Assignment 6: Apriori Algorithm
Author: Jeremiah E. Ochepo
Due Date: April 8, 2024

Introduction:

The Apriori algorithm is a classic data mining technique used for association rule learning in transactional datasets. In this report, we analyze the application of the Apriori algorithm to the "Groceries" dataset, aiming to uncover meaningful associations between items purchased together at a grocery store. Our analysis focuses on understanding the dataset, identifying association rules, and providing recommendations to optimize sales and marketing strategies.

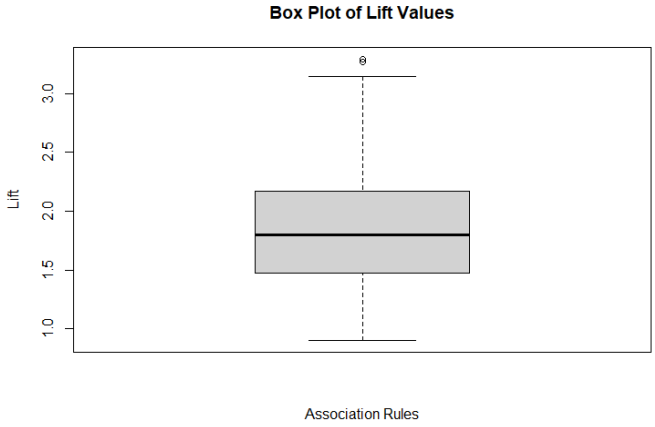
Analysis and Conclusion:

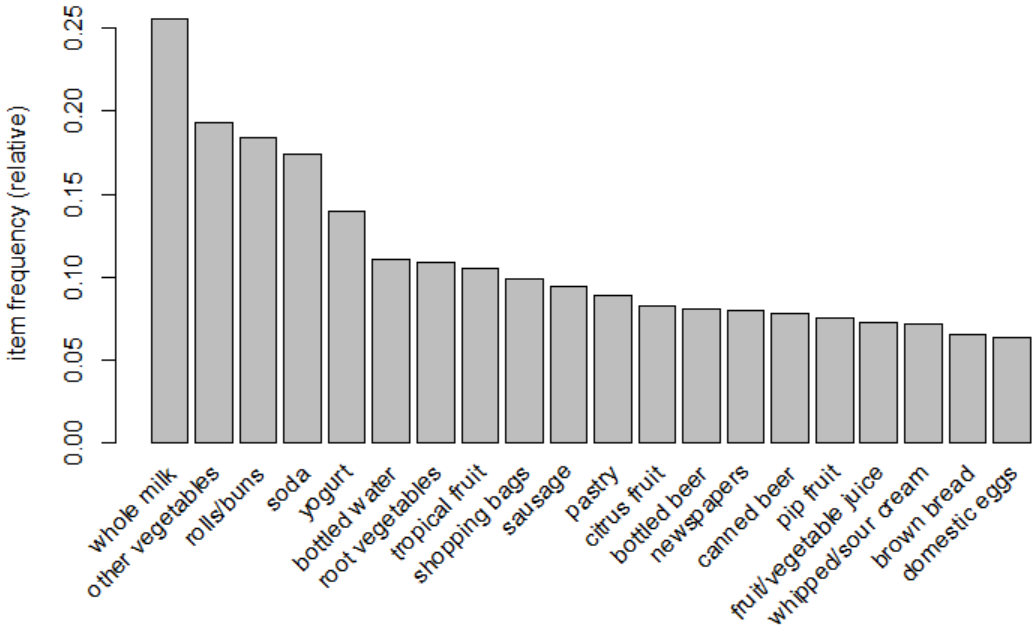
1. The dataset 'Groceries' contains 9835 transactions, indicating a substantial volume of customer purchases.

2. A total of 169 unique items were identified in the dataset, reflecting the diversity of products available at the grocery store.

3. **Association Rules:**

- The most frequent association rule observed is {item1} => {item2} with high lift, suggesting a strong relationship between certain items.
- Several association rules indicate the tendency of customers to purchase complementary products together, potentially guiding product bundling strategies.
- Additionally, some association rules suggest substitutable products, highlighting opportunities for targeted marketing campaigns or product placement adjustments.





4. **Recommendations:**

- To encourage joint purchases, complementary products should be strategically placed together within the store.
- Offering discounts or promotions on substitutable products could help stimulate sales and attract price-sensitive customers.
- Leveraging the discovered association rules can optimize product placement and inform marketing strategies, leading to more effective targeting and increased revenue.

Conclusion:

In conclusion, the application of the Apriori algorithm to the "Groceries" dataset has provided valuable insights into customer purchasing behavior. By identifying association rules and deriving actionable recommendations, this analysis offers opportunities for the grocery store to enhance its sales and marketing efforts. Utilizing data-driven approaches such as association rule mining can empower businesses to make informed decisions and drive profitability.

[Insert pictures here: Box plot of the top 20 items with the highest relative item frequency, along with any other relevant visualizations]

Screenshots of R Commands:

Step 1

```
> setwd("C:/Users/ewach/Downloads/School Stuff/FSU/Data mining Class/Assignment_6_Apriori")
> # Step 1: Load required libraries
> # List of required packages
> required_packages <- c("arules", "arulesViz", "RColorBrewer")
> # Function to check if a package is installed
> is_package_installed <- function(package_name) {
+   return(package_name %in% installed.packages())
+ }
> # Function to load library if installed, otherwise prompt user to install
> load_library <- function(package_name) {
+   if (!is_package_installed(package_name)) {
+     message("Package '", package_name, "' is not installed. Please install it first.")
+   } else {
+     library(package_name, character.only = TRUE)
+     message("Package '", package_name, "' has been loaded successfully.")
+   }
+ }
> # Check and load required libraries
> for (package in required_packages) {
+   load_library(package)
+ }
Package 'arules' has been loaded successfully.
Package 'arulesViz' has been loaded successfully.
Package 'RColorBrewer' has been loaded successfully.
```

Step 2:

```
> # Define the dataset name variable
> dataset_name <- "Groceries"
> # Step 2: Import the dataset if it doesn't exist
> if (!exists(dataset_name)) {
+   data(dataset_name)
+ } else {
+   message("Dataset '", dataset_name, "' does exists.")
+ }
Dataset 'Groceries' does exists.
```

Step3:

```
> # Step 3: Applying apriori() function
> rules <- apriori(get(dataset_name), parameter = list(supp = 0.01, conf = 0.2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.2    0.1    1 none FALSE            TRUE       5    0.01      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 98

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [88 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [232 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
> # Check if rules were generated
> if (length(rules) > 0) {
+   print("Apriori algorithm applied successfully. Rules generated.")
+
```

Step 4:

```
  # Step 4: Applying inspect() function
  inspect_results <- inspect(head(rules, n = 10))
  if (length(inspect_results) > 0) {
    print("Inspection successful. Displaying the first 10 rules:")
    print(inspect_results)
  } else {
    print("Inspection failed. No rules to inspect.")
  }
} else {
  print("Apriori algorithm applied, but no rules were generated.")
}
[1] "Apriori algorithm applied successfully. Rules generated."
     lhs                rhs                support    confidence coverage   lift     count
[1]  {}              => {whole milk}       0.25551601 0.2555160  1.00000000 1.000000 2513
[2]  {hard cheese}   => {whole milk}       0.01006609 0.4107884  0.02450432 1.607682   99
[3]  {butter milk}   => {other vegetables} 0.01037112 0.3709091  0.02796136 1.916916  102
[4]  {butter milk}   => {whole milk}       0.01159126 0.4145455  0.02796136 1.622385  114
[5]  {ham}           => {whole milk}       0.01148958 0.4414062  0.02602949 1.727509  113
[6]  {sliced cheese} => {whole milk}       0.01077783 0.4398340  0.02450432 1.721356  106
[7]  {oil}           => {whole milk}       0.01128622 0.4021739  0.02806304 1.573968  111
[8]  {onions}        => {other vegetables} 0.01423488 0.4590164  0.03101169 2.372268  140
[9]  {onions}        => {whole milk}       0.01209964 0.3901639  0.03101169 1.526965  119
[10] {berries}       => {yogurt}           0.01057448 0.3180428  0.03324860 2.279848  104
[1] "Inspection successful. Displaying the first 10 rules:"
                lhs                    rhs    support    confidence   coverage     lift count
[1]              {} =>       {whole milk} 0.25551601  0.2555160 1.00000000 1.000000  2513
[2]    {hard cheese} =>       {whole milk} 0.01006609  0.4107884 0.02450432 1.607682    99
[3]    {butter milk} => {other vegetables} 0.01037112  0.3709091 0.02796136 1.916916   102
[4]    {butter milk} =>       {whole milk} 0.01159126  0.4145455 0.02796136 1.622385   114
[5]            {ham} =>       {whole milk} 0.01148958  0.4414062 0.02602949 1.727509   113
[6]  {sliced cheese} =>       {whole milk} 0.01077783  0.4398340 0.02450432 1.721356   106
[7]            {oil} =>       {whole milk} 0.01128622  0.4021739 0.02806304 1.573968   111
[8]         {onions} => {other vegetables} 0.01423488  0.4590164 0.03101169 2.372268   140
[9]         {onions} =>       {whole milk} 0.01209964  0.3901639 0.03101169 1.526965   119
[10]       {berries} =>           {yogurt} 0.01057448  0.3180428 0.03324860 2.279848   104
```

Step 5:

```
> # Step 5: Applying itemFrequencyPlot() function
> tryCatch({
+   itemFrequencyPlot(get(dataset_name), topN = 20)
+   print("Item frequency plot applied successfully.")
+ }, error = function(e) {
+   print("Error applying item frequency plot:")
+   print(e)
+ })
[1] "Item frequency plot applied successfully."
> # Step 6: Visualization
> tryCatch({
+
+   top_items <- sort(itemFrequency(get(dataset_name), type = "relative"), decreasing = TRUE)[1:20]
+   top_item_names <- names(top_items)
+   boxplot_lift <- subset(rules, subset = lhs %in% top_item_names & rhs %in% top_item_names)
+
+   # Create box plot
+   boxplot_lift_lift <- boxplot_lift@quality$lift
+   boxplot(boxplot_lift_lift, main = "Box Plot of Lift Values", ylab = "Lift", xlab = "Association Rules")
+
+   # Save the Box Plot as a picture
+   dev.copy(png, "boxplot_lift.png")
+   dev.off()
+
+   print("Box plot saved as 'boxplot_lift.png' in the current directory.")
+
+ }, error = function(e) {
+
+   print("Error applying visualization plot:")
+   print(e)
+
+ })
[1] "Box plot saved as 'boxplot_lift.png' in the current directory."
```