HarvardX PH125.9x Data Science: Capstone

Andy Birch

10/05/2021

Contents

```r
# Install and load packages
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(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")
if(!require(broom)) install.packages("broom", repos = "http://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kableExtra", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(recosystem)) install.packages("recosystem", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(stringr)
library(ggthemes)
library(broom)
library(kableExtra)
library(lubridate)
library(recosystem)
```

1. Introduction

This analysis makes use of the MovieLens 10m dataset, which contains over ten million movie ratings provided by the GroupLens research group at the Department of Computer Science and Engineering at the University of Minnesota[1].

The specific task is to build a recommendation system that can accurately predict the rating that a user will give for a movie. The real life application of such a prediction would be to allow a company that sells or streams movies to make tailored recommendations that could drive sales and or engagement with the company.

This report is being submitted as part of the capstone project for the HarvardX PH125.9x Data Science course. The target for the final algorithm is to achieve an average root mean squared error (RMSE) of 0.86 or below.

[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872]

2. Data

i. Dataset checks

The script provided in the course materials downloads the data as a zip file from the GroupLens website[2] and prepares a single tidy file, *movielens*. This dataset contains 10,000,054 rows, with each representing a review for an individual movie (*movieId*) supplied by an individual user (*userId*). There are a total of 6 columns with the following attributes:

	Type	Distinct_values	Missing_values
userId	integer	69,878	0
movieId	numeric	10,677	0
rating	numeric	10	0
timestamp	integer	7,096,905	0
title	character	10,676	0
genres	character	797	0

Given that there a potential total number of 746,087,406 combinations of *moveId* and *userId*, the dataset with 10,000,054 records is very sparse, representing 1.34% of the potential combinations.

There are no missing values to contend with, but we can see a disparity in the number of unique values for *movieId* and *title*. We can see that each *movieId* always refers to a single *title*:

```r
# List any movieId that has multiple titles
movielens %>% group_by(movieId) %>% summarise(titles = n_distinct(title)) %>% filter(titles !=1)
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: movieId <dbl>, titles <int>
```

However one *title* has two different *movieId*s:

```r
# List any title that has multiple movieIds
(multi_movieId <- movielens %>% group_by(title) %>% summarise(movies = n_distinct(movieId)) %>% filter(
```

title	movieId	reviews
War of the Worlds (2005)	34,048	2,758
War of the Worlds (2005)	64,997	31

The use of a second *movieId* appears to be a mistake, but it has only been used for 1.11% of the reviews. Therefore it will have little impact on our ability to predict ratings for this title and will have an extremely neglible impact the overall predicitve power of any algorithm.

Finally lets check that each record is a unique combination of a *movieId* and a *userId*:

```r
# Check if each row is a unique combination of userId and movieId
n_distinct(paste0(movielens$movieId, "-", movielens$userId)) == nrow(movielens)
```

```
## [1] TRUE
```

Having completed these simple checks, we know that the data is of sufficient quality to conduct the analysis. The next step is to remove a 10% sample of records (“*validation*”) to be held out of **any** analysis and used solely for the final validation of model accuracy. The remaining 90% of records are stored in the *edx* dataset. Any records in *validation* that contain a *movieId* or *userId* that is not in the training set are moved to *edx* to ensure that we can always make a rating prediction.

We can now conduct exploratory data analysis for each data field, this will inform the model building process

[2] http://files.grouplens.org/datasets/movielens/

ii. movieId

The mean number of reviews per movie is 843, but the distribution is heavily positively skewed:

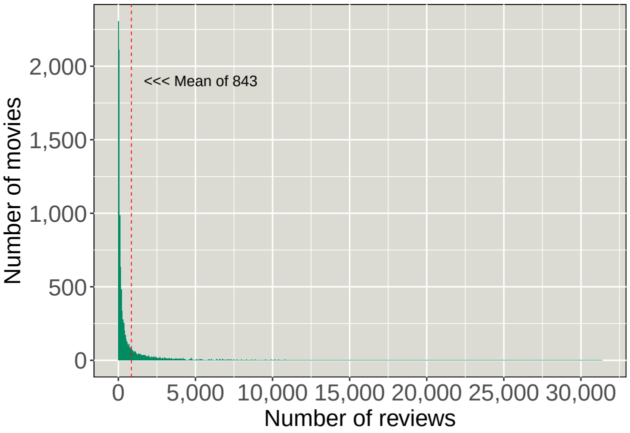

Figure 1: Number of reviews per movie

Looking at the summary statistics for the number of reviews per movie, the first quartile only have 30 reviews or fewer and the median number of reviews is 122:

minimum	q1	median	mean	q3	maximum
1	30	122	842.94	565	31,362

Further, when looking at the movies with an average rating of four or greater we can see a significant majority have a low number of reviews:

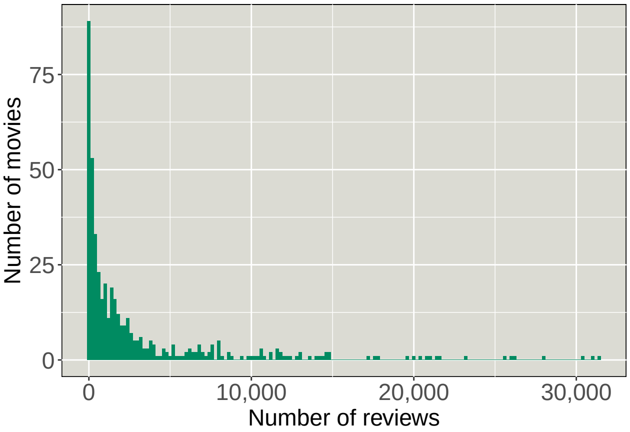

Figure 2: Number of reviews for films with average rating of four or greater

Of the 504 movies with a rating of four or higher, 71 only have three reviews or fewer. These movies appear to be quite obscure and their high rating should be treated with caution:

movieId	title	n_reviews	avg_rating
3,226	Hellhounds on My Trail (1999)	1	5.0
7,452	Mickey (2003)	1	4.5
7,823	Demon Lover Diary (1980)	1	4.5
38,320	Valerie and Her Week of Wonders (Valerie a týden divu) (1970)	1	4.5
42,783	Shadows of Forgotten Ancestors (1964)	1	5.0

In contrast, movies with a high number of reviews are instantly recognisable and highly critically acclaimed titles whose average rating is justified:

movieId	title	n_reviews	avg_rating
296	Pulp Fiction (1994)	31,362	4.15
356	Forrest Gump (1994)	31,079	4.01
593	Silence of the Lambs, The (1991)	30,382	4.20
318	Shawshank Redemption, The (1994)	28,015	4.46
110	Braveheart (1995)	26,212	4.08

This needs to be considered when building the model, as we should have less confidence in the average rating for movies with a small number of reviews. Failing to adjust for this issue could lead to overfitting, which can lead to the selection of a model that has good accuracy for the training set, but is poor for the validation set.

Regularisation is a popular technique for reducing the risk of overfitting and it will be used in the final model. Regularisation works by shrinking the coefficient of (in this case) *movieId* towards zero when there are a small number of observations.

iii. userId

The number of reviews per user is also positively skewed, with a much lower average than the number of reviews per movie:

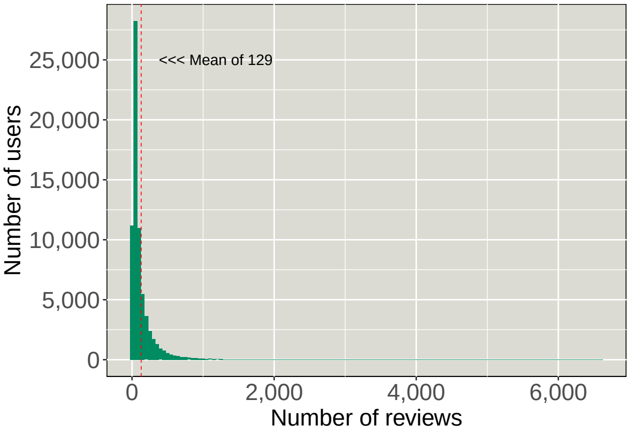

Figure 3: Number of reviews per user

The mean is within the interquartile range, which was not the case for the number of reviews per movie, and users had to have at least 10 reviews to be included, so the skew is less extreme:

minimum	q1	median	mean	q3	maximum
10	32	62	128.8	141	6,616

Whilst the skew for reviews per user is less extreme, it is still strong and regularisation would likely improve prediction accuracy.

iv. genres

The *genre* field is a combination individual genres from a list of 20 options:

```
##      [,1]                 [,2]          [,3]       [,4]      [,5]
## [1,] "(no genres listed)" "Children"    "Drama"    "IMAX"    "Sci-Fi"
## [2,] "Action"             "Comedy"      "Fantasy"  "Musical" "Thriller"
## [3,] "Adventure"          "Crime"       "Film-Noir" "Mystery" "War"
## [4,] "Animation"          "Documentary" "Horror"   "Romance" "Western"
```

The most complex value for *genre* has 8 different elements:

genres	sub_genres
Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller	8
Adventure|Animation|Children|Comedy|Crime|Fantasy|Mystery	7
Adventure|Animation|Children|Comedy|Fantasy|Musical|Romance	7
Adventure|Animation|Children|Comedy|Drama|Fantasy|Mystery	7
Adventure|Animation|Children|Comedy|Fantasy|Romance	6

There are 797 unique values in the *genre* field and the most popular have a large number of reviews:

genres	n_reviews	n_movies
Drama	733,296	1,815
Comedy	700,889	1,047
Comedy|Romance	365,468	379
Comedy|Drama	323,637	551
Drama|Romance	259,355	412

In theory the *genre* could provide useful predictive power, but the definition of a movie's *genre* is largely subjective. Whilst the individual genre elements are quite intuitive (drama, comedy etc), it is very difficult to quantify their existence in a movie.

We may expect to find films in the same franchise would have the same *genres*, but this is not always the case, as illustrated with a couple of examples:

title	genres	n_reviews
Terminator, The (1984)	Action|Sci-Fi|Thriller	15,737
Terminator 2: Judgment Day (1991)	Action|Sci-Fi	25,984
Terminator 3: Rise of the Machines (2003)	Action|Adventure|Sci-Fi	3,540

title	genres	n_reviews
Star Trek: The Motion Picture (1979)	Adventure|Sci-Fi	4,649
Star Trek II: The Wrath of Khan (1982)	Action|Adventure|Sci-Fi|Thriller	8,355
Star Trek III: The Search for Spock (1984)	Action|Adventure|Sci-Fi	5,392
Star Trek IV: The Voyage Home (1986)	Adventure|Comedy|Sci-Fi	6,648
Star Trek V: The Final Frontier (1989)	Action|Sci-Fi	3,622
Star Trek VI: The Undiscovered Country (1991)	Action|Mystery|Sci-Fi	5,789
Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	14,550
Star Trek: First Contact (1996)	Action|Adventure|Sci-Fi|Thriller	10,186
Star Trek: Insurrection (1998)	Action|Drama|Romance|Sci-Fi	3,822
Star Trek: Nemesis (2002)	Action|Drama|Sci-Fi|Thriller	1,470

Whilst there are common elements between the genres within each franchise, they are generally distinct from each other. Therefore it looks unlikely that *genre* will provide stronger predictive power for *rating* than *movieId* and *userId*.

v. timestamp

The timestamp records when the review was submitted, measured in the number of seconds since 1st Jan 1970 (UTC). The movie *title* also contains the year that the movie was released, so this is extracted to calculate the age of the movie at the time of each review.

The earliest review date is 09-Jan-1995 and the most recent is 05-Jan-2009. Plotting the number of reviews against the date, grouped by year and month, shows an unexpected pattern:

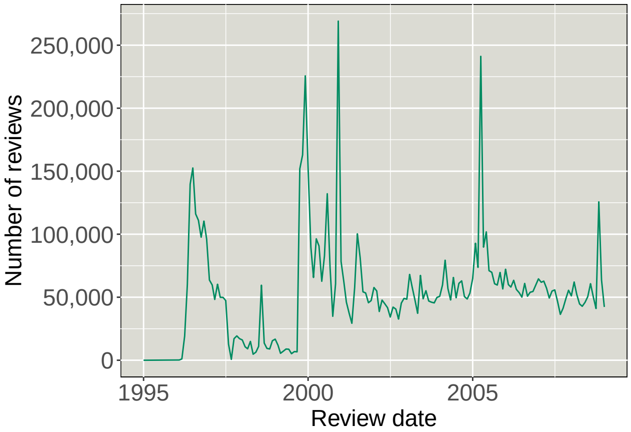

Figure 4: Number of reviews per month

We would expect to see a smooth change in the number of reviews over time, probably increasing as more users are added. There is no explanation given for the sudden changes in the number of reviews, there could be data classification errors or it could be how the records were chosen from the larger MovieLens dataset. Regardless, it raises concerns about the predictive power of the timestamp of the review.

We can still explore the data to see if any patterns emerge over time, starting with the prevalence of the different ratings over time. The following chart shows the proportion of ratings in each year that were given each rating:

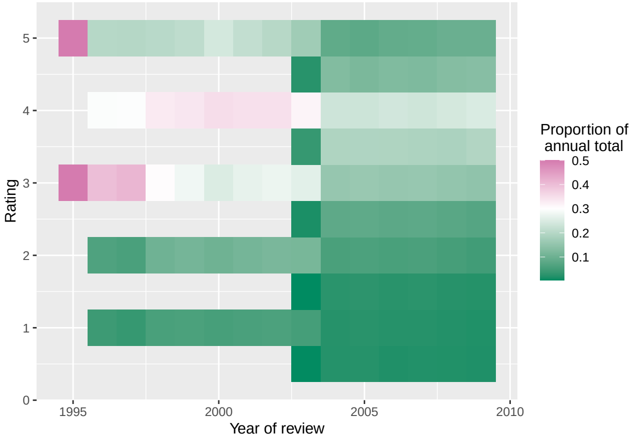

Figure 5: Prevalence of ratings over time

We can immediately see that half point ratings were only introduced part way through the time period, on 12-Feb-2003 to be precise. This helps to explain why the distribution of ratings across the whole time period has lower values for half point ratings than might be expected:

When the distribution of ratings is split to before and after 12-Feb-2003, both datasets are far more normally distributed:

The following chart provides a more detailed look at the change in average rating since 1997 (prior years excluded due to extreme volatility). Each data point is the average for an individual month, with a smoothed regression line:

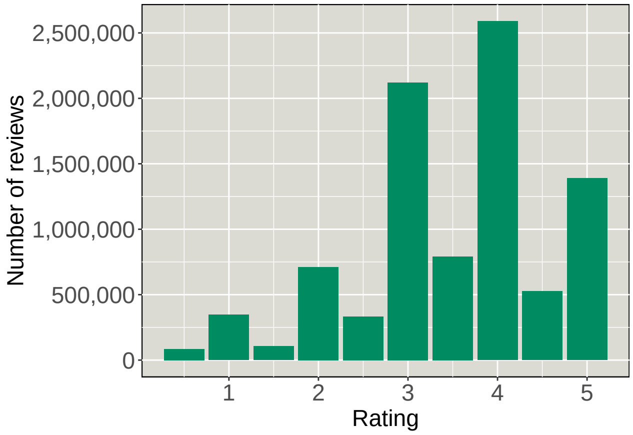

Figure 6: Number of reviews per rating

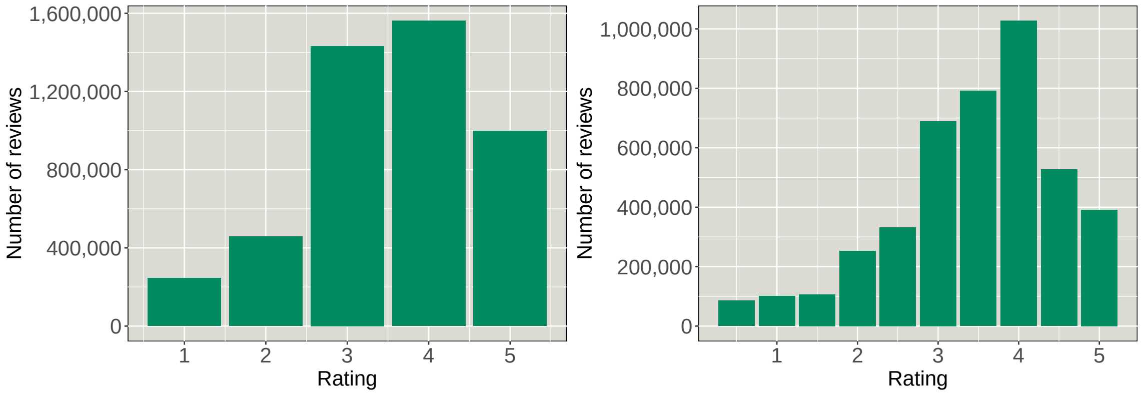

Figure 7: Number of reviews per rating prior to (left) and after introduction of half star ratings (right)

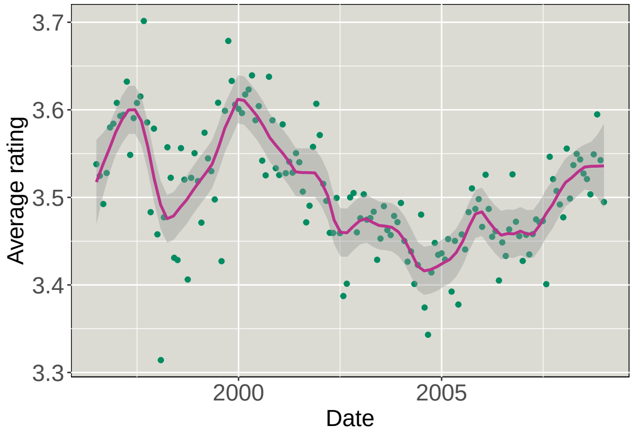

Figure 8: Average rating over time

There are some strong movements in the average rating, but these appear to be erratic with low predictive power.

Rather than looking at how all ratings change over time, we can look at how the average rating for individual movies evolve over time, by using the age of the film at the time of the review. The following chart shows the pattern for the five movies with the largest number of reviews:

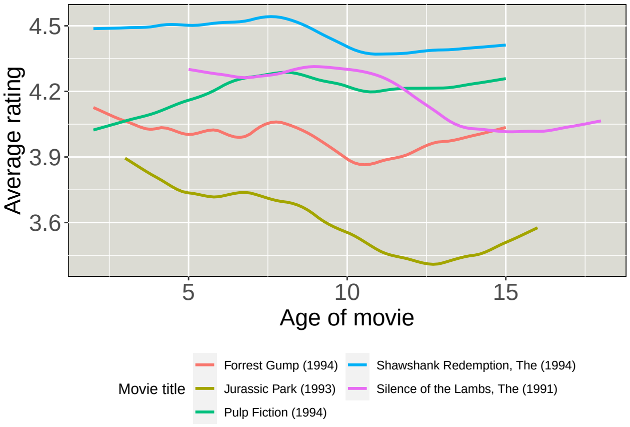

Figure 9: Average rating over time for five most frequently reviewed movies

Whilst this chart only looks at five movies, they have jointly received 150,198 reviews. Therefore the lack of pattern in this chart suggests that age of movie at time of review will not be a powerful predictor.

vi. Choice of input data

Given the findings above, the data fields that appear to show the most predictive power are the *movieId* and *userId*, so these will be the first variables we use to build models.

3. Modelling

i. Data preparation

Potential models will be trained using 90% of the *edx* dataset, with the remainder being used to test each model. The course instructions stipulated that root mean squared error (RMSE) be used as the loss function test model accuracy:

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

Where:

$y_{i,u}$ = actual rating given for movie i by user u

$\hat{y}_{i,u}$ = predicted rating for movie i by user u

N = number of movie ratings

ii. Naive model

The use of a naive model is always a good starting point for any machine learning process. It sets a benchmark performance that more sophisticated models need to beat to justify their complexity. In this case, the use of the mean average of all known ratings (μ) can be used as the single value for all predictions. Any difference to the actual rating is assumed to be due to random variation:

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

Where:

μ = mean average of all actual ratings

$\epsilon_{i,u}$ = random variation for rating of movie i by user u

The RMSE for this model is 1.06, which is large, but not unsurprising given the simplicity of the approach. We will need to make better use of the data if we are to beat the target RMSE of 0.86.

iii. Movie bias

Only using μ to predict the rating fails to recognise the range of average ratings that different movies recieve:

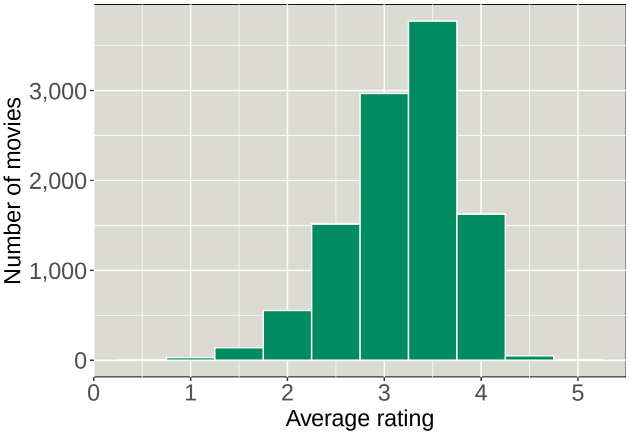

Figure 10: Distribution of average movie ratings

We can calculate a *movie bias* (b_i) by taking the average difference between the actual rating ($y_{i,u}$)and μ for each *movieId*:

$$b_i = \frac{\sum_u (y_{i,u} - \mu)}{N}$$

Our model now becomes:

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

Where:

μ = mean average of all actual ratings

b_i = movie effect

$\epsilon_{i,u}$ = random variation for rating of movie i by user u

Adding the movie bias, reduces the RMSE, but it is still far in excess of the target:

method	RMSE
Mean movie rating	1.06
Movie bias	0.94

iv. User bias

Just as there was a range of average ratings for different movies, the same is true of users. It makes intuitive sense that some users will tend to be more harsh in their ratings than others. The following chart shows the distribution of residuals ($y_{u,i} - \mu - b_i$):

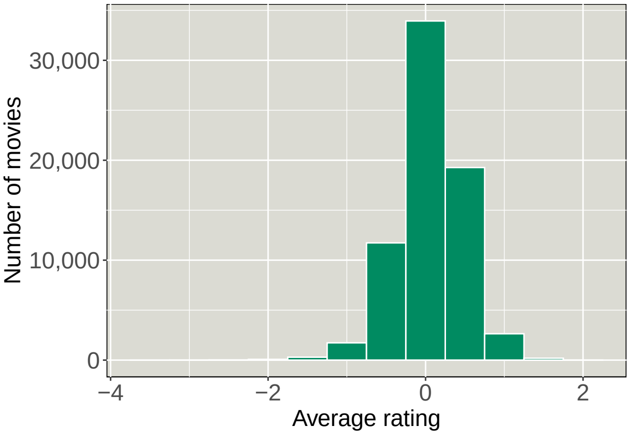

Figure 11: Distribution of residuals for movie bias model

We can add the user bias to the model:

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

Where:

μ = mean average of all actual ratings

b_i = movie effect

b_u = user specific effect

$\epsilon_{i,u}$ = random variation for rating of movie i by user u

The RMSE is now very slightly below the target level:

method	RMSE
Mean movie rating	1.06
Movie bias	0.94
User bias	0.86

This is not sufficient however. This result is a random variable as the training and test sets were created using sampling, a different sample may produce a worse RMSE. Furthermore, the true test is the assessment against the *validation* set which is also a random sample and the model could be less accurate for that data.

v. Using recosystem

So far the models have only considered how each movie is rated by different users and how each user rates different movies. There is a wealth of additional information that can be gleaned by identifying similarities between users and between movies, matrix factorisation is one approach that can be used to achieve this.

Essentially the objective is to use the observed values in the matrix of *movieId* and *userId* to predict the unknown values. Since we have already calculated the bias for *movieId* and *userId*, we want to predict the unknown values for the residuals: $Y_{i,u} - \mu - b_i - b_u$

This indicative example shows what the problem looks like:

	Movie 1	Movie 2	Movie 3	...	Movie n
User 1	??	??	0.4	...	??
User 2	??	3	??	...	-0.5
User 3	-1.2	??	??	...	??
...	...	...	...	...	...
User m	0.3	-0.5	??	...	??

Our model will then be the following:

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

Where:

μ = mean average of all actual ratings

b_i = movie effect

b_u = user specific effect

$\alpha_{i,u}$ = movie - user interaction effect

$\epsilon_{i,u}$ = random variation for rating of movie i by user u

Whilst there are numerous matrix factorisation techniques available, the *LIBMF* library has been shown to perform strongly, making use of modern multicore processors to perform parallel computations to improve speed[3]. The *recosystem* package is a wrapper for LIBMF that simplifies data preparation and model building and has been chosen for this project.

After the initial data preparation, the model parameters need to be tuned using cross validation. The values recommended by the package authors were used initially:

Parameter	Description	Tuning options
dim	Number of latent factors	10, 20, 30
lrate	Learning rate, which can be thought of as the step size in gradient descent	0.1, 0.2
costp_l1	L1 regularization cost for user factors	0
costq_l1	L1 regularization cost for movie factors	0
nthread	Number of threads for parallel computing	4
niter	Number of iterations	10

Note that both *movieId* and *userId* are regularised, addressing the issue of extreme values from small numbers of observations highlighted earlier.

The parameters that gave the smallest loss were then used to train the final model, which can in turn be used to make our predictions for the residuals. In a small number of cases the prediction lies outside the range of true potential ratings (0.5 to 5), so as a final step, our predictions are capped between 0.5 and 5.

The results of this model were very impressive and comfortably beat the target RMSE:

There may be opportunities to further improve the model by allowing the tuning process to consider a wider range of values. The optimal parameters from the initial tuning included the smallest learning rate and the largest number of latent factors, perhaps reducing the former and / or increasing the latter would provide another improvement in accuracy.

Applying such changes would significantly increase the processing time however, which is unnecessary given the performance of the current model. Given the impressive RMSE of this model, it will be used as our final model.

[3] https://github.com/yixuan/recosystem/blob/master/README.md

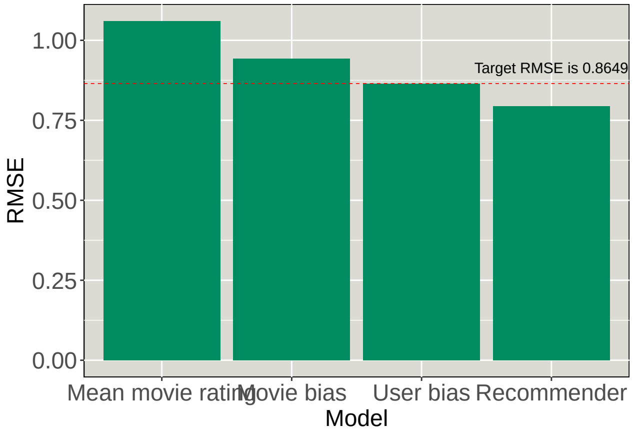

Figure 12: RMSE results for each model

4. Results

The final assessment of our chosen model is performed against the *validation* set which has been held out of all analysis and testing thus far. The RMSE for the validation set is NA, which is significantly better than the target of 0.86, therefore the model has achieved our stated objective.

5. Conclusions and further work

A dataset of over ten million movie review ratings from MovieLens was used to build models to predict ratings for other movie reviews. The objective was to develop a model that would achieve an RMSE of 0.86 or lower.

The simplest models calculated and applied bias terms for the movie and for the user, they produced good results, but were not accurate enough to beat the target RMSE. Our final model used matrix factorisation to identify similarities between movies and between users to draw on more rating information to make predictions. This significantly improved accuracy and comfortably beat the target RMSE.

There are two areas that could improve accuracy further and could warrant additional work

Splitting pre and post half star

All ratings prior to 12-Feb-2003 were limited to whole numbers, with half point ratings introduced from this time. If the model is intended to be used to make predictions on future reviews where half point ratings are available, then it may improve accuracy to only use known ratings from 12-Feb-2003 or later. This approach was not tested in this assessment as the exercise was to assess all the reviews in the dataset provided, including many from prior to the introduction of half point ratings.

Further tuning of recosystem

The initial range of parameters used to tune the recosystem model provided an excellent RMSE which achieved our objective, therefore no further tuning was undertaken. However it is noted that the learning rate and number of latent factors selected were at the extremes of the range made available, so extending the range of parameters to allow for an even more gradual rate of learning and / or including more latent factors could improve accuracy further.