Vanessa Fotso
UMGC SUMMER 2020
DATA 630

Exercise 3 : Association Rules Mining

I. Introduction

The Credit Approval dataset analyzed in this report was taken from the archives of the Machine Learning repository of the University of California, Irvine (UCI). The data concern credit card applications. It is a multivariate dataset with a mix of categorical, integer and real number attributes as characteristics, along with some missing values. The dataset includes 690 cases representing credit card applicants, and 16 variables, 15 of which representing various attributes of applicants, and one (approved variable) representing the outcome of the application. In this analysis, we will be running the Apriori algorithm on the dataset to find the correlation among the attributes by identifying which combinations of applicants' characteristics lead to a positive credit approval status.

II. Data Pre-processing: Load the Credit Approval data in RStudio. You may use the Tools menu or you may run read.csv command.

```
> # preview the data frame credit
> head(credit)
  Key Male   Age  Debt Married BankCustomer EducationLevel Ethnicity
1   1    b 30.83 0.000       u            g              w         v
2   2    a 58.67 4.460       u            g              q         h
3   3    a 24.50 0.500       u            g              q         h
4   4    b 27.83 1.540       u            g              w         v
5   5    b 20.17 5.625       u            g              w         v
6   6    b 32.08 4.000       u            g              m         v
  YearsEmployed PriorDefault Employed CreditScore DriversLicense Citizen
1          1.25            t        t           1              f       g
2          3.04            t        t           6              f       g
3          1.50            t        f           0              f       g
4          3.75            t        t           5              t       g
5          1.71            t        f           0              f       s
6          2.50            t        f           0              t       g
  Zipcode Income class
1     202      0     +
2      43    560     +
3     280    824     +
4     100      3     +
5     120      0     +
6     360      0     +
```

Figure 1: Credit Approval Data Preview

A. **What data pre-processing does the Apriori method require for Credit Approval data? Include the commands you ran, the output, and the output interpretation in the report. For each command, explain its purpose.**

The data preview in figure 1 above shows the presence of the unique identifier values (key variable). As this column is irrelevant to our analysis, we will first start by dropping this column using the following command: credit$key <- NULL

```
> # remove the irrelevant key values
> credit$Key <- NULL
> # Data preview after dropping the key column
> head(credit)
  Male   Age  Debt Married BankCustomer EducationLevel Ethnicity YearsEmploy
1    b 30.83 0.000       u            g              w         v           1.
2    a 58.67 4.460       u            g              q         h           3.
3    a 24.50 0.500       u            g              q         h           1.
4    b 27.83 1.540       u            g              w         v           3.
5    b 20.17 5.625       u            g              w         v           1.
6    b 32.08 4.000       u            g              m         v           2.
  PriorDefault Employed CreditScore DriversLicense Citizen ZipCode Income c
1            t        t           1              f       g     202      0
2            t        t           6              f       g      43    560
3            t        f           0              f       g     280    824
4            t        t           5              t       g     100      3
5            t        f           0              f       s     120      0
6            t        f           0              t       g     360      0
> |
```

Figure 2: Credit Approval Preview after Removing the Key Column

In order to conduct the Apriori rules method, all variables in the dataset should be categorical (discrete or factor). We will check each variable type in the dataset by inspecting the data structure.

```
> # Display the data structure
> str(credit)
'data.frame':   690 obs. of  16 variables:
 $ Male          : Factor w/ 3 levels "","a","b": 3 2 2 3 3 3 3 2 3 3 ...
 $ Age           : num  30.8 58.7 24.5 27.8 20.2 ...
 $ Debt          : num  0 4.46 0.5 1.54 5.62 ...
 $ Married       : Factor w/ 4 levels "","l","u","y": 3 3 3 3 3 3 3 3 4 4 ...
 $ BankCustomer  : Factor w/ 4 levels "","g","gg","p": 2 2 2 2 2 2 2 2 4 4 ...
 $ EducationLevel: Factor w/ 15 levels "","aa","c","cc",..: 14 12 12 14 14 11 13 4 10
 $ Ethnicity     : Factor w/ 10 levels "","bb","dd","ff",..: 9 5 5 9 9 9 5 9 5 9 ...
 $ YearsEmployed : num  1.25 3.04 1.5 3.75 1.71 ...
 $ PriorDefault  : Factor w/ 2 levels "f","t": 2 2 2 2 2 2 2 2 2 2 ...
 $ Employed      : Factor w/ 2 levels "f","t": 2 2 1 2 1 1 1 1 1 1 ...
 $ CreditScore   : int  1 6 0 5 0 0 0 0 0 0 ...
 $ DriversLicense: Factor w/ 2 levels "f","t": 1 1 1 2 1 2 2 1 1 2 ...
 $ Citizen       : Factor w/ 3 levels "g","p","s": 1 1 1 1 3 1 1 1 1 1 ...
 $ ZipCode       : int  202 43 280 100 120 360 164 80 180 52 ...
 $ Income        : int  0 560 824 3 0 0 31285 1349 314 1442 ...
 $ class         : Factor w/ 2 levels "-","+": 2 2 2 2 2 2 2 2 2 2 ...
```

Figure 3: Credit Approval Data Structure.

Figure 3 reveals that the variables age, debt, years employed, credit score, zip code and income are numeric. We will run unsupervised discretization filter with the k-means method of binning (to preserve the original values distribution) to convert the continuous variables age, debt , years employed, credit score and income to factor variables, and the factor function to categorize the zip code variable as it has distinct values. We will then use the summary function on each variable to display its statistics or count per category or class.

```
> # running k-means discretization method on age, debt, yearsemployed, creditscore, and
me variables
> credit$Age <- discretize(credit$Age, method = "cluster", breaks = 6)
> summary(credit$Age)
  [13.8,22)   [22,28.4) [28.4,35.9) [35.9,44.7) [44.7,56.3) [56.3,80.2]        NA's
        148         191         140         104          60          35          12
>
> credit$Debt <- discretize(credit$Debt, method = "cluster", breaks = 6)
> summary(credit$Debt)
   [0,2.05) [2.05,4.66)  [4.66,8.2)  [8.2,12.1) [12.1,17.9)   [17.9,28]
        292         148          91          95          48          16
>
> credit$YearsEmployed <- discretize(credit$YearsEmployed, method = "cluster", breaks =
> summary(credit$YearsEmployed)
   [0,0.731) [0.731,1.97)  [1.97,3.88)  [3.88,6.85)  [6.85,12.1)  [12.1,28.5]
         305          151          106           72           34           22
>
> credit$CreditScore <- discretize(credit$CreditScore, method = "cluster", breaks = 4)
> summary(credit$CreditScore)
   [0,4.12) [4.12,11.3) [11.3,34.2)   [34.2,67]
        554         104          30           2
>
> credit$Income <- discretize(credit$Income, method = "cluster", breaks = 6)
> summary(credit$Income)
       [0,1.91e+03) [1.91e+03,7.77e+03) [7.77e+03,2.05e+04) [2.05e+04,3.98e+04)
                618                  56                  11                   2
[3.98e+04,7.53e+04)     [7.53e+04,1e+05]
                  2                    1
>
```

Figure 4: Age, Debt, Years Employed, Credit Score and Income Variables Statistics after Discretization

From figure 4, we can see that most applicants are between 22 and 28 years old, have a debt value between 0 and 2.05 units, have been employed for less than a year, have a credit score less than 4.12 and an income below 1,910 units

```
> # running Factor function on zipcode variable
> credit$Zipcode <- factor(credit$Zipcode)
> summary(credit$Zipcode)
      0     120     200     160      80     100     280     180     140     240 
    132      35      35      34      30      30      22      18      16      14 
    300     260      60     220     400     340     360     380      40      70 
     13      11       9       9       9       7       7       5       4       4 
    132     144     232     420     440     520      96     128     150     164 
      4       4       4       4       4       4       3       3       3       3 
    181     216     272     290     460     480      20      50      73      88 
      3       3       3       3       3       3       2       2       2       2 
    110     112     129     130     136     145     154     168     210     225 
      2       2       2       2       2       2       2       2       2       2 
    252     312     330     350     352     370     396     399     500     560 
      2       2       2       2       2       2       2       2       2       2 
     17      21      22      24      28      29      30      32      43      45 
      1       1       1       1       1       1       1       1       1       1 
     52      56      62      75      76      86      93      94      99     102 
      1       1       1       1       1       1       1       1       1       1 
    121     141     152     156     163     167     170     171     174     178 (oth
      1       1       1       1       1       1       1       1       1       1 
   NA's 
     13 
```

Figure 5: Zip code Variable Statistics After Running the Factor Function

Figure 5 shows that most applicants live in the zone 0 area.

III. Run the method with default arguments and store the generated rules in a variable called rules.

A. Include the command, the output, and the output interpretation in the report. Discuss the number of returned rules and the default arguments.

```
> # Run the method with default parameters and save it into the variable rules
> rules <- apriori(credit)
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen t
        0.8    0.1    1 none FALSE            TRUE       5     0.1      1     10
  ext
 TRUE

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

Absolute minimum support count: 69

set item appearances ...[0 item(s)] done [0.03s].
set transactions ...[243 item(s), 690 transaction(s)] done [0.07s].
sorting and recoding items ... [34 item(s)] done [0.00s].
creating transaction tree ... done [0.14s].
checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.30s].
writing ... [16758 rule(s)] done [0.41s].
creating S4 object  ... done [0.23s].
Warning message:
In apriori(credit) :
  Mining stopped (maxlen reached). Only patterns up to a length of 10 returned!
> rules
set of 16758 rules
```

Figure 6: Credit Approval Apriori Output

The Apriori algorithm generated a set 16,758 rules, taking 690 rows as a method input. The algorithm rejected any rules that did not meet the minimum support and confidence. For the default parameters, the minimum confidence is set to 0.8, the minimum support set to 0.1 and each rule cannot contain more than 10 items.

B. Run the inspect command to display the first 10 rules and interpret the output, including the returned rules and metrics. Include the command, the output, and interpretation in the report

```
> # inspect the first 10 rules
> inspect(rules[1:10])
     lhs                            rhs                        support   confidence cove
[1]  {}                          => {CreditScore=[0,4.12)}     0.8028986 0.8028986  1.00
[2]  {}                          => {Income=[0,1.91e+03)}      0.8956522 0.8956522  1.00
[3]  {}                          => {Citizen=g}                0.9057971 0.9057971  1.00
[4]  {EducationLevel=q}          => {BankCustomer=g}           0.1043478 0.9230769  0.11
[5]  {EducationLevel=q}          => {Married=u}                0.1043478 0.9230769  0.11
[6]  {EducationLevel=q}          => {Income=[0,1.91e+03)}      0.1000000 0.8846154  0.11
[7]  {EducationLevel=q}          => {Citizen=g}                0.1101449 0.9743590  0.11
[8]  {YearsEmployed=[4.4,10.6)} => {PriorDefault=t}           0.1028986 0.8255814  0.12
[9]  {YearsEmployed=[4.4,10.6)} => {Citizen=g}                0.1144928 0.9186047  0.12
[10] {Debt=[7.46,11.4)}          => {BankCustomer=g}           0.1028986 0.8068182  0.12
     lift      count
[1]  1.0000000 554
[2]  1.0000000 618
[3]  1.0000000 625
[4]  1.2272121  72
[5]  1.2272121  72
[6]  0.9876774  69
[7]  1.0756923  76
[8]  1.5779811  71
[9]  1.0141395  79
[10] 1.0726484  71
```

Figure 7: The First 10 Rules Generated.

Figure 7 shows the first 10 out of 16,758 rules generated by the Apriori method. The first three rules have no items in the antecedent that lead to the consequent in the right-hand side. Although the min support and confidence was met, these rules cannot be valid as their lift value is equal to one, which means both the left- and right-hand side in these 3 rules are independent from each other. We will have to change the min length parameter to eliminate the rules with blank itemset. The lift for rules 4, 5, 7, 8, 9 and 10 is greater than one; thus, the itemset on the left and right hand side are dependent on each other. Rule 5 for example reveals that 10% of the total applications show that applicants with the married status of "u" have an education level of ‘q’ and 92% of applicants who has a marital status of ‘u’ have an education level of ‘q’.

IV. Run the method with 2 different combinations of confidence, support, and minimum length values.

A. For each run

- **Specify the input parameters you used.**
- **Include the command and the command output.**
- **Discuss how many rules and how many items were returned.**
- **Run the inspect command to preview the first 10 rules. What is the strongest rule, and why? Include the command, the output, and output interpretation in the report.**

```
> # first run:
> rules <- apriori(credit, parameter= list(supp=0.4, conf=0.7, minlen=2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target
        0.7    0.1    1 none FALSE            TRUE       5     0.4      2     10  rules
  ext
 TRUE

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

Absolute minimum support count: 276

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[243 item(s), 690 transaction(s)] done [0.05s].
sorting and recoding items ... [16 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 done [0.02s].
writing ... [237 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules
set of 237 rules
```

Figure 8: Run 1 - Apriori Output with min-supp = 0.4, min-confidence = 0.7 and minlen = 2

```
> inspect(rules[1:10])
     lhs                   rhs                      support   confidence coverage
[1]  {Employed=t}       => {Citizen=g}              0.4231884 0.9898305  0.4275362
[2]  {class=+}          => {PriorDefault=t}         0.4115942 0.9250814  0.4449275
[3]  {PriorDefault=t}   => {class=+}                0.4115942 0.7867036  0.5231884
[4]  {class=+}          => {Citizen=g}              0.4159420 0.9348534  0.4449275
[5]  {DriversLicense=t} => {Income=[0,1.91e+03)}    0.4130435 0.9018987  0.4579710
[6]  {DriversLicense=t} => {Citizen=g}              0.4159420 0.9082278  0.4579710
[7]  {PriorDefault=f}   => {class=-}                0.4434783 0.9300912  0.4768116
[8]  {class=-}          => {PriorDefault=f}         0.4434783 0.7989556  0.5550725
[9]  {PriorDefault=f}   => {CreditScore=[0,4.12)}   0.4666667 0.9787234  0.4768116
[10] {PriorDefault=f}   => {Income=[0,1.91e+03)}    0.4579710 0.9604863  0.4768116
     lift     count
[1]  1.092773 292
[2]  1.768161 284
[3]  1.768161 284
[4]  1.032078 287
[5]  1.006974 285
[6]  1.002684 287
[7]  1.675621 306
[8]  1.675621 306
[9]  1.218988 322
[10] 1.072388 316
```

Figure 9: First 10 rules for Run 1

The first run was done with the min-supp = 0.4, min-confidence = 0.7 and min-length = 2. In this run, the method returns 237 rules with 243 items. All the rules below the support of 0.4 and the confidence of 0.7 were omitted. We set the min-length to 2 to eliminate the blank itemset observed earlier. Out of the 10 rules in figure 9, the stronger rule is rule 3 as it has the highest lift value (lift = 1.77), the highest degree of correlation between the antecedent (prior default = t) and the consequent (class = +).

```
> #second run:
> rules <- apriori(credit, parameter= list(supp=0.5, conf=0.8, minlen=2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen targe
        0.8    0.1    1 none FALSE            TRUE       5     0.5      2     10  rule
   ext
 TRUE

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

Absolute minimum support count: 345

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[243 item(s), 690 transaction(s)] done [0.02s].
sorting and recoding items ... [12 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [63 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules
set of 63 rules
```

Figure 10: Run 2 - Apriori Output with min-supp = 0.5, min-confidence = 0.8 and minlen = 2

```
> inspect(rules[1:10])
     lhs                           rhs                      support   confidence coverage
[1]  {class=-}                  => {CreditScore=[0,4.12)} 0.5362319 0.9660574  0.555072
[2]  {class=-}                  => {Income=[0,1.91e+03)}  0.5376812 0.9686684  0.555072
[3]  {YearsEmployed=[0,1.4)} => {CreditScore=[0,4.12)} 0.5130435 0.9030612  0.568115
[4]  {YearsEmployed=[0,1.4)} => {Income=[0,1.91e+03)}  0.5217391 0.9183673  0.568115
[5]  {YearsEmployed=[0,1.4)} => {Citizen=g}            0.5086957 0.8954082  0.568115
[6]  {Employed=f}               => {CreditScore=[0,4.12)} 0.5724638 1.0000000  0.572463
[7]  {Employed=f}               => {Income=[0,1.91e+03)}  0.5405797 0.9443038  0.572463
[8]  {Ethnicity=v}              => {Income=[0,1.91e+03)}  0.5217391 0.9022556  0.578260
[9]  {Ethnicity=v}              => {Citizen=g}            0.5217391 0.9022556  0.578260
[10] {Male=b}                   => {CreditScore=[0,4.12)} 0.5463768 0.8055556  0.678260
     lift      count
[1]  1.2032123 370
[2]  1.0815230 371
[3]  1.1247513 354
[4]  1.0253616 360
[5]  0.9885306 351
[6]  1.2454874 395
[7]  1.0543198 373
[8]  1.0073728 360
[9]  0.9960902 360
[10] 1.0033093 377
```

Figure 11: First 10 rules for Run 2

The second run with the parameter specified in figure 10 returned 63 rules with the strongest rule being rule 6 in figure 11 (lift = 1.25). This rule has a confidence = 1, which means that all applicants who have a credit score between 0 and 4.12 have an employment status of “f”.

B. **How does changing confidence, support, and minimum length values affect the returned rules?**

Increasing the confidence, support and minimum length values reduced the number of rules the algorithm returns as it improve the strength of the rules.

C. **What are the differences between support, confidence, and lift metrics for identifying the strongest rules?**

Lift gives the degree of correlation between the antecedent and the consequent, it reveals how the antecedent affects the consequent.

Support and confidence measure the strength of the rule. Support provides the fraction of observation that contain the specified itemset, while the confidence value gives the probability in which the consequent occurs with the occurrence of the antecedent.

V. Generate the rules that have only the class=’+’ or class=’-’ on the right-hand side. Store the rules in the variable rules. (Hint - See the

generating rules for the specified itemsets section in the Word Document. You may need to adjust the confidence and support values)

A. Include the command, the output, and output interpretation in the report.

```
> # rules with "+" or "-" on the right hand side
> rules<-apriori(credit, parameter= list(supp=0.2, conf=0.8, minlen=2), appearance=li
c("class=-", "class=+"), default="lhs"))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen targ
        0.8    0.1    1 none FALSE            TRUE       5     0.2      2     10  rul
  ext
 TRUE

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

Absolute minimum support count: 138

set item appearances ...[2 item(s)] done [0.00s].
set transactions ...[243 item(s), 690 transaction(s)] done [0.08s].
sorting and recoding items ... [26 item(s)] done [0.03s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 7 done [0.06s].
writing ... [208 rule(s)] done [0.01s].
creating S4 object  ... done [0.04s].
> rules
set of 208 rules
```

Figure 12: Apriori Output with Consequent Class = + or -

The algorithm generated 208 rules with the consequent class = “+” or class = “-“ , taking 243 items as an input and satisfying the minimum support of 0.2 and the minimum confidence of 0.8

B. Run the inspect command to preview the first 10 rules. Include the command and output in the report.

```
> inspect(rules[1:10])
     lhs                                    rhs       support   confidence coverage
[1]  {PriorDefault=f}                    => {class=-} 0.4434783 0.9300912  0.476811
[2]  {PriorDefault=t,Employed=t}         => {class=+} 0.3000000 0.9078947  0.330434
[3]  {PriorDefault=t,DriversLicense=f}   => {class=+} 0.2130435 0.8166667  0.260869
[4]  {Married=u,PriorDefault=t}          => {class=+} 0.3521739 0.8209459  0.428985
[5]  {BankCustomer=g,PriorDefault=t}     => {class=+} 0.3521739 0.8209459  0.428985
[6]  {PriorDefault=t,Citizen=g}          => {class=+} 0.3971014 0.8058824  0.492753
[7]  {PriorDefault=f,DriversLicense=f}   => {class=-} 0.2608696 0.9278351  0.281159
[8]  {YearsEmployed=[0,1.4),PriorDefault=f} => {class=-} 0.3420290 0.9440000  0.362318
[9]  {PriorDefault=f,Employed=f}         => {class=-} 0.3492754 0.9198473  0.379710
[10] {Ethnicity=v,PriorDefault=f}        => {class=-} 0.2710145 0.9444444  0.286956
     lift     count
[1]  1.675621 306
[2]  2.040545 207
[3]  1.835505 147
[4]  1.845123 243
[5]  1.845123 243
[6]  1.811267 274
[7]  1.671557 180
[8]  1.700679 236
[9]  1.657166 241
[10] 1.701480 187
>
```

Figure 13: First 10 Rules with Consequent Approved = + or – (approved or disapproved)

C. What do the returned rules suggest about the credit approval decision? What are the strongest rules?

The output in figure 13 suggests that rule 2 is the strongest rule leading to an approved application (class +), while rule 10 is the strongest rule leading to a denied application (class -). Rule 2 reveals that 30% of the applications shows that applicants who were approved had a prior default and employed status of "t" and 90% of applicants who got approved had an employment status and prior default status of 't'; and the degree of correlation between the antecedent and the consequent is 2.04. The data also show that ethnicity is one of the determining factor of credit approval.

VI. Prune the returned rules

A. Why do we prune the returned rules?

We prune the returned rules to remove the redundant rules and assure that no superset of infrequent itemset is generated or tested.

B. Run the following commands on a variable that stores the rules class='+' or class='-' on the right-hand side to find the redundant rules and to eliminate them. Discuss the output of which(redundant) command. Include the output and discussion in the report.

```
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)] <- F
redundant <- colSums(subset.matrix, na.rm=T) >= 1
which(redundant)
```

```
> redundant <- colSums(subset.matrix, na.rm=T) >= 1
> which(redundant)
                                        {Married=u,BankCustomer=g,PriorDefault=t,Employed=t,class=+}
                                                                                                   3
                                             {Married=u,PriorDefault=t,Employed=t,Citizen=g,class=+}
                                                                                                   4
                                        {BankCustomer=g,PriorDefault=t,Employed=t,Citizen=g,class=+}
                                                                                                   5
                              {Married=u,BankCustomer=g,PriorDefault=t,Employed=t,Citizen=g,class=+}
                                                                                                   6
                                                       {PriorDefault=t,Employed=t,Citizen=g,class=+}
                                                                                                   8
                                             {PriorDefault=t,Employed=t,Income=[0,1.91e+03),class=+}
                                                                                                   9
                                   {PriorDefault=t,Employed=t,Citizen=g,Income=[0,1.91e+03),class=+}
                                                                                                  10
                                         {Married=u,BankCustomer=g,PriorDefault=t,Citizen=g,class=+}
                                                                                                  13
                                                 {Male=b,Married=u,PriorDefault=t,Citizen=g,class=+}
                                                                                                  14
                                            {Male=b,BankCustomer=g,PriorDefault=t,Citizen=g,class=+}
                                                                                                  15
                                  {Male=b,Married=u,BankCustomer=g,PriorDefault=t,Citizen=g,class=+}
                                                                                                  16
                                                   {Married=u,BankCustomer=g,PriorDefault=t,class=+}
                                                                                                  20
                                    {Married=u,PriorDefault=t,Citizen=g,Income=[0,1.91e+03),class=+}
                                                                                                  22
                               {BankCustomer=g,PriorDefault=t,Citizen=g,Income=[0,1.91e+03),class=+}
                                                                                                  23
                     {Married=u,BankCustomer=g,PriorDefault=t,Citizen=g,Income=[0,1.91e+03),class=+}
                                                                                                  24
                                                           {Male=b,Married=u,PriorDefault=t,class=+}
                                                                                                  25
                                                      {Male=b,BankCustomer=g,PriorDefault=t,class=+}
                                                                                                  26
                                            {Male=b,Married=u,BankCustomer=g,PriorDefault=t,class=+}
                                                                                                  27
```

Figure 14: Redundant Rules Preview

Figure 14 shows the preview of the list of redundant rules, rules that are subset of larger rules. The which() function returns the position of elements in the vector for which value is TRUE.

C. Run the following commands to remove the redundant rules and to display the remaining rules. Which rules remain? Include the commands, the output, and output interpretation in the report.

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

```
> rules.pruned <- rules.sorted[!redundant]
> inspect(rules.pruned)
     lhs                          rhs         support confidence  coverage     lift count
[1]  {Married=u,
      PriorDefault=t,
      Employed=t}              => {class=+} 0.2579710  0.9222798 0.2797101 2.072876   178
[2]  {BankCustomer=g,
      PriorDefault=t,
      Employed=t}              => {class=+} 0.2579710  0.9222798 0.2797101 2.072876   178
[3]  {PriorDefault=t,
      Employed=t}              => {class=+} 0.3000000  0.9078947 0.3304348 2.040545   207
[4]  {Married=u,
      PriorDefault=t,
      Citizen=g}               => {class=+} 0.3405797  0.8453237 0.4028986 1.899913   235
[5]  {BankCustomer=g,
      PriorDefault=t,
      Citizen=g}               => {class=+} 0.3405797  0.8453237 0.4028986 1.899913   235
[6]  {PriorDefault=t,
      DriversLicense=f,
      Citizen=g}               => {class=+} 0.2014493  0.8373494 0.2405797 1.881991   139
[7]  {Married=u,
      PriorDefault=t}          => {class=+} 0.3521739  0.8209459 0.4289855 1.845123   243
[8]  {BankCustomer=g,
      PriorDefault=t}          => {class=+} 0.3521739  0.8209459 0.4289855 1.845123   243
[9]  {PriorDefault=t,
      DriversLicense=f}        => {class=+} 0.2130435  0.8166667 0.2608696 1.835505   147
[10] {Ethnicity=v,
      PriorDefault=t,
      Citizen=g}               => {class=+} 0.2159420  0.8097826 0.2666667 1.820033   149
```

Figure 15: Preview of Remaining Rules After Removing the Redundant Rules

We are left with 43 rules after pruning the redundant one. From the top two rules (1 &2), we can see that the combinations **{Married=u, PriorDefault=t, Employed=t},** and **{BankCustomer=g,**

PriorDefault=t, Employed=t} equally predict the approved application with a degree of correlation of lift = 2.07.

VII. Rules visualization

A. Choose any visualization method discussed in the tutorial to visualize the pruned rules in step 6. Explain how the plot represents the rules and the metrics for ranking rules. How do we use the plot to identify the strongest rules? Include the commands, the plot, and plot discussion in the report.

Command: `plot(rules.pruned, method="paracoord", control = list(reorder = TRUE))`

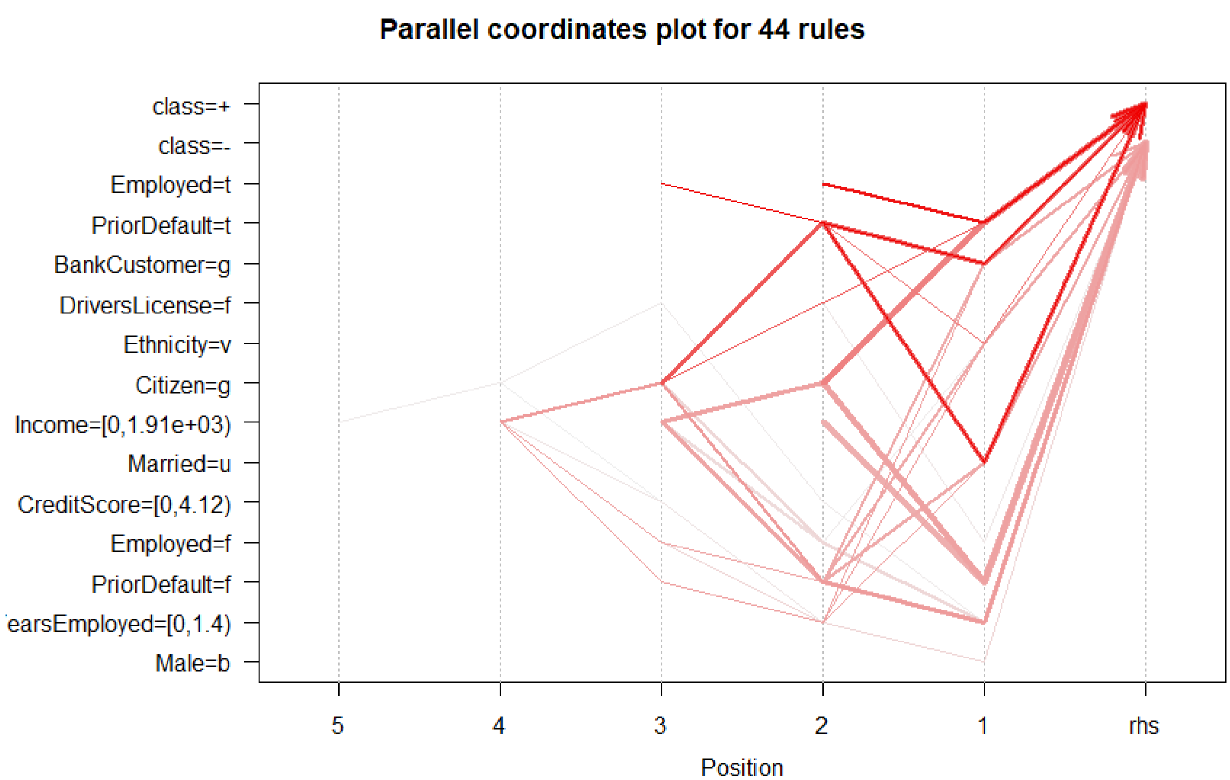

Figure 16: Parallel Coordinates Plot

Parallel coordinates plot is an individual rule representation. It helps visualizing which characteristics lead to an approved or denied application. In the above plot, the third red arrow from the top suggest that applicants with the following characteristics (citizen = g, prior default = t and married = u) are likely to be approved for credit.

VIII. Summary

A. Why do we consider more than one metric to identify the strongest rules?

The goal with association rules mining is to identify rules that are useful to the user. Thus, considering more than one metric helps measure the usefulness of the rules and prevent errors.

B. Which part of this exercise did you find the most challenging, and why? What approach did you take to resolve the challenge?

The most challenging step was the data preprocessing, identifying the correct method to discretize the numeric variables.