Movielens

Andy Fan

2024-09-22

Introduction

The goal of this project is to use a unique movie ID and user ID combination to predict the rating of a movie, given by the specific user. The provided code loads a large movielens dataset and split it into training and test set.

Step 1: Transform raw data into useful predictors.

Total of (3) predictors will be used for this exercise, see below:

- Predictor 1: Average rating of a specific movie
- Predictor 2: Average rating from a specific user
- Predictor 3: Average rating of similar movies ("similar" is defined by same genres)

Here's the function that transform raw data to predictors:

```
process_data <- function(data){

  # Predictor 1: all ratings from user u
  user_avg <- data %>%
    group_by(userId) %>%
    summarise(user_avg = mean(rating))

  # Predictor 2: all ratings of movie i
  movie_avg <- data %>%
    group_by(movieId) %>%
    summarise(movie_avg = mean(rating))

  # Predictor 3: ratings of movies similar to i
  genres_avg <- data %>%
    group_by(genres) %>%
    summarise(genres_avg = mean(rating))

  # create data parameters for training and testing
  modified_data <- data %>%
    left_join(user_avg, by = "userId") %>%
    left_join(movie_avg, by = "movieId") %>%
    left_join(genres_avg, by = "genres") %>%
    select(rating, user_avg, movie_avg, genres_avg)
}
```

We will then use this function to transform the train and test dataset:

```
train_data <- process_data(edx)
test_data <- process_data(final_holdout_test)
```

Here's how the transformed data looks like:

```
head(train_data)
```

```
##   rating user_avg movie_avg genres_avg
## 1      5        5  2.858586   3.414486
## 2      5        5  3.129334   3.461289
## 3      5        5  3.418011   3.346671
## 4      5        5  3.349677   3.507407
## 5      5        5  3.337457   3.154399
## 6      5        5  2.487787   2.875053
```

Step 2: Use transformed data to train model

In this step we will use the transformed data to train a linear model, and use cross-validation to assess model accuracy.

In this step the training control is setup using cross validation, with 75% of data used for training in each iteration, and 10 total iterations.

```
control <- trainControl(method = "cv", p = 0.75, number = 10)
```

This next code will train a linear model. This code will take a long time to run due to large dataset.

```
fit <- train(rating ~ ., data = train_data,
             method = "glm",
             trControl = control,
             metric = "RMSE")
```

Step 3: Predictions

In this step we will use the trained model to predict movie ratings from the test dataset.

This code pulls the movie ratings from test dataset for final accuracy assessment:

```
# pull the correct rating data from final test data, for RMSE calculation
final_rating <- test_data %>%
  pull(rating)
```

This code removes the ratings from test dataset, leaving only predictors for the model to use:

```
# pull test parameters from final test data
test_parameters <- test_data %>%
  select(-rating)
```

This code uses the model to predict movie ratings:

```
# predict rating using model
rating_hat <- predict(fit, test_parameters)
```

Step 4: RMSE

In this step we calculate RMSE for the predictions, using this function:

```
# function to calculate RMSE
RMSE <- function(y_hat, y){
```

```
  sqrt(mean((y_hat - y)^2))
}
```

Then we calculate the final RMSE:

```
# calculate RMSE
RMSE(rating_hat, final_rating)
```

```
## [1] 0.8452249
```

Final RMSE = 0.8452249