HarvardX - Data Science: Capstone Course - MovieLens Project

James Melanson

2021-07-16

Contents

1 Introduction

1.1 Purpose

Media streaming services, such as Netflix and Spotify, contain a wide variety of media in their libraries. Further, different users of these services have different preferences in what they view or listen to. These services recommend videos and create personalized playlists for users. However, how can these recommendations be produced?

The aim of this project is to produce a machine learning-based model to predict a user's rating of a movie, using information from a dataset containing 10 million movie ratings. The accuracy of the model was assessed by measuring the difference between users' true ratings, contained in a test set that was held out during the model's development, and the model's predictions of user ratings. Predictions of users' ratings were able to be made that were informed by information present within the dataset, allowing for the prediction model's use as a movie recommendation system.

1.2 Dataset

The MovieLens 10M dataset contains movie ratings extracted from the MovieLens movie recommendation service, located at the following link: https://movielens.org/. Released in January 2009, the MovieLens 10M dataset contains 10 million movie ratings from 72,000 users on 10,000 movies.

This dataset contains three data-containing files: "movies.dat", "ratings.dat", and "tags.dat", which refer to the movies contained within the dataset, ratings of movies given by users, and movie tag information given by users, respectively.

1.2.1 Movies.dat File Structure

The movies.dat file contains information related to the movies contained within the MovieLens 10M dataset, such as a movie's title and release year, the movie's unique MovieLens identification number, and the genres to which the movie belongs.

1.2.2 Ratings.dat File Structure

The ratings.dat file contains a sample of the ratings given to movies by users in the MovieLens movie recommendation service. Information included within ratings.dat include a user's unique identification number, the user's movie rating, the MovieLens identification number of the movie for which the rating was given, as well as the time at which this rating was given; referred to as the timestamp.

1.2.3 Tags.dat File Structure

The tags.dat file contains user-generated tags for movies contained within the MovieLens 10M dataset. This file was not used as part of the analysis performed in this document.

1.3 Goal of the project

The goal of this project was to produce a prediction model for movie ratings, using machine learning methodology. As part of this project, I trained a machine learning algorithm on a portion of the MovieLens 10M dataset, which effectively created a movie recommendation system.

This algorithm recommends movies to users by predicting the rating a user will give a movie before they have viewed it, using movie ratings data from other users. The quality of the movie recommendation system

was measured by comparing its predictions of users' movie ratings to the users' true ratings, using the root mean squared error (RMSE). Importantly, the algorithm was blinded to the users' true ratings during its development.

The RMSE is defined as follows:

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

where:

N = the number of observations

$\hat{y}$ = predicted user rating by the machine learning algorithm

y = actual user rating

Thus, the RMSE is a measure of the error present in the predictions generated by the model. The lesser the difference between $\hat{y}$ and y, the lower the RMSE and the better the model is predicting the values of y.

The goal of this project is to train a machine learning algorithm on the training set, that when tested on the validation set, produces a RMSE that is less than 0.86490. This target RMSE was provided by the HarvardX team.

1.4 Key steps that were performed

The following key steps were performed as part of this analysis:

1. Data retrieval using R from the GroupLens MovieLens 10M Dataset page: http://files.grouplens.org/datasets/movielens/ml-10m.zip
2. Data cleaning
3. Splitting of the MovieLens data into a training set (edx) and a hold-out set that was used to assess the accuracy of the final model (validation)
4. Splitting of the edx training set into two subsets: a training set (train) and test set (test_set) to develop and test models
5. Development and testing of machine learning algorithms to predict users' movie ratings, using the root mean squared error (RMSE) as a metric to compare different algorithms

   Note: Steps 1, 2, and 3 were accomplished using a script provided by the HarvardX team.

2 Methods

2.1 Data Retrieval

The data used for this analysis was retrieved using R from the GroupLens MovieLens 10M Dataset page, available at: http://files.grouplens.org/datasets/movielens/ml-10m.zip. A temporary file name was created using the `tempfile` function, and the file was downloaded from its link using the `download.file` function to the file path created in the first step.

2.2 Data Cleaning

2.2.1 Data Cleaning of Movies.dat

The Movies.dat file was cleaned by splitting the lines present in the file into three columns, based on the presence of two double colons "::". The columns generated from this process were named as follows: "movieId", "title", and "genres".

2.2.2 Data Cleaning of Ratings.dat

The Ratings.dat file was cleaned by substituting the two double colons "::" separating the data elements within a line, with tabs to create a data structure akin to a tab-separated value file. This file was then passed to `fread`, which automatically detected the number of columns and rows of the previously processed data to create a table of clean data in the data.frame format. Finally, the columns were named as follows: "userId", "movieId", "rating", "timestamp".

2.2.3 Table Joining

To create a table containing users' ratings of movies and information about those movies, the `movies` data.frame was joined to the `ratings` data.frame with the `left_join` function; using "movieId" as the column linking the two tables.

2.3 Partitioning of the MovieLens Data into Training and Validation Sets

The `caret` package was used to partition the MovieLens dataset into training and validation sets. The training set, `edx`, was used to train the machine learning algorithm and consisted of 90% of the MovieLens data. The validation set, `validation`, consisted of 10% of the MovieLens data and was held out during model development to test the accuracy of the final machine learning algorithm.

Following this, the edx dataset was partitioned into training and validation sets. The training set, `train`, was used for model development and consisted of 90% of the edx data. The validation set, `test_set`, was used for validation during model development and contained 10% of the edx data.

> Note: When partitioning the data into training and test sets, records were moved from the validation set and into the training set. This was performed to ensure that user IDs and movie IDs present in the validation set were also present in the training set.

2.4 Root mean squared error (RMSE)

The RMSE is defined as follows:

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

where:

N = the number of observations

$\hat{y}$ = predicted user rating by the machine learning algorithm

y = actual user rating

2.5 Data Exploration and Visualization

2.5.1 edX Dataset

The edX dataset contains 9000055 ratings of 10677 movies from 69878 different users. Some of the most rated movies in this dataset are very-well known movies, as shown in the table below.

Table 1: Top 10 Movies by Number of Ratings in the edX Dataset

Title	n
Pulp Fiction (1994)	31,362
Forrest Gump (1994)	31,079
Silence of the Lambs, The (1991)	30,382
Jurassic Park (1993)	29,360
Shawshank Redemption, The (1994)	28,015
Braveheart (1995)	26,212
Fugitive, The (1993)	25,998
Terminator 2: Judgment Day (1991)	25,984
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25,672
Apollo 13 (1995)	24,284

The movie ratings in this dataset range from 0.5 stars to 5 stars, which is visualized in Figure 1. Of note, is that half-star ratings are less common than whole star ratings.

Table 2: Distribution of Movie Ratings in edx Dataset

Movie Rating	Frequency
0.5	85,374
1	345,679
1.5	106,426
2	711,422
2.5	333,010
3	2,121,240
3.5	791,624
4	2,588,430
4.5	526,736
5	1,390,114

Finally, the following table provides an overview of how many movie ratings are contained within the edX dataset across a group of selected genres.

Table 3: Number of Movie Ratings in the edx Dataset for Selected Genres

Genre	n
Comedy	3,540,930
Drama	3,910,127
Thriller	2,325,899
Romance	1,712,100

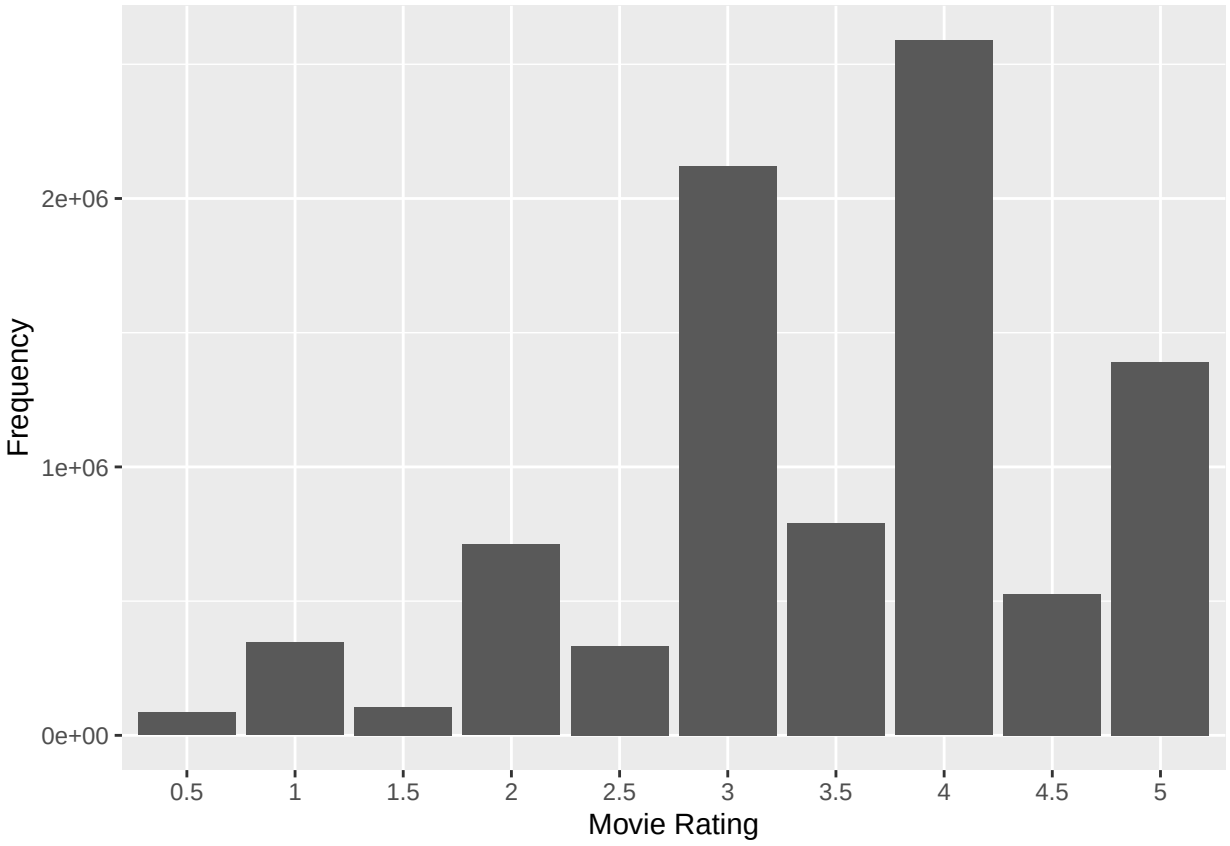

Figure 1: Distribution of Movie Ratings in the edX Dataset

2.5.2 Validation Dataset

The validation dataset consists of 999999 ratings of 9809 movies from 68534 different users. Further exploration of this data was not performed as the final machine learning-based prediction model would be used to predict the ratings in this dataset as if they were not known. The accuracy of the final model was tested by calculating the RMSE between the true ratings contained within the `validation` dataset, and the predicted ratings produced by the model.

2.6 Modelling Approach

2.6.1 Effects

As part of the model I developed to predict users' ratings of movies, I accounted for the following effects:

- movies (b_{movie}), to account for movies that are generally viewed as better or worse than others
- users (b_{user}), to account for the variability that exists between users, i.e., whether some users tend to rate higher or lower than other users
- genres (b_{genre}), to account for the variability that exists between genres
- week (b_{week}), to account for the variability that exists in the ratings of the data set across time
- year released ($b_{year\_released}$), to account for variability that exists in ratings with respect to the year a movie was released
- time post-movie release (b_{time}), to account for the variability that exists in ratings with respect to how long it was reviewed post-release

2.6.1.1 Movie-specific effects (b_{movie})

Some movies are, on average, rated higher than others. Within this project, a movie-specific effect may be described as a score assigned to a movie that reflects how that movie's ratings generally compare to other movies' ratings. Thus, in predicting a user's future rating of a movie that has many "very good" ratings, it would be logical to predict that this user will also like the movie and rate it as "very good". Likewise, if the distribution of a movie's ratings is generally higher than that of other movies' distributions, it would be sensible to predict that a future user's rating will lie within this higher distribution.

To calculate movie-specific effects, the following procedure was used:

1. The average of all movie ratings was calculated; which we may call μ
2. The ratings were grouped by movie and the average difference between these movie ratings and μ was calculated to calculate the movie-specific effect
3. Step 2 was repeated across all of the movies present in the data set

This allowed for the identification of movies that were, on average, rated higher than the average movie score.

2.6.1.2 User-specific effects (b_{user})

User-specific effects were calculated in a method analogous to the strategy used to calculate movie-specific effects:

1. The average of all movie ratings was calculated and called μ
2. The ratings were grouped by user, and the difference between these movie ratings and μ after removing the variability that we already know exists due to movie-specific effects was calculated to calculate the user-specific effect

3. Step 2 was repeated across all of the movies present in the data set

This allowed for the identification of users who tended to like all the movies they watched, and those who were more negative in their ratings; relative to other users. This information is useful in the recommendation system, as we may be able to predict that a movie with a four-star average, will only be given a three-star rating if it is being reviewed by a tough critic.

2.6.1.3 Genre-specific effects (b_{genre})

Genre-specific effects were calculated in a method analogous to the strategy used to calculate movie-specific effects:

1. The average of all movie ratings was calculated and called μ
2. The ratings were grouped by genre, and the difference between these movie ratings and μ after removing the variability that exists due to movie-specific and user-specific effects was calculated to calculate the genre-specific effect
3. Step 2 was repeated across all of the movies present in the data set

This allowed for the identification of genres that were generally liked and disliked.

2.6.1.4 Week-specific effects (b_{week})

Week-specific effects were calculated in a method analogous to the strategy used to calculate movie-specific effects:

1. The average of all movie ratings was calculated and called μ
2. The ratings were grouped by the week the review was given, and the difference between these movie ratings and μ after removing the variability that exists due to movie, user, and genre-specific effects was calculated to calculate the week-specific effect
3. Step 2 was repeated across all of the movies present in the data set

By calculating week-specific effects, the variability that existed across time in the ratings of the data set was able to be incorporated into the recommendation system.

2.6.1.5 Release year-specific effects ($b_{year\_released}$)

Release year-specific effects were calculated in a method analogous to the strategy used to calculate movie-specific effects:

1. The average of all movies was calculated and called μ
2. The ratings were grouped by the year the movie was released, and the difference between these movie ratings and μ after removing the variability that exists due to movie, user, genre, and week-specific effects was calculated to calculate the release year-specific effect
3. Step 2 was repeated across all of the movies present in the data set

Calculating release date-specific effects allowed for the variability due to release year to be included in the prediction model.

2.6.1.6 Time post-movie release-specific effects (b_{time})

The amount of time that elapsed between when a movie was released and when a movie rating was given by a user, in years, was calculated. Time post-movie release-specific effects were calculated in a method analogous to the strategy used to calculate movie-specific effects:

1. The average of all movies was calculated and called μ
2. The difference in time, in years, between the release date of a movie and the year a movie rating was given was calculated
3. The ratings were grouped by years elapsed post-movie release, and the difference between these movies ratings and μ after removing the variability that exists due to movie, user, genre, week, and release year-specific effects was calculated
4. Step 3 was repeated across all of the movies present in the data set

By calculating the time elapsed between a movie's release and a movie's review by a user, variability due to this effect can be included in the prediction model. For example, movies reviewed soon after their release may be viewed more favourably than movies reviewed years after their release.

2.6.1.7 Regularization

Regularization was used to penalize large b's that were calculated from samples that had a low number of samples. The penalization term, λ, was a tuning parameter whose value(s) were empirically tested to find RMSE minimums of the models. Regularization was applied using the following method:

$$b_x(\lambda) = \frac{1}{\lambda + n_x} \sum_{u=1}^{n_x} (Y_{u,m} - \mu)$$

where:

- b_x is the x-specific effect
- λ is the penalization term used to shrink the effect estimates
- $Y_{u,m}$ is a user u's rating of a movie m
- μ is the average of all movie ratings

2.6.2 Models

2.6.2.1 Model 1: Predict the average user's rating One of the simplest models of predicting a user's movie rating, would be to predict the average of all users' ratings. Thus, the model would be described by the following equation:

$$Y_{u,m} = \mu + \epsilon$$

where:

- $Y_{u,m}$ is the model's movie rating prediction for user u and movie m
- μ is the average of all movie ratings
- ϵ is an error independently sampled from a distribution centered at zero

Although this model may be used to predict a user's rating of a particular movie, it does not take advantage of much information present in the data set. Nonetheless, this will serve as a starting point from which improvements in modelling can be compared.

2.6.2.2 Model 2: Movie effect

$$Y_{u,m} = \mu + b_{movie} + \epsilon$$

where:

$$b_{movie} = mean(Y_{u,m} - \mu)$$

- μ is the average of all movie ratings
- b_{movie} is the movie-specific effect
- ϵ is an error independently sampled from a distribution centered at zero

Model 2 adds a movie effect to the prediction model. This model will add a movie-specific effect b_{movie} to μ to produce a movie rating prediction Y.

2.6.2.3 Model 3: Movie effect (regularized)

$$Y_{u,m} = \mu + b_{movie} + \epsilon$$

where:

$$b_{movie}(\lambda) = \frac{1}{\lambda + n_m} \sum_{u=1}^{n_m} (Y_{u,m} - \mu)$$

- μ is the average of all movie ratings
- b_{movie} is the movie-specific effect
- ϵ is an error independently sampled from a distribution centered at zero

Model 3 is an extension of Model 2 that incorporates regularization. This model adds a movie-specific effect b_{movie} to μ to produce a movie rating prediction Y as in Model 2, but penalizes movie-specific effects with a factor λ.

2.6.2.4 Model 4: Movie effects + User effects (regularized)

$$Y_{u,m} = \mu + b_{movie} + b_{user} + \epsilon$$

where:

$$b_{movie}(\lambda) = \frac{1}{\lambda + n_m} \sum_{u=1}^{n_m} (Y_{u,m} - \mu)$$

$$b_{user}(\lambda) = \frac{1}{\lambda + n_u} \sum_{u=1}^{n_u} (Y_{u,m} - \mu)$$

- μ is the average of all movie ratings
- b_{movie} is the movie-specific effect
- b_{user} is the user-specific effect
- ϵ is an error independently sampled from a distribution centered at zero

Model 4 contains movie effects and user effects. This model adds movie-specific (b_{movie}) and user-specific (b_{user}) effects to μ to product a movie rating prediction Y. Further, Model 4 penalizes movie and user effects with a lambda that is common to both effects.

2.6.2.5 Model 5: Movie effects + User effects (regularized, with separate lambdas for movie effects and user effects)

$$Y_{u,m} = \mu + b_{movie} + b_{user} + \epsilon$$

where:

$$b_{movie}(\lambda_1) = \frac{1}{\lambda_1 + n_m} \sum_{u=1}^{n_m} (Y_{u,m} - \mu)$$

$$b_{user}(\lambda_2) = \frac{1}{\lambda_2 + n_u} \sum_{u=1}^{n_u} (Y_{u,m} - \mu)$$

- μ is the average of all movie ratings
- b_{movie} is the movie-specific effect
- b_{user} is the user-specific effect
- ϵ is an error independently sampled from a distribution centered at zero

Model 5 is an extension of Model 4, wherein the lambda that penalizes each effect is coded by a different variable. This model adds movie-specific (b_{movie}) and user-specific (b_{user}) effects to μ to product a movie rating prediction Y.

2.6.2.6 Model 6: Movie effects + User effects + Genre effects (regularized, with separate lambdas for movie, user, and genre effects)

$$Y_{u,m} = \mu + b_{movie} + b_{user} + b_{genre} + \epsilon$$

where:

$$b_{movie}(\lambda_1) = \frac{1}{\lambda_1 + n_m} \sum_{u=1}^{n_m} (Y_{u,m} - \mu)$$

$$b_{user}(\lambda_2) = \frac{1}{\lambda_2 + n_u} \sum_{u=1}^{n_u} (Y_{u,m} - \mu)$$

$$b_{genre}(\lambda_3) = \frac{1}{\lambda_3 + n_g} \sum_{u=1}^{n_g} (Y_{u,m} - \mu)$$

- μ is the average of all movie ratings
- b_{movie} is the movie-specific effect
- b_{user} is the user-specific effect
- b_{genre} is the genre-specific effect
- ϵ is an error independently sampled from a distribution centered at zero

Model 6 contains movie, user, and genre-specific effects. This model adds movie-specific (b_{movie}), user-specific (b_{user}), and genre-specific (b_{genre}) effects to μ to product a movie prediction Y. Further, the lambdas that penalize each effect are specific to each effect.

2.6.2.7 Model 7: Movie effects + User effects + Genre effects + Week effects (regularized)

$$Y_{u,m} = \mu + b_{movie} + b_{user} + b_{genre} + b_{week} + \epsilon$$

where:

$$b_{movie}(\lambda) = \frac{1}{\lambda + n_m} \sum_{u=1}^{n_m} (Y_{u,m} - \mu)$$

$$b_{user}(\lambda) = \frac{1}{\lambda + n_u} \sum_{u=1}^{n_u} (Y_{u,m} - \mu)$$

$$b_{genre}(\lambda) = \frac{1}{\lambda + n_g} \sum_{u=1}^{n_g} (Y_{u,m} - \mu)$$

$$b_{week}(\lambda) = \frac{1}{\lambda + n_w} \sum_{u=1}^{n_w} (Y_{u,m} - \mu)$$

- μ is the average of all movie ratings
- b_{movie} is the movie-specific effect
- b_{user} is the user-specific effect
- b_{genre} is the genre-specific effect
- b_{week} is the week-specific effect
- ϵ is an error independently sampled from a distribution centered at zero

Model 7 expands on Model 6 to include a week-specific effect. This model adds movie-specific (b_{movie}), user-specific (b_{user}), genre-specific (b_{genre}), and week-specific (b_{week}) effects to μ to product a movie prediction Y. The lambda used to penalize the effects is common between the effects.

2.6.2.8 Model 8: Movie effects + User effects + Genre effects + Week effects + Year released effects (regularized)

$Y_{u,m} = \mu + b_{movie} + b_{user} + b_{genre} + b_{week} + b_{year\_released} + \epsilon\$$

where:

$$b_{movie}(\lambda) = \frac{1}{\lambda + n_m} \sum_{u=1}^{n_m} (Y_{u,m} - \mu)$$

$$b_{user}(\lambda) = \frac{1}{\lambda + n_u} \sum_{u=1}^{n_u} (Y_{u,m} - \mu)$$

$$b_{genre}(\lambda) = \frac{1}{\lambda + n_g} \sum_{u=1}^{n_g} (Y_{u,m} - \mu)$$

$$b_{week}(\lambda) = \frac{1}{\lambda + n_w} \sum_{u=1}^{n_w} (Y_{u,m} - \mu)$$

$$b_{year\_released}(\lambda) = \frac{1}{\lambda + n_y} \sum_{u=1}^{n_y} (Y_{u,m} - \mu)$$

- μ is the average of all movie ratings
- b_{movie} is the movie-specific effect
- b_{user} is the user-specific effect
- b_{genre} is the genre-specific effect
- b_{week} is the week-specific effect
- $b_{year\_released}$ is the year released-specific effect
- ϵ is an error independently sampled from a distribution centered at zero

Model 8 contains movie, user, genre, week, and year released-specific effects. This model adds movie-specific (b_{movie}), user-specific (b_{user}), genre-specific (b_{genre}), week-specific (b_{week}), and year released-specific ($b_{year\_released}$) effects to μ to product a movie prediction Y. Further, the lambdas that penalize each effect are common between the effects.

2.6.2.9 Model 9: Movie effects + User effects + Genre effects + Week effects + Year released + Time Post-Movie Release effects (regularized)

$$Y_{u,m} = \mu + b_{movie} + b_{user} + b_{genre} + b_{week} + b_{year\_released} + b_{time} + \epsilon$$

where:

$$b_{movie}(\lambda) = \frac{1}{\lambda + n_m} \sum_{u=1}^{n_m} (Y_{u,m} - \mu)$$

$$b_{user}(\lambda) = \frac{1}{\lambda + n_u} \sum_{u=1}^{n_u} (Y_{u,m} - \mu)$$

$$b_{genre}(\lambda) = \frac{1}{\lambda + n_g} \sum_{u=1}^{n_g} (Y_{u,m} - \mu)$$

$$b_{week}(\lambda) = \frac{1}{\lambda + n_w} \sum_{u=1}^{n_w} (Y_{u,m} - \mu)$$

$$b_{year\_released}(\lambda) = \frac{1}{\lambda + n_y} \sum_{u=1}^{n_y} (Y_{u,m} - \mu)$$

$$b_{time}(\lambda) = \frac{1}{\lambda + n_t} \sum_{u=1}^{n_t} (Y_{u,m} - \mu)$$

- μ is the average of all movie ratings
- b_{movie} is the movie-specific effect
- b_{user} is the user-specific effect
- b_{genre} is the genre-specific effect
- b_{week} is the week-specific effect
- $b_{year\_released}$ is the year released-specific effect
- b_{time} is the time post-movie release-specific effect
- ϵ is an error independently sampled from a distribution centered at zero

Model 9 is an extension of Model 8 that includes a time post-movie release-specific effect. This model adds movie-specific (b_{movie}), user-specific (b_{user}), genre-specific (b_{genre}), week-specific (b_{week}), year released-specific ($b_{year\_released}$), and time post-movie release-specific (b_{time}) effects to μ to product a movie prediction Y. The lambdas that penalize each effect are common between the effects.

2.6.2.9.1 Model 9.1 Model 9.1 is the first iteration of this model, which took a larger step size of 0.25 across the lambda search space of five through seven.

2.6.2.9.2 Model 9.2 Model 9.2 is the second iteration of Model 9. In Model 9.1, it appeared that there could be a minimum at the bottom of the RMSE versus lambda curve that fell between the common lambda being set at 5.1 and 5.3. Thus, the lambda search space was restricted to this range and the step size was reduced to 0.05 to get increased resolution on this part of the RMSE versus lambda curve.

3 Results

3.1 Model 1: Predict the average user's rating

In Model 1, the average rating of movies in the training set `train`, was used as a prediction for all ratings of movies in the test set `test_set`. The RMSE of Model 1 was 1.0600537.

Model	RMSE	RMSE - Target RMSE
1: Predict the average	1.060054	0.1951537

3.2 Model 2: Movie effect

In Model 2, movie effects were added to the average rating of movies in the training set `train`, to produce ratings predictions for movies in the test set `test_set`. The RMSE of Model 2 was 0.9429615.

Model	RMSE	RMSE - Target RMSE
1: Predict the average	1.0600537	0.1951537
2: Movie effect	0.9429615	0.0780615

3.3 Model 3: Movie effect (regularized)

In Model 3, regularized movie effects were added to the average rating of movies in the training set `train`, to produce ratings predictions for movies in the test set `test_set`. The RMSE of Model 3 was 0.9429377.

Model	RMSE	RMSE - Target RMSE
1: Predict the average	1.0600537	0.1951537
2: Movie effect	0.9429615	0.0780615
3: Movie effect (regularized)	0.9429377	0.0780377

3.4 Model 4: Movie effects + User effects (regularized)

In Model 4, regularized movie and user effects were added to the average rating of movies in the training set `train`. This was performed to produce ratings predictions for movies in the test set `test_set`. The RMSE of Model 4 was 0.8641362.

Model	RMSE	RMSE - Target RMSE
1: Predict the average	1.0600537	0.1951537
2: Movie effect	0.9429615	0.0780615
3: Movie effect (regularized)	0.9429377	0.0780377
4: Movie and user effects (regularized)	0.8641362	-0.0007638

3.5 Model 5: Movie effects + User effects (regularized, with separate lambdas for movie effects and user effects)

In Model 5, regularized movie and user effects were added to the average rating of movies in the training set `train`. This was performed to produce ratings predictions for movies in the test set `test_set`. The RMSE of Model 5 was 0.8641484.

Model	RMSE	RMSE - Target RMSE
1: Predict the average	1.0600537	0.1951537
2: Movie effect	0.9429615	0.0780615
3: Movie effect (regularized)	0.9429377	0.0780377
4: Movie and user effects (regularized)	0.8641362	-0.0007638
5: Movie and user effects (regularized, with separate lambdas for movie effects and user effects)	0.8641484	-0.0007516

3.6 Model 6: Movie effects + User effects + Genre effects (regularized, with separate lambdas for movie, user, and genre effects)

In Model 6, regularized movie, user, and genre effects were added to the average rating of movies in the training set `train`. This was performed to produce ratings predictions for movies in the test set `test_set`. The RMSE of Model 6 was 0.8638229.

Model	RMSE	RMSE - Target RMSE
1: Predict the average	1.0600537	0.1951537
2: Movie effect	0.9429615	0.0780615
3: Movie effect (regularized)	0.9429377	0.0780377
4: Movie and user effects (regularized)	0.8641362	-0.0007638
5: Movie and user effects (regularized, with separate lambdas for movie effects and user effects)	0.8641484	-0.0007516
6: Movie, user, and genre effects (regularized, with seperate lambdas for movie, user, and genre effects)	0.8638229	-0.0010771

3.7 Model 7: Movie effects + User effects + Genre effects + Week effects (regularized)

In Model 7, regularized movie, user, genre, and week effects were added to the average rating of movies in the training set `train`. This was performed to produce ratings predictions for movies in the test set `test_set`. The RMSE of Model 7 was 0.8636771.

Model	RMSE	RMSE - Target RMSE
1: Predict the average	1.0600537	0.1951537
2: Movie effect	0.9429615	0.0780615
3: Movie effect (regularized)	0.9429377	0.0780377
4: Movie and user effects (regularized)	0.8641362	-0.0007638
5: Movie and user effects (regularized, with separate lambdas for movie effects and user effects)	0.8641484	-0.0007516
6: Movie, user, and genre effects (regularized, with seperate lambdas for movie, user, and genre effects)	0.8638229	-0.0010771
7: Movie, user, genre, and week effects (regularized)	0.8636771	-0.0012229

3.8 Model 8: Movie effects + User effects + Genre effects + Week effects + Year released effects (regularized)

In Model 8, regularized movie, user, genre, week, and year released effects were added to the average rating of movies in the training set `train`. This was performed to produce ratings predictions for movies in the test set `test_set`. The RMSE of Model 8 was 0.8634169.

3.9 Model 9: Movie effects + User effects + Genre effects + Week effects + Year released + Time Post-Movie Release effects (regularized)

In Model 9, regularized movie, user, genre, week, year released, and time post-movie release effects were added to the average rating of movies in the training set `train`. This was performed to produce ratings predictions for movies in the test set `test_set`.

3.9.1 Model 9.1

The RMSE of Model 9.1 was 0.8631303.

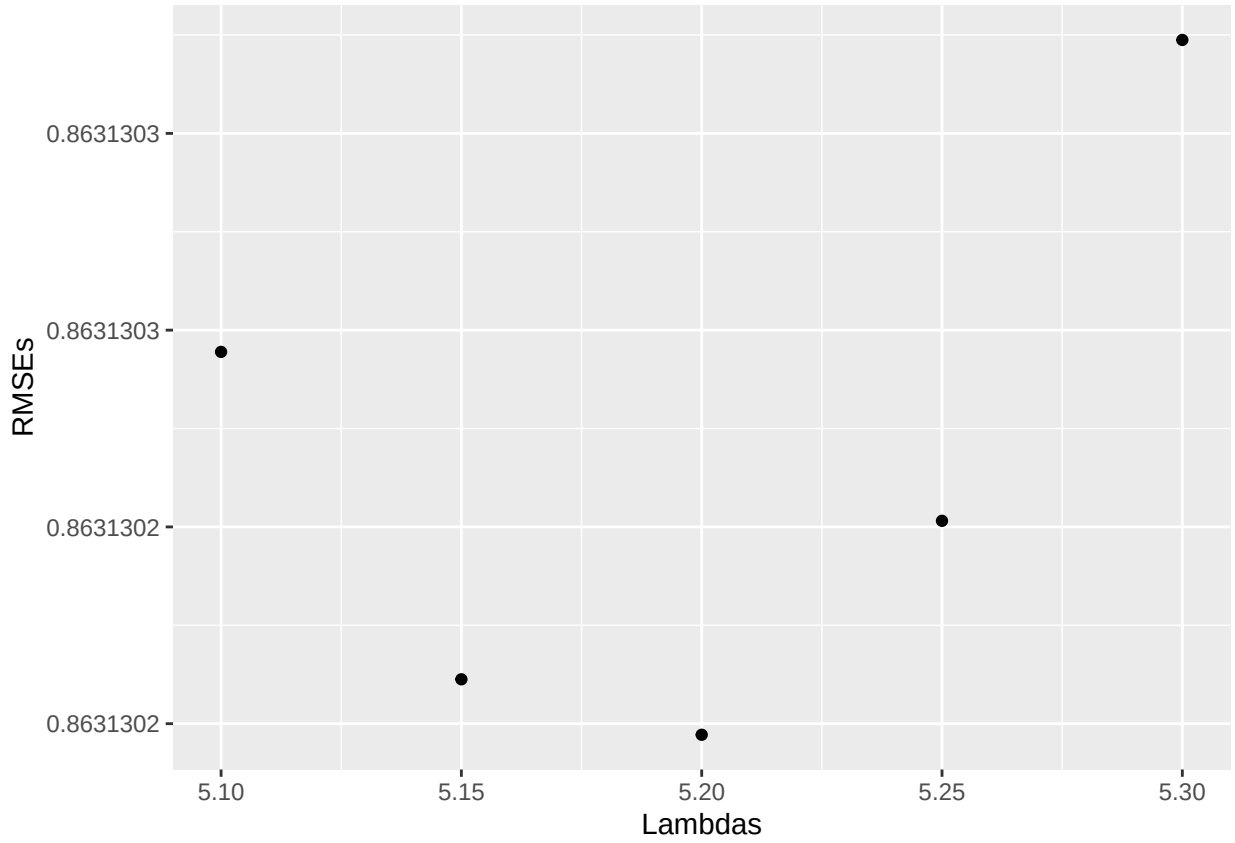

Figure 2: Model 9.1 - RMSEs versus Lambdas

Model	RMSE	RMSE - Target RMSE
1: Predict the average	1.0600537	0.1951537

Model	RMSE	RMSE - Target RMSE
2: Movie effect	0.9429615	0.0780615
3: Movie effect (regularized)	0.9429377	0.0780377
4: Movie and user effects (regularized)	0.8641362	-0.0007638
5: Movie and user effects (regularized, with separate lambdas for movie effects and user effects)	0.8641484	-0.0007516
6: Movie, user, and genre effects (regularized, with seperate lambdas for movie, user, and genre effects)	0.8638229	-0.0010771
7: Movie, user, genre, and week effects (regularized)	0.8636771	-0.0012229
8: Movie, user, genre, week, and year released effects (regularized)	0.8634169	-0.0014831
9.1: Movie, user, genre, week, year released, and time post-movie release effects (regularized, version 1)	0.8631303	-0.0017697

3.9.2 Model 9.2

The RMSE of Model 9.2 was 0.8631302.

Model	RMSE	RMSE - Target RMSE
1: Predict the average	1.0600537	0.1951537
2: Movie effect	0.9429615	0.0780615
3: Movie effect (regularized)	0.9429377	0.0780377
4: Movie and user effects (regularized)	0.8641362	-0.0007638
5: Movie and user effects (regularized, with separate lambdas for movie effects and user effects)	0.8641484	-0.0007516
6: Movie, user, and genre effects (regularized, with seperate lambdas for movie, user, and genre effects)	0.8638229	-0.0010771
7: Movie, user, genre, and week effects (regularized)	0.8636771	-0.0012229
8: Movie, user, genre, week, and year released effects (regularized)	0.8634169	-0.0014831
9.1: Movie, user, genre, week, year released, and time post-movie release effects (regularized, version 1)	0.8631303	-0.0017697
9.2: Movie, user, genre, week, year released, and time post-movie release effects (regularized, version 2)	0.8631302	-0.0017698

3.10 Validation of the model's accuracy against the hold-out test set

Validation of the final model, Model 9.2, was performed using the hold out test set created by the HarvardX team's script. The RMSE of Model 9.2's predicted ratings on the hold-out test set `validation` was 0.864173.

Model	RMSE	RMSE - Target RMSE
9.2: Movie, user, genre, week, year released, and time post-movie release effects (regularized, version 2)	0.8631302	-0.0017698
9.2: Validation using hold-out test set `validation`	0.8641730	-0.0007270

4 Conclusion

The aim of this project was to produce a machine learning-based prediction model that could serve as a movie recommendation system. Using a model that included regularized movie, user, genre, week, year released, and time post-movie release effects, a RMSE of 0.864173 was achieved; below the target RMSE of `0.86490`.

One of the limitations of this set of models is that they rely solely on the variation captured within the defined effects to produce a prediction. Further, another limitation with this set of models is that they discourage the use of separate lambdas for each of the different effects; as the calculations quickly become computationally intensive.

Future work on this recommendation system could utilize more advanced machine learning methods such as principal component analysis to reduce the RMSE obtained by the prediction model. However, this may come with challenges of model interpretability and determining what each principal component represents.

5 References

Grolemund, G., Wickham, H. (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL https://www.jstatsoft.org/v40/i03/.

GroupLens Research Lab - University of Minnesota, Twin Cities. (2021, March 2). MovieLens 10M Dataset. GroupLens. https://grouplens.org/datasets/movielens/10m/

Irizarry, R. A. (2021, July 3). Introduction to Data Science. Rafalab.Github.Io. https://rafalab.github.io/dsbook/

Kuhn, M. (2021). caret: Classification and Regression Training. R package version 6.0-88. https://CRAN.R-project.org/package=caret

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D. A., François, R., ... & Yutani, H. (2019). Welcome to the Tidyverse. Journal of open source software, 4(43), 1686.

6 Appendix

6.1 CreateTrainandValidationSets.R

This script was provided by the HarvardX team to perform retrieval of the GroupLens MoviesLens 10M data set, cleaning of the data, and creation of the training and validation sets used for training the machine learning algorithm.

```
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

# Note: this process could take a couple of minutes

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")

library(tidyverse)
library(caret)
library(data.table)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# if using R 3.6 or earlier:
# <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
#                                     title = as.character(title),
#                                     genres = as.character(genres))
# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
```

```r
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```