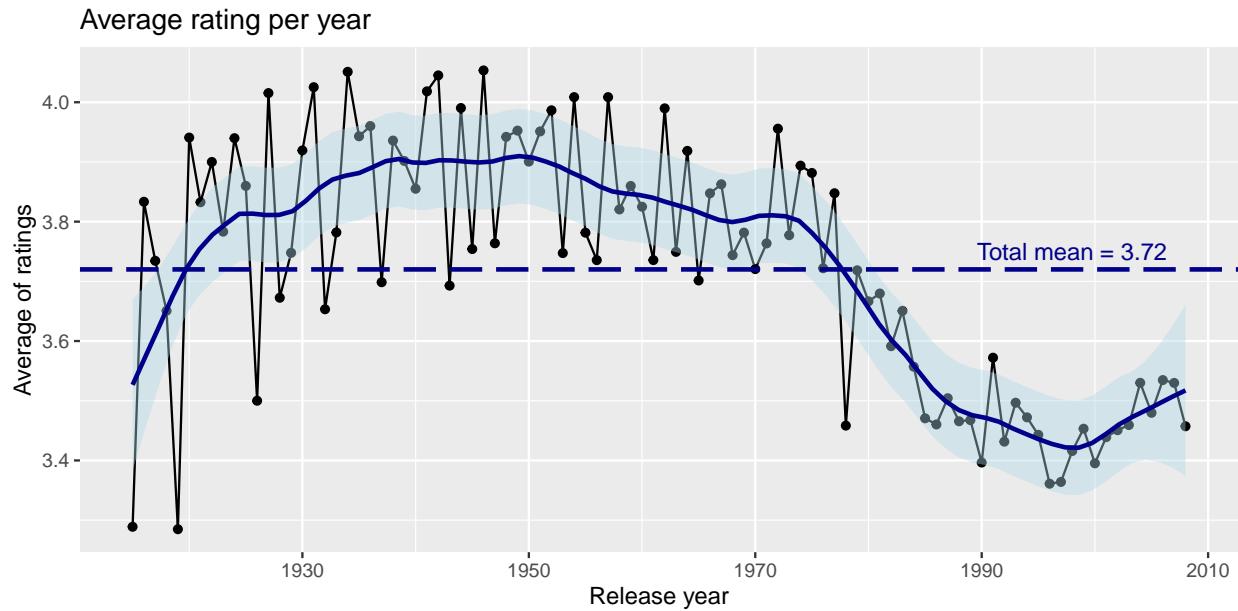


A user bias can be confirmed. The mean rating per user is 3.6 but there are users which systematically rate higher or lower than that.

Following with the perspective on movies, the number of ratings per movie is shown:



An uneven contribution of movies per release year could be determined from the table summary. If users tend to have a bias this could be also true for old or young movies. To investigate the average rating per release year is shown:



The average of the yearly average ratings is 3.72 and old movies are clearly above this mean. For the case of movies age discriminates and a bias from the release year can be determined. To summarize, this data exploration indicates variable dependencies. The rating of a movie is biased from the user and the release year has an influence on the average movie rating.

2.2 Modelling Approach

To compare the different models, the root mean square error is defined as a function:

```
#define the loss function
rmse <- function(act, fc){
  sqrt(mean((act - fc)^2))}
```

This evaluates the error by comparing the actual rating and the forecasted prediction for all instances.

To provide a baseline, the first forecast is done with the average of all ratings:

```
## Baseline Model----

#Calculate input parameters for forecast function
avg_rating <- mean(train$rating)

#Calculate error measure to compare models
baseline_rmse <- RMSE(train$rating, avg_rating)
```

Prediction Model	RMSE
Baseline (avg. rating)	1.060331

This means the prediction of ratings is always done with the average of the whole historic data set. This is a naive forecast and leads to a RMSE of >1.

To account for the movie specific bias, a linear regression increment is added to the forecast function:

```
## Model 1: Movie effect Model----

#Calculate input parameters for forecast function
movie_avgs <- train %>%
  group_by(movieId) %>%
  summarize(movie_avgs = sum(rating - avg_rating)/(n()))

#Create Forecast
fc <- train %>%
  left_join(movie_avgs, by = "movieId") %>%
  mutate(fc = avg_rating + movie_avgs) %>% .$fc
```

The forecast now depends on the overall average of all ratings and the average of previous ratings of a movie.

Prediction Model	RMSE
Baseline (avg. rating)	1.0603313
Movie Effect Model	0.9423475

Movie bias improves the RMSE, as variability in ratings can be explained to some extent from historic ratings per movie.

The user specific bias can also be added as a linear regression increment:

```
## Model 2: Movie + User effect Model----

#Calculate input parameters for forecast function
user_avgs <- train %>%
  left_join(movie_avgs, by="movieId") %>%
  group_by(userId) %>%
  summarize(user_avgs = sum(rating - avg_rating - movie_avgs)/(n()))

#create Forecast
fc <- train %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  mutate(fc = avg_rating + movie_avgs + user_avgs) %>% .$fc
```

With both increments included, the forecast now also depends on previous ratings of the user. This accounts for user bias and improves the RMSE

Prediction Model	RMSE
Baseline (avg. rating)	1.0603313
Movie Effect Model	0.9423475
Movie + User Effect Model	0.8567039

Data analysis suggested that the release year also has an impact on the rating. The release year is also added to the regression function:

```
## Model 3: Movie + User + Year effect Model----

#Calculate input parameters for forecast function
year_avgs <- train %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  group_by(year) %>%
  summarize(year_avgs = sum(rating - avg_rating - movie_avgs - user_avgs)/(n()))
```

```
#Create Forecast
fc <- train %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  left_join(year_avgs, by="year") %>%
  mutate(fc = avg_rating + movie_avgs + user_avgs + year_avgs) %>% .$fc
```

Prediction Model	RMSSE
Baseline (avg. rating)	1.0603313
Movie Effect Model	0.9423475
Movie + User Effect Model	0.8567039
Movie + User + Year Effect Model	0.8563777

All regression increments are now included without weighting in the forecast term. But the impact of variables becomes more stable with a high number of observations. Currently for example a user with very low number of ratings has the same impact on the forecast as the average rating per movie from many ratings.

To not overestimate the influence from a variable with low number of estimations a tuning parameter is introduced that factors in the count of observations and lowers the regression impact of those with low number of previous observations.

The forecast is produced for a series of regulator values. As a initial value the regulator is filled with values from 0 to 5.

```
## Model 4: Regularized Movie + User + Year effect Model----

#open regulator search, set first window from 0 to 5
regulator <- seq(0,.5,.5)

#apply forecast function to all regulator values
rmse_reg <- sapply(regulator, function(y){

  #Calculate input parameters for forecast function
  movie_avgs <- train %>%
    group_by(movieId) %>%
    summarize(movie_avgs = sum(rating - avg_rating)/(n()+y))

  user_avgs <- train %>%
    left_join(movie_avgs, by="movieId") %>%
    group_by(userId) %>%
    summarize(user_avgs = sum(rating - avg_rating - movie_avgs)/(n()+y))

  year_avgs <- train %>%
```

```

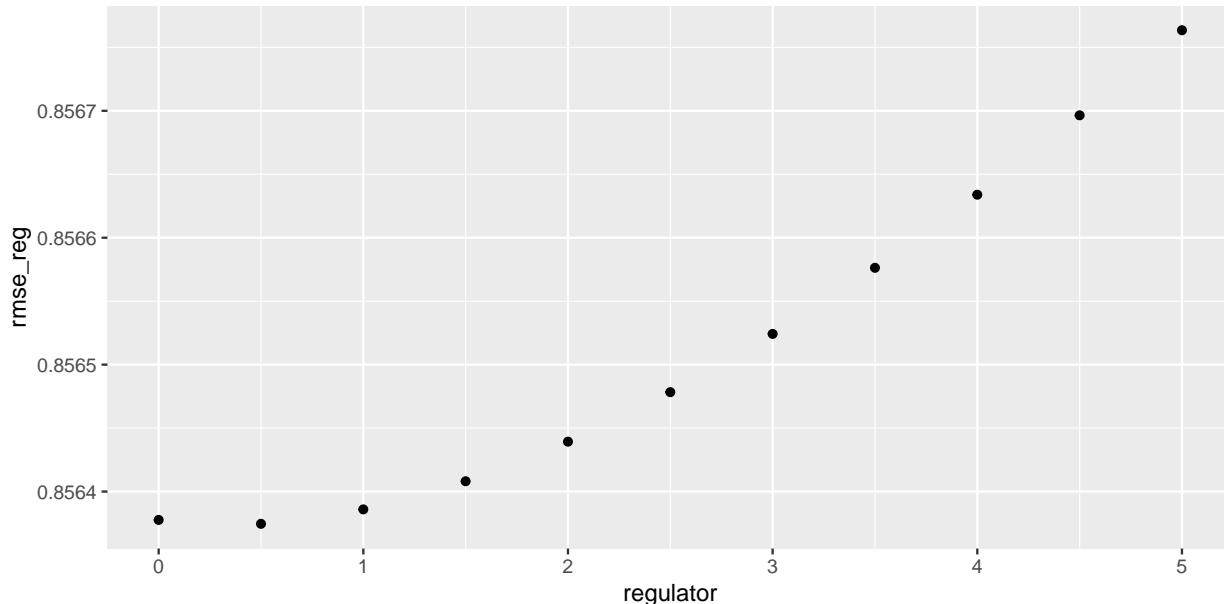
left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  group_by(year) %>%
  summarize(year_avgs = sum(rating - avg_rating - movie_avgs - user_avgs)/(n() + y))

#Create Forecast
fc <- train %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  left_join(year_avgs, by="year") %>%
  mutate(fc = avg_rating + movie_avgs + user_avgs + year_avgs) %>% .$fc

#Calculate error measure for all regulator values
return(RMSE(train$rating,fc))
})

```

The regulating values produce following RMSEs:

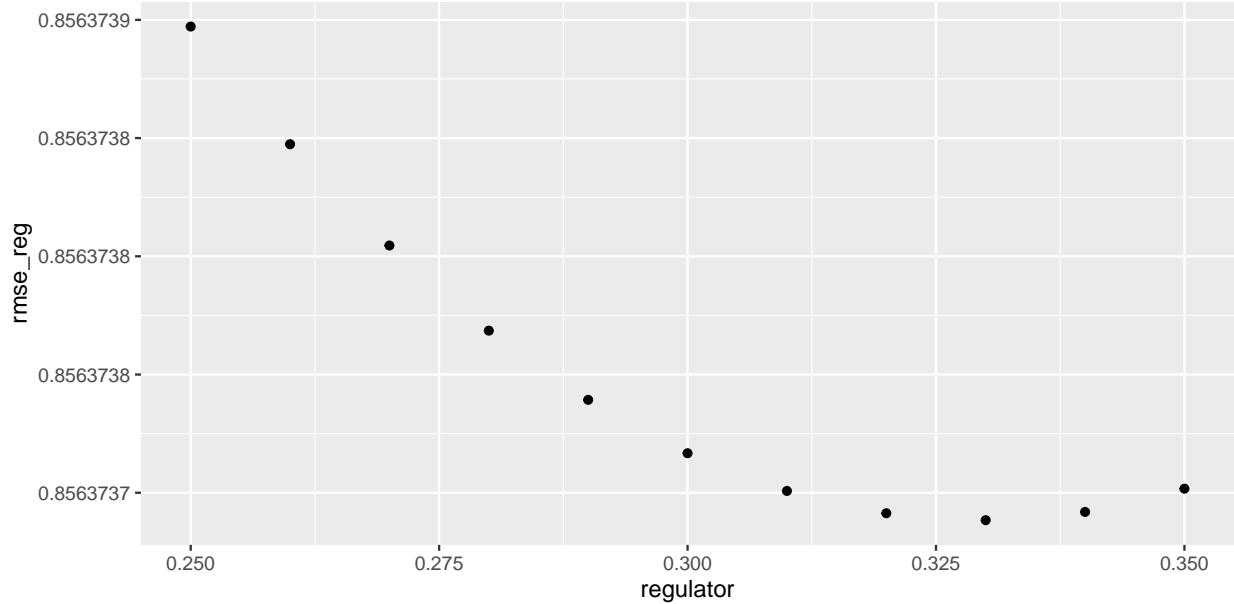


The preliminary solution is the regulating value is:

```
## RMSE of 0.8563746 with regulator of 0.5
```

To further improve, the window of valid values is narrowed down and the optimal tuning parameter on 0.01 digit level is chosen:

```
# show RMSE results per regulator
qplot(regulator, rmse_reg)
```



The final tuning parameter for model 4 is set to:

```
## RMSE of  0.8563737  with regulator of  0.33
```

This improves the model to some extent as the RMSE lowers, but asymptotically.

Prediction Model	RMSE
Baseline (avg. rating)	1.0603313
Movie Effect Model	0.9423475
Movie + User Effect Model	0.8567039
Movie + User + Year Effect Model	0.8563777
Regularized Movie + User + Year Effect Model	0.8563737

With further inclusion of variable dependencies to the linear regression model the forecast improves and the RMSE value is lower.

3 Results

Within the model approach chapter the training data set was used. To test for true accuracy, the models are run with the validation test set.

All regression elements lower the RMSE, therefore they contribute in explaining predicted outcomes. The regularization is, after including other variables, a tuning parameter that improves the regression, but by a narrow margin.

The final model predicts the validation set ratings with a RMSE of 0.8251.

Prediction Model	RMSE (validation set)
Baseline (avg reating)	1.061202
Movie Effect Model	0.938309
Movie + User Effect Model	0.825177
Movie + User + Year Effect Model	0.824827
Regularized Movie + User + Year Effect Model	0.825152

4 Conclusion

The validation prediction shows overall good error measures, compared to the whole rating scale values. But results of the validation show that the fine tunes regularized model is less accurate than the Model 3. The regularization parameter therefore can not explain all hidden and further dependencies. Future analysis should concentrate on explaining more in depth dependencies with matrix factorization and principle component analysis for example. Also, linear regression is seen as the most basic concept and more advances Machine Learning algorithms could reveal further potential in forecast accuracy.