Data 613 - Project 1

Anna Moy & Natalie Kalukeerthie

2025-06-08

Overview

Briefly describe the recommender system that you're going to build out from a business perspective, e.g. "This system recommends data science books to readers."

For this project, we are building a recommender system for Yelp-style restaurant reviews. Users rate restaurants from 1 to 5, but not every user rates every restaurant (in this project our item will be 'restaurants'). Our goal is to predict missing ratings, using two simple models:

- A **Raw Average Model** that predicts all ratings using the global mean.
- A **Baseline Model** that adjusts predictions based on both user and restaurant (item) biases.

Our core idea is that evaluating how well each model performs (using RMSE), we aim to learn how bias-aware predictions improve recommendation accuracy. We simulate reviews from 6 Yelp users (Alice, Ben, Cindy, David, Ella, Frank) for 5 restaurants (Pasta Place, Sushi Spot, Burger Barn, Curry Corner, Taco Town). Keep in mind that some users have not rated every restaurant.

```r
# Load library
library(reshape2)
```

```r
# Create Dataset with 6 users and 5 restaurants
ratings_df <- data.frame(
  user = c(
    rep("Alice", 5),
    rep("Ben", 5),
    rep("Cindy", 5),
    rep("David", 5),
    rep("Ella", 5),
    rep("Frank", 5)
  ),
  restaurant = rep(c("Pasta_Place", "Sushi_Spot", "Burger_Barn", "Curry_Corner", "Taco_Town"), 6),
  rating = c(
    5, NA, 4, NA, 4,       # Alice
    4, 3, 5, 3, 4,         # Ben
    4, 2, NA, NA, 3,       # Cindy
    2, 2, 3, 1, 2,         # David
    4, NA, 5, 4, 5,        # Ella
    4, 2, 5, 4, 4          # Frank
  )
)
```

We pivot the data into a matrix format (User-Item Matrix), which helps visualize missing ratings.

```r
#Transform data frame into a user-item matrix

#pivot the long format into a matrix for user, item and ratings
matrix <- dcast(ratings_df, user ~ restaurant, value.var = "rating")

# row labels will be the based on user
rownames(matrix) <- matrix$user

# remove the first column since it is repeating duplicate value
matrix <- matrix[, -1]

#print the matrix
print(matrix)
```

```
##       Burger_Barn Curry_Corner Pasta_Place Sushi_Spot Taco_Town
## Alice           4           NA           5         NA         4
## Ben             5            3           4          3         4
## Cindy          NA           NA           4          2         3
## David           3            1           2          2         2
## Ella            5            4           4         NA         5
## Frank           5            4           4          2         4
```

Next, we will split the data into a test (20% of data) and training (80% of data) dataset

```r
set.seed(42)
# Take the data and split into training 80% and testing 20% dataframe
train_indices <- sample(1:nrow(ratings_df), size = 0.8 * nrow(ratings_df))

train_df <- ratings_df[train_indices, ]
test_df <- ratings_df[-train_indices, ]
```

Our first model will be the raw average predictor, this model predicts the same score (the global mean) for all missing values.

```r
# Find the Raw average and calculate RMSE

# Raw Average on Training set and ignore NA
global_mean <- mean(train_df$rating, na.rm = TRUE)


# Add the raw avg onto the training and test dataset
train_df$raw_avg <- global_mean
test_df$raw_avg <- global_mean

# RMSE function
rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

train_fil <- train_df[!is.na(train_df$rating), ]
test_fil <- test_df[!is.na(test_df$rating), ]
```

```r
# Calculate RSME which takes the difference of ratings - avg mean for training and test set
rmse_train_raw <- rmse(train_fil$rating, train_fil$raw_avg)
rmse_test_raw <- rmse(test_fil$rating, test_fil$raw_avg)

rmse_train_raw
```

```
## [1] 1.133719
```

```r
rmse_test_raw
```

```
## [1] 1.182748
```

Our second model 2 is the **Baseline Model** assumes that each user and item (restaurant) has a tendency to rate higher or lower than average. This model adjusts predictions using:

- **User bias**: How much a user rates higher/lower than average.
- **Item bias**: How popular (or unpopular) a restaurant is compared to others.

We predict the rating as is: Predicted Rating = Global Mean + User Bias + Item Bias

This is a considered a collaborative filtering approach, where bias correction leads to more personalized predictions.

To prevent overfitting, we applied regularization to user and item bias estimates. This penalizes users or restaurants with very few ratings, reducing the risk of exaggerated bias values. For example, if a user has rated only one item extremely high, regularization ensures that this does not overly skew their predicted ratings across the system.

```r
#Finds out the bias values for all columns and rows in grid

library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Set regularization strength
lambda <- 10

# Compute regularized user bias
user_bias_df <- train_df[!is.na(train_df$rating), ] %>%
  group_by(user) %>%
  summarise(
    num_ratings = n(),
    sum_diff = sum(rating - global_mean),
    user_bias = sum_diff / (num_ratings + lambda)
  )

# Compute regularized restaurant bias
restaurant_bias_df <- train_df[!is.na(train_df$rating), ] %>%
  group_by(restaurant) %>%
  summarise(
    num_ratings = n(),
    sum_diff = sum(rating - global_mean),
    restaurant_bias = sum_diff / (num_ratings + lambda)
  )

# Find the mean for all bia users in training data
#user_avg <- aggregate(rating ~ user, data = train_df, mean)
# Take the user bias values and the difference from the raw mean
#user_avg$user_bias <- user_avg$rating - global_mean

#user_avg

# Find the mean for all items in the training data
#restaurant_avg <- aggregate(rating ~ restaurant, data = train_df, mean)

# Take the item values and the difference frm the raw mean
#restaurant_avg$restaurant_bias <- restaurant_avg$rating - global_mean

#restaurant_avg
```

```
#Baseline Predictor

# Merge regularized user and restaurant bias into training set
train_df <- merge(train_df, user_bias_df[, c("user", "user_bias")], by = "user", all.x = TRUE)
train_df <- merge(train_df, restaurant_bias_df[, c("restaurant", "restaurant_bias")], by = "restaurant"

# Predict using regularized baseline model
train_df$pred_baseline <- global_mean + train_df$user_bias + train_df$restaurant_bias

# Repeat for test set
test_df <- merge(test_df, user_bias_df[, c("user", "user_bias")], by = "user", all.x = TRUE)
test_df <- merge(test_df, restaurant_bias_df[, c("restaurant", "restaurant_bias")], by = "restaurant",

test_df$pred_baseline <- global_mean + test_df$user_bias + test_df$restaurant_bias

# Merge user bias into the training data set by user
#train_df <- merge(train_df, user_avg[, c("user", "user_bias")], by = "user", all.x = TRUE)

#Merger item into the training data set by item
#train_df <- merge(train_df, restaurant_avg[, c("restaurant", "restaurant_bias")], by = "restaurant", a

#train_df
```

```
#Baseline Predictor for Training data avg + user bias + item bias
#train_df$pred_baseline <- global_mean + train_df$user_bias + train_df$restaurant_bias

# Repeat same steps for testing dataset
#test_df <- merge(test_df, user_avg[, c("user", "user_bias")], by = "user", all.x = TRUE)
#test_df <- merge(test_df, restaurant_avg[, c("restaurant", "restaurant_bias")], by = "restaurant", all

#test_df$pred_baseline <- global_mean + test_df$user_bias + test_df$restaurant_bias
```

```
# Baseline Predictor RMSE

#remove the NA in the row
train_filtered <- train_df[!is.na(train_df$rating), ]
test_filtered <- test_df[!is.na(test_df$rating), ]

# Using the RMSE function taking the difference between rating and baseline predictor for training and
rmse_train_base <- rmse(train_filtered$rating, train_filtered$pred_baseline)
rmse_test_base <- rmse(test_filtered$rating, test_filtered$pred_baseline)


rmse_train_base
```

```
## [1] 0.8429098
```

```
rmse_test_base
```

```
## [1] 1.020685
```

```
cat("Raw Average Model:\n")
```

```
## Raw Average Model:
```

```
cat("  RMSE Train:", round(rmse_train_raw, 3), "\n")
```

```
##   RMSE Train: 1.134
```

```
cat("  RMSE Test: ", round(rmse_test_raw, 3), "\n\n")
```

```
##   RMSE Test:  1.183
```

```
cat("Baseline Predictor Model:\n")
```

```
## Baseline Predictor Model:
```

```
cat("  RMSE Train:", round(rmse_train_base, 3), "\n")
```

```
##   RMSE Train: 0.843
```

```r
cat("  RMSE Test: ", round(rmse_test_base, 3), "\n")
```

```
##   RMSE Test:  1.021
```

```r
result <- (1- (rmse_test_base/rmse_test_raw)) * 100

cat("  RMSE Test Increased or Decreased: ", round(result, 3), "\n")
```

```
##   RMSE Test Increased or Decreased:  13.702
```

The Raw Average Model uses the overall average rating to predict user ratings, resulting in:

`Training RMSE: 1.134`

`Testing RMSE: 1.183`

The Baseline Predictor Model, which adjusts predictions by adding user and restaurant biases to the global average, resulted in:

`Training RMSE: 0.843 (lower than raw average)`

`Testing RMSE: 1.021 (lower than raw average)`

The Baseline Predictor Model improves test RMSE by about 13.7% compared to the Raw Average Model, indicating better generalization on unseen data. We incorporated regularization into our model to reduce over-fitting of the data and this process actually decreased our baseline predictor model's test and training RMSE since we shrunk extreme bias values toward the global average.

RMSE is used to measure how close our predictions are compared to teh actual ratings. The lower RMSE of the baseline predictor confirms it outperforms the raw average model, indicating meaningful improvement. Sushi Spot and Curry Corner have negative item biases, suggesting they are consistently rated below average by users. Burger Barn and Taco Town on the other hand received higher average ratings from users which indicates customer satisfaction.

The users Alice and Ella will provide higher ratings compared to others on average which are positive user biases. The negative user biases would be David and Cindy.