

Student Name:

Submission Date:

DBST 667 – Data Mining

Dr. Irene Tsapara

Week 3 Individual Exercise

Deliverables: Two Files: (1) Submit this lab report with answers to all questions including output screenshots into the 'Individual Exercises Week 3' assignment folder. (2) Submit an R script that contains all commands with comments that briefly describe each commands purpose.

Grading: This exercise is worth 2% of the course grade. All questions must be answered in your own words with any paraphrased references properly cited using in-text citations and a reference list as needed. In addition, grammatical and spelling errors may affect the grade.

Part 2 – Run an exercise on the credit approval dataset you used for the week 2 exercise last week, completing this report and providing the commands, output screenshots, and discussion/interpretation as requested. Ensure that all commands are saved in this report and in an R script.

- a. **Introduction:** Describe the expected output and behavior of the Apriori method and what it will accomplish for the credit approval data. Use the knowledge gained through the week 3 tutorial and week 3 lectures. Provide a one-paragraph, masters-level response.

b. Data Pre-Processing: Load the credit approval data into R Studio using the read.csv command.

- i. What data pre-processing does the Apriori method require for the credit approval data? Include the commands you ran and the output screenshots.**

Command: >

Output:

c. Apriori Method – Default Parameters:

- i. Run the Apriori method with the default parameters and store the generated rules in a variable called ‘rules’. Include the command, the output screenshot, and provide a one-paragraph, masters-level discussion of the returned rules and the default arguments.**

Command: >

Output:

Discussion:

- ii. **Run the inspect command to display the first 10 rules. Include the command, the output screenshot, and interpretation of the returned rules and metrics.**

Command: >

Output:

Interpretation:

- d. **Apriori Method – Two Runs with Non-Default Parameters using different combinations of confidence, support, and minimum length values. For each run, specify the input parameters used (include the command, output screenshot, and discuss the rules returned) and then run the inspect command to preview the first 10 rules (include the command, output screenshot, and discuss which of the returned rules is the strongest and why):**

- i. **1st Non-Default Parameter Run:**

Command: >

Output:

Discussion:

Inspect Command: >

Output:

Discussion:

ii. 2nd Non-Default Parameter Run:

Command: >

Output:

Discussion:

Inspect Command: >

Output:

Discussion:

- iii. **How does changing the confidence, support, and minimum length values affect the returned rules? Provide a one-paragraph, masters-level response.**

- iv. What are the differences between the confidence, support, and lift metrics for identifying the strongest rules?
- e. **Apriori Method – Include only rules that have class='+' or class='-' on the right-hand side. Store the output in the variable 'rules'. (Hint: See the generating rules for the specified item sets section of the week 3 tutorial. You may need to adjust the confidence and support values). Include the command and output screenshot:**

Command: >

Output:

- i. **Run the inspect command to preview the first 10 rules. Include the command, output screenshot, a discussion identifying the strongest rules.**

Command: >

Output:

Discussion:

- ii. **What do the returned rules suggest about the credit approval decision? Even though the attribute names are abstracted, provide a discussion about how the rules with abstracted attribute names, the strongest rules, and any other factors you wish to include positively or negatively impacted the credit approval decision (i.e. class='+' and class='-'). Provide a one-paragraph, masters-level response.**

f. Apriori Method – Prune the Returned Rules:

- i. **Why do we as data analysts prune the rules returned by the Apriori method?**

- ii. Run the following commands on the variable ‘rules’ (generated from 2.e above) to find the redundant rules. Include the output screenshot from the commands and provide a discussion about the rules that were removed.

```
> rules.sorted<-sort(rules, by = "lift")
> inspect(rules.sorted)
> subset.matrix<-is.subset(rules.sorted, rules.sorted)
> subset.matrix[lower.tri(subset.matrix, diag=T)]<-NA
> redundant<-colSums(subset.matrix, na.rm=T)>=1
> which(redundant)
```

Output:

Discussion:

- iii. Run the following commands to remove the redundant rules and display the remaining rules. Include the commands, the output screenshot, and a discussion on which rules remain.

```
> rules.pruned<-rules.sorted[!redundant]
> inspect(rules.pruned)
```

Output :

Discussion :

- g. Rules Visualization – Choose any visualization method discussed in the tutorial to visualize the pruned rules in the previous step. Provide the command, output screenshot of the visualization, and a discussion about how effectively the plot represents the rules and the metrics for ranking rules.**

Command: >

Output:

Discussion:

References