MovieLens Capstone Project

Jebbe Schellevis

6 September 2021

1 Introduction

One of the most popular fields within data science is machine learning. It allows data scientists to use (large) datasets to train algorithms and use them to predict future or (yet) unknown events. In this project, machine learning is applied to predict a user's (person's) rating for a specific movie. A practical application for such algorithms is in a movie recommendation system, such as those employed by Netflix. This report describes the data, methodology, and results for the machine learning project mentioned.

1.1 Goal

The goal of this project is to develop an algorithm that predicts a user's rating for a movie, given predicting factors such as the movie in question, its genre and other ratings for the same movie and from the same user. The quality of the algorithm will be evaluated using a final evaluation dataset, for which the actual given ratings are known. The measure of quality used is the RMSE (Root Mean Squared Error), in short the average difference between the predicted and actual movie ratings. The objective is to minimize the RMSE as that implies the algorithm predicts close to the actual ratings. An RMSE of around ~0.865 is considered a good result for this analysis.

1.2 Dataset

The data used in this machine learning project is (a subset of) the MovieLens dataset obtained from http://grouplens.org/datasets/movielens. It contains 10 million (10M) movie ratings from the MovieLens movie recommendation site, found at http://movielens.org. For each movie rating, the dataset includes several characteristics:

Table 1: Dataset preview

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

For use in this project the 10M dataset is split into a working and (final) evaluation set in a 90%/10% ratio. This is described in more detail in chapter 2.

1.3 Approach

The main steps in this data science project are:

1. Load and clean dataset
2. Split dataset into training, test and final evaluation sets
3. Train different machine learning algorithms and evaluate their predictive power
4. Select best prediction algorithm and obtain results on final evaluation set
5. Summarize methodology and results in this report

A more detailed, technical explanation of steps 1 through 4 is provided in chapters 2 and 3.

2 Methodology

This chapter describes the methodology used in this data science project. For each of the aforementioned steps, it details the methods and functions used. Some of the code used is (visibly) included in this report, but only when it supports the storyline and/or adds to the understanding of the reader. The full body of code including commentary can be found in a separate file `movielens.r`.

2.1 General and report

This analysis was done using the language R, mostly used for statistical analysis. Scripts were developed in RStudio 1.3.959 on Windows 10. The version of R used is 4.0.2. This report was set up in R Markdown, a markup language that allows for R code and regular text to be mixed into a readable document.

2.2 Load and clean dataset

The code for loading and cleaning the dataset was provided by the edX course developers. Sequentially, it involves (a) downloading the data from the website mentioned in chapter 1.2 (actually in two files: one for the ratings and one for the movie details); (b) reading both data files line by line and splitting them into columns before putting them into a matrix or data table; (c) setting the right column types and titles; and finally (d) joining the matrix (converted to data frame) and data table into a single data frame with all ratings and movies data.

2.3 Split dataset

The first step in splitting the dataset was also provided by the edX course developers. It involves using the `createDataPartition()` function to split the 10 million row dataset randomly into two chunks of 90% (the working set) and 10% (the validation set). Next, the code checks to make sure all users and movies present in the final validation set are also present in the working set. This ensures that the algorithms used for prediction can be trained optimally for the final validation.

The second step in splitting the dataset involves splitting the working set (90% of the data, or approximately 9 million rows) into train and test sets. This again is done using `createDataPartition()` function, this time using a 50%/50% partitioning of the working set. This results in two datasets: 'train' and 'test', both containing approximately 4.5 million rows:

```
# Create train and test sets in 50/50 ratio
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.5, list = FALSE)
train <- edx[-test_index,]
test <- edx[test_index,]

# Show dimensions for train and test sets
dim(train)
```

```
## [1] 4500027       6
```

```
dim(test)
```

```
## [1] 4500028       6
```

2.4 Train algorithms

After preparing and splitting the data, several algorithms were trained to predict movie ratings based on the training set. The complexity of the algorithms was gradually increased in this process up to the point where the RMSE (based on the test set) neared the goal of ~0.865. The textbook 'Introduction to Data Science: Data Analysis and Prediction Algorithms with R' by Rafael A. Irizarry was used as a basis for suggestions on models to use and code structure.

Predictors used

Based on the fields available in the dataset, there are a number of possible predictors that can be used to train a prediction model. Below a short consideration is given on the perceived predictive value of each field or data point in the dataset:

- **Movie**: probably the most important predictor as some movies are generally considered better than others
- **Year of release**: not considered a good predictor in itself, as good and bad movies are of all times.
- **Genre**: might be a predictor, but has strong correlation with the movie itself
- **User**: considered a predictor as people usually rate consistently
- **Date and time of rating**: day or time are not deemed a good predictor in themselves (why would a rating in the weekends be higher or lower than on working days?). But relative to the movie's year of release it might be. Movies might be rated worse as time passes (only the best movies stay 'hot')

Models

In this project, four different models (M1 through M4) were developed and trained to predict movie ratings. Each next model builds upon the previous and adds complexity. This paragraph describes the setup of each of the models.

M1 - Movie as predictor The first model is a very simple prediction model, and only one step more advanced than simply predicting the same (average) score for each movie. The M1 model calculates the average deviation per movie from the overall average rating of all movies and uses this for prediction. In effect, the predicted rating only takes into account the movie being rated, not any other factors.

In this model, first, the overall average rating of all movies in the training set is calculated:

```r
mean(train$rating)
```

```
## [1] 3.512439
```

Next, the deviation from the overall average is calculated for each individual movie in the training set. This step is not strictly necessary (we could also use the nominal rating for each movie), but is convenient to prepare for later models.

```r
movie_avgs <- train %>%
    group_by(movieId) %>%
    summarize(movie_effect = mean(rating - overall_avg_rating))
```

This results in a set of movie-specific ratings that can be used to predict future ratings. To give an example: movie The Net (1995) has an average rating of 3.15. Considering the overall average, the movie-specific deviation is -0.36. This deviation plus the overall average is the rating that will be predicted for all future ratings involving this movie.

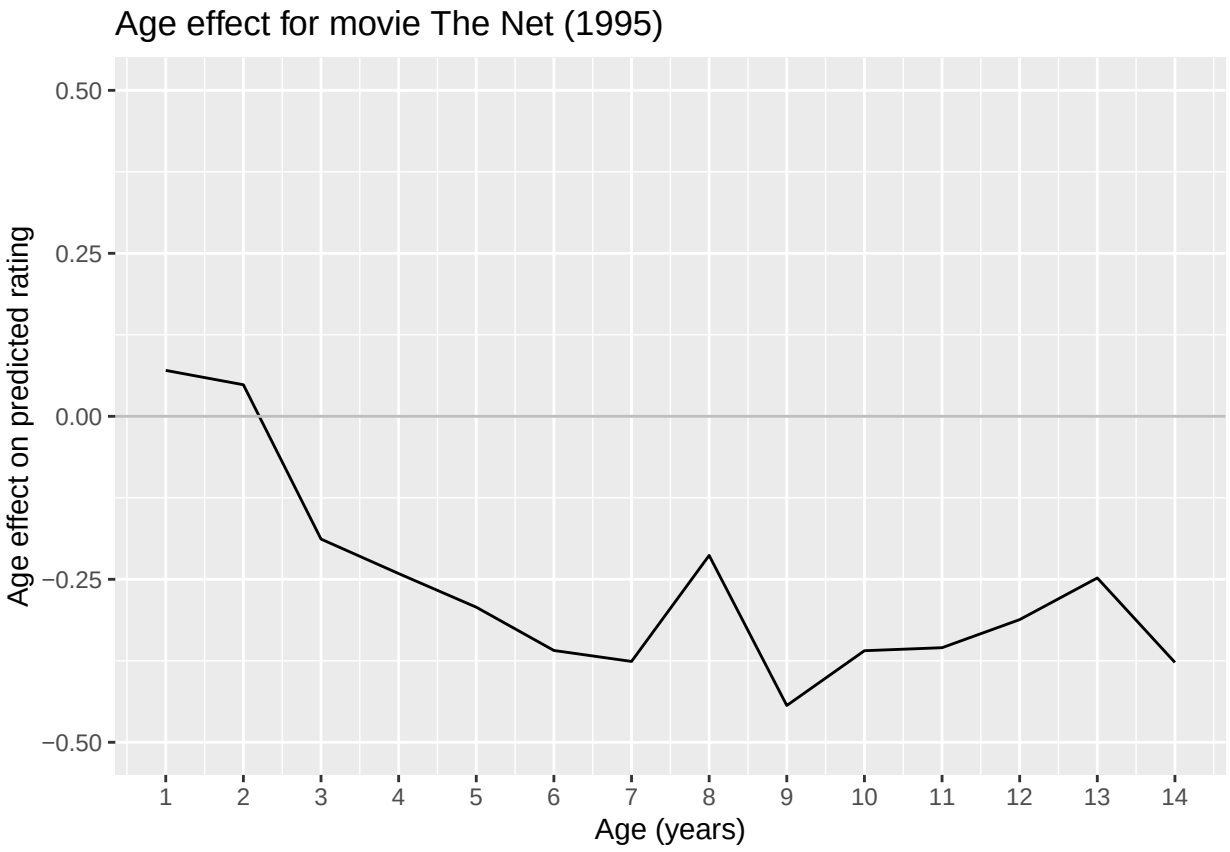
M2 - Movie and user as predictors The second model (M2) builds on the first model and adds a second predictor. Instead of only looking at the movie rated, it also takes into account which user submitted the rating. This is relevant because users tend to rate consistently higher or lower than other users.

To build this model we take each rating in the training set, subtract the overall average rating, and then subtract the movie-specific deviation from the average rating. The residual value is then averaged per user and labeled the 'user effect'.

```r
user_avgs <- train %>%
    left_join(movie_avgs, by='movieId') %>%
    group_by(userId) %>%
    summarize(user_effect = mean(rating - overall_avg_rating - movie_effect))
```

As with M1, the result is a set of values that can be used to predict future ratings. Returning to the example from M1: the prediction for a future rating of movie The Net is not only based on the movie itself (average rating 3.15), but also on the user that will be doing the future rating. Let's assume the user effect for a particular user is 0.2 (she rates consistently higher than other users), then the rating predicted will be: 3.51 (overall average) + -0.36 (movie effect) + 0.2 (user effect) = 3.35.

M3 - Movie, user and age as predictors The third model (M3) adds one more predictor: the movie's age at the time of the rating. This makes sure time effect is taken into account when making predictions. For example, in the first two years after The Net (1995) came out, it enjoyed higher ratings than in the next 12 years:



To use the movie's age at time of rating, we first need to extract the movie's year of release from the movie's title. Next we can extract the year in which the rating is done from the rating timestamp and then subtract the movie's year of release:

```
train <- train %>% mutate(
  movie_year = as.numeric(str_extract(title, "(?<=\\()\\d{4}(?=\\))")),
  movie_age = year(anytime(train$timestamp))-movie_year)
```

Next, we take each rating in the training set and subtract the overall average rating, then subtract the movie-specific deviation from the average rating, and finally subtract the user-specific effect. By averaging the residual by movie and movie age (in years) at the time of rating, we end up with an effect score for each movie and each possible movie age (in years).

```
age_avgs <- train %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    group_by(movie_age) %>%
    summarize(age_effect = mean(rating - overall_avg_rating - movie_effect - user_effect))
```

In our example of The Net this would work as follows: the prediction for a future rating will be: 3.51 (overall average) + -0.36 (movie effect) + 0.2 (user effect) + -0.38 (age effect for The Net after 14 years) = 2.97.

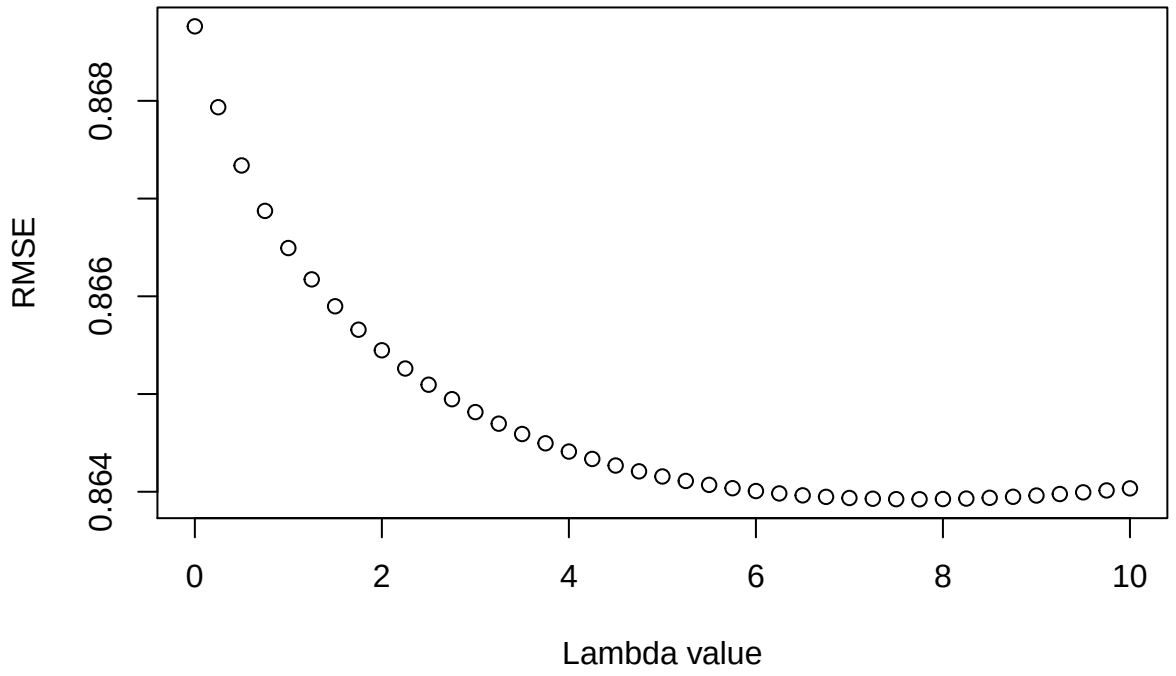
M4 - Movie, user and age as predictors (regularized) The fourth and final model (M4) uses the same predictors as M3: movie, user and age. But where prediction factors are nominal in M1 through M3, M4 uses so-called regularized factors. Regularization is a technique that can be used to normalize outliers

that are based on small sample sizes. Some movies only have a few ratings, and some users have only rated a few movies. In those cases, the resulting effect factor can be distorted due to the small sample. Using regularization helps normalize these distortions, resulting in better factors and thus a more accurate prediction.

To apply regularization averages are not calculated as simple averages, but averages over the number of ratings plus a penalty factor called lambda. This is shown in the code snippet below, where `l` is lambda. This technique ensures that the factor for small sample sizes reverts to zero, while not materially affecting factors for large sample sizes.

```
movie_avgs <- train %>%
      group_by(movieId) %>%
      summarize(movie_effect = sum(rating - overall_avg_rating)/(n()+l))
```

The optimal value for lambda is different for each project and dataset. To find the optimal value, we can run the model several times with different values for lambda (this is called tuning). In this project lambda values between 0 and 10 were tried, with intermediate steps of 0.25. For each lambda value the RMSE of the model on the test set is calculated. The lambda value for which RMSE is lowest is the optimal value. The plot below shows RMSE for tested lambda values and indicates that a lambda between 7 and 8 is optimal.



More specifically, the value of lambda with that yields the lowest RMSE is:

```
# Return lambda value that yields lowest RMSE with M4
lambda_tuning[which.min(m4_tuning_rmse)]
```

```
## [1] 7.75
```

After tuning, the M4 model is re-run using regularized values based on a lambda value of 7.75. The resulting factors differ slightly from the ones produced by the M3 model.

2.5 Evaluate best prediction algorithm

The four models are consecutively trained on the training set and then tested against the test set, resulting in an RMSE for each model. The table below shows the resulting RMSEs for the four models.

Table 2: Model results

Model	RMSE
M1 - Movie as predictor	0.9440085
M2 - Movie and user as predictors	0.8697242
M3 - Movie, user and age as predictors	0.8687621
M4 - Movie, user and age as predictors (regularized)	0.8639232

Based on the RMSE, it appears that the final and most complex model (M4) is the best model to predict movie ratings on the test set. This model takes into account several predictors (movie, user, and movie age) and is regularized to better deal with small sample sizes.

3 Results

For the final performance assessment, the best performing model (M4) is run against the evaluation dataset. This set contains 1 million records and has not been used in training or intermediate testing of the machine learning models.

The result of this final validation is an RMSE of:

```
## [1] 0.8639279
```

4 Conclusion

This final chapter summarizes the results of this data science project and puts these in a perspective of limitations and recommendations for further analysis.

4.1 Summary of results

In this data science project, four machine learning models were developed to predict movie ratings based on predictors such as the movie in question, the user giving the rating and the movie's age at the time of rating. The models were trained on the MovieLens dataset supplied by GroupLens, containing 10 million movie ratings.

Of the four models trained and tested, the final and most complex model yielded the most accurate predictions of movie ratings. This model includes movie, user and movie age as predictors and uses regularized factors to penalize small sample sizes.

Running this model against the final validation dataset resulted in an RMSE of 0.86393.

4.2 Limitations of the analysis

Given the nature of this project, the analysis and results presented have some limitations. The first is that it only includes three predictors: movie, user and the movie's age. Other data points, such as the movie's genre, were discarded due to time constraints. Including other or more predictors can improve the accuracy of the model and result in better predictions.

A second limitation is that only basic machine learning techniques were used in this analysis. More advanced techniques, such as matrix factorization, can improve the predictive power of a given model, but are also more complex and time-consuming to implement.

A third limitation is that the movie's age is not a suitable predictor for future ratings. In this project the goal was to predict the ratings in the validation set, which are historical rather than future ratings. That means the model was indeed trained in the combinations of movie and age presented by the validation set. When used as a predictive model for future ratings, the model will not be trained in movie-age combinations it encounters (as those ratings have not been recorded yet), resulting in less accurate predictions.

4.3 Recommendations for future analysis

If one would want to build on or improve this analysis in the future, the limitations from the previous paragraph provide some guidance into how to do this. There are two recommendations for future analysis:

1. Thoroughly analyze correlation between all available predictors and movie rating, to assess whether additional predictors could improve the predictive power of the model.
2. Apply more advanced techniques, such as matrix factorization to improve the predictive power of the model.