**CS5610**

**ADVANCED R FOR DATA SCIENCE**

**PROJECT REPORT**

**INSTRUCTOR**

**Dr. Wassnaa Al-Mawee**

A logo with brown and yellow letters

Description automatically generated

**SARAT CHANDRA GOPINEEDI (WIN: 976330583)**

**SRIRAM KONERU (WIN: 426755762)**

**VISHNU KODITHALA (WIN: 074925479)**

**SREEJA BATCHU (WIN: 947328825)**

**Department of Statistics**

**Western Michigan University**

**Kalamazoo MI**

**49008-5278 USA (269) 387-1420**

1. **MOTIVATION AND OVERVIEW**

The objective is to create a Shiny application that provides personalized grant recommendations to users based on their preferences and historical ratings. The primary goal is to help users identify funding opportunities that match their interests and background, ultimately facilitating better access to relevant grants.

**MOTIVATION**

The process of identifying suitable grants can be time-consuming and challenging, especially given the vast number of opportunities available across different industries and sectors. By providing personalized recommendations, this application aims to simplify and expedite the search for appropriate funding opportunities. Utilizing user data, including preferences and historical ratings, allows for a more targeted approach to recommending grants. This can improve the chances of users finding opportunities that align with their goals and needs. By offering a tailored list of recommendations, the application can help users connect with grant providers in industries that are relevant to them. This may increase the chances of successful grant applications and overall user satisfaction.

**PROJECT GOALS**

We seek to deliver an efficient and personalized grant recommendation system, leveraging data and machine learning techniques to help users identify and connect with funding opportunities that match their interests and expertise. Through this application, the goal is to simplify the grant-seeking process and improve access to relevant opportunities for users. We utilize a user-based collaborative filtering (UBCF) model to generate personalized grant recommendations based on users historical interactions and preferences. The app presents the top 10 recommended grants to users based on their preferences and industry choice. This ensures that users receive a concise and relevant list of opportunities.

1. **RELATED WORK**

The project is inspired by several concepts and existing applications in the domains of recommendation systems and machine learning. Some of the key areas and works that influenced the project are

**RESEARCH PAPERS**

Awan, M. J., Khan, R. A., Nobanee, H., Yasin, A., Anwar, S. M., Naseem, U., & Singh, V. P. (2021). A recommendation engine for predicting movie ratings using a big data approach. *Electronics*, *10*(10), 1215.

Biswas, A., Vineeth, K. S., & Jain, A. (2020, January). Development of product recommendation engine by collaborative filtering and association rule mining using machine learning algorithms. In *2020 Fourth International Conference on Inventive Systems and Control (ICISC)* (pp. 272-277). IEEE.

**SHINY APPLICATIONS**

The Shiny app "Movie Recommendations with Shiny" presented by RStudio serves as a notable example of how recommender systems can be effectively integrated into a Shiny application, providing a basis for the design and structure of our project. Shiny applications demonstrate the use of dynamic user interfaces and server logic, motivating the creation of an application with similar capabilities.

**RECOMMENDATION PLATFORMS**

Platforms such as Amazon, Spotify offer personalized recommendations. These platforms provided inspiration for the concept of matching users with funding opportunities based on their preferences.

**CLASSROOM DISCUSSIONS AND COURSEWORK**

Concepts and techniques covered in advanced R programming courses, such as data manipulation with dplyr and working with data frames, influenced the approach to data preprocessing and analysis in the project. Classroom discussions around the challenges and best practices in designing recommendation systems also provided insights that shaped the project's goals and implementation.

1. **INITIAL QUESTIONS**

**How can we provide personalized grant recommendations to users based on their interests and preferences?**

The goal was to develop a recommendation system that would use collaborative filtering to match users with grants that align with their interests and industry preferences.

**What is the most effective approach for implementing a user-based collaborative filtering (UBCF) model in a Shiny application?**

We wanted to explore how to use the recommenderlab package to implement a UBCF model in a Shiny app and determine the best practices for making accurate and relevant recommendations.

**How can we create a user-friendly and intuitive interface for interacting with the recommendation system?**

We sought to design a Shiny app that would allow users to easily authenticate themselves, input their preferences, and receive recommendations without confusion or complexity.

**How can we ensure that the recommendations are relevant and valuable to the users?**

Beyond just providing recommendations, we wanted to ensure that the top recommended grants aligned with the users interests and preferences.

1. **DATA**

The data consists of two main datasets, one containing information about grant opportunities and another containing user ratings for grants.

**GRANT OPPORTUNITIES DATA**

**Features**:

* **grant\_id**: A unique identifier for each grant opportunity.
* **grant\_name**: The name of the grant opportunity.
* **industry**: The industry or sector to which the grant opportunity belongs.

The data is imported from a CSV file using the read.csv() function. Column names are adjusted to make them more descriptive and consistent.

**USER RATINGS DATA**

**Features**:

* **user\_id**: A unique identifier for each user.
* **grant\_name**: The name of the grant for which the rating is provided.
* **rating**: The user's rating for the grant (e.g., -1 for dislike, 0 for neutral, 1 for like).

A user DataFrame was created with a set number of users (**n\_users = 100**). Random ratings are generated using the sample() function, with probabilities assigned to different rating values. A ratings DataFrame (**ratings\_df**) is created by combining user IDs, grant names, and ratings. The ratings DataFrame is converted to a realRatingMatrix for use with the **recommenderlab** package.

**SPLITTING DATA FOR EVALUATION**

An evaluation scheme is set up using evaluationScheme() from recommenderlab, which splits the data into training and test sets (70% training, 30% test). The training data is used to build the recommendation model, while the test data is used to evaluate its performance. When generating recommendations, the user predictions are joined with the grant opportunities data to filter recommendations based on the user's preferred industry. Data is sorted by predicted ratings to present the top recommendations.

Overall, data import and wrangling involved importing data from CSV files, creating synthetic user ratings, converting data into appropriate formats, and splitting data for model evaluation. Proper data preparation was crucial for the successful implementation and evaluation of the recommendation system.

1. **EXPLORATORY DATA ANALYSIS**

**VISUALIZATIONS AND STATISTICAL SUMMARIES**

We used a bar chart showing the distribution of grants across different industries to help identify which industries have more or fewer opportunities. Also, a plot of grant occurrences (grant names) highlighted popular or less common grants in the data. We also used basic statistics such as count, mean, and median for each industry and observe any imbalances in grant availability across industries.

A bar chart or histogram showed us the frequency of different ratings (-1, 0, 1) across all the users which gave us insight into the overall user sentiment toward grants. Plotting the number of ratings per grant revealed popular or less-rated grants. Observing the relationship between user preferences and industries provided us insight into the distribution of user interests across industries.

**DATA ANALYSIS CONCLUSIONS**

Observations of imbalances in industry distributions or rating frequencies influenced in the model and analysis choices. Identifying varying user engagement levels helped us in addressing potential issues with sparse user data. The patterns in user-grant ratings guided us in the choice of recommendation algorithms and methods.

Assessing correlations and similarities between user preferences and grant characteristics guided us in the selection of appropriate filtering methods (user-based vs. item-based collaborative filtering).

The exploratory data analysis provided the foundation for the statistical analyses and modelling choices used in the project. It guided the decision to use a user-based collaborative filtering approach. Additionally, the Exploratory Data Analysis highlighted the importance of industry-specific recommendations, shaping the design of the application to prioritize user preferences and enhance the relevance of recommendations.

1. **DATA ANALYSIS**

We applied a user-based collaborative filtering (UBCF) model using the recommenderlab package to generate grant recommendations based on user preferences and historical ratings.

**USER BASED COLLABORATIVE FILTERING (UBCF)**

UBCF is a type of recommendation system that leverages similarities between users to predict their preferences. In this approach, users who have similar preferences are identified, and the preferences of similar users are used to make predictions for a target user. We implemented the UBCF model using the Recommender() function from recommenderlab with parameters to normalize the data and optimize the model. The model was trained on the training data (70% of the ratings data) and used to predict ratings for the test data.

The UBCF method was chosen for its simplicity, effectiveness, and flexibility in providing personalized recommendations based on user preferences and historical ratings. While other methods such as item-based collaborative filtering and matrix factorization were considered, they were ultimately not used due to the ease of use and interpretability of UBCF, as well as its ability to align with the project's goals of providing personalized recommendations to users based on their preferences.

1. **SUMMARY**

**CONCLUSIONS**

The user ratings data showed a range of preferences towards different grants, with most ratings being neutral (0) and a smaller proportion of likes (1) and dislikes (-1). The grant opportunities data displayed variation in the availability of grants across different industries, with some industries having more opportunities than others.

By applying a user-based collaborative filtering (UBCF) model using the **recommenderlab** package, we were able to generate personalized grant recommendations based on user preferences and similarities with other users.

* **Approach**: UBCF model predicted user preferences for grants and filtered recommendations based on the user's preferred industry.
* **Justification**: The model's effectiveness is justified by its ability to generate personalized recommendations based on user similarities and the approach's widespread use in recommendation systems.

The Shiny application allowed users to authenticate themselves and specify their preferred industry to receive tailored recommendations.

* **Approach**: By providing a simple interface for inputting user IDs and selecting preferred industries, users could easily navigate the app and view recommended grants.
* **Justification**: This streamlined interface maximized user engagement and made the application accessible to users with different levels of experience.

The application presented the top 10 recommended grants based on the user's preferences and predicted ratings.

* **Approach**: The recommendations were ranked in descending order of predicted rating, ensuring users saw the most relevant and highest-rated grants first.
* **Justification**: This ranking method aligned with the goal of providing users with the most promising grant opportunities.

**LIMITATIONS**

The presence of sparse user-grant interactions may limit the model's accuracy for certain users or grants. This could lead to less reliable recommendations for users with minimal historical data.

Users may have different rating scales (e.g., some may rate grants more harshly or leniently), which can affect the model's predictions. Normalization methods were used to mitigate this bias, but it could still impact results.

Some industries may be overrepresented or underrepresented in the grant opportunities data, affecting the range and quality of recommendations available in certain industries.

Overall, the project successfully implemented a user-based collaborative filtering model to generate personalized grant recommendations in a Shiny application. While the model effectively provided tailored recommendations based on user preferences, there were some limitations due to sparse data and potential biases in user ratings.

Despite these limitations, the application provides a valuable tool for helping users discover grant opportunities that align with their interests and industry preferences. Future improvements could involve incorporating more comprehensive data sources and exploring hybrid recommendation models to enhance the quality of recommendations.

1. **CODE**

# Load necessary libraries

library(shiny)

library(dplyr)

library(recommenderlab)

# Load the dataset from a CSV file

# Modify the file path according to your file location

grant\_opportunities\_df <- read.csv("D:/Data/Class Materials/Advanced R Programming/R Project/TEST Funding Opportunities.csv")

colnames(grant\_opportunities\_df) <- c("grant\_id", "grant\_name", "industry")

# Create a user DataFrame

n\_users <- 100

users\_df <- data.frame(user\_id = 1:n\_users)

# Create random ratings

set.seed(123) # For reproducibility

ratings <- sample(c(-1, 0, 1), size = n\_users \* nrow(grant\_opportunities\_df), replace = TRUE, prob = c(0.1, 0.8, 0.1))

# Create a ratings DataFrame

ratings\_df <- data.frame(# Load necessary libraries

library(shiny)

library(dplyr)

library(recommenderlab)

# Load the dataset from a CSV file

# Modify the file path according to your file location

grant\_opportunities\_df <- read.csv("D:/Data/Class Materials/Advanced R Programming/R Project/TEST Funding Opportunities.csv")

colnames(grant\_opportunities\_df) <- c("grant\_id", "grant\_name", "industry")

# Create a user DataFrame

n\_users <- 100

users\_df <- data.frame(user\_id = 1:n\_users)

# Create random ratings

set.seed(123) # For reproducibility

ratings <- sample(c(-1, 0, 1), size = n\_users \* nrow(grant\_opportunities\_df), replace = TRUE, prob = c(0.1, 0.8, 0.1))

# Create a ratings DataFrame

ratings\_df <- data.frame(

  user\_id = rep(users\_df$user\_id, each = nrow(grant\_opportunities\_df)),

  grant\_name = rep(grant\_opportunities\_df$grant\_name, n\_users),

  rating = ratings

)

# Convert the ratings DataFrame to a realRatingMatrix

ratings\_matrix <- as(ratings\_df, "realRatingMatrix")

# Create an evaluation scheme for splitting data

split\_data <- evaluationScheme(ratings\_matrix, method = "split", train = 0.7, given = -1, goodRating = 1)

# Get the training and test data

train\_data <- getData(split\_data, "train")

test\_data <- getData(split\_data, "unknown")

# Create a UBCF model

model\_ubcf <- Recommender(train\_data, method = "UBCF", param = list(normalize = "Z-score"))

# Define the Shiny UI

ui <- fluidPage(

  tabsetPanel(

    id = "main\_tabs",

    tabPanel("Authentication",

      fluidPage(

        titlePanel("User Authentication"),

        sidebarLayout(

          sidebarPanel(

            numericInput("user\_id", "User ID (1-100):", value = NULL, min = 1, max = 100),

            selectInput("industry", "Preferred Industry:", choices = unique(grant\_opportunities\_df$industry)),

            actionButton("authenticate", "Authenticate")

          ),

          mainPanel(

            textOutput("auth\_message")

          )

        )

      )

    ),

    tabPanel("Recommendations",

      fluidPage(

        titlePanel("Top 10 Recommended Grants"),

        sidebarLayout(

          sidebarPanel(),

          mainPanel(

            tableOutput("recommendations\_table")

          )

        )

      )

    )

  )

)

# Define the Shiny server

server <- function(input, output, session) {

  observeEvent(input$authenticate, {

    # Validate user ID and industry inputs

    if (is.null(input$user\_id) || input$industry == "") {

      output$auth\_message <- renderText("Please enter a valid user ID and preferred industry.")

      return()

    }

    # Redirect to the recommendations tab

    updateTabsetPanel(session, "main\_tabs", selected = "Recommendations")

  })

  # Generate recommendations for the given user ID and industry preference

  output$recommendations\_table <- renderTable({

    # Get user ID and preferred industry from input

    user\_id <- input$user\_id

    preferred\_industry <- input$industry

    # Make predictions using the UBCF model

    predicted\_ratings <- predict(model\_ubcf, test\_data, type = "ratings")

    # Convert predicted ratings to a matrix

    predicted\_matrix <- as(predicted\_ratings, "matrix")

    # Extract predictions for the specific user ID

    user\_predictions <- predicted\_matrix

userid,������,

    # Create a data frame of predictions and grant names

    recommendations <- data.frame(

      grant\_name = names(user\_predictions),

      predicted\_rating = user\_predictions

    )

    # Join with grant opportunities data frame to get industry information

    recommendations <- recommendations %>%

      inner\_join(grant\_opportunities\_df, by = "grant\_name") %>%

      filter(industry == preferred\_industry) %>%

      arrange(desc(predicted\_rating))

    # Select the top 10 recommendations

    top\_10\_recommendations <- recommendations

1:10,1:10,

    # Display the top 10 recommended grants in a table format

    top\_10\_recommendations %>%

      select(grant\_name, predicted\_rating) %>%

      rename("Grant Title" = grant\_name, "Predicted Rating" = predicted\_rating)

  })

}

# Run the Shiny app

shinyApp(ui = ui, server = server)

  user\_id = rep(users\_df$user\_id, each = nrow(grant\_opportunities\_df)),

  grant\_name = rep(grant\_opportunities\_df$grant\_name, n\_users),

  rating = ratings

)

# Convert the ratings DataFrame to a realRatingMatrix

ratings\_matrix <- as(ratings\_df, "realRatingMatrix")

# Create an evaluation scheme for splitting data

split\_data <- evaluationScheme(ratings\_matrix, method = "split", train = 0.7, given = -1, goodRating = 1)

# Get the training and test data

train\_data <- getData(split\_data, "train")

test\_data <- getData(split\_data, "unknown")

# Create a UBCF model

model\_ubcf <- Recommender(train\_data, method = "UBCF", param = list(normalize = "Z-score"))

# Define the Shiny UI

ui <- fluidPage(

  tabsetPanel(

    id = "main\_tabs",

    tabPanel("Authentication",

      fluidPage(

        titlePanel("User Authentication"),

        sidebarLayout(

          sidebarPanel(

            numericInput("user\_id", "User ID (1-100):", value = NULL, min = 1, max = 100),

            selectInput("industry", "Preferred Industry:", choices = unique(grant\_opportunities\_df$industry)),

            actionButton("authenticate", "Authenticate")

          ),

          mainPanel(

            textOutput("auth\_message")

          )

        )

      )

    ),

    tabPanel("Recommendations",

      fluidPage(

        titlePanel("Top 10 Recommended Grants"),

        sidebarLayout(

          sidebarPanel(),

          mainPanel(

            tableOutput("recommendations\_table")

          )

        )

      )

    )

  )

)

# Define the Shiny server

server <- function(input, output, session) {

  observeEvent(input$authenticate, {

    # Validate user ID and industry inputs

    if (is.null(input$user\_id) || input$industry == "") {

      output$auth\_message <- renderText("Please enter a valid user ID and preferred industry.")

      return()

    }

    # Redirect to the recommendations tab

    updateTabsetPanel(session, "main\_tabs", selected = "Recommendations")

  })

  # Generate recommendations for the given user ID and industry preference

  output$recommendations\_table <- renderTable({

    # Get user ID and preferred industry from input

    user\_id <- input$user\_id

    preferred\_industry <- input$industry

    # Make predictions using the UBCF model

    predicted\_ratings <- predict(model\_ubcf, test\_data, type = "ratings")

    # Convert predicted ratings to a matrix

    predicted\_matrix <- as(predicted\_ratings, "matrix")

    # Extract predictions for the specific user ID

    user\_predictions <- predicted\_matrix

userid,������,

    # Create a data frame of predictions and grant names

    recommendations <- data.frame(

      grant\_name = names(user\_predictions),

      predicted\_rating = user\_predictions

    )

    # Join with grant opportunities data frame to get industry information

    recommendations <- recommendations %>%

      inner\_join(grant\_opportunities\_df, by = "grant\_name") %>%

      filter(industry == preferred\_industry) %>%

      arrange(desc(predicted\_rating))

    # Select the top 10 recommendations

    top\_10\_recommendations <- recommendations

1:10,1:10,

    # Display the top 10 recommended grants in a table format

    top\_10\_recommendations %>%

      select(grant\_name, predicted\_rating) %>%

      rename("Grant Title" = grant\_name, "Predicted Rating" = predicted\_rating)

  })

}

# Run the Shiny app

shinyApp(ui = ui, server = server)