DATA-612 Recommender Systems

Project 1: Global Baseline Predictors and RMSE

Author: Bikash Bhowmik, Rupendra Shrestha

08 Jun 2025

Contents

1 Instruction **1**

2 Introduction **2**

3 Data Processing **2**

4 Train Dataset **3**

5 Test Dataset **3**

6 Create a User-Item Matrix **4**

7 Calculate User Bias **5**

8 Calculate Item Bias for User & Item **6**

9 Calculate the Baseline Predictor **7**

10 Calculate RMSE Calculation **7**

11 Summary **8**

12 Future Improvements **8**

1 Instruction

We'll attempt to predict ratings with very little information. We'll first look at just raw averages across all (training dataset) users. We'll then account for "bias" by normalizing across users and across items. We'll be working with ratings in a user-item matrix, where each rating may be (1) assigned to a training dataset, (2) assigned to a test dataset, or (3) missing.

- 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."
- Find a dataset, or build out your own toy dataset. As a minimum requirement for complexity, please include numeric ratings for at least five users, across at least five items, with some missing data.
- Load your data into (for example) an R or pandas dataframe, a Python dictionary or list of lists, (or another data structure of your choosing). From there, create a user-item matrix.
- If you choose to work with a large dataset, you're encouraged to also create a small, relatively dense "user-item" matrix as a subset so that you can hand-verify your calculations.
- Break your ratings into separate training and test datasets.
- Using your training data, calculate the raw average (mean) rating for every user-item combination.
- Calculate the RMSE for raw average for both your training data and your test data.
- Using your training data, calculate the bias for each user and each item.
- From the raw average, and the appropriate user and item biases, calculate the baseline predictors for every user-item combination.
- Calculate the RMSE for the baseline predictors for both your training data and your test data.
- Summarize your results.

2 Introduction

This project focuses on building a simple recommender system using global baseline predictors on a simulated dataset of book ratings by users.

The baseline recommender estimates unknown ratings by considering three components:

- The overall average rating (global mean),
- A user bias — the tendency of a user to rate higher or lower than average,
- An item bias — how a book tends to be rated across all users.

Mathematically:

Baseline Estimate $= \mu + b_u + b_i$

Where: - μ = global average rating, - b_u = bias of user u, - b_i = bias of item i.

Baseline recommenders are often the first step in building scalable systems. They are interpretable, efficient, and lay the groundwork for more complex techniques like matrix factorization or deep learning.

This project aims to:

- Simulate sparse user-item rating data,
- Predict unknown ratings using baseline predictors,
- Evaluate accuracy using Root Mean Square Error (RMSE).

3 Data Processing

We create a sample dataset representing ratings given by 10 users to 10 books. Ratings are randomly generated on a scale of 1 to 5, with some values intentionally set as missing (NA) to simulate real-world sparsity. The dataset is then split into training and test sets, ensuring that each contains a mix of observed and missing ratings. Additional missing values are introduced to better reflect incomplete user behavior. Finally, meaningful row and column labels are assigned to represent user and book names, preparing the data for visualization and analysis.

```r
# random sample of 100 ratings
set.seed(612)
df <- matrix(sample(1:5, 100, replace = TRUE), nrow = 10)

# sample dataset for splitting
split_df <- sample(1:length(df), 10, replace = FALSE)

# split the data into train_dfing dataset
train_df <- df
train_df[split_df] <- NA

# split the data into train_dfing dataset
test_df <- df
test_df[-split_df] <- NA

# create some missing values for both dataset
set.seed(612)
missing_df <- sample(1:length(df), 10, replace = FALSE)
df[missing_df] <- NA
train_df[missing_df] <- NA
test_df[missing_df] <- NA

# name of the books
users <- c("User_1","User_2","User_3","User_4","User_5","User_6","User_7","User_8","User_9","User_10")
rownames(df) <- users
rownames(train_df) <- users
rownames(test_df) <- users

# name of the users
colname <- c("Book_1","Book_2","Book_3","Book_4","Book_5","Book_6","Book_7","Book_8","Book_9","Book_10")
colnames(df) <- colname
colnames(train_df) <- colname
colnames(test_df) <- colname

# print the matrix
kable(df, caption = "User-Book Ratings", booktabs = TRUE) %>%
  kable_styling(
    latex_options = c("HOLD_position", "striped", "condensed"),
    position = "center"
  ) %>%
  row_spec(0, bold = TRUE, color = "white", background = "#ea7872")
```

Table 1: User-Book Ratings

	Book_1	Book_2	Book_3	Book_4	Book_5	Book_6	Book_7	Book_8	Book_9	Book_10
User_1	4	1	2	4	2	1	2	5	NA	2
User_2	4	5	4	4	3	4	4	2	5	1
User_3	5	NA	4	5	2	1	NA	1	3	1
User_4	4	NA	NA	2	4	5	3	4	NA	5
User_5	1	2	2	3	5	5	4	1	4	NA
User_6	5	5	NA	3	3	1	4	4	5	NA
User_7	2	2	4	3	5	4	3	5	1	5
User_8	3	4	5	2	2	5	4	2	5	4
User_9	1	1	5	2	2	2	5	4	4	NA
User_10	3	1	5	1	2	5	2	4	3	4

4 Train Dataset

This section presents the training dataset, which is derived from the full user-book rating matrix by randomly selecting a subset of entries to retain while setting the rest to missing (NA). These observed values will be used to calculate the global average, user and item biases, and to train our prediction model. The remaining missing values simulate unknown ratings that the model will attempt to estimate.

```r
kable(train_df, caption = "Training Dataset", booktabs = TRUE) %>%
  kable_styling(
    latex_options = c("HOLD_position", "striped", "condensed"),
    position = "center"
  ) %>%
  row_spec(0, bold = TRUE, color = "white", background = "#ea7872")
```

Table 2: Training Dataset

	Book_1	Book_2	Book_3	Book_4	Book_5	Book_6	Book_7	Book_8	Book_9	Book_10
User_1	NA	1	2	4	2	1	2	5	NA	2
User_2	4	NA	4	4	3	4	NA	2	5	1
User_3	NA	NA	4	5	2	NA	NA	1	3	NA
User_4	4	NA	NA	NA	4	5	3	4	NA	5
User_5	1	2	2	3	5	5	4	1	4	NA
User_6	5	5	NA	NA	3	1	4	4	NA	NA
User_7	2	2	4	3	5	4	3	5	1	5
User_8	3	4	5	2	2	5	4	2	5	4
User_9	1	1	5	2	2	2	5	4	4	NA
User_10	3	1	5	1	2	NA	2	4	3	4

5 Test Dataset

Building a test dataset

```r
kable(test_df, caption = "Test Dataset", booktabs = TRUE) %>%
  kable_styling(
    latex_options = c("HOLD_position", "striped", "condensed"),
    position = "center"
  ) %>%
  row_spec(0, bold = TRUE, color = "white", background = "#ea7872")
```

Table 3: Test Dataset

	Book_1	Book_2	Book_3	Book_4	Book_5	Book_6	Book_7	Book_8	Book_9	Book_10
User_1	4	NA	NA	NA	NA	NA	NA	NA	NA	NA
User_2	NA	5	NA	NA	NA	NA	4	NA	NA	NA
User_3	5	NA	NA	NA	NA	1	NA	NA	NA	1
User_4	NA	NA	NA	2	NA	NA	NA	NA	NA	NA
User_5	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
User_6	NA	NA	NA	3	NA	NA	NA	NA	5	NA
User_7	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
User_8	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
User_9	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
User_10	NA	NA	NA	NA	NA	5	NA	NA	NA	NA

6 Create a User-Item Matrix

In this section, we generate a user-item matrix where each missing rating is replaced with the overall average rating from the training dataset. Using the replicate function, we fill the matrix with the global mean to create a baseline prediction model. This simple approach assumes that all users rate items similarly and serves as a benchmark for evaluating more advanced methods.

```
# raw average
raw_avg <- round(mean(train_df, na.rm = TRUE), 2)

# user-item matrix for raw avearge
user_item <- matrix(replicate(100, raw_avg), 10)
rownames(user_item) <- rownames(train_df)
colnames(user_item) <- colnames(train_df)

kable(user_item, caption = "User-Item Matrix", booktabs = TRUE) %>%
  kable_styling(
    latex_options = c("HOLD_position", "striped", "condensed"),
    position = "center"
  ) %>%
  row_spec(0, bold = TRUE, color = "white", background = "#ea7872")
```

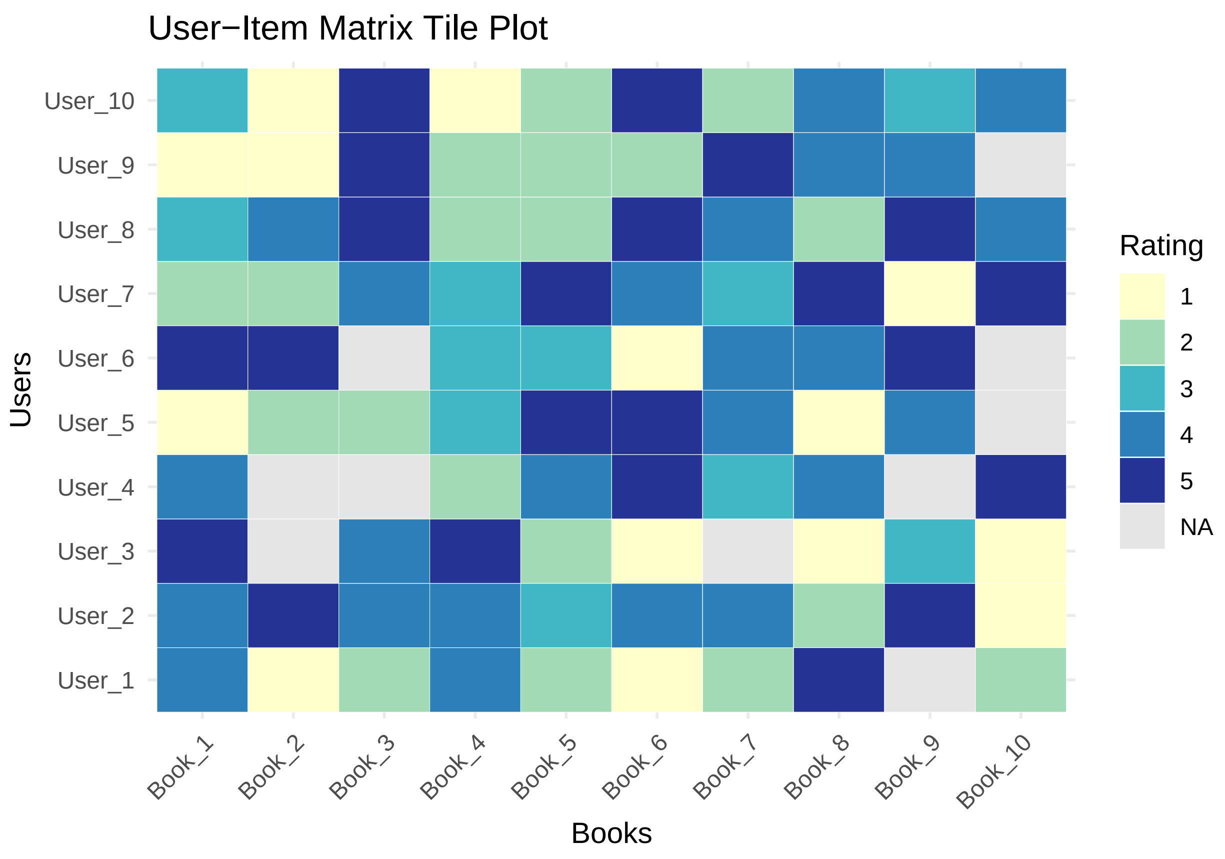
Table 4: User-Item Matrix

	Book_1	Book_2	Book_3	Book_4	Book_5	Book_6	Book_7	Book_8	Book_9	Book_10
User_1	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
User_2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
User_3	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
User_4	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
User_5	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
User_6	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
User_7	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
User_8	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
User_9	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
User_10	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2

```
# Melt the original matrix (with raw average or actual ratings)
df_long <- melt(df, varnames = c("User", "Book"), value.name = "Rating")

# Tile plot (categorical coloring)
```

```r
ggplot(df_long, aes(x = Book, y = User, fill = as.factor(Rating))) +
  geom_tile(color = "white") +
  scale_fill_brewer(palette = "YlGnBu", na.value = "gray90", name = "Rating") +
  labs(title = "User-Item Matrix Tile Plot",
       x = "Books", y = "Users") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



7 Calculate User Bias

Each user has different rating behavior—some users rate generously, others more critically. The User bias represents how much a user's average rating deviates from the global average.

- Positive user bias → this user rates items higher than average.
- Negative user bias → this user rates items lower than average.

Calculating bias for each user using row Means function.

```r
# bias for each user
user_bias <- round((rowMeans(train_df, na.rm = TRUE) - raw_avg), 2)

kable(user_bias, caption = "User Bias", booktabs = TRUE) %>%
  kable_styling(
    latex_options = c("HOLD_position", "striped", "condensed"),
    position = "center"
  ) %>%
  row_spec(0, bold = TRUE, color = "white", background = "#ea7872")
```

Table 5: User Bias

	x
User_1	-0.83
User_2	0.17
User_3	-0.20
User_4	0.97
User_5	-0.20
User_6	0.47
User_7	0.20
User_8	0.40
User_9	-0.31
User_10	-0.42

8 Calculate Item Bias for User & Item

Calculating bias for each item using colMeans function

- User Bias reflects how a particular user typically rates items compared to the global average. A positive bias means the user tends to give higher ratings than average; a negative bias indicates lower ratings.
- Item Bias reflects how a particular item (book) is generally rated. For example, a book that's consistently rated higher than the average across users will
  have a positive bias.

These biases help correct for systemic tendencies in the data, improving the accuracy of predictions.

```r
# raw average
raw_avg <- round(mean(train_df, na.rm = TRUE), 2)

# bias for each item
item_bias <- round((colMeans(train_df, na.rm = TRUE) - raw_avg), 2)

kable(item_bias, caption = "Item Bias", booktabs = TRUE) %>%
  kable_styling(
    latex_options = c("HOLD_position", "striped", "condensed"),
    position = "center"
  ) %>%
  row_spec(0, bold = TRUE, color = "white", background = "#ea7872")
```

Table 6: Item Bias

	x
Book_1	-0.33
Book_2	-0.91
Book_3	0.67
Book_4	-0.20
Book_5	-0.20
Book_6	0.17
Book_7	0.17
Book_8	0.00
Book_9	0.37
Book_10	0.30

The baseline recommender system implemented here is based on a global baseline predictor model. It estimates a user's rating of an item by combining three components:

- **Global Average ( ):** the overall average rating in the training dataset.
- **User Bias (b):** how much a user's average rating deviates from the global average.
- **Item Bias (b):** how much an item's average rating deviates from the global average.

The formula used to compute the predicted rating is:

$$\hat{r}_{ui} = \mu + b_u + b_i$$

Where: - $\hat{r}_{ui}$: predicted rating for user u and item i - μ: global average rating - b_u: bias of user u - b_i: bias of item i

This simple model improves prediction accuracy by adjusting for consistent user and item tendencies. It's a foundational approach used in many recommender systems as a baseline for comparison.

9 Calculate the Baseline Predictor

Calculating baseline predictors for every user-item combination

The baseline predicted rating for a user u and item i is:

$$\hat{r}_{ui} = \mu + b_u + b_i$$

Where:
- μ: global average rating
- b_u: user bias
- b_i: item bias

```r
# calculate every user-item biases combination
com <- apply(expand.grid((as_tibble(user_bias))[[1]], (as_tibble(item_bias))[[1]]), 1, sum)

# baseline predictors for every user-item combination
baseline <- (replicate(100, raw_avg) + com)
baseline <- matrix(baseline, 10)

rownames(baseline) <- rownames(train_df)
colnames(baseline) <- colnames(train_df)

kable(
  baseline,
  caption = "Baseline Predictors"
) %>%
  kable_styling(
    latex_options = c("striped", "hold_position"),
    position = "center"
  ) %>%
  row_spec(0, bold = TRUE, color = "white", background = "#ea7872")
```

Table 7: Baseline Predictors

	Book__1	Book__2	Book__3	Book__4	Book__5	Book__6	Book__7	Book__8	Book__9	Book__10
User__1	2.04	1.46	3.04	2.17	2.17	2.54	2.54	2.37	2.74	2.67
User__2	3.04	2.46	4.04	3.17	3.17	3.54	3.54	3.37	3.74	3.67
User__3	2.67	2.09	3.67	2.80	2.80	3.17	3.17	3.00	3.37	3.30
User__4	3.84	3.26	4.84	3.97	3.97	4.34	4.34	4.17	4.54	4.47
User__5	2.67	2.09	3.67	2.80	2.80	3.17	3.17	3.00	3.37	3.30
User__6	3.34	2.76	4.34	3.47	3.47	3.84	3.84	3.67	4.04	3.97
User__7	3.07	2.49	4.07	3.20	3.20	3.57	3.57	3.40	3.77	3.70
User__8	3.27	2.69	4.27	3.40	3.40	3.77	3.77	3.60	3.97	3.90
User__9	2.56	1.98	3.56	2.69	2.69	3.06	3.06	2.89	3.26	3.19
User__10	2.45	1.87	3.45	2.58	2.58	2.95	2.95	2.78	3.15	3.08

10 Calculate RMSE Calculation

Calculate RMSE Calculation for Baseline Predictors

Calculating RMSE for baseline predictors for training and testing data

Root Mean Square Error (RMSE) measures the average difference between predicted and actual ratings. A lower RMSE indicates better prediction accuracy.

round((sqrt(mean((x - y)^2, na.rm = TRUE))), 2)

```r
# function to calculate RMSE
rmse <- function(x, y) {
  round((sqrt(mean((x - y)^2, na.rm = TRUE))), 2)
}

# rmse for train_df dataset
rmse1 <- rmse(train_df, raw_avg)

# rmse for test_df dataset
rmse2 <- rmse(test_df, raw_avg)

# rmse for baseline predictors
rmse3 <- rmse(test_df, baseline)
rmse4 <- rmse(train_df, baseline)
```

11 Summary

In this project, we explored the foundation concept of global baseline predictors in recommender systems using a simulated user-book rating dataset. We began by constructing a user-item matrix and splitting the data into training and test sets, introducing missing values to mimic real-world sparsity. We calculated the global average rating and used it to generate a basic prediction model. Then, we incorporated user and item biases to enhance prediction accuracy through baseline predictors. RMSE scores were computed for both raw averages and bias-adjusted predictions, demonstrating how accounting for individual user and item tendencies improves performance. This simple yet powerful approach lays the groundwork for more sophisticated recommendation algorithms.

```r
# summary of the result
kable(
  cbind(rmse1, rmse2, rmse3, rmse4),
  col.names = rep(c("Train", "Test"), 2),
  caption = "Summary",
  booktabs = TRUE
) %>%
  add_header_above(c("Raw Average" = 2, "Baseline Predictor" = 2)) %>%
  kable_styling(
    latex_options = c("striped", "bordered", "hold_position"),
    position = "center"
  ) %>%
  row_spec(0, bold = TRUE, color = "white", background = "#ea7872")
```

Table 8: Summary

Raw Average		Baseline Predictor	
Train	**Test**	**Train**	**Test**
1.4	1.59	1.88	1.25

Train RMSE Higher Than Test RMSE

In this case, it may be due to:

- The small dataset: there's a high chance of random variation.
- Some high-error values might have been retained in the train set.
- Sparsity: fewer known values to learn from can cause the model to underperform during training.

Although test RMSE often exceeds train RMSE, with small-scale or toy datasets, the reverse can occur because of sampling noise or overfitting to sparse data.

12 Future Improvements

While baseline predictors are simple and effective for sparse data, the model can be significantly enhanced with the following techniques:

1. Regularization

Add penalties to large user/item biases when there's limited data. This helps avoid overfitting: - Example: shrink user biases if a user rated only 1-2 books.

2. Matrix Factorization

Learn latent features of users and items to predict ratings (e.g., via SVD or ALS). It's the foundation of collaborative filtering.

3. Hybrid Recommenders

Use additional content features like book genres, user demographics, or textual reviews alongside ratings.

4. Neural Approaches

Deep learning models can capture non-linear patterns and are especially useful with large datasets.

Implementing these techniques will improve scalability and prediction power, particularly in real-world systems.