MovieLens Project

Aly O'Mahoney

10/11/2020

Abstract

In 2006 Netflix launched an open competition - improve their in-house recommendation algorithm (Cinematch) by at least 10% and win $1,000,000. Three years later the competition was won by BellKor's Pragmatic Chaos, a seven person group consisting of statisticians, machine-learning experts and computing engineers [1].

A summary of the techniques used by BellKor's Pragmatic Chaos is available here. It discusses how techniques such as neighbourhood models, matrix factorisation and regression were used to achieve the winning result. Although gradient boosted decision trees were used to combine over 500 models, the winners noted the following [2]:

> However, we would like to stress that it is not necessary to have such a large number of models to do well. The plot below shows RMSE as a function of the number of methods used. One can achieve our winning score (RMSE=0.8712) with less than 50 methods, using the best 3 methods can yield RMSE < 0.8800, which would land in the top 10. Even just using our single best method puts us on the leaderboard with an RMSE of 0.8890. The lesson here is that having lots of models is useful for the incremental results needed to win competitions, but practically, excellent systems can be built with just a few well-selected models.

Introduction

This report aims to construct a movie recommendation system via machine learning methods. Approach 1: Simple Bias accounts for basic movie to movie and user to user variability. Approach 2: Regularised Bias is similar, however regularisation is implemented. This turns out to be only a minor improvement on non-regularised bias. Approach 3: Collaborative Filtering uses a nearest neighbours style approach. It makes use of the idea that some users are similar to other users, and that some movies are similar to other movies. Final Model (Results) retrains the best performing model and assesses its performance using a validation set which has not been used for model construction or selection at any point in this report.

The data being worked with is the MovieLens 10M Dataset. It was created by GroupLens, a research group in the Department of Computer Science and Engineering at the University of Minnesota [3]. It contains over ten million ratings for over ten thousand movies from various users.

For this report, the MovieLens 10M Dataset is split into a training and a test set (`edx` and `validation` respectively). Only the `edx` data set is used for model construction. The `validation` data set is used only for assessing the performance of the *final* model. `edx` is split into `edx_train` and `edx_test`. Various models are constructed using `edx_train` and their performances are assessed using `edx_test`. The best performing model is then retrained using `edx` and assessed using `validation`. This way, `validation` has no effect on which model is selected to be the final model. The R code used to construct these data sets, models and plots is available in this GitHub repo.

Data Exploration

The structure of the `edx` data set is shown below. This section aims to visualise and explore the data being worked with.

```
## 'data.frame':    9000055 obs. of  9 variables:
##  $ userId      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId     : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating      : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp   : int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885
##  $ title       : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres      : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Actio
##  $ release_year: num  1992 1995 1995 1994 1994 ...
##  $ year_rated  : num  1996 1996 1996 1996 1996 ...
##  $ month_rated : Ord.factor w/ 12 levels "Jan"<"Feb"<"Mar"<..: 8 8 8 8 8 8 8 8 8 8 ...
```

Overview of Ratings

Figure 1 shows the distribution of the ratings, with the mean rating 3.51 indicated by the red dashed line. The plot indicates that, although ratings are on a scale of 0.5-5, users tent to rate movies more positively than negatively. It is also apparent that integer ratings are more common than half star ratings.

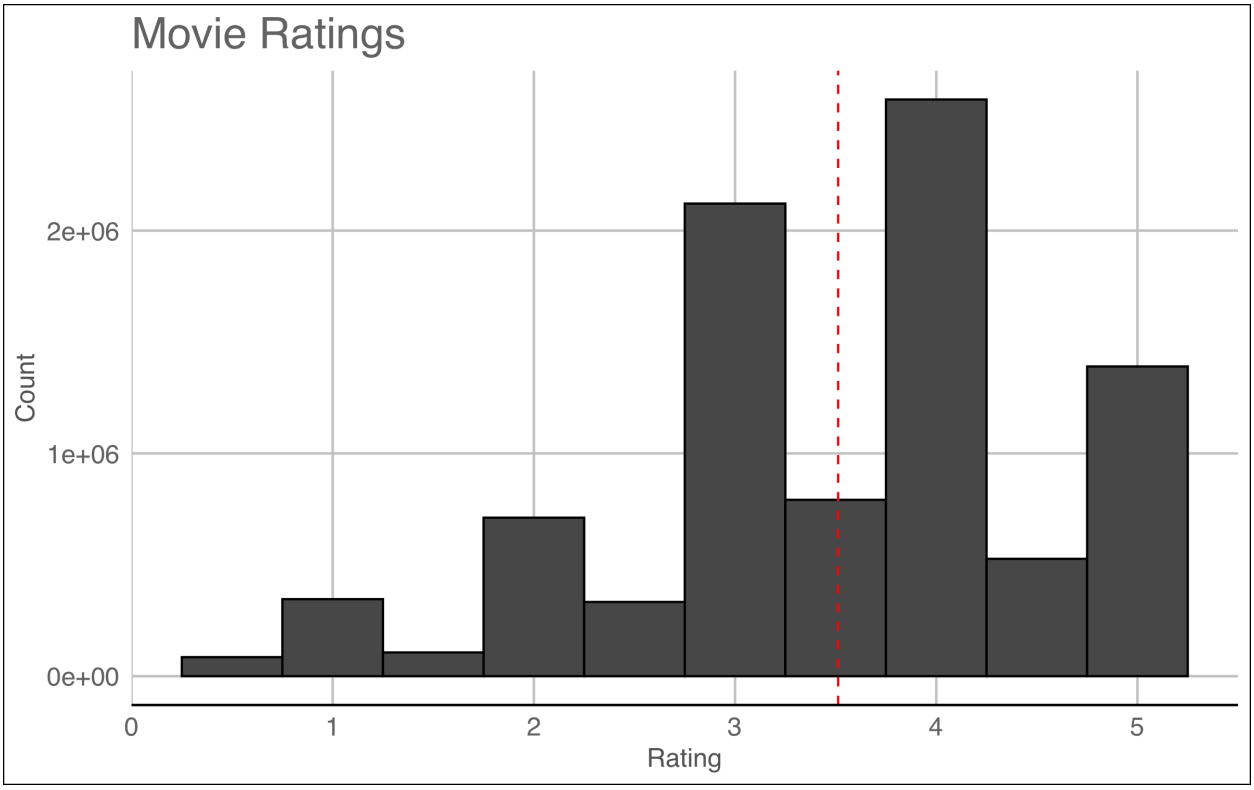


Figure 1: Histogram of all movie ratings in edx. The red dashed line represents the mean movie rating.

Release Year

The year in which a movie was released may not appear to be significant. However, Figure 2 illustrates that there is some kind of trend. Were there lots of really good movies release in the 40s and 50s? Why are films getting worse as time goes on?

An extremely important observation is the vertical spread of the data across time. Mean ratings for the time period 1920-1950 are spread further from the red line than ratings post 1990 for example. This is due to there being more movies (and more ratings) made per year as time has progressed.

Perhaps that explains part of the story, although there has undoubtedly been a decrease in movie ratings over the past fifty years. With the advances of the internet and affordable technology maybe the prospect of making a movie isn't that out of reach for most people. In their recent release of the iPhone 12 Pro, Apple mentioned that the likes of *American Idol* and *Mythic Quest* have been partially filmed on an iPhone. Another example is Steven Soderbergh's *Unsane*, which was filmed entirely on an iPhone 7 Plus. The fact that nearly anyone with a smart phone and an idea can make a movie is revolutionary. It does however beg the question - are more low quality movies produced because of this? It might be somewhat of a far fetched idea, but it's food for thought.

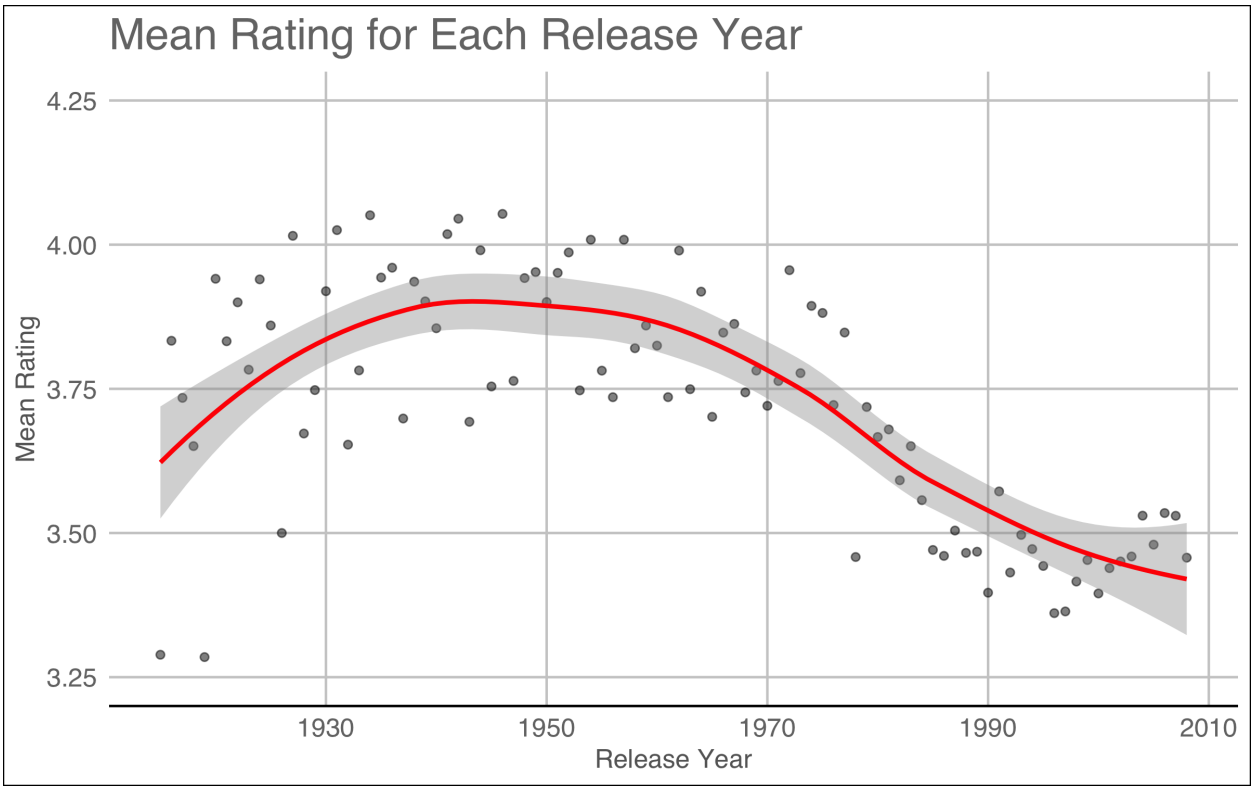


Figure 2: Scatterplot of mean movie rating for each release year. Note - the grey shaded area does not account for the size or spread of the sample from which the mean ratings were calculated.

Rating Year

Figure 3 is perhaps slightly less interesting. It does however suggest that people's attitude towards rating films has not changed much over the years.

What about the point sitting up at the top left with a rating of four? Well, only one user in the `edx` data set was active in 1995, and they only rated two movies. It is probably safe to class this as an outlier.

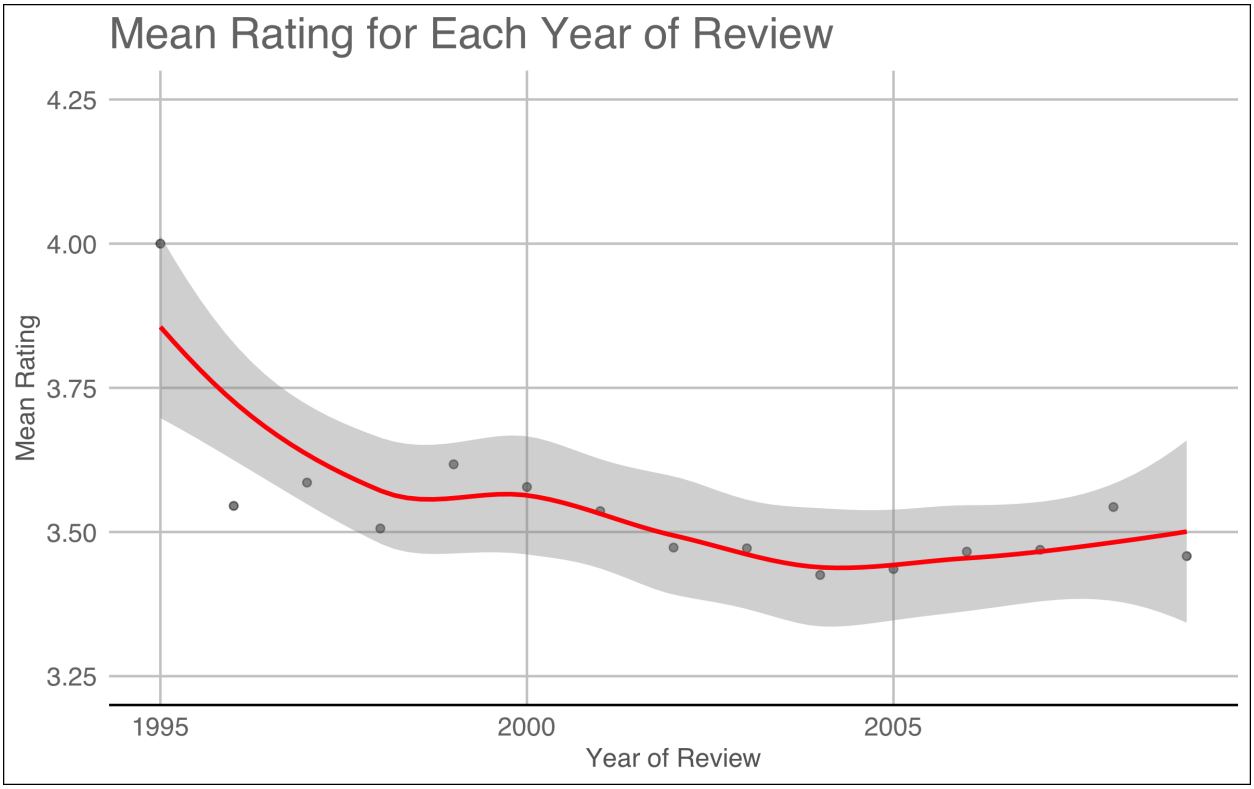


Figure 3: Scatterplot of mean movie rating for each year of review. Note - the grey shaded area does not account for the size or spread of the sample from which the mean rating was calculated.

Rating Month

Speaking of uninteresting plots...

If you ever had your suspicions that people rated movies more generously around Christmas time or during the Summer, Figure 4 is for you.

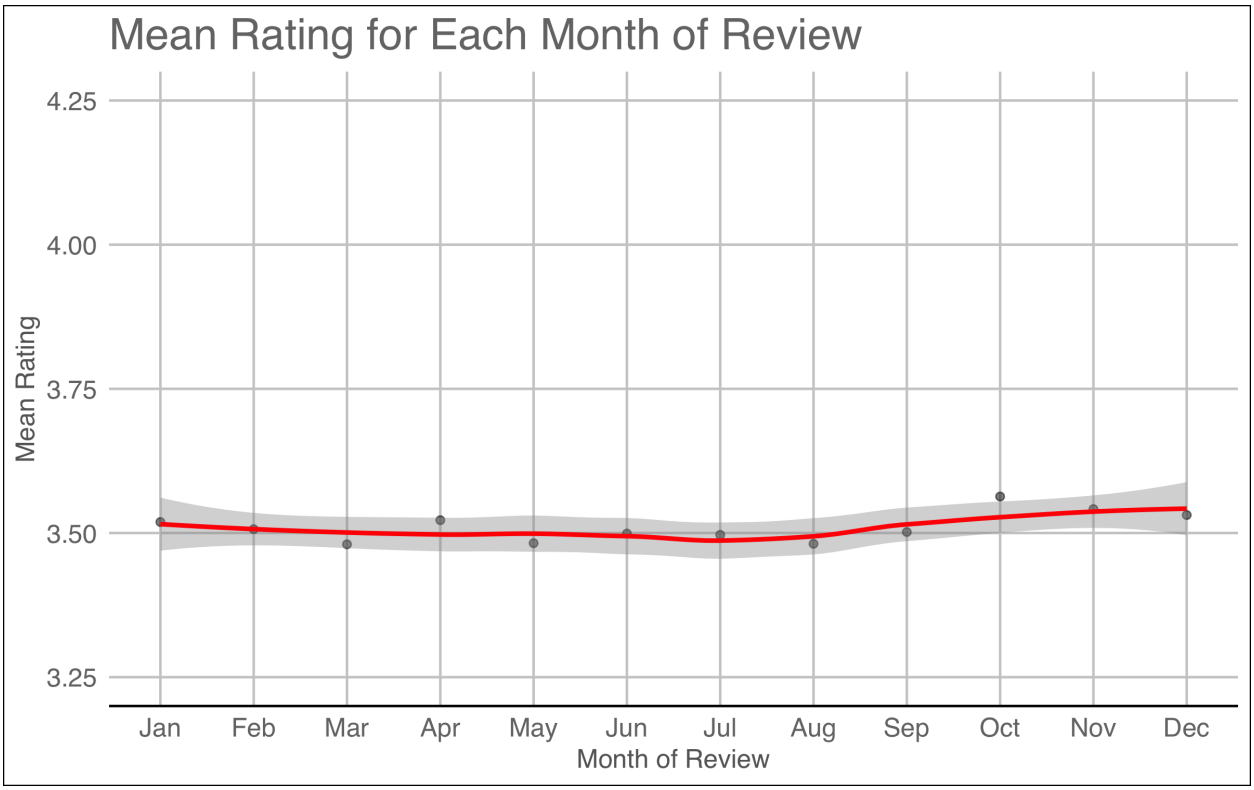


Figure 4: Scatterplot of mean movie rating for each month of review. Note - the grey shaded area does not account for the size or spread of the sample from which the mean rating was calculated.

Genre

Figure 5 illustrates the impact genre has on a movie's rating. The impact isn't actually too significant. The range between Sci-Fi and Drama is only about 0.25. The main takeaways from Figure 5 are probably that there weren't many IMAX, documentary or film-noir movies rated, and horror movies are generally pretty bad. Maybe that last point is unfair. After all, if a horror movie isn't scary then it's deemed to be poor. If it does its job and leaves the user terrified, would they be in a rush to give it a five star review?

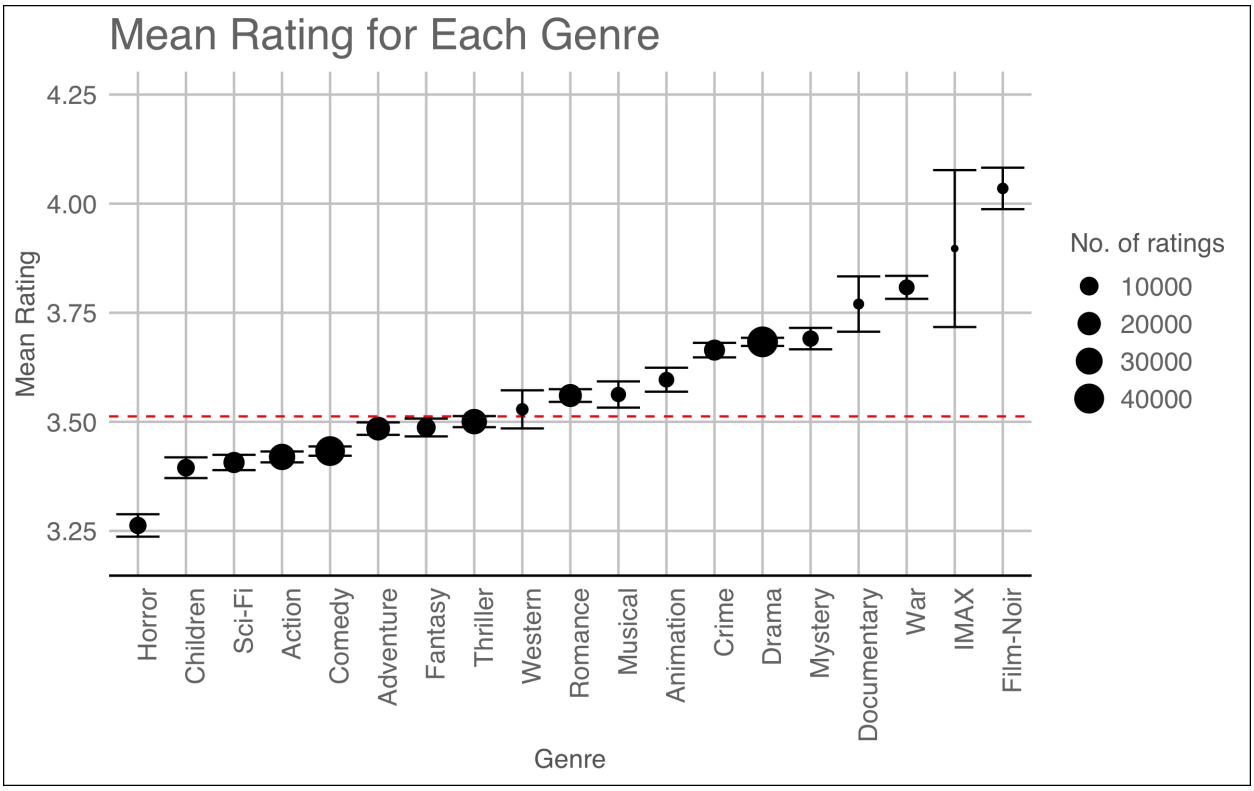


Figure 5: Mean rating for each genre (random sample of 100,000 observations).

Approach 1: Simple Bias

This section of the report constructs models using `edx_train` which will predict the movie ratings in `edx_test`. Before any models are constructed, the way in which their performance is measured needs to be considered. A common loss term used to asses a model's performance is RMSE (also used in the Netflix competition). RMSE is used to assess all of the models in this report.

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(\hat{y}_i - y_i)^2}{N}} \tag{1}$$

where N is the sample size, $\hat{y}_i$ are the predicted values and y_i are the corresponding observations.

Average Movie Rating

The first, and most simple, algorithm used to guess a user's rating for a given film would be to guess the average rating for all movies. The average movie rating across `edx_train` is 3.51. A model which uses only this information for prediction is as follows:

$$r_{u,m} = \mu + \epsilon_{u,m} \tag{2}$$

where $r_{u,m}$ is the rating given by user u for movie m, μ is the average movie rating (3.51) and $\epsilon_{u,m}$ is the error term for the corresponding movie and user. The table below shows that the RMSE of this algorithm is around 1.0601. The next section will try and improve on this.

Table 1: RMSE result after first model.

Method	**RMSE**
Just the Average	1.060054

Movie Bias

An obvious factor to consider is that some movies are better (rated more highly) than others. This is highlighted in Figure 6 below. Instead of using only the average movie rating for predictions, it may be more intuitive to use each movie's average rating. The model is then

$$r_{u,m} = \mu + \beta_m + \epsilon_{u,m} \tag{3}$$

where β_m is the bias for movie m.Note that accounting for movie bias now means that there is no interest in investigating release year, as the value of information given by release year is captured in calculating the bias for each movie.

Table 2 below shows the improved RMSE of 0.943 when accounting for movie to movie variability.

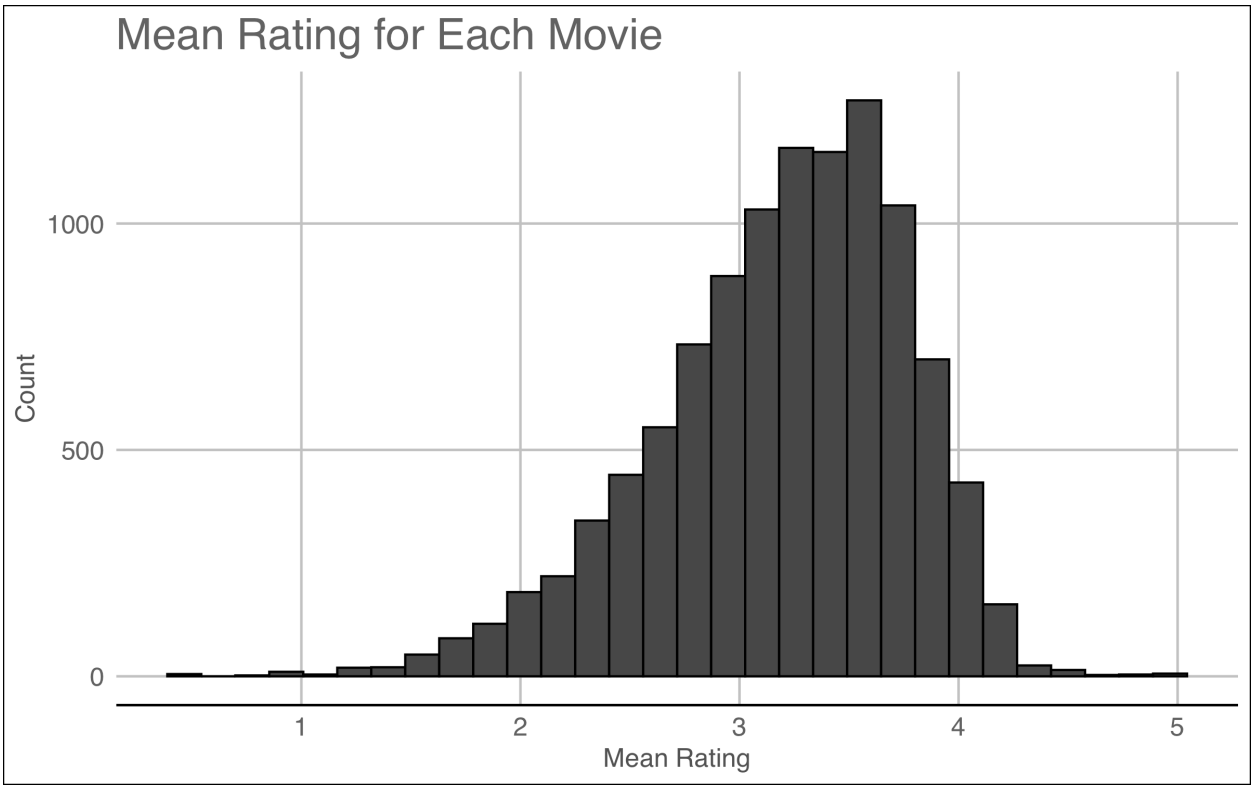


Figure 6: Histogram of mean ratings for each movie

Table 2: RMSE results after second model.

Method	**RMSE**
Just the Average	1.0600537
Movie Bias	0.9429615

Movie and User Bias

Not only do different movies get rated differently, but different users have different rating tendencies. Some users are generous whereas others are harsh. This is highlighted in Figure 7.

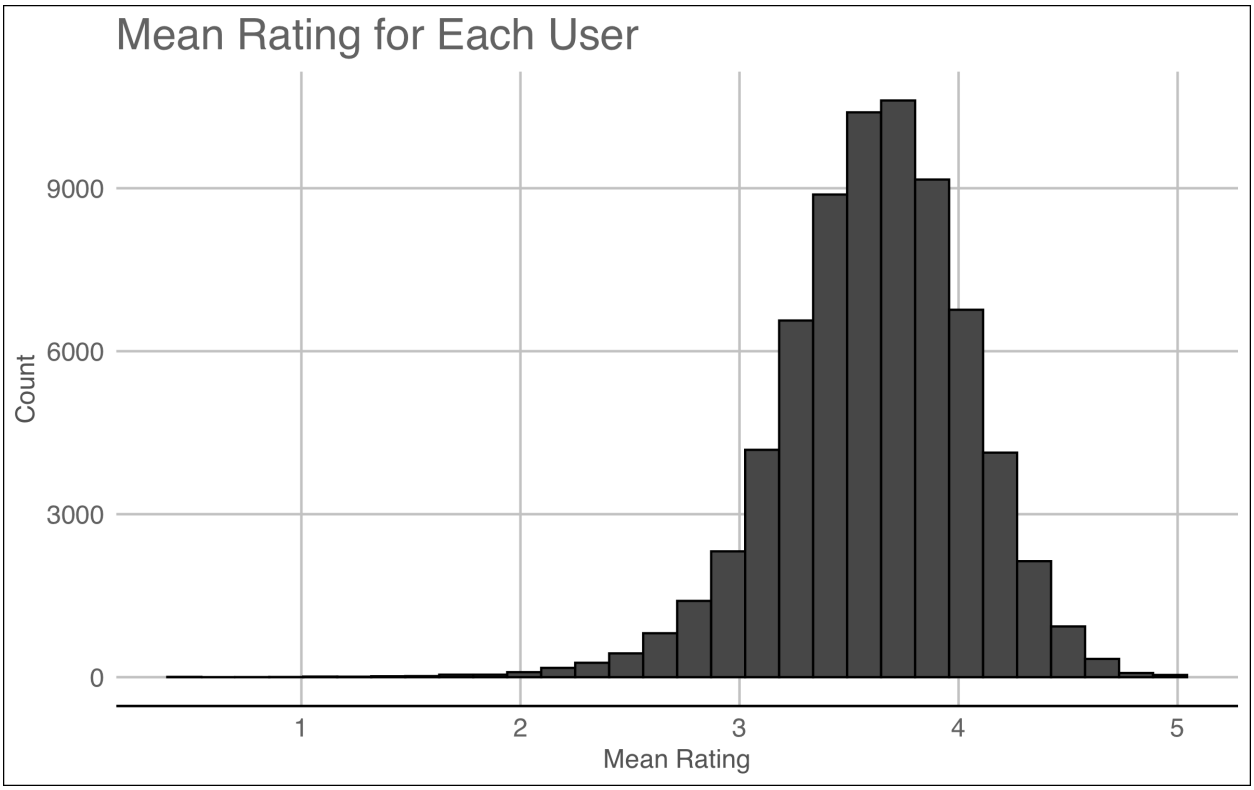


Figure 7: Histogram of mean ratings for each user.

In the same way that the previous model accounts for movie to movie variability, this model accounts for user to user variability. Thus, the model is

$$r_{u,m} = \mu + \beta_m + \beta_u + \epsilon_{u,m} \tag{4}$$

where β_u is the bias for user u.

One small issue with this model is that it can actually predict values outwith the interval $[0.5, 5]$. Therefore, any predictions above/below the interval are assigned a value of 5/0.5 respectively.

This model reduces the RMSE to 0.8645.

Table 3: RMSE results after third model.

Method	**RMSE**
Just the Average	1.0600537
Movie Bias	0.9429615
Movie + User Bias	0.8644818

Approach 2: Regularised Bias

To motivate the idea behind regularised bias (regularised regression) the five best and worst movies - according to the Movie Bias model - are listed in Table 4 and Table 5 below respectively. They're not very convincing lists. What these movies have in common is that they are quite obscure - they have been rated very few times. One way to think about this idea is: are you more likely to buy a product which one person has rated five stars, or a product which over a thousand people have rated four and a half stars? The number of reviews gives credibility to the score.

Table 4: Top five movies according to "Movie Bias" model.

Title	β_m	**Number of ratings**
Hellhounds on My Trail (1999)	1.487544	1
Satan's Tango (Sátántangó) (1994)	1.487544	2
Shadows of Forgotten Ancestors (1964)	1.487544	1
Fighting Elegy (Kenka erejii) (1966)	1.487544	1
Sun Alley (Sonnenallee) (1999)	1.487544	1

Table 5: Worst five movies according to "Movie Bias" model.

Title	β_m	**Number of ratings**
Besotted (2001)	-3.012456	2
Hi-Line, The (1999)	-3.012456	1
Accused (Anklaget) (2005)	-3.012456	1
Confessions of a Superhero (2007)	-3.012456	1
War of the Worlds 2: The Next Wave (2008)	-3.012456	2

Regularised Movie Bias

Resularised regression is a machine learning algorithm which penalises parameter estimates which come from small sample sizes and are deemed to be somewhat unreliable. In the Movie Bias model the following equation is minimised to obtain the least squares estimate (LSE) for each movie bias:

$$\frac{1}{N}\sum_{u,m}(r_{u,m}-\mu-\beta_m)^2. \tag{5}$$

To estimate the parameters for the new model using regularised regression, the following equation is minimised:

$$\frac{1}{N}\sum_{u,m}(r_{u,m}-\mu-\beta_m)^2+\lambda\sum_{m}\beta_m^2 \tag{6}$$

where λ is a tuning parameter which determines how much 'unreliable' parameter estimates are penalised.

With the use of calculus, it is easy to show that the values of β that minimise this equation are defined as

$$\hat{\beta}_m(\lambda)=\frac{1}{\lambda+n_m}\sum_{u=1}^{n_m}(R_{u,m}-\hat{\mu}) \tag{7}$$

where n_m is the number of ratings for movie m.

Note that when n_m is small (not many people have rated the movie), λ is having a more significant impact. When n_m is large (many people have rated the movie), the effect λ has becomes negligible. The equation also illustrates that a large value of λ results in more shrinking (more skepticism).

This model is similar to the Movie Bias model however regularised regression is used to construct the movie bias parameter estimates. 5-fold cross-validation, on `edx_test`, is used to choose an optimal λ. Figure 8 illustrates the mean RMSE obtained with each λ with 2.4 being the optimal choice.

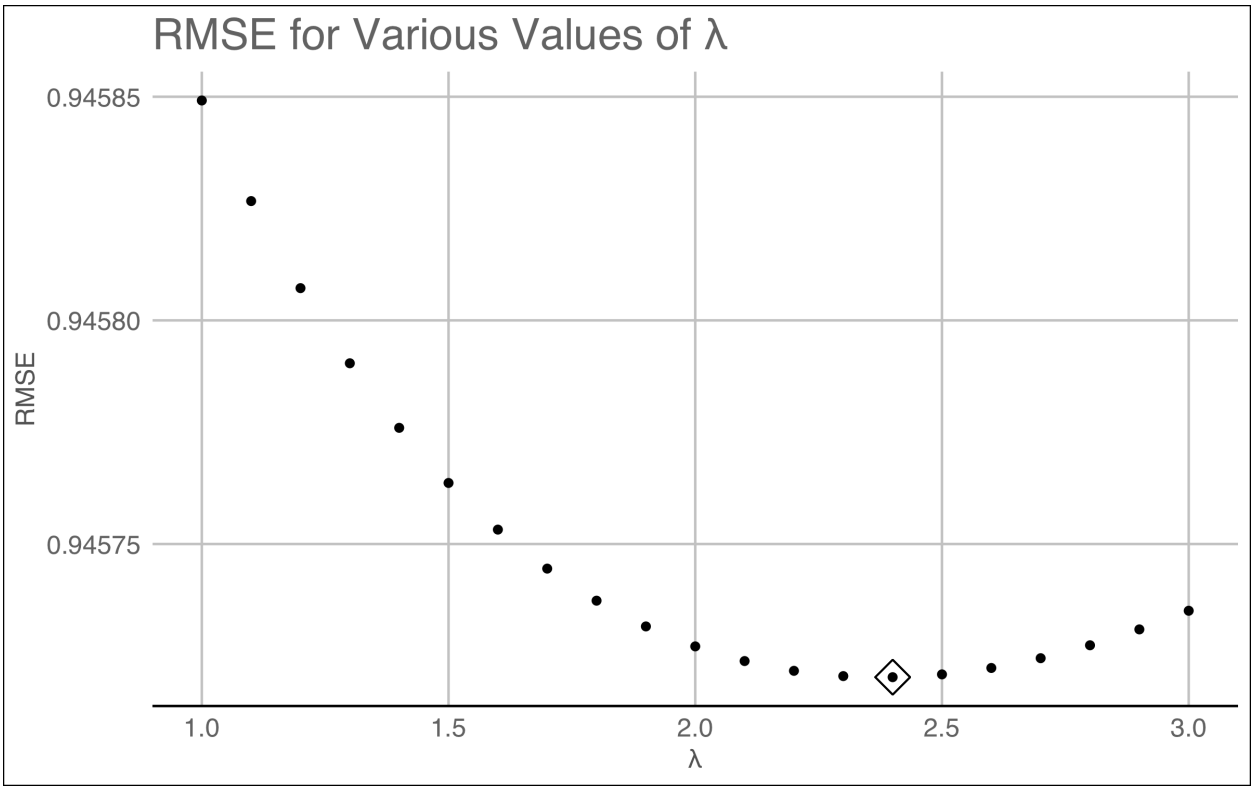


Figure 8: Scatterplot of RMSE for each λ (5-fold cross-validation). The optimal value of $\lambda = 2.3$ is highlighted.

The five best and worst movies when accounting for regularisation are shown in Table 6 and Table 7 respectively. These results are more in line with expectations. In addition to this, the number of ratings for each movie is much higher in comparison to Table 4 and Table 5.

Table 6: Top five movies according to "Regularised Movie Bias" model.

Title	β_m	**Number of ratings**
Shawshank Redemption, The (1994)	0.9440210	28015
Godfather, The (1972)	0.9040596	17747
Usual Suspects, The (1995)	0.8539909	21648
Schindler's List (1993)	0.8515313	23193
More (1998)	0.8244364	7

Table 7: Worst five movies according to "Regularised Movie Bias" model.

Title	β_m	**Number of ratings**
SuperBabies: Baby Geniuses 2 (2004)	-2.633308	56
From Justin to Kelly (2003)	-2.603988	199
Disaster Movie (2008)	-2.542397	32
Pokémon Heroes (2003)	-2.425194	137
Barney's Great Adventure (1998)	-2.342446	208

Table 8 shows that the new model is only a slight improvement over the "Movie Bias" model. The reason for this is that only a few predictions really benefit from using regularised regression. The bias estimates for movies such as *The Shawshank Redemption* and *The Godfather* don't really differ from the first model because they have a large number of ratings. The bias estimates for movies like *Hellhounds on My Trail* and *Satan's Tango* (see Table 4) are significantly altered, however these movies account for such a small proportion of the ratings.

Table 8: RMSE results after fourth model.

Method	**RMSE**
Just the Average	1.0600537
Movie Bias	0.9429615
Movie + User Bias	0.8644818
Regularised Movie Bias	0.9429398

Regularised Movie and User Bias

The idea of regularised regression can be applied to users as well as movies. When accounting for two regularised parameters there are a couple of routes to take. The following equation can be minimised:

$$\frac{1}{N}\sum_{u,m}(r_{u,m}-\mu-\beta_m-\beta_u)^2+\lambda\left(\sum_m\beta_m^2+\sum_u\beta_u^2\right) \tag{8}$$

which would mean finding an optimal λ which simultaneously regularises both parameters. Another route is to fix the previously calculated value of λ for the movie bias and find another λ which regularises the user bias. Since it offers more flexibility, this model uses the latter approach. Thus, the equation to be minimised is

$$\frac{1}{N}\sum_{u,m}(r_{u,m}-\mu-\beta_m-\beta_u)^2+\lambda\sum_u\beta_u^2 \tag{9}$$

where β_m are the **regularised** parameter estimates.

Like before, the parameter estimates are defined by the following equation:

$$\hat{\beta}_u(\lambda)=\frac{1}{\lambda+n_u}\sum_{m=1}^{n_u}(R_{u,m}-\hat{\mu}-\hat{\beta}_m) \tag{10}$$

where n_u is the number of ratings provided by user u.

Figure 9 illustrates the mean RMSE obtained with each λ with 5.1 being the optimal choice. Again, this model is only a slight improvement over the Movie and User Bias model. The RMSE is reduced to 0.864.

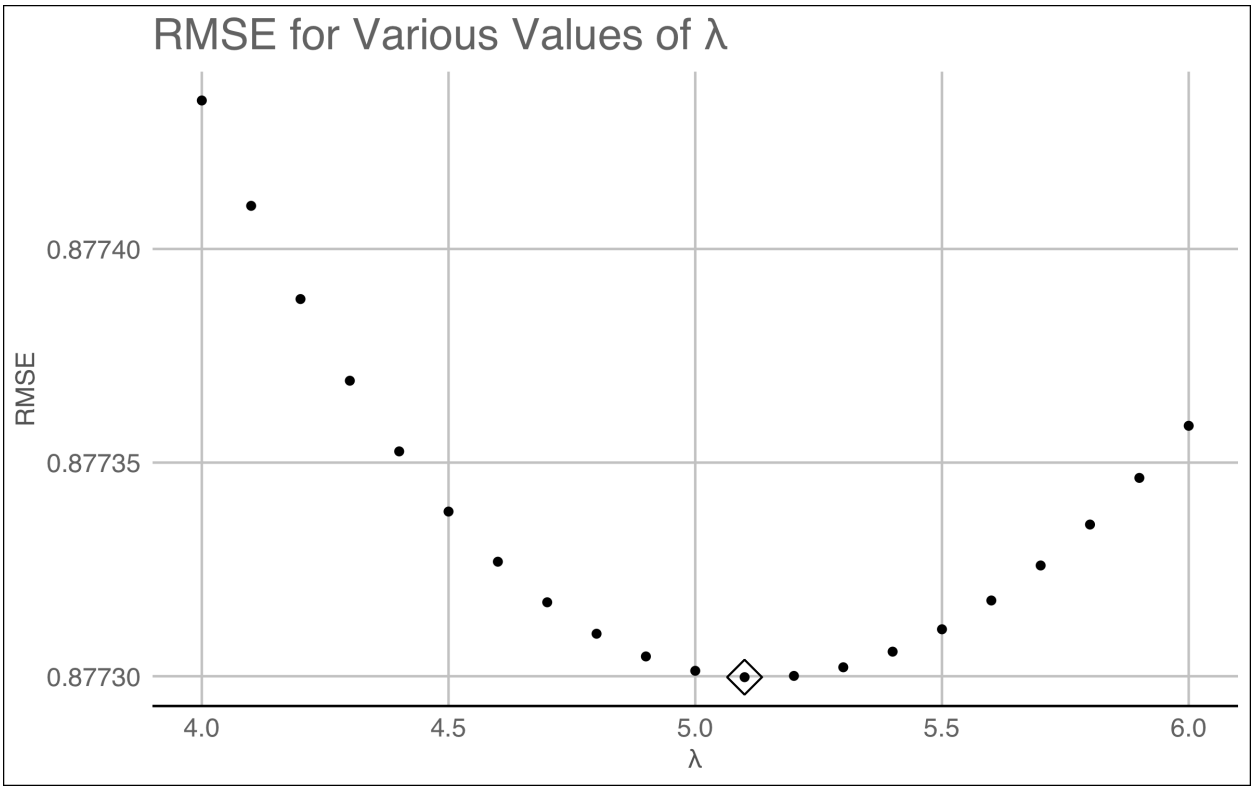


Figure 9: Scatterplot of RMSE for each λ (5-fold cross-validation). The optimal value of $\lambda = 5.1$ is highlighted.

Table 9: RMSE results after fifth model.

Method	**RMSE**
Just the Average	1.0600537
Movie Bias	0.9429615
Movie + User Bias	0.8644818
Regularised Movie Bias	0.9429398
Regularised Movie + User Bias	0.8640398

Approach 3: Collaborative Filtering

Collaborative filtering (CF) is a popular machine learning method used in recommendation systems. It first helps to image that ratings are stored in a matrix (a row for every user and a column for every movie). As not every user rates every movie, this matrix is going to be extremely sparse. The goal of CF is to fill in these missing values.

CF is commonly split into two groups: **user-based CF** and **item-based CF**. The motivation behind UBCF is that some users are similar to others, therefore are likely to have similar rating tendencies. For example, if you and your sibling like the same kind of films as each other, and your sibling gives a five star rating to a movie that you haven't seen, it's likely that you will enjoy the movie. The motivation behind IBCF is that some movies are similar to others. For example, if you gave *Harry Potter and the Philosopher's Stone* a one star rating it's probably true that you wouldn't enjoy *Harry Potter and the Chamber of Secrets*, because those two movies are arguably quite similar.

The following two models use UBCF and IBCF respectively. The third model is an ensemble of the two CF models.

User-Based Collaborative Filtering

The first stage in constructing a user-based CF model is to determine what it means for users to be similar to each other. A popular measure, used in this report, is the cosine similarity. By taking R to be a user-movie rating matrix, the cosine similarity between two users A and B is defined as

$$sim(A, B) = \frac{\mathbf{u}_A \cdot \mathbf{u}_B}{\|\mathbf{u}_A\|\|\mathbf{u}_B\|} \tag{11}$$

where $\mathbf{u}_U$ is the row vector in R for user U. This measure is calculated using only the movies that both users rated. Once the similarities have been determined, the top k users similar to each individual user are used to calculate their predicted ratings. The prediction for the rating given by user A for movie m is given by

$$\hat{r}_{A,m} = \frac{\sum_{U \in S} sim(A, U) r_{U,m}}{\sum_{U \in S} |sim(A, U)|} \tag{12}$$

Where S is the set of k users most similar to user A. This is a weighted mean of the ratings given to movie m by all of the users similar to user A - the ratings of users very similar to user A are given more weight than the ratings given by users who are not as similar.

The idea behind UBCF is relatively simple, however problems arise when trying to implement it on large data sets. A common R package for recommender systems is **Recommendarlab** [4]. Although this package is very user-friendly, is does not scale well at all. In most cases it's actually impossible to use Recommenderlab with a data set as large as `edx_train` because it runs out of memory.

In Stefan Nikolić's blog [5] he presents a solution to tackle this issue. As well as using an optimized approach for calculating the similarities [6], one of the key features is that the matrix is split up into 'chunks'. Figure 10 below is a screenshot from Nikolić's blog. It illustrates how the initial matrix, `edx_train` in the case of this report, is split up into chunks in preparation for similarities being calculated. This step is then iterated until the entire matrix has been accounted for.

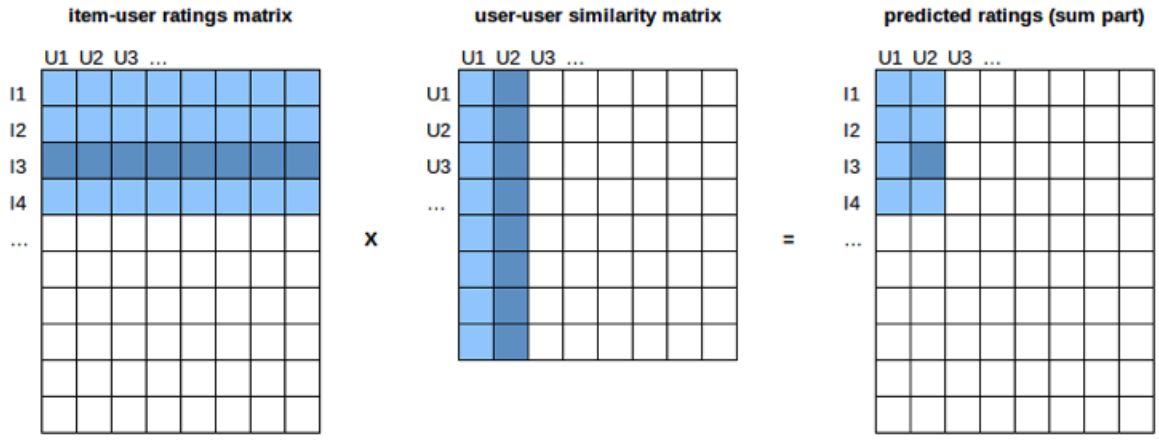


Figure 10: Nikolić's illustration of splitting the matrix into chunks.

This method is used to construct the sixth model in this report: User-Based Collaborative Filtering. The chosen parameters are highlighted below in Table 10. To further improve this model, these parameters could be optimised using CV. However, due to the time it takes to run the models this report only uses one set of parameters. At a later date (and with a more powerful computer) this report will be updated with optimised tuning parameters. Nikolić actually includes a script `test.R` in his GitHub repo which uses CV to assess model performance (using RMSE) which is convenient for optimising tuning parameters.

Table 10: Parameters used for UBCF.

Parameter	**Value**
k	100
Row chunk size	15000
Column chunk size	1000

There is one small problem which needs addressing before the performance of the model can be measured. Due to the nature of the algorithm, some similarities cannot be measured. This is due to the matrix being so sparse. Therefore, some of the predictions will be missing values. A quick and easy fix for this is to replace these missing values with predictions from the Regularised Movie and User Bias model. Thus, the predictions obtained from the UBCF model are defined as

$$\hat{r}_{u,m} = \begin{cases} \frac{\sum_{U \in S} sim(u,U) r_{U,m}}{\sum_{U \in S} |sim(u,U)|} & \text{if } S \neq \emptyset \\ \mu + \beta_m + \beta_u & \text{otherwise} \end{cases}$$

where S is the set of users that have rated movie m out of the k users most similar to user u, $sim(u,U)$ is the cosine similarity between user u and user U, and β_m and β_u are the **regularised** movie and user biases respectively, calculated in Regularised Movie and User Bias.

IMPORTANT NOTE

Before the similarities are calculated, the overall mean and regularised row and column means are subtracted from the matrix. This means that the UBCF model is dealing with a residual matrix, taking "Regularised Movie and User Bias" to be the base model. Once the similarities and residual predictions are calculated, μ, b_m and b_u are then added to the residual predictions to achieve the actual predictions. This takes place in all CF models in this report, but it is not highlighted in the formulae as is begins to look messy.

This model appears to be a deterioration on the previous model, resulting in an RMSE of 0.8802. The following section takes a similar approach, only this time considering similarities between movies.

Table 11: RMSE results after sixth model.

Method	**RMSE**
Just the Average	1.0600537
Movie Bias	0.9429615
Movie + User Bias	0.8644818
Regularised Movie Bias	0.9429398
Regularised Movie + User Bias	0.8640398
User-Based Collaborative Filtering	0.8802492

Item-Based Collaborative Filtering

This model is constructed in almost exactly the same way as the previous model. The underlying motivations and mathematics are the same, only this model uses similarities between movies and not users. It might be easiest to imagine the previous model construction but with a movie-user matrix instead of a user-movie matrix. The parameters used for this model are highlighted below in Table 12. Again, these should ideally be optimised using cross validation.

The RMSE is much better for this model, yielding a result of 0.8187.

Table 12: Parameters used for IBCF.

Parameter	**Value**
k	100
Row chunk size	18000
Column chunk size	5500

Table 13: RMSE results after seventh model.

Method	**RMSE**
Just the Average	1.0600537
Movie Bias	0.9429615
Movie + User Bias	0.8644818
Regularised Movie Bias	0.9429398
Regularised Movie + User Bias	0.8640398
User-Based Collaborative Filtering	0.8802492
Item-Based Collaborative Filtering	0.8186541

User-Item-Based Collaborative Filtering (Ensemble)

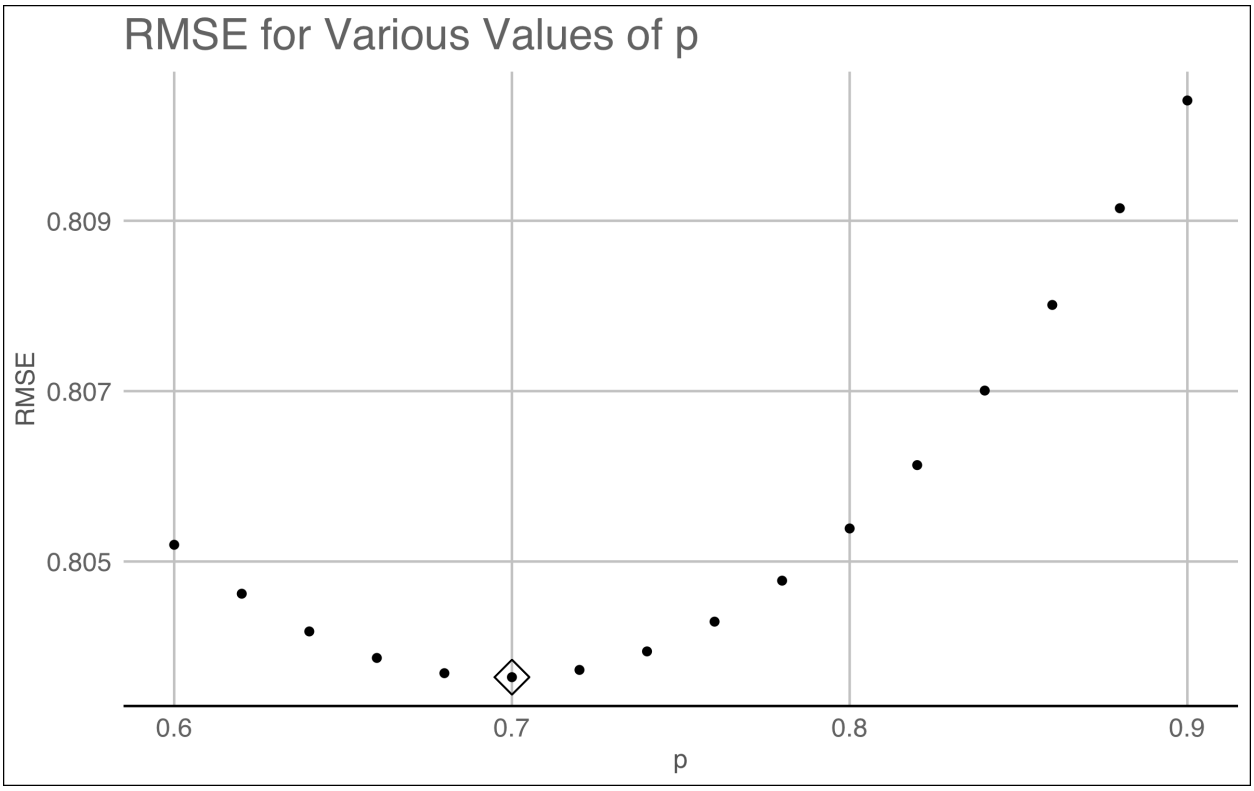
This model further develops the idea of CF by ensembling the two models previously constructed. A naive approach would be to take the mean prediction of each model, however Table 13 indicates that the IBCF model is far superior. For this reason, it is of interest to consider a **weighted mean**. The model is then defined as

$$\hat{r}_{u,m} = (1-p)\hat{r}_{u,m}^{UBCF} + p\hat{r}_{u,m}^{IBCF} \tag{13}$$

where $\hat{r}_{u,i}^{UBCF}$ and $\hat{r}_{u,i}^{IBCF}$ are predictions from the UBCF and IBCF models respectively and $0 \leq p \leq 1$ is the weight given to the IBCF model. Figure 11 shows the RMSE with various values of p when assessing the model on `edx_test`, indicating that $p = 0.7$ is the optimal choice. Therefore, this model is defined as

$$\hat{r}_{u,m} = 0.3\hat{r}_{u,m}^{UBCF} + 0.7\hat{r}_{u,m}^{IBCF} \tag{14}$$

Table 14 shows a significantly improved RMSE of 0.8036.



Note that it would be more appropriate to perform CV on `edx_train` to select the optimal p, however due to the time it takes to construct the CF models various values of p are simply tested against `edx_test`. Despite not using CV in this instance, the final hold out test set, `validation`, still hasn't been used so the final assessment is still fair.

Table 14: RMSE results after eighth model.

Method	**RMSE**
Just the Average	1.0600537
Movie Bias	0.9429615
Movie + User Bias	0.8644818
Regularised Movie Bias	0.9429398
Regularised Movie + User Bias	0.8640398
User-Based Collaborative Filtering	0.8802492
Item-Based Collaborative Filtering	0.8186541
User-Item-Based Collaborative Filtering (Ensemble)	0.8036421

Final Model (Results)

The previous sections construct models which are assessed using `edx_test` which is a subset of `edx`. The best performing model is User-Item-Based Collaborative Filtering (Ensemble) which yields an RMSE of 0.8036. Therefore, the final model takes the same approach. The final model is constructed using the entire `edx` data set and is then evaluated using `validation`. Note that `validation` is not used at any point in this report so it is a fair assessment.

The same parameters as highlighted in Table 10 and Table 12 are used in this final model. In his blog, Nikolić highlights the parameters used in his model (on a very similar data set) which are in line with those used in this report.

Though it is quite a mouthful, the model is as follows:

$$\hat{r}_{u,m} = 0.3\hat{r}_{u,m}^{UBCF} + 0.7\hat{r}_{u,m}^{IBCF} \tag{15}$$

where

$$\hat{r}_{u,m}^{UBCF} = \begin{cases} \frac{\sum_{U \in S} sim(u,U) r_{U,m}}{\sum_{U \in S} |sim(u,U)|} & \text{if } S \neq \emptyset \\ \mu + \beta_m + \beta_u & \text{otherwise,} \end{cases}$$

$$\hat{r}_{u,m}^{IBCF} = \begin{cases} \frac{\sum_{M \in T} sim(m,M) r_{u,M}}{\sum_{M \in T} |sim(m,M)|} & \text{if } S \neq \emptyset \\ \mu + \beta_m + \beta_u & \text{otherwise} \end{cases}$$

where

- S is the set of users that have rated movie m out of the 100 users most similar to user u,
- T is the set of movies that have been rated by user u out of the 100 movies most similar to movie m,
- $sim(u, U)$ is the cosine similarity between user u and user U,
- $sim(m, M)$ is the cosine similarity between movie m and movie M,
- β_m and β_u are the **regularised** movie and user biases respectively.

Note that all of these values are calculated using `edx` and not `edx_train`. This means that, for example, the values for β_m are slightly different in the final model compared to Regularised Movie and User Bias.

As shown in Table 15, this final model achieves an RMSE of 0.7968.

Table 15: Table 15: RMSE of final model.

Method	**RMSE**
User-Item-Based Collaborative Filtering (Ensemble)	0.7968132

Conclusion

This report discusses a few methods used to construct recommendation systems. The best performing model is User-Item-Based Collaborative Filtering (Ensemble) which yields an RMSE of 0.7968 when trained on `edx` and tested on `validation`. This is an improvement on the first model's RMSE by around 25%.

Although Nikolić's scalable approach to constructing CF models is convenient, it would be much more preferable to calculate the similarity measures using the entire matrix instead of splitting it into chunks. This is the main limitation of the final model. A further limitation is that, in this report, the tuning parameters are not optimised. The chunk sizes are more dependent on computing power, however k could easily be optimised using cross validation with enough computing power.

Those interested in a more advanced approach to construct recommender systems should consult the recosystem package. It provides the means to implement matrix factorisation on large data sets. It is a scalable alternative to the likes of recommenderlab. The high level idea of matrix factorisation is that different movies have different features, and those features appeal to some users and not to others. These features are 'latent factors', they don't have predetermined definitions. Luis Serrano explains the concept well in this video. Matrix factorisation using recosystem improves on this report's best performing model's RMSE by just under 2%.

Many thanks are due to Rafael Irizarry, the course instructor of HarvardX's Professional Certificate in Data Science, and to the teaching staff who were always at hand to answer questions and queries raised by students. This edX series has been thoroughly enjoyable and valuable. Irizarry delivered engaging lectures and provided a range of useful coding examples throughout the series.

Another thank you is due to Stefan Nikolić and the team he worked with to create the scripts used in Approach 3: Collaborative Filtering. This report gives all credit to Nikolić for being able to implement CF on such a large data set.

References

[1] Lohr, S. *Netflix Awards $1 Million Prize and Starts a New Contest.* https://bits.blogs.nytimes.com/2009/09/21/netflix-awards-1-million-prize-and-starts-a-new-contest/ (date last accessed - 15/10/2020)

[2] Chen, E. *Winning the Netflix Prize: A Summary.* http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/ (date last accessed - 15/10/2020)

[3] MovieLens *MovieLens 10M Dataset* https://grouplens.org/datasets/movielens/10m/ (date last accessed - 15/10/2020)

[4] Hahsler, M. *recommenderlab: A Framework for Developing and Testing Recommendation Algorithms. R package version 0.2-6.* (2020)

[5] Nikolić, S. *Improved R implementation of collaborative filtering* https://blog.smartcat.io/2017/improved-r-implementation-of-collaborative-filtering/ (date last accessed - 15/10/2020)

[6] Nikolić, S. *Recommender Systems: Matrix operations for fast calculation of similarities* https://blog.smartcat.io/2016/recommender-systems-matrix-operations-for-fast-calculation-of-similarities/ (date last accessed - 15/10/2020)