MovieLens Capstone

Verne Cazaubon

2023-12-03

Introduction

In October 2006, Netflix launched a challenge to create a recommendation system that beat Cinematch (Netflix's in-house recommendation system) by at least ten (10) percent. That is, the new system had to better predict the movies that customers would like. The grand prize was one million United States Dollars!

So what is a recommendation system? And are there any practical examples of its uses?

Recommendation systems use ratings that users give an item on buying and/or using them to make specific recommendations. Companies like Amazon collect massive data sets of user ratings on the products sold to them. The data sets are subsequently used to predict high rated items for a given user and then recommended them to the user.

Similarly, Netflix uses movie ratings provided by users to predict a high rated movie for a given user and then recommends it to the user. Usually the movie ratings are on a 1-5 scale, where 5 represents an excellent movie and 1 suggests it to be a poor one. For this project, the ratings are from 0.5 to 5.

Description of dataset

The MovieLens data set included in the course material is a subset of a larger data set. The stripped down version contains approximately ten (10) million data records. Each record is comprised of six (6) variables: userId, movieId, rating, timestamp, title, and genres.

Summary of goal of project

The aim of this project is to create a movie recommendation system using concepts taught throughout the previous eight (8) courses of edX's HarvardX Data Science Professional Certificate program. This recommendation system utilizes a supervised machine learning algorithm to predict user ratings of movies given certain features.

Key steps performed

An outline of the key programming steps performed are as follows:

1. Install packages if necessary, and load the required libraries.
2. Download the files.
3. Combine the data into one file.
4. Visualize the data. Gain insight into variables.
5. Separate the data into a training set and a test set.
6. Analyse the data using different machine learning algorithms.
7. Evaluate the performance of these algorithms, and choose an algorithm as the final model.
8. Run the final_holdout_test on the chosen model.

Analysis

Processes and techniques used in data cleaning

The data were downloaded from the grouplens website. Two files were downloaded, one containing the ratings data and another containing the movie data. The "movieId" variable linked the two files. The two files were joined into one, based on this link variable. Then, the data were partitioned into an edx data set and a final_holdout_test data set. All unnecessary data frames were then deleted to preserve memory. There was very little data cleaning to be done.

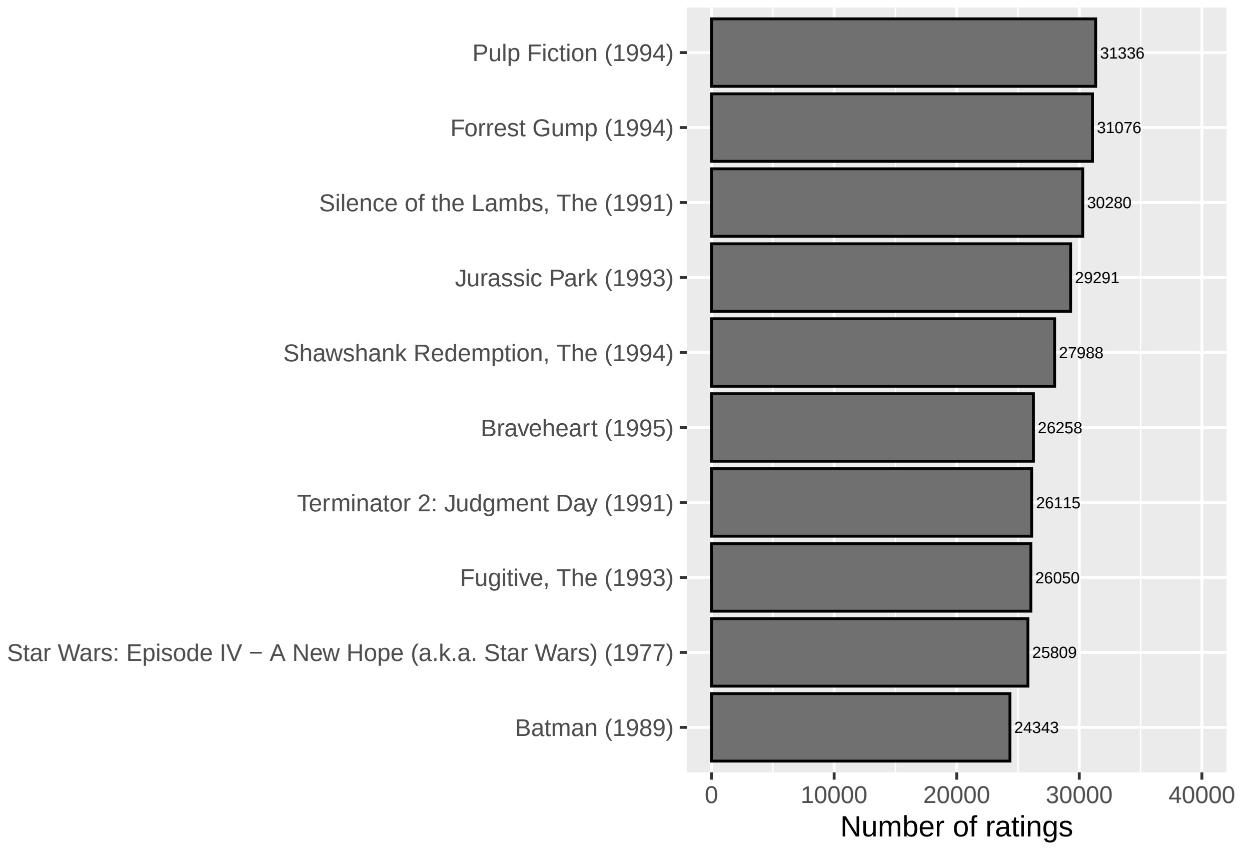
Processes and techniques used in data exploration and visualization

By examining the structure and head of the edx data set we note that it contains the six variables "userID", "movieID", "rating", "timestamp", "title", and "genres". Each row represent a single rating of a user for a single movie. The first six (6) records can be seen in the table below. There are no missing values (NAs) in the data set. It contains 9000061 records.

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	231	5	838983392	Dumb & Dumber (1994)	Comedy
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

Movies

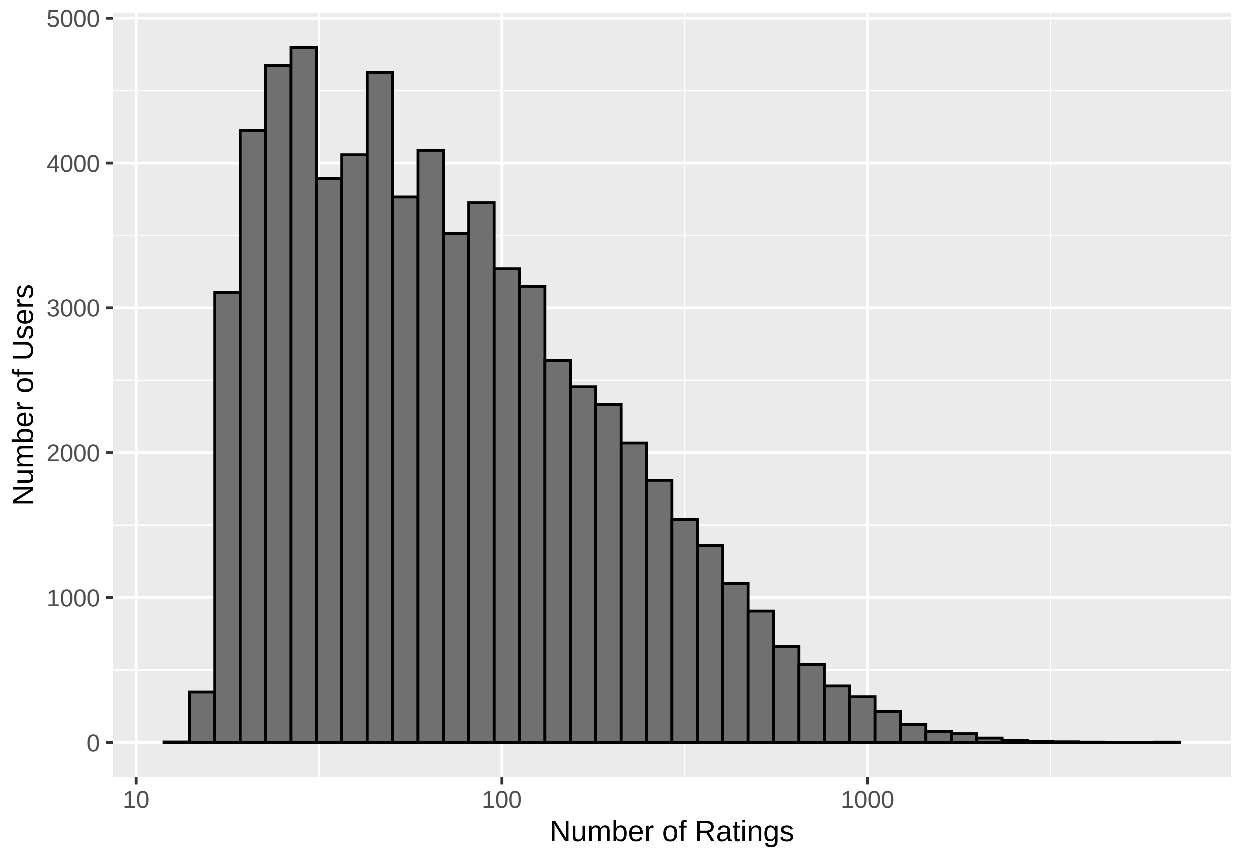
The top 10 titles in the data set are as follows:



There are 10677 distinct movies in the data set.

Users

The distribution of user ratings is as follows:



There are 69878 distinct users in the data set.

Genres

The genres sorted by count are as follows:

genres	count
Drama	3909401
Comedy	3541284
Action	2560649
Thriller	2325349
Adventure	1908692
Romance	1712232
Sci-Fi	1341750
Crime	1326917
Fantasy	925624
Children	737851
Horror	691407
Mystery	567865
War	511330
Animation	467220
Musical	432960
Western	189234
Film-Noir	118394
Documentary	93252

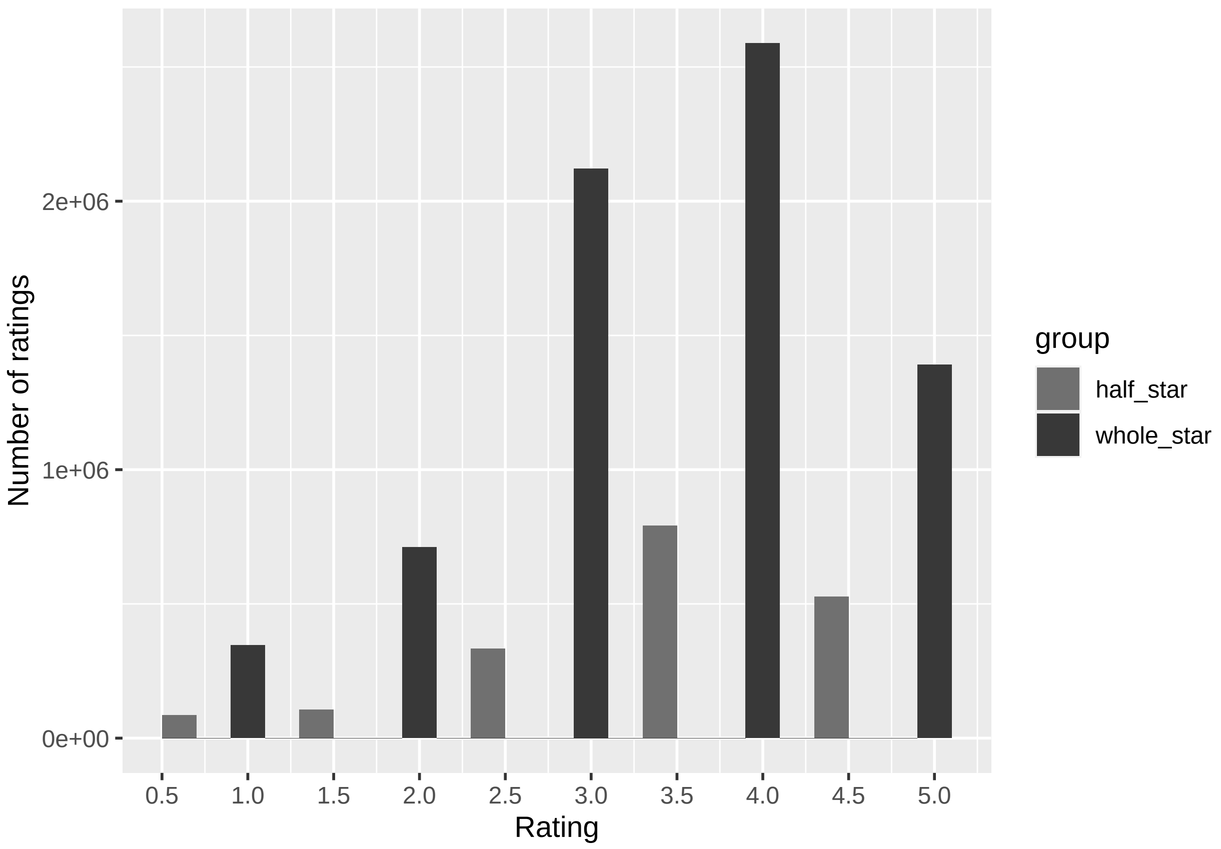
genres	count
IMAX	8190
(no genres listed)	6

Ratings

A 6-number summary of the ratings is:

rating
Min. :0.500
1st Qu.:3.000
Median :4.000
Mean :3.512
3rd Qu.:4.000
Max. :5.000

The distribution of ratings is as follows:



Insights gained

From the above visualizations it was observed that the most popular movies are older movies. The top 10 movies were made in years ranging from 1977 to 1995.

The distribution of number of user ratings is right-skewed. The majority of users have less than 100 ratings.

The most popular genres are drama, comedy, and action, whereas the least favorite are film-noir, documentary, and IMAX.

There are more whole star ratings than there are half star ratings. The most common rating is 4 stars.

Modeling approach

RMSE will be used as the loss function to compare the performance of the machine learning algorithms employed. The formula is RMSE $= \sqrt{\frac{\sum_{u,i}(\hat{y}_{u,i}-y_{u,i})^2}{N}}$ where $\hat{y}_{u,i}$ represents the prediction of the rating of movie i by user u. $y_{u,i}$ is the observed rating of movie i by user u. And N is the total number of user-movie combinations. The RMSE can be interpreted similar to standard deviation. If the RMSE $= 1$, as an example, this means that the average error between the predicted value is the observed value is one star. This project aims to get a RMSE < 0.86490.

The machine learning techniques utilized are regression models, and a recommender package that uses matrix factorization to address the problem. We now consider these more in depth.

Naive model

This very basic model predicts that every user will give every movie the average/mean rating. For our training set this mean is equal to 3.512354. It assumes that any deviation from this value is explained by random variation. This model is represented by the equation:

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

where μ is the true rating for all movies and users, and $\epsilon_{u,i}$ represents the independent errors per user per movie sampled from the same distribution that is centered at zero. The result of this model is

Model	RMSE
Naive - mean user rating	1.059

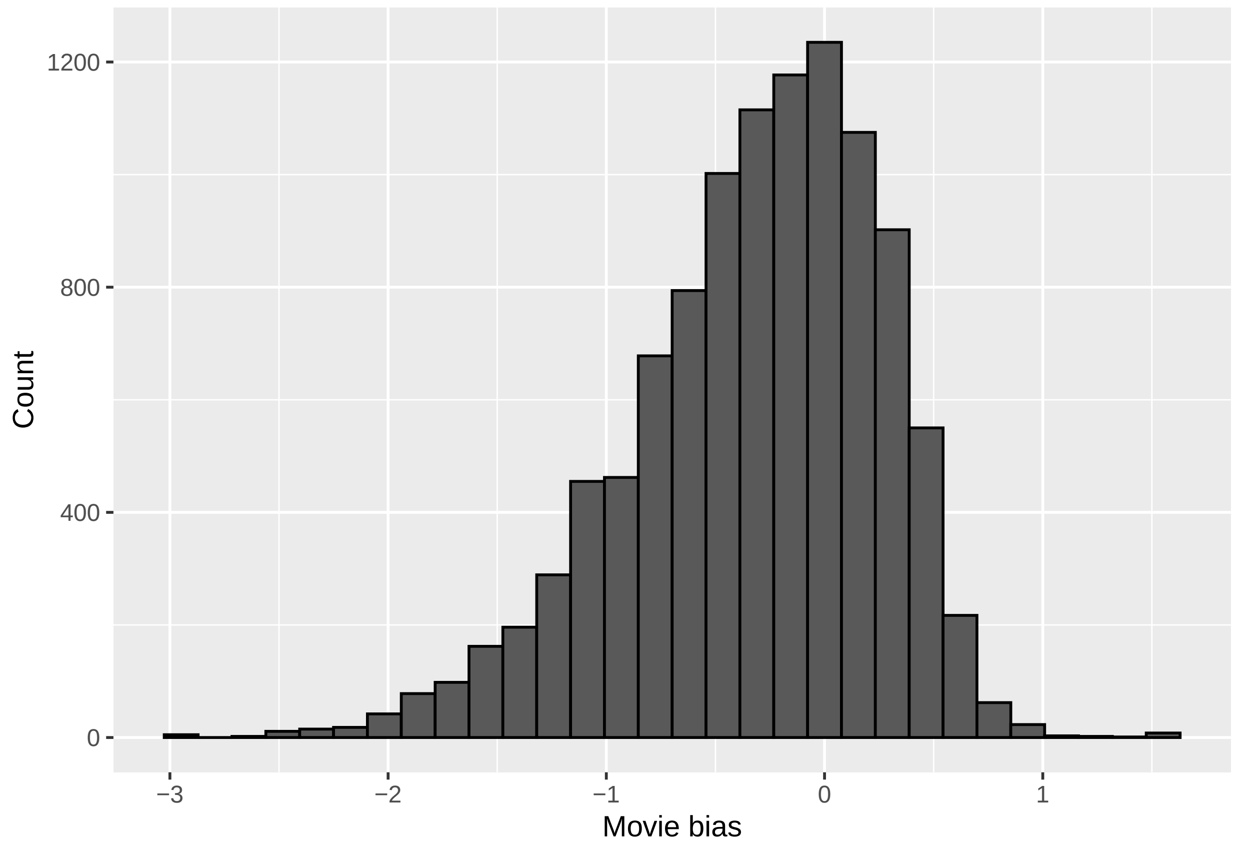
Movie effect

This model adds a new term, b_i, representing the average rating for movie i. This model is represented by the equation

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

Note that b_i is the average of $Y_{u,i} - \mu$ for each movie i.

We see below that the $\hat{b}_i$ form a left skewed normal distribution ranging from -3 to 1.5 with most values around zero. This is not unusual as $\hat{\mu}$ was 3.512354 and we saw earlier that the ratings are distributed between 0.5 and 5.



The results of models to this point are

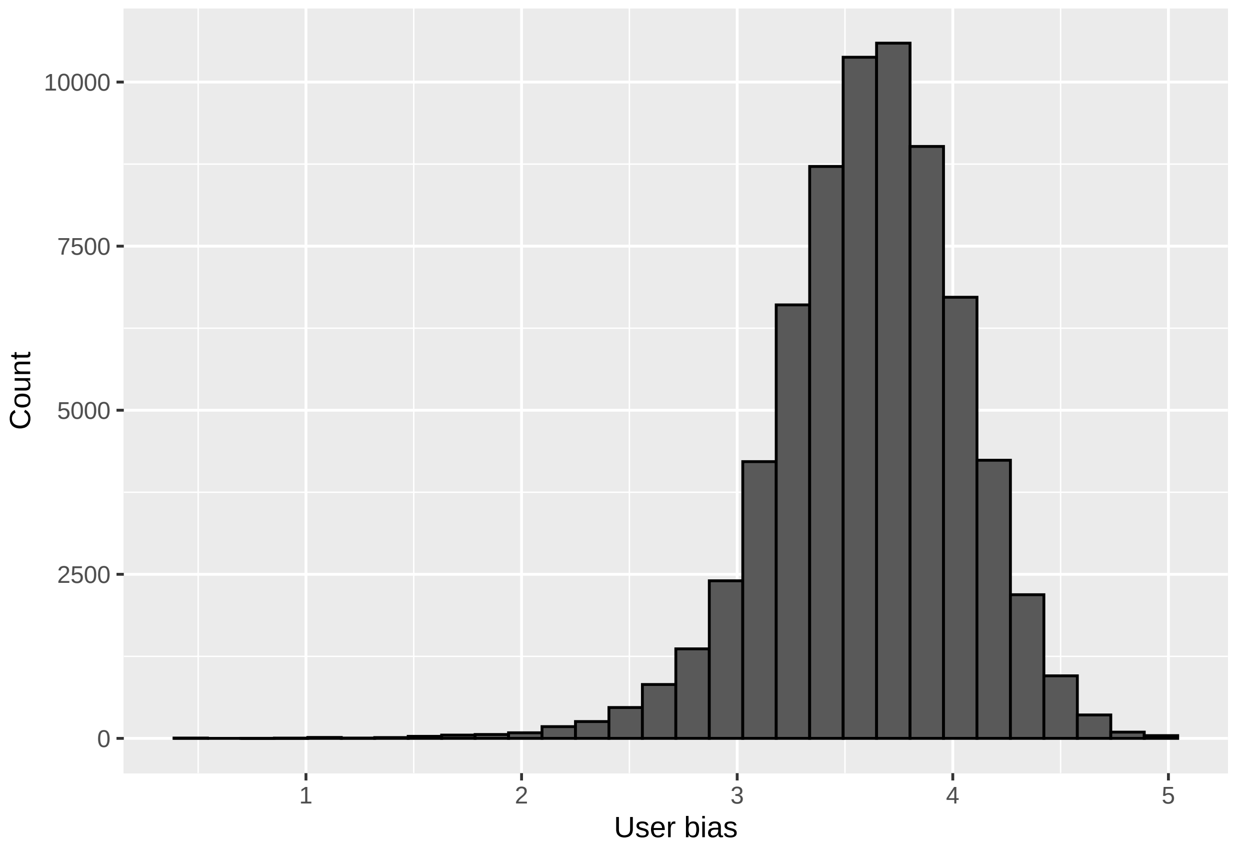
Model	RMSE
Naive - mean user rating	1.0590002
Movie Effect	0.9426564

Movie + user effect

This model adds a new term, b_u representing the average rating for user u. This model is represented by the equation

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

Note that b_u is the average of $Y_{u,i} - \mu - b_i$ for each user u. We will only include users who have rated at least 100 movies. This distribution appears slightly skewed also, although not as much as with $\hat{b}_i$. As expected, the distribution ranges from 0.5 to 5.



The results of models to this point are

Model	RMSE
Naive - mean user rating	1.0590002
Movie Effect	0.9426564
Movie + User Effect	0.8646047

Regularized movie effect

Regularization penalizes extreme estimates (whether large or small) that are formed using small sample sizes, as is the case when a small number of users rate a movie. In such cases, the prediction is shrunk to a more conservative value in an effort to decrease errors and thereby reduce RMSE.

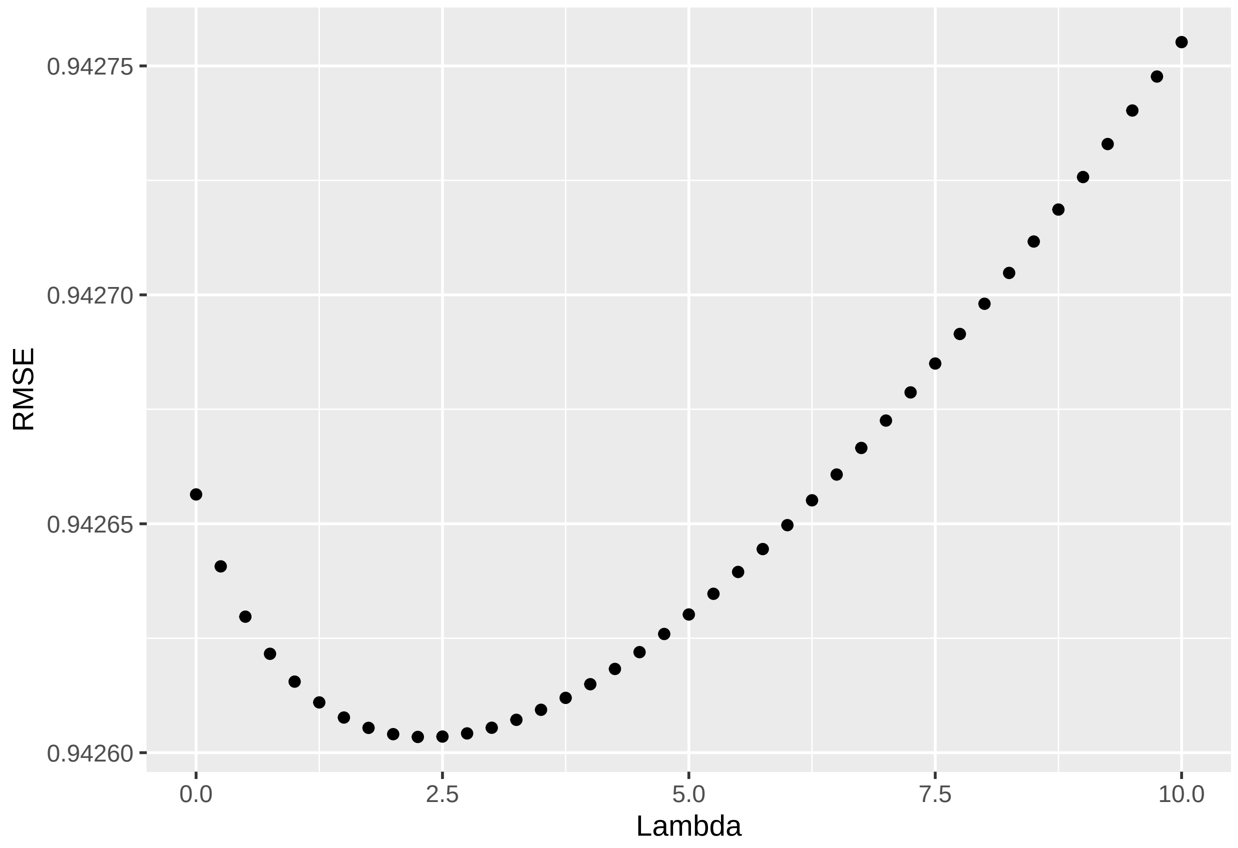
Using regularization to estimate the movie effect, we estimate the b's by minimizing the equation $\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i)^2+\lambda\sum_i b_i^2$.

The optimal penalty λ to use can be found using cross-validation. It will minimize the equation

$\hat{b}_i(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i}-\hat{\mu})$

where n_i is the number of ratings for movie i.

We apply cross validation to obtain the optimal penalty, λ. The plot below shows the value of λ producing the lowest RMSE.

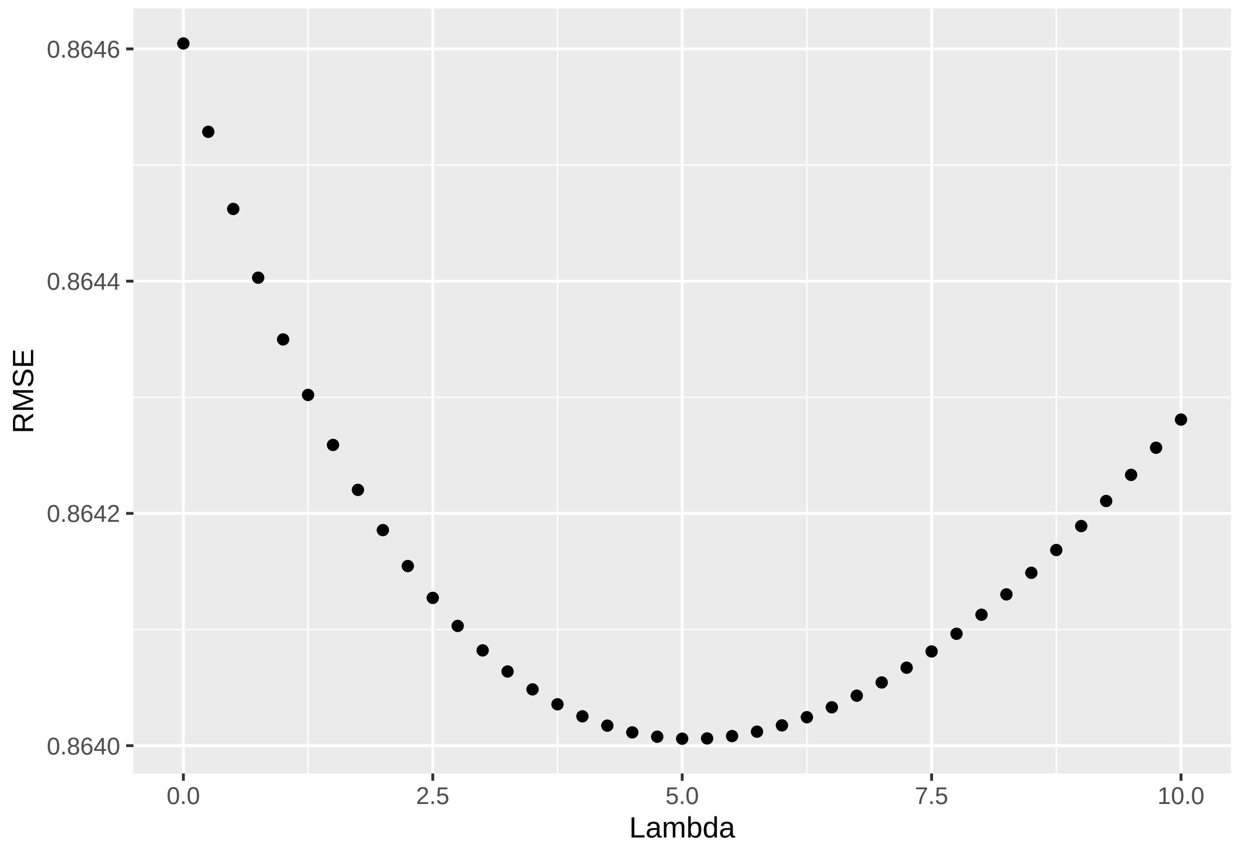


The optimal value of λ was found to be 2.25. The results of models to this point are

Model	RMSE
Naive - mean user rating	1.0590002
Movie Effect	0.9426564
Movie + User Effect	0.8646047
Regularized Movie Effect	0.9426034

Regularized movie + user effect

Using regularization to estimate the movie and user effect, we estimate the b's by minimizing the equation $\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2)$.
The optimal penalty λ to use can be found using cross-validation. It will minimize the equations
$\hat{b}_i(\lambda) = \frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i} - \hat{\mu})$ and
$\hat{b}_u(\lambda) = \frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i} - \hat{\mu} - \hat{b}_i)$.
We apply cross validation to obtain the optimal penalty, λ. The plot below shows the value of λ producing the lowest RMSE.



The optimal value of λ was found to be 5. The results of models to this point are

Model	RMSE
Naive - mean user rating	1.0590002
Movie Effect	0.9426564
Movie + User Effect	0.8646047
Regularized Movie Effect	0.9426034
Regularized Movie + User Effect	0.8640060

Matrix Factorization

For this method data are processed as a large and sparse matrix, then decomposed into two smaller dimensional matrices with latent features and less sparsity. To make the process more efficient the recosystem package will be used. We start by converting data into the recosystem format, find the best tuning parameters, train and finally test it.

The results of models to this point are

Model	RMSE
Naive - mean user rating	1.0590002
Movie Effect	0.9426564
Movie + User Effect	0.8646047
Regularized Movie Effect	0.9426034
Regularized Movie + User Effect	0.8640060
Matrix Factorization	0.7845206

Results

Modeling results

The results of all models are as follows:

Model	RMSE
Matrix Factorization (on final_holdout_test)	0.7814768
Matrix Factorization	0.7845206
Regularized Movie + User Effect	0.8640060
Movie + User Effect	0.8646047
Regularized Movie Effect	0.9426034
Movie Effect	0.9426564
Naive - mean user rating	1.0590002

Discussion of model performance

The winner is clear among this group of models. Matrix factorization produces a RMSE far below the target of 0.86490. It's RMSE is 0.7845206.

All models improved on the baseline model's performance. The regularized models also beat out their corresponding counterparts proving that regularization is effective. It can be seen that the regularized move + user effect model performed just below the 0.86490 target. However, this may not have been the case if the same model ran on the final_holdout_test data set.

Conclusion

Brief summary of report

This project examined numerous algorithms to predict the ratings of movies by users of a subset of MovieLens data. The algorithms' performances were evaluated using root mean squared error (RMSE). A linear regression model utilizing regularization (regularized movie + user effect) provided very good performance but could not outdo the matrix factorization method. Although they both performed below the target of 0.86490, the matrix factorization method also beat the target with the final_holdout_test data set. It's RMSE on the final_holdout_set was 0.7814768.

Potential impact

Performance gains on any recommendation system would result in algorithms better able to predict what we like based on our past ratings. This would reduce the need to choose from an overwhelming number of goods and services recommendations that we are bombarded with on a daily basis, particularly e-commerce and media related.

Limitations

The best performing algorithm in this project was very computationally demanding having to manipulate structures containing tens of millions of items. That algorithm takes approximately 50 minutes to run on an old i7-5500U system with 15.7 GB of usable RAM. If we can get similar performance from an algorithm that takes ${\frac{1}{10}}^{th}$ of the time, we would go with the latter, even at the expense of performance.

Future work

In the future we will consider other effects such as movie + user + genre effect and movie + user + time effect and if possible regularized versions of these. We will also consider the use of principal component analysis, support vector machines, eXtreme gradient boosting, and other recommender systems. An ensemble model will also be created.