Data Science Capstone - Movielense project

Julia Schröder

23/02/2021

1. Introduction

1a. Description of dataset

The dataset used in this analysis is called "movielens" and contains 10 million ratings of movies by users. The dataset was split into the `train` (90%) and `validation` sets (10%) according to the task. The analyses going forward will only use and describe the `train` dataset. It comprises 23,371,423 ratings of 10,677 movies of 19 genres (plus category "no genres listed") given by 69,878 users.

1b. Summary of the project's goal

Goal of this project was to build a movie recommendation system. It will predict movie ratings by training a machine learning (ML) algorithm using the available ratings from the 10M version of the MovieLens dataset. Herefore, the `train` dataset was divided into a training set comprising 90% of the data and a test dataset with 10% of the data. The different features were added sequentially when building the model using the training data in order to improve the model performance, which was evaluated using the residual mean squared error (RMSE). The final result of the RMSE was obtained using the `validation` dataset according to the task.

2. Methods

The methods applied in this project were taken from the Machine Learning course on EdX.org and the book "Introduction to Data Science" by Prof. Rafael A. Irizarry. Data cleaning, exploratory data analysis, model selection and evaluation were performed in R using the packages from the `tidyverse`.

At first, the data was cleaned using the `stringr` package: The genres were unnested by splitting the genres column by the "|" string and joining the data back using the movieId column.

Next, the data was split according to the task into `train` (90%) and `validation` datasets (10%).

An exploratory data analysis was performed to find the key features of the dataset necessary to build the model for the recommendation system. It was postulated that rating are influenced by the following effects:

1. movie
2. user
3. genre

The hypotheses were analyzed and visualized using `ggplot`.

Next, the model was build sequentially. The basic model was simply based on the mean rating of the training dataset where every movie is predicted to have the same rating. In the next step, the effect of the rated movie was included by adding the average residual effect of the rating per movie to the model. Afterwards, the effect of the user was included by adding the average residual effect of the rating per user. Since genre also has an effect on the rating, the sum of the residual effect of the genres per movie was added, since some movies are categorized in multiple genres to the model. To avoid skewing the data, a regularized approach was used where large estimates with small sample sizes are penalized when calculating the effects. The parameter λ was tuned using cross-validation with values between 0 and 10 in increments of 0.5.

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

After each step, the residual mean squared error (RMSE) was calculated using the test dataset:

$$RMSE = \sqrt{\frac{1/N}{\sigma} u, i(\hat{y}_{u,i} - y_{u,i})^2}$$

3. Results

3.1 Data cleaning

The genre information were reformatted by extracting nested genres.

```r
# the nested genre information was split into multiple rows per movie with one genre each
genres <- str_split(movielens$genres, fixed("|"), simplify = TRUE) %>%
  as.data.frame() %>%
  bind_cols(select(movielens, movieId)) %>%
  distinct(.keep_all = TRUE) %>%
  gather(col, movie_genre, -movieId) %>%
  filter(movie_genre != "") %>%
  select(-col)

# the genre information was added back to the original dataset and only relevant columns were kept
movielens <- movielens %>%
  left_join(genres, by = "movieId") %>%
  rename(movie_title = title) %>%
  select(userId, rating, movieId, movie_title, movie_genre)
```

This is the head of the final dataframe used for further analyses:

userId	rating	movieId	movie_title	movie_genre
1	5	122	Boomerang (1992)	Comedy
1	5	122	Boomerang (1992)	Romance
1	5	185	Net, The (1995)	Action
1	5	185	Net, The (1995)	Crime
1	5	185	Net, The (1995)	Thriller
1	5	231	Dumb & Dumber (1994)	Comedy
1	5	292	Outbreak (1995)	Action
1	5	292	Outbreak (1995)	Drama
1	5	292	Outbreak (1995)	Sci-Fi
1	5	292	Outbreak (1995)	Thriller

3.2 Split in training and validation data

The data was split into `train` and `validation` data using the code provided in the task:

```r
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")

test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
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)
```

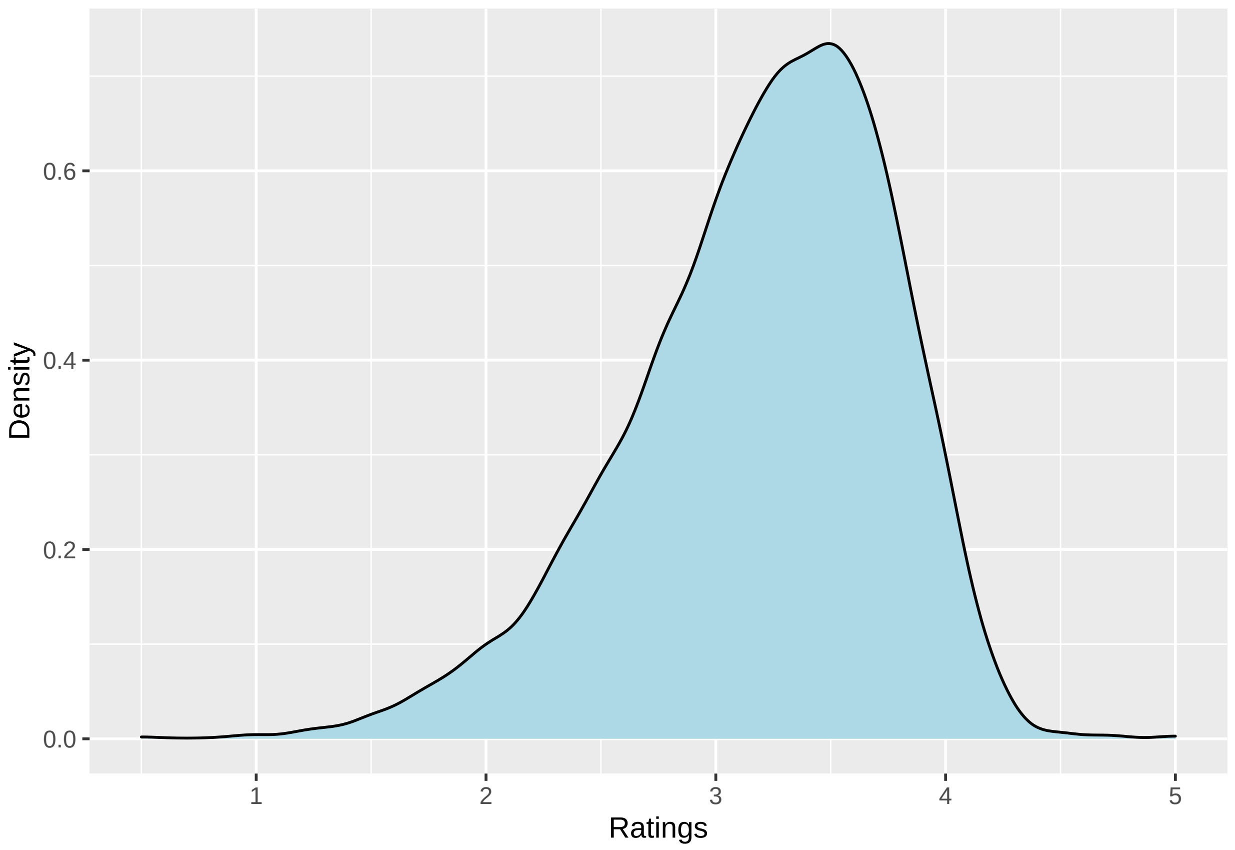
```
## Joining, by = c("userId", "rating", "movieId", "movie_title", "movie_genre")
```

```r
edx <- rbind(edx, removed)

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

3.3 Exploratory data analysis

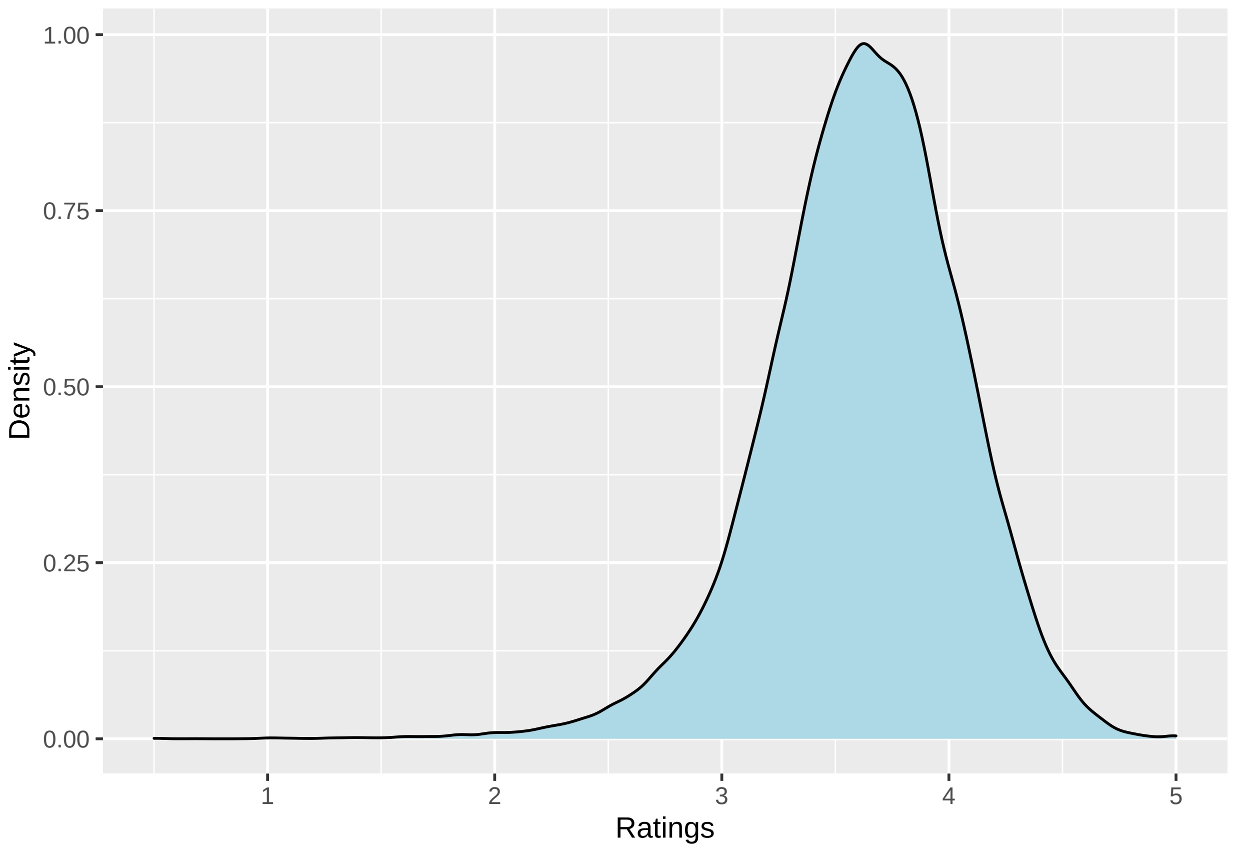
For the exploratory analysis, the effects of the rated movie, the user rating the movie and the genre of the rated movie were analyzed.

This is the number of movies per rating:



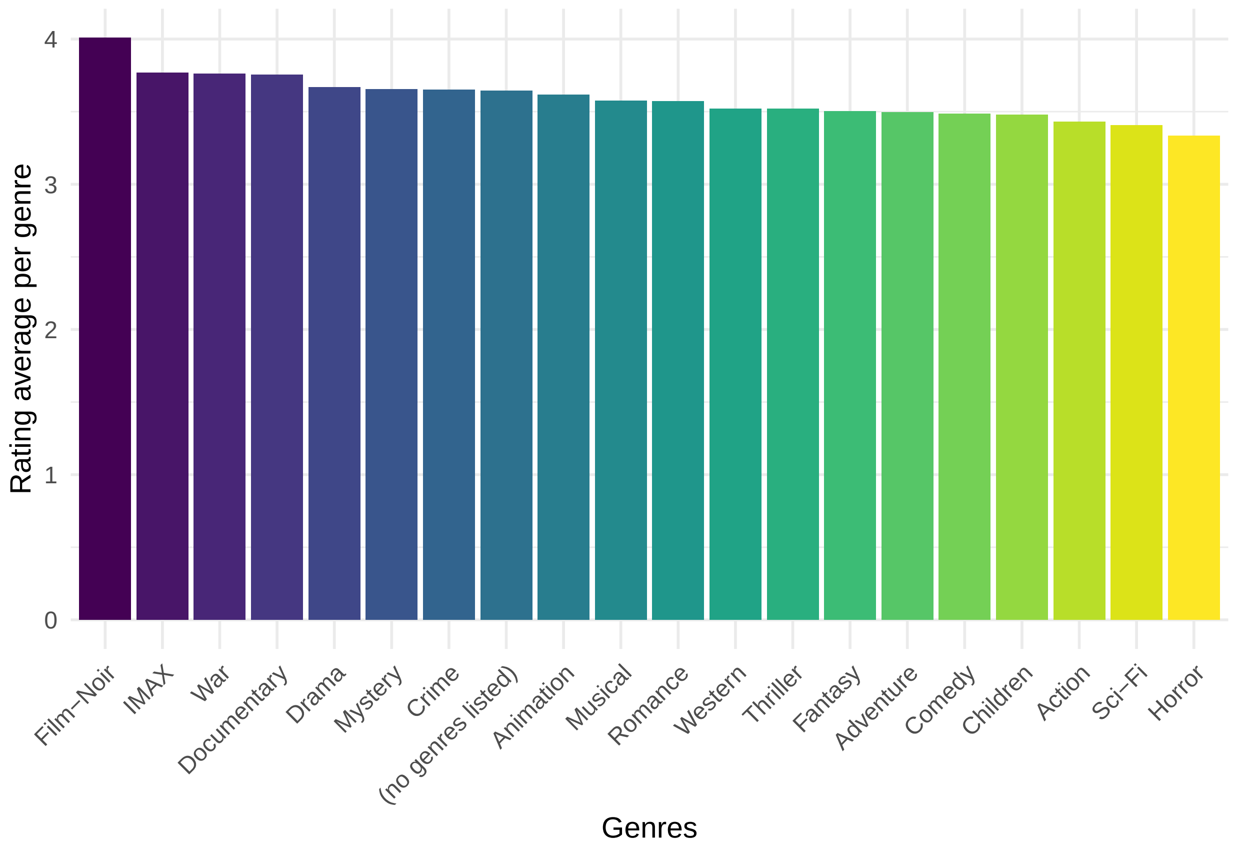
The distribution shows an effect of the movie on the ratings. Often-rated movies have an above-average rating. The mean rating of the movies differ suggesting an effect of the movie on the rating.

This is the distribution of users per rating:



The distribution shows an effect of the user on the ratings. Most users have a mean rating above average. The mean rating between users differs suggesting an effect of the user on the rating.

These are the average ratings per genre:



These data show an effect of the genre of the movie on the rating. All genres are in average rated above the mean, however, some higher than others suggesting an effect of the kind of genre on the rating.

3.4 Data setup

The `edx` dataset was split into training and test data by splitting off 10% of the dataset for testing.

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(edx$rating, times=1, p=0.1, list=FALSE)
temp <- edx[test_index,]
edx_train <- edx[-test_index,]

# exclude users and movies that do not appear in training data
edx_test <- temp %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

# add rows removed from test set back into training set
removed <- anti_join(temp, edx_test)
```

```
## Joining, by = c("userId", "rating", "movieId", "movie_title", "movie_genre")
```

```
edx_train <- rbind(edx_train, removed)

rm(temp, removed)
```

3.5 Modelling

3.5.1 Basic model

The most basic model for predicting the movie ratings $Y_{u,i}$ would be the average of all ratings $\hat{\mu}$ and the independent sampling errors $\epsilon_{u,i}$:

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

```r
# basic model: mu is the mean of all ratings calculated in the training set
mu <- mean(edx_train$rating)

# write function to calculate RMSE
rmse <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
}

# calculate RMSE for basic model with the ratings from the test set and mu
rmse_mu <- rmse(edx_test$rating, mu)
```

$\hat{\mu}$ is 3.5268879 and the RSME is 1.0521194.

method	RMSE
mu_hat	1.052

3.5.2 Movie effect

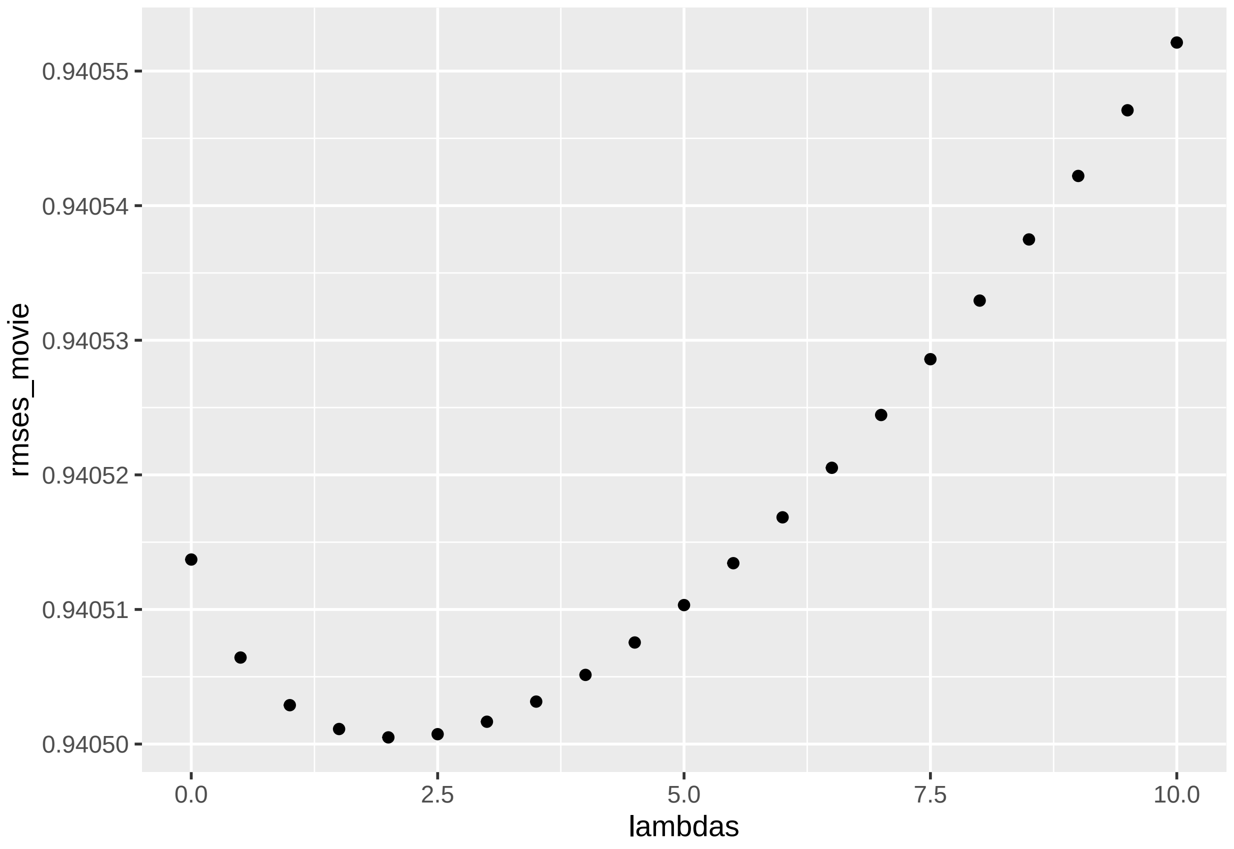
Since certain movies are rated higher than others, the addition of a movie-specific effect b_i is added to the model:

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

This effect can be estimated by calculating the average of the true rating $Y_{u,i}$ per movie minus the average rating $\hat{\mu}$.

Using cross-validation to tune the parameter λ, the value of λ with the smallest RMSE was chosen for the model.

```r
# setting of values for tuning of lambda
lambdas <- seq(0,10,0.5)

rmses_movie <- sapply(lambdas, function(l){
  # calculate mu in training data
  mu <- mean(edx_train$rating)

  # calculate penalized movie effect for different lambda values in training data
  b_movie <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_movie = sum(rating - mu)/(n()+l), .groups="keep")

  # calculate predictions using mu and movie effects in test data
  preds <- edx_test %>%
    left_join(b_movie, by = "movieId") %>%
    mutate(pred = mu + b_movie) %>%
    .$pred

  # calculate RMSE for each lambda
  rmse(edx_test$rating, preds)
})

# plot RMSEs for movie effect against lambda values
qplot(lambdas, rmses_movie)
```



In this case, the λ is 2. Adding the movie-specific effect to the model improves the RMSE to 0.9405005.

method	RMSE
mu_hat	1.0521
+ movie effect (regularized)	0.9405

3.5.3 User effect

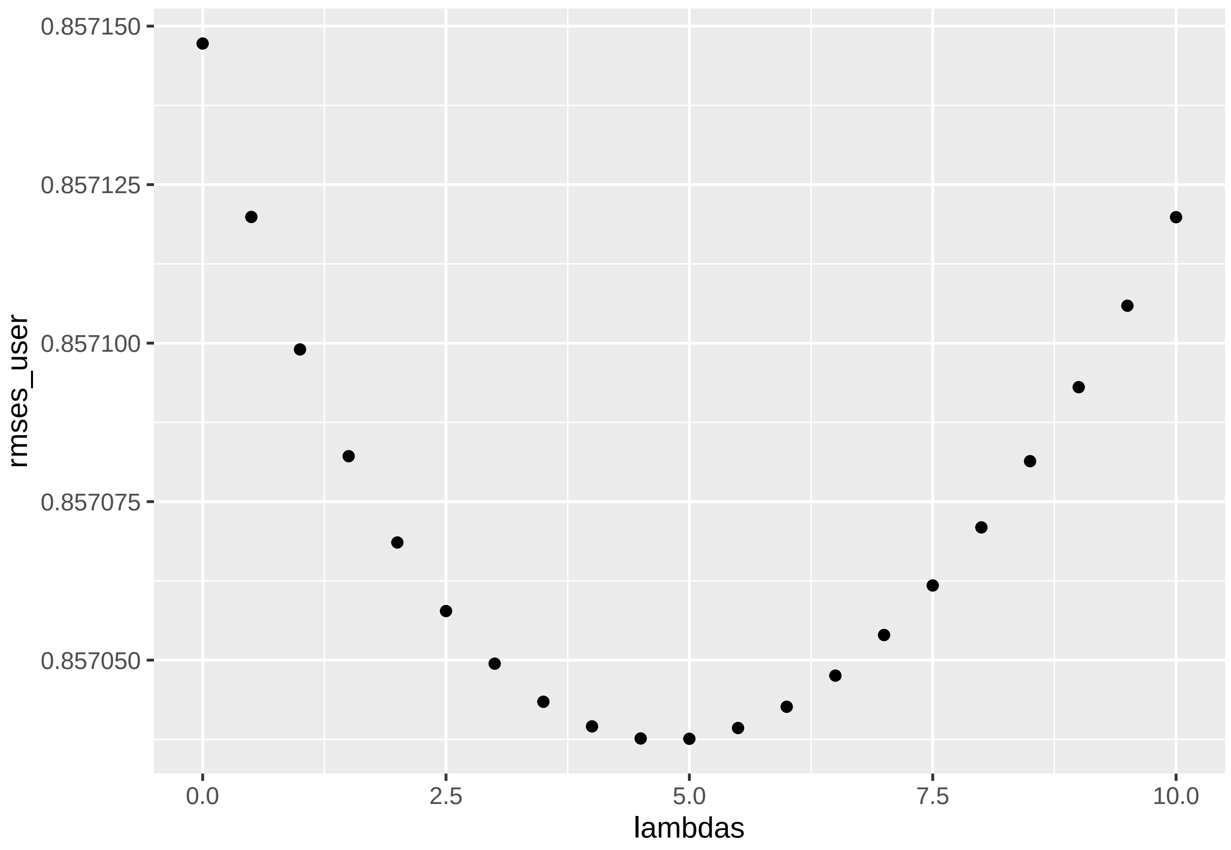
Similarily, certain users generally affect the rating of a movie by generally rating higher or lower. We can model this as the user effect b_u:

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

This effect can be estimated by computing the average of each rating $Y_{u,i}$ per user minus the average rating $\hat{\mu}$ and minus the movie effect b_i.

Again, cross-validation was used to tune the parameter λ.

```r
rmses_user <- sapply(lambdas, function(l){
  # calculate mu in training data
  mu <- mean(edx_train$rating)

  # calculate penalized movie effect for different lambda values in training data
  b_movie <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_movie = sum(rating - mu)/(n()+l), .groups="keep")

  # calculate penalized user effect for different lambda values in training data
  b_user <- edx_train %>%
    left_join(b_movie, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_user = sum(rating - mu - b_movie)/(n()+l), .groups="keep")

  # calculate predictions using mu, movie and user effects in test data
  preds <- edx_test %>%
    left_join(b_movie, by = "movieId") %>%
    left_join(b_user, by = "userId") %>%
    mutate(pred = mu + b_movie + b_user) %>%
    .$pred

  # calculate RMSE for each lambda
  rmse(edx_test$rating, preds)
})

# plot RMSEs agains lambda values
qplot(lambdas, rmses_user)
```



In this case, the λ is 5. Adding the user-specific effect to the model improves the RMSE further to 0.8570376.

method	RMSE
mu_hat	1.0521
+ movie effect (regularized)	0.9405
+ user effect (regularized)	0.8570

3.5.4 Genre effect

To model the effect of the genre on the rating, the effect of each genre per movie has to be added since some movies are assigned to more than one genre. We can model this as the genre effect b_k:

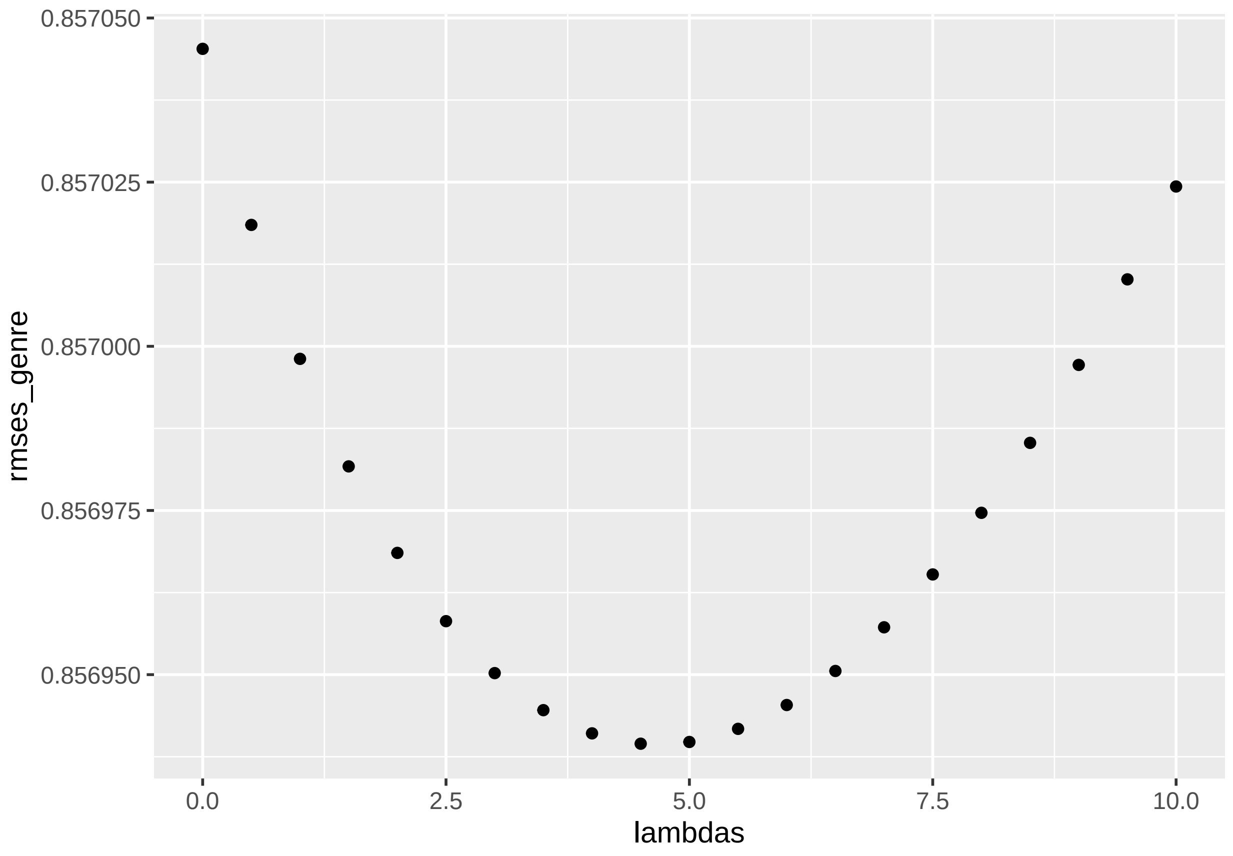
$$Y_{u,i} = \mu + b_i + b_u + \sum_{k=1}^{K} x_{u,i}^k \beta_k + \epsilon_{u,i} \ , \ with \ x_{u,i}^k = 1 \ if \ g_{u,i} \ is \ genre \ k$$

This effect can be estimated by computing the average of each rating $Y_{u,i}$ per genre minus the average rating $\hat{\mu}$, minus the movie effect b_i and minus the user effect b_u.

Cross-validation was used to tune the parameter λ.

```
rmses_genre <- sapply(lambdas, function(l){
  # calculate mu in training data
  mu <- mean(edx_train$rating)

  # calculate penalized movie effect for different lambda values in training data
  b_movie <- edx_train %>%
    group_by(movieId) %>%
```

```r
    summarize(b_movie = sum(rating - mu)/(n()+l), .groups="keep")

  # calculate penalized user effect for different lambda values in training data
  b_user <- edx_train %>%
    left_join(b_movie, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_user = sum(rating - mu - b_movie)/(n()+l), .groups="keep")

  # calculate penalized genre effect for different lambda values in training data
  b_genre <- edx_train %>%
    left_join(b_movie, by = "movieId") %>%
    left_join(b_user, by = "userId") %>%
    group_by(movie_genre) %>%
    summarize(b_genre = sum(rating - mu - b_movie - b_user)/(n()+l), .groups="keep")

  # calculate predictions using mu, movie, user and genre effects in test data
  preds <- edx_test %>%
    left_join(b_movie, by = "movieId") %>%
    left_join(b_user, by = "userId") %>%
    left_join(b_genre, by = "movie_genre") %>%
    group_by(movieId, userId) %>%
    mutate(pred = mu + b_movie + b_user + sum(b_genre)) %>%
    .$pred

  # calculate RMSE for each lambda
  rmse(edx_test$rating, preds)
})

# plot RMSEs against lambda values
qplot(lambdas, rmses_genre)
```



In this case, the λ is 4.5. Adding the cumulative genre-specific effects to the model improves the RMSE further to 0.8569395.

method	RMSE
mu_hat	1.0521
+ movie effect (regularized)	0.9405
+ user effect (regularized)	0.8570
+ genre effect (regularized)	0.8569

3.6 Final model

The final model that performed best on the training data, as shown by the lowest RMSE in the test data, incorporates the movie-, user- and genre-specific effects. The change in the RMSE when adding the genre-specific effect is low but still improves the RMSE.

Using the final model with the optimized parameter λ of 4.5, the model can now be trained on the entire `train` dataset before being applied to the final `validation` dataset as indicated by the task. This last step will provide the final RMSE.

```r
lambda <- lambdas[which.min(rmses_genre)]

# calculate mu of the train dataset
mu <- mean(edx$rating)

# calculate movie effect on the train dataset
b_movie <- edx %>%
  group_by(movieId) %>%
  summarize(b_movie = sum(rating - mu)/(n()+lambda), .groups="keep")

# calculate user effect on the train dataset
b_user <- edx %>%
  left_join(b_movie, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_user = sum(rating - mu - b_movie)/(n()+lambda), .groups="keep")

# calculate genre effect on the train dataset
b_genre <- edx %>%
  left_join(b_movie, by = "movieId") %>%
  left_join(b_user, by = "userId") %>%
  group_by(movie_genre) %>%
  summarize(b_genre = sum(rating - mu - b_movie - b_user)/(n()+lambda), .groups="keep")

# calculate predictions using mu, movie, user and genre effects in validation dataset
preds <- validation %>%
  left_join(b_movie, by = "movieId") %>%
  left_join(b_user, by = "userId") %>%
  left_join(b_genre, by = "movie_genre") %>%
  group_by(movieId, userId) %>%
  mutate(pred = mu + b_movie + b_user + sum(b_genre)) %>%
  .$pred

# calculate final RMSE with true ratings from validation dataset and obtained predictions
rmse_final <- rmse(validation$rating, preds)
```

The RMSE of the final model is 0.8567866.

4. Conclusion

In this report, a movie recommendation system was built using the movielens data. The aim was to build a model with an RMSE below 0.86490. The final model fits this criterion. It incorporates the movie-, user- and genre-specific effects that the exploratory data analysis has shown to be relevant. The RMSE could be further reduced by incoporating more effects, such as the year of the movie or the date and time of the rating.

4.1 Limitations

A few limitations can be noded about the presented project. The effects and predictions could have been obtained using a linear regression model. However, this would be too computationally expensive. Furthermore, the cross-validation could have been performed using more finely incremented values of lambda, which was also too computationally expensive. Finally, other forms of machine learning algorithms could have been used, such as k-nearest neighbors or random forest approaches.