Association Rules Mining – Credit Approval Process

When talking about association rules, typically, people use the market basket analysis (MBA) to describe the process. Basically, retailers use MBA techniques to identify unique relationships between items people buy together in any given transaction which can help with their sales strategies. A recent assessment by the data analytics and advisory firm Quantzig, enabled a German food retailer to increase quarterly sales by 50%, reduce marketing cost by 15%, and prompted the retailer to reconfigure the store layout, all of this, by exploiting MBA's association rules (businesswire.com, 2020).

The following is an association rules credit-approval process report using RStudio. The purpose of the same is to find patterns (rules) between the items captured in a .csv file named "CreditApproval" while using the Apriori algorithm. The findings could ultimately be employed to improve operations and prioritization of supporting programs in the credit approval process, marketing initiatives, and/or applications' sifting.

Data Pre-processing. During pre-processing, the working directory was set to the location of the .csv file, which was loaded into RStudio as "cadf". A quick glimpse of the first six rows [head() command] of the dataframe shows the original 17 attributes, as well as some of the observations. The original names and values of this referred file have been altered to maintain the anonymity of the data (Dua, D. and Graff, C. (2019).

```
# Set working directory
setwd("~/OneDrive/UMGC/DATA630/Week3/Exercise3")
# Load intended file
cadf <- read.csv("CreditApproval.csv", head =TRUE, sep=",", as.is=FALSE)
# Preview vectored cadf
head(cadf)
```

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

```
# Preview cadf structure
str(cadf)
```

```
> str(cadf)
'data.frame':   690 obs. of  17 variables:
 $ Key           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ 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 14 ...
 $ 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 ...
```

The data frame structure confirms 17 variables and 690 observations with a mix of different type of variables. Importantly, unique identifiers, like "Key" may affect the Apriori algorithm and it is recommendable to remove them during this pre-processing phase. Importantly, the algorithm only works with categorical data so discretization will be handy for this dataset. Also, any irrelevant variable –those not adding any value or insight, can be removed.

```
# Descriptive statistics
summary(cadf)
```

```
> # Descriptive statistics
> summary(cadf)
      Key         Male          Age              Debt          Married BankCustomer EducationLevel   Ethnicity   YearsEmployed    PriorDefault
 Min.   :  1.0    :  12   Min.   :13.75   Min.   : 0.000    :  6     :  6       c      :137    v      :399   Min.   : 0.000   f:329
 1st Qu.:173.2   a:210   1st Qu.:22.60   1st Qu.: 1.000   l:  2   g :519       q      : 78    h      :138   1st Qu.: 0.165   t:361
 Median :345.5   b:468   Median :28.46   Median : 2.750   u:519   gg:  2       w      : 64    bb     : 59   Median : 1.000
 Mean   :345.5           Mean   :31.57   Mean   : 4.759   y:163   p :163       i      : 59    ff     : 57   Mean   : 2.223
 3rd Qu.:517.8           3rd Qu.:38.23   3rd Qu.: 7.207                        aa     : 54            :  9   3rd Qu.: 2.625
 Max.   :690.0           Max.   :80.25   Max.   :28.000                        ff     : 53    j      :  8   Max.   :28.500
                         NA's   :12                                            (Other):245    (Other): 20
 Employed  CreditScore   DriversLicense Citizen    Zipcode         Income          class
 f:395     Min.   : 0.0   f:374          g:625   Min.   :   0   Min.   :     0.0   -:383
 t:295     1st Qu.: 0.0   t:316          p:  8   1st Qu.:  75   1st Qu.:     0.0   +:307
           Median : 0.0                  s: 57   Median : 160   Median :     5.0
           Mean   : 2.4                          Mean   : 184   Mean   :  1017.4
           3rd Qu.: 3.0                          3rd Qu.: 276   3rd Qu.:   395.5
           Max.   :67.0                          Max.   :2000   Max.   :100000.0
                                                 NA's   :13
```

Descriptive statistics. The summary command was used to display the descriptive statistics of all the variables within the set. It highlights the minimum, maximum, mean, median, and quartiles of the continues values, while highlighting facts about the discrete variables and NA values across some of the variables. Out of these descriptive statistics, the following steps were

taken: 1. remove the “Key” variable, 2. Identified and counted missing values, 3. Calculated missing value rate, removed variables as needed, and 4. Renamed the “Male” variable to “Sex”.

```
#Remove the Key variable
cadf$Key<-NULL
summary(cadf$Key)
> cadf$Key<-NULL
> summary(cadf$Key)
Length  Class   Mode
     0   NULL   NULL
Warning message:
Unknown or uninitialised column: `Key`.
#How many values are missing
sum(is.na(cadf))
# Identify and count column(s) with missing values
colSums(is.na(cadf))
```

```
> #How many values are missing
> sum(is.na(cadf))
[1] 25
> # Identify and count column(s) with missing values
> colSums(is.na(cadf))
         Male           Age          Debt       Married   BankCustomer EducationLevel     Ethnicity  YearsEmployed  PriorDefault
            0            12             0             0             0             0             0             0             0
     Employed CreditScore DriversLicense       Citizen       Zipcode        Income         class
            0             0             0             0            13             0             0
>
```

Next, for easier understanding, I renamed the “Male” variable to “Sex”.

```
> # Rename Male variable to Sex
> cadf<-dplyr::rename(cadf, Sex = Male)
> str(cadf)
tibble [690 × 16] (S3: tbl_df/tbl/data.frame)
 $ Sex           : Factor w/ 3 levels "","a","b": 3 2 2 3 3 3 3 2 3 3 ...
 $ Age           : num [1:690] 30.8 58.7 24.5 27.8 20.2 ...
 $ Debt          : num [1:690] 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 14 ...
```

```
> # Calculate percentage of missing values
> m_perc<- sum(sum(is.na(cadf))/nrow(cadf))
> m_perc
[1] 0.03623188
```

Finally, as shown by the above calculation, only 3% of the observations were missing values, so I decided to exclude those rows with the na.omit command and created a new vector called “n_cadf”.

```
> # Handling missing values by removing the observations
> n_cadf<-na.omit(cadf)
> summary(n_cadf)
 Sex          Age             Debt        Married BankCustomer EducationLevel   Ethnicity   YearsEmployed    PriorDefault Employed
  : 12   Min.   :13.75   Min.   : 0.000    :  0     :  0       c      :135   v      :388   Min.   : 0.000   f:314        f:376
 a:203   1st Qu.:22.60   1st Qu.: 1.010   l:  2   g :508       q      : 77   h      :138   1st Qu.: 0.165   t:352        t:290
 b:451   Median :28.50   Median : 2.750   u:508   gg:  2       w      : 64   bb     : 55   Median : 1.000
         Mean   :31.57   Mean   : 4.798   y:156   p :156       i      : 56   ff     : 54   Mean   : 2.222
         3rd Qu.:38.25   3rd Qu.: 7.207                       aa     : 53   j      :  8   3rd Qu.: 2.585
         Max.   :80.25   Max.   :28.000                       ff     : 50   z      :  8   Max.   :28.500
                                                              (Other):231   (Other): 15
  CreditScore     DriversLicense Citizen    Zipcode           Income         class
 Min.   : 0.000   f:359          g:609   Min.   :   0.00   Min.   :     0.0   -:367
 1st Qu.: 0.000   t:307          p:  2   1st Qu.:  75.25   1st Qu.:     0.0   +:299
 Median : 0.000                  s: 55   Median : 160.00   Median :     5.0
 Mean   : 2.459                          Mean   : 182.12   Mean   :   998.6
 3rd Qu.: 3.000                          3rd Qu.: 271.00   3rd Qu.:   399.0
 Max.   :67.000                          Max.   :2000.00   Max.   :100000.0
```

Discretization. Another step taken was transforming the rest of the variables from numeric values to discrete factors. As mentioned before, this step was imperative in preparation for the Apriori algorithm.

```
# Age discretized by Equal Intervals
summary(n_cadf$Age) #Before
n_cadf$Age<-discretize(n_cadf$Age, method="interval", breaks=10)
summary(n_cadf$Age) #After
summary(n_cadf)

# YearsEmployed discretized by Equal Frequency Method
summary(n_cadf$YearsEmployed) # Before
n_cadf$YearsEmployed<-discretize(n_cadf$YearsEmployed, method="frequency",
breaks=5)
summary(n_cadf$YearsEmployed) # After

# Debt discretized by K-means Clustering
summary(n_cadf$Debt) # Before
n_cadf$Debt<-discretize(n_cadf$Debt, method="cluster", breaks=6)
summary(n_cadf$Debt) # After
```

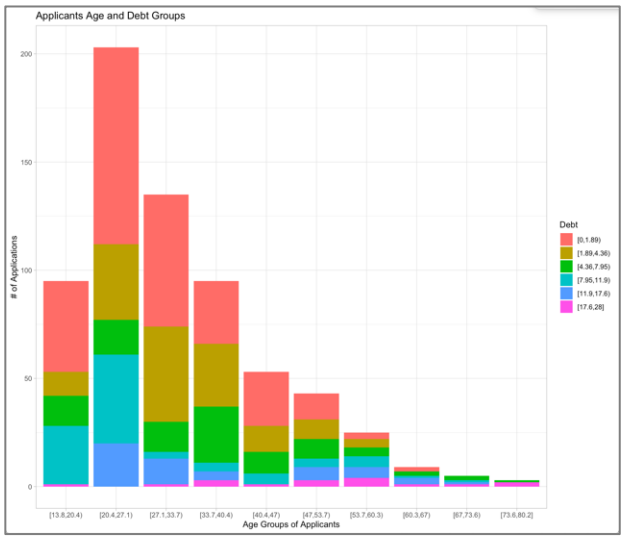
```
> summary(n_cadf$Age) #Before
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  13.75   22.60   28.50   31.57   38.25   80.25
> n_cadf$Age<-discretize(n_cadf$Age, method="interval", breaks=10)
> summary(n_cadf$Age) #After
[13.8,20.4) [20.4,27.1) [27.1,33.7) [33.7,40.4)   [40.4,47)   [47,53.7) [53.7,60.3)   [60.3,67)   [67,73.6) [73.6,80.2]
         95         203         135          95          53          43          25           9           5           3
> # YearsEmployed discretized by Equal Frequency Method
> summary(n_cadf$YearsEmployed) # Before
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.165   1.000   2.222   2.585  28.500
> n_cadf$YearsEmployed<-discretize(n_cadf$YearsEmployed, method="frequency", breaks=5)
> summary(n_cadf$YearsEmployed) # After
  [0,0.125) [0.125,0.5)   [0.5,1.5)   [1.5,3.5)  [3.5,28.5]
        119         124         139         147         137
>
> # Debt discretized by K-means Clustering
> summary(n_cadf$Debt) # Before
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   1.010   2.750   4.798   7.207  28.000
> n_cadf$Debt<-discretize(n_cadf$Debt, method="cluster", breaks=6)
> summary(n_cadf$Debt) # After
   [0,1.89) [1.89,4.36) [4.36,7.95) [7.95,11.9) [11.9,17.6)   [17.6,28]
        265         144          98          91          51          17
```

```
#factor the rest of usable variable
n_cadf$Zipcode<-factor(n_cadf$Zipcode)
n_cadf$CreditScore<-factor(n_cadf$CreditScore)
n_cadf$Debt<-factor(n_cadf$Debt)
n_cadf$Income<-factor(n_cadf$Income)
summary(n_cadf)
```

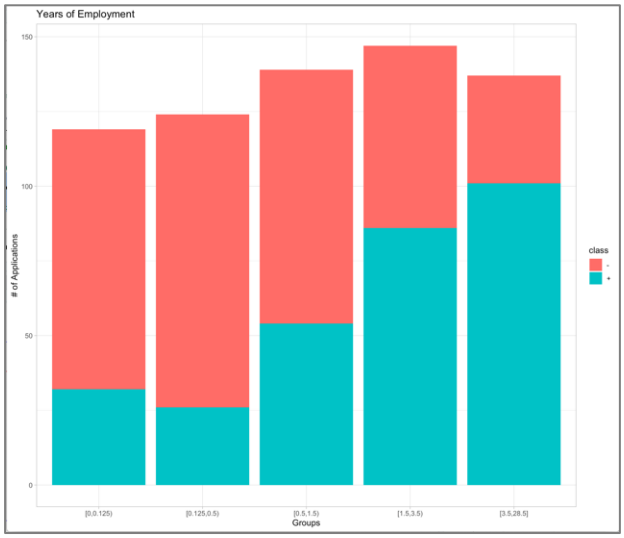
```
> summary(n_cadf)
 Sex             Age              Debt     Married BankCustomer EducationLevel   Ethnicity       YearsEmployed PriorDefault Employed
  : 12   [20.4,27.1):203   [0,1.89)   :265    :  0     :  0       c      :135    v      :388   [0,0.125)  :119   f:314        f:376
 a:203   [27.1,33.7):135   [1.89,4.36):144    l:  2   g :508       q      : 77    h      :138   [0.125,0.5):124   t:352        t:290
 b:451   [13.8,20.4): 95   [4.36,7.95): 98    u:508   gg:  2       w      : 64    bb     : 55   [0.5,1.5)  :139
         [33.7,40.4): 95   [7.95,11.9): 91    y:156   p :156       i      : 56    ff     : 54   [1.5,3.5)  :147
         [40.4,47)  : 53   [11.9,17.6): 51                         aa     : 53    j      :  8   [3.5,28.5] :137
         [47,53.7)  : 43   [17.6,28]  : 17                         ff     : 50    z      :  8
         (Other)    : 42                                           (Other):231    (Other): 15
  CreditScore  DriversLicense Citizen    Zipcode       Income    class
 0      :376   f:359          g:609   0      :129   0      :279   -:367
 1      : 71   t:307          p:  2   120    : 35   1      : 28   +:299
 2      : 43                  s: 55   160    : 34   500    : 10
 3      : 27                          200    : 34   1000   : 10
 6      : 22                          80     : 30   2      :  9
 11     : 19                          100    : 30   5      :  8
 (Other):108                          (Other):374   (Other):322
>
```

Observations of some of the discretized variables:

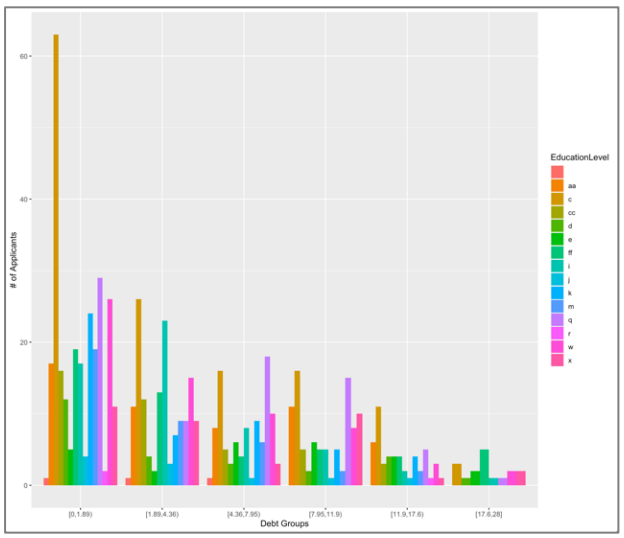
The highest Age-group of applicants range between 20 and 27 years. Among this Age-group most of the applicants were distinguished between 0 to 1.89 Debt ratio.



In regards of years of employment as related to the classes, the main group was between 1.5 – 3.5 years of employment with a majority of the applicants cataloged as "class +". Additionally, the group greater than 3.5 years of employment had the majority of the applicants identified as "class +".



Based on the discretized Debt attribute, a significant number of applicants were part of the 0 – 1.89 debt group and with an "Education Level" of "c".



Apriori Method. After completing all the basic pre-processing steps, I proceed to run the Apriori algorithm with its default arguments, and storing the crafted rules in a variable named "rules".

```
> # Run Apriori with default parameters
> rules<-apriori(n_cadf)
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.8    0.1    1 none FALSE            TRUE       5     0.1      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: 66

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[699 item(s), 666 transaction(s)] done [0.00s].
sorting and recoding items ... [32 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 7 8 done [0.01s].
writing ... [4666 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules
set of 4666 rules
```

Per the above image, using the algorithm's default parameters, Apriori identified 4666 rules after analyzing 600 transactions and the command saved the results in a new variable called "rules". These 4666 rules are the result of Apriori's default configuration of 0.8 in accuracy, a minimum number of items or minlen of 1, and a minimum support constrain of 0.1.

After generating the rules, I used the inspect() command to preview the first 10 rules. The display showed that 609 observations fell within the first rule, but no left-hand antecedent to link with the right-hand consequent.

```
> inspect(rules[1:10])
     lhs                          rhs              support   confidence coverage  lift      count
[1]  {}                        => {Citizen=g}      0.9144144 0.9144144  1.0000000 1.0000000 609
[2]  {CreditScore=1}           => {Employed=t}     0.1066066 1.0000000  0.1066066 2.2965517  71
[3]  {CreditScore=1}           => {Citizen=g}      0.1051051 0.9859155  0.1066066 1.0781933  70
[4]  {EducationLevel=q}        => {Married=u}      0.1066066 0.9220779  0.1156156 1.2088659  71
[5]  {EducationLevel=q}        => {BankCustomer=g} 0.1066066 0.9220779  0.1156156 1.2088659  71
[6]  {EducationLevel=q}        => {Citizen=g}      0.1126126 0.9740260  0.1156156 1.0651910  75
[7]  {Age=[33.7,40.4)}         => {Sex=b}          0.1141141 0.8000000  0.1426426 1.1813747  76
[8]  {Age=[33.7,40.4)}         => {Citizen=g}      0.1321321 0.9263158  0.1426426 1.0130153  88
[9]  {Age=[13.8,20.4)}         => {Citizen=g}      0.1306306 0.9157895  0.1426426 1.0015038  87
[10] {YearsEmployed=[0,0.125)} => {Citizen=g}      0.1621622 0.9075630  0.1786787 0.9925073 108
```

Follow this initial preview, I ran the method with different parameters across the confidence and minimum length. The first altered method I ran was with a support of 0.4 and minimum confidence of 0.7. The bellow display shows how by incorporating these parameters I was able to slim the model to 80 rules with a higher accuracy.

```
> rules <- apriori(n_cadf, parameter= list(supp=0.4, conf=0.7))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.7    0.1    1 none FALSE            TRUE       5     0.4      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: 266

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[699 item(s), 666 transaction(s)] done [0.00s].
sorting and recoding items ... [15 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [80 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>
```

Out of this model, two rules caught my attention, rule #5 and rule #6. These were characterized by a lift greater than 1.77 across the "class" and "PriorDefault" attributes. Additionally, the same variables were found in rules #7, #9, and #10, making me think that these variables were going to be among the final top rules.

```
> inspect(rules[1:10])
     lhs                    rhs                 support   confidence coverage  lift      count
[1]  {}                  => {Married=u}         0.7627628 0.7627628  1.0000000 1.0000000 508
[2]  {}                  => {BankCustomer=g}    0.7627628 0.7627628  1.0000000 1.0000000 508
[3]  {}                  => {Citizen=g}         0.9144144 0.9144144  1.0000000 1.0000000 609
[4]  {Employed=t}        => {Citizen=g}         0.4309309 0.9896552  0.4354354 1.0822830 287
[5]  {class=+}           => {PriorDefault=t}    0.4204204 0.9364548  0.4489489 1.7718151 280
[6]  {PriorDefault=t}    => {class=+}           0.4204204 0.7954545  0.5285285 1.7718151 280
[7]  {class=+}           => {Citizen=g}         0.4249249 0.9464883  0.4489489 1.0350759 283
[8]  {DriversLicense=t}  => {Citizen=g}         0.4204204 0.9120521  0.4609610 0.9974166 280
[9]  {PriorDefault=f}    => {class=-}           0.4429429 0.9394904  0.4714715 1.7049064 295
[10] {class=-}           => {PriorDefault=f}    0.4429429 0.8038147  0.5510511 1.7049064 295
```

For the second model, I changed the minlen to 4, skipping the first 3 rules, while maintaining the support and confidence at 0.4 and 0.7 respectively. This resulted in decreasing to 9 rules for all the 666 transactions.

```
> rules <- apriori(n_cadf, parameter= list(supp=0.4, conf=0.7, minlen=4))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.7    0.1    1 none FALSE            TRUE       5     0.4      4     10  rules TRUE

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

Absolute minimum support count: 266

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[699 item(s), 666 transaction(s)] done [0.00s].
sorting and recoding items ... [15 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [9 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>
```

Out of these 9 rules, 7 of them had a lift greater than 1.0 with a confidence level greater than 90%. My assumption at this point was that any variation among the parameters of the Apriori model would have a direct impact to the number of rules generated. Metrics like support, confidence, and lift would help you locking-into the strongest combined-rules of the model. The following snap-shot displays a copy of the final 9 rules.

```
> inspect(rules[1:9])
    lhs                                                rhs              support   confidence coverage  lift      count
[1] {Married=u,BankCustomer=g,PriorDefault=t}       => {Citizen=g}      0.4084084 0.9444444  0.4324324 1.0328407 272
[2] {Married=u,PriorDefault=t,Citizen=g}            => {BankCustomer=g} 0.4084084 1.0000000  0.4084084 1.3110236 272
[3] {BankCustomer=g,PriorDefault=t,Citizen=g}       => {Married=u}      0.4084084 1.0000000  0.4084084 1.3110236 272
[4] {Married=u,BankCustomer=g,Ethnicity=v}          => {Citizen=g}      0.4099099 0.9039735  0.4534535 0.9885819 273
[5] {Married=u,Ethnicity=v,Citizen=g}               => {BankCustomer=g} 0.4099099 1.0000000  0.4099099 1.3110236 273
[6] {BankCustomer=g,Ethnicity=v,Citizen=g}          => {Married=u}      0.4099099 1.0000000  0.4099099 1.3110236 273
[7] {Sex=b,Married=u,BankCustomer=g}                => {Citizen=g}      0.4564565 0.9020772  0.5060060 0.9865080 304
[8] {Sex=b,Married=u,Citizen=g}                     => {BankCustomer=g} 0.4564565 1.0000000  0.4564565 1.3110236 304
[9] {Sex=b,BankCustomer=g,Citizen=g}                => {Married=u}      0.4564565 1.0000000  0.4564565 1.3110236 304
>
```

The subsequent step consisted in exploring rules associated to the class, "+" or "-" on the right-hand side as a consequence. For that reason I specified the appearance parameter as a right-hand combination only containing the "class = -" and "class = +" factors in addition of setting the confidence to 80% and support to no less than 30%.

```
# Generate rules that have only the "-", or "+" class
rules<-apriori(n_cadf, parameter=list(supp=0.3, conf=0.8, minlen=1),
appearance=list(rhs=c("class=-", "class=+"), default="lhs"))
# Inspect first 10 generated rules
inspect(rules[1:10])
```

```
> # Generate rules that have only the "-", or "+" class
> rules<-apriori(n_cadf, parameter=list(supp=0.3, conf=0.8, minlen=1), appearance=list(rhs=c("class=-", "class=+"), default="lhs"))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.8    0.1    1 none FALSE            TRUE       5     0.3      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: 199

set item appearances ...[2 item(s)] done [0.00s].
set transactions ...[492 item(s), 666 transaction(s)] done [0.00s].
sorting and recoding items ... [18 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 done [0.00s].
writing ... [20 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The results updated the previous rules variable, this time with precisely 20 rules for the all the observations with an 0.8 confidence and 0.3 support levels. To further explore these 20 rules I used the inspect command to assess the first 10 rules.

```
> # Inspect first 10 generated rules
> inspect(rules[1:10])
     lhs                                rhs       support   confidence coverage  lift     count
[1]  {PriorDefault=f}                => {class=-} 0.4429429 0.9394904  0.4714715 1.704906 295
[2]  {PriorDefault=t,Employed=t}     => {class=+} 0.3063063 0.9066667  0.3378378 2.019532 204
[3]  {BankCustomer=g,PriorDefault=t} => {class=+} 0.3588589 0.8298611  0.4324324 1.848453 239
[4]  {Married=u,PriorDefault=t}      => {class=+} 0.3588589 0.8298611  0.4324324 1.848453 239
[5]  {PriorDefault=t,Citizen=g}      => {class=+} 0.4054054 0.8108108  0.5000000 1.806020 270
[6]  {PriorDefault=f,CreditScore=0}  => {class=-} 0.3483483 0.9317269  0.3738739 1.690818 232
[7]  {PriorDefault=f,Employed=f}     => {class=-} 0.3483483 0.9317269  0.3738739 1.690818 232
[8]  {BankCustomer=g,PriorDefault=f} => {class=-} 0.3108108 0.9409091  0.3303303 1.707481 207
[9]  {Married=u,PriorDefault=f}      => {class=-} 0.3108108 0.9409091  0.3303303 1.707481 207
[10] {PriorDefault=f,Citizen=g}      => {class=-} 0.3948949 0.9528986  0.4144144 1.729238 263
>
```

The inspect command gave me a holistic picture of the rules, as limited by the class variable. I noticed that irrespective of the class value (+ or -), some of the rules were sharing a similar lift value. Furthermore, the "PriorDefault" variable was across all of the captured rules, highlighting the importance of such values throughout the credit approval decision process. The rule with the strongest lift of 2.019 was rule #2, involving "PriorDefault = t", and "Employed = t" with "class = +". Lastly, after observing rule #2 with the higher lift, I noticed these were not sorted in descending order, which prompted me to my next step of pruning the rules.

Prune the returned rules. Pruning or trimming the obtained rules is imperative in order to improve the quality of the rules, as some rules may be redundant. The first step was to sort the rules by their respective "lift". For that purpose I used the sort() command to sort by "lift" and save that sorting as rules.sorted, then reviewed the first 10 rules.

```
# Prune the returned rules #######
rules.sorted <- sort(rules, by="lift")

inspect(rules.sorted[1:10])
```

```
> rules.sorted <- sort(rules, by="lift")
> inspect(rules.sorted[1:10])
     lhs                                                rhs       support   confidence coverage  lift     count
[1]  {PriorDefault=t,Employed=t}                     => {class=+} 0.3063063 0.9066667  0.3378378 2.019532 204
[2]  {PriorDefault=t,Employed=t,Citizen=g}           => {class=+} 0.3048048 0.9062500  0.3363363 2.018604 203
[3]  {BankCustomer=g,PriorDefault=t,Citizen=g}       => {class=+} 0.3468468 0.8492647  0.4084084 1.891673 231
[4]  {Married=u,PriorDefault=t,Citizen=g}            => {class=+} 0.3468468 0.8492647  0.4084084 1.891673 231
[5]  {Married=u,BankCustomer=g,PriorDefault=t,Citizen=g} => {class=+} 0.3468468 0.8492647  0.4084084 1.891673 231
[6]  {BankCustomer=g,PriorDefault=t}                 => {class=+} 0.3588589 0.8298611  0.4324324 1.848453 239
[7]  {Married=u,PriorDefault=t}                      => {class=+} 0.3588589 0.8298611  0.4324324 1.848453 239
[8]  {Married=u,BankCustomer=g,PriorDefault=t}       => {class=+} 0.3588589 0.8298611  0.4324324 1.848453 239
[9]  {PriorDefault=t,Citizen=g}                      => {class=+} 0.4054054 0.8108108  0.5000000 1.806020 270
[10] {PriorDefault=f,Citizen=g}                      => {class=-} 0.3948949 0.9528986  0.4144144 1.729238 263
>
```

Followed the sorting steps, I created a subset matrix to help me identifying those rules with redundancy in order to exclude them.

```
# List redundant rules
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)
```

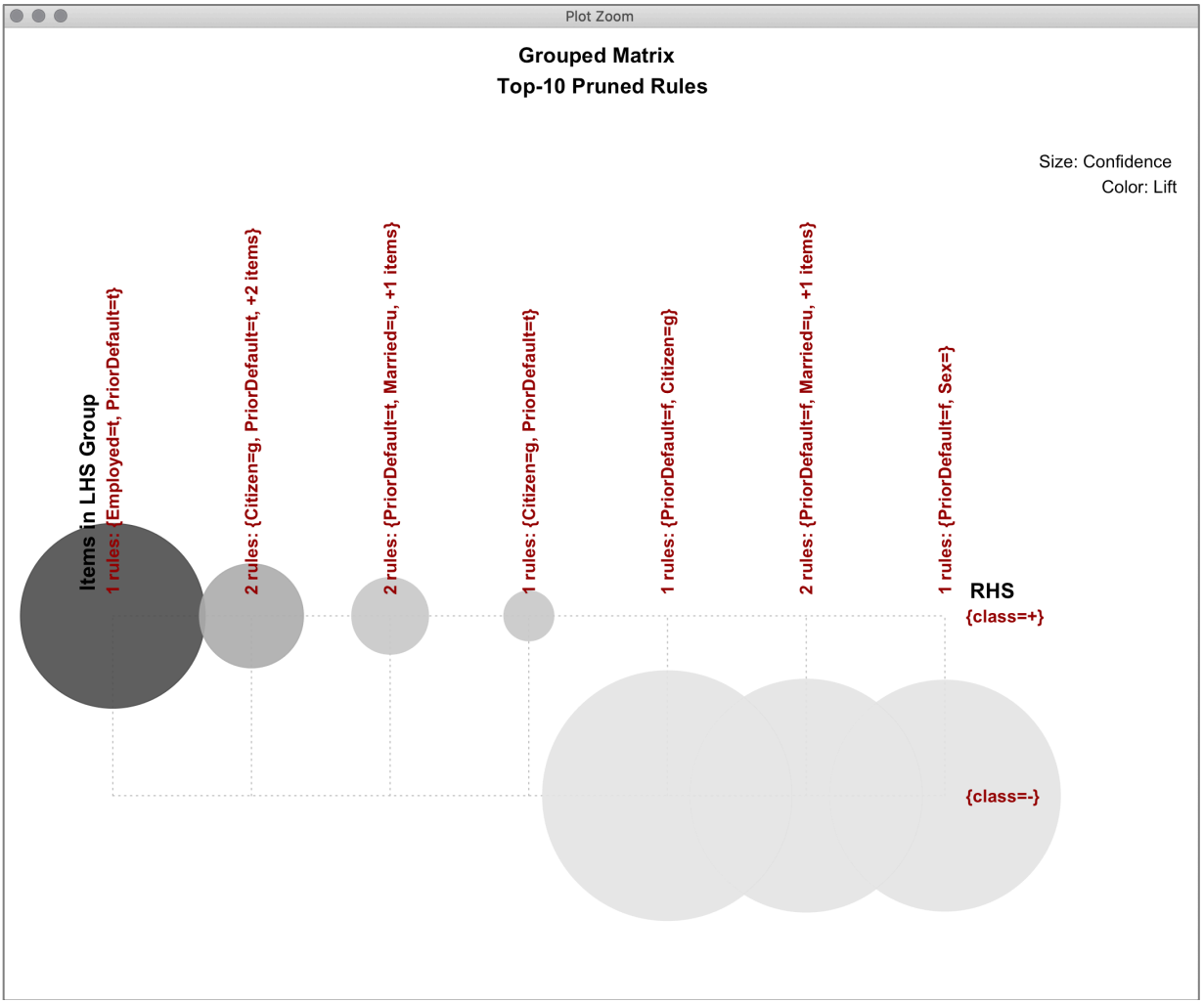
```
> # List redundant rules
> subset.matrix <- is.subset(rules.sorted, rules.sorted)
> subset.matrix[lower.tri(subset.matrix, diag=T)] <- F
> redundant <- colSums(subset.matrix, na.rm=T) >= 1
> which(redundant)
             {PriorDefault=t,Employed=t,Citizen=g,class=+} {Married=u,BankCustomer=g,PriorDefault=t,Citizen=g,class=+}
                                                        2                                                           5
         {Married=u,BankCustomer=g,PriorDefault=t,class=+}            {PriorDefault=f,CreditScore=0,Citizen=g,class=-}
                                                        8                                                          11
             {PriorDefault=f,Employed=f,Citizen=g,class=-} {PriorDefault=f,Employed=f,CreditScore=0,Citizen=g,class=-}
                                                       12                                                          13
         {Married=u,BankCustomer=g,PriorDefault=f,class=-}                    {PriorDefault=f,CreditScore=0,class=-}
                                                       16                                                          18
                        {PriorDefault=f,Employed=f,class=-}          {PriorDefault=f,Employed=f,CreditScore=0,class=-}
                                                       19                                                          20
>
```

The which( ) command results underlined exactly 10 out of 20 of the rules redundant. Afterwards the [!redundant] command was used to exclude the redundant rules and save the remaining rules as the “rules.pruned” vector. Finally, the inspect command was used against this created “rules.pruned”, to produce the last top-10 rules of the model.

```
> inspect(rules.pruned)
     lhs                                          rhs       support   confidence coverage  lift     count
[1]  {PriorDefault=t,Employed=t}               => {class=+} 0.3063063 0.9066667  0.3378378 2.019532 204
[2]  {BankCustomer=g,PriorDefault=t,Citizen=g} => {class=+} 0.3468468 0.8492647  0.4084084 1.891673 231
[3]  {Married=u,PriorDefault=t,Citizen=g}      => {class=+} 0.3468468 0.8492647  0.4084084 1.891673 231
[4]  {BankCustomer=g,PriorDefault=t}           => {class=+} 0.3588589 0.8298611  0.4324324 1.848453 239
[5]  {Married=u,PriorDefault=t}                => {class=+} 0.3588589 0.8298611  0.4324324 1.848453 239
[6]  {PriorDefault=t,Citizen=g}                => {class=+} 0.4054054 0.8108108  0.5000000 1.806020 270
[7]  {PriorDefault=f,Citizen=g}                => {class=-} 0.3948949 0.9528986  0.4144144 1.729238 263
[8]  {BankCustomer=g,PriorDefault=f}           => {class=-} 0.3108108 0.9409091  0.3303303 1.707481 207
[9]  {Married=u,PriorDefault=f}                => {class=-} 0.3108108 0.9409091  0.3303303 1.707481 207
[10] {PriorDefault=f}                          => {class=-} 0.4429429 0.9394904  0.4714715 1.704906 295
>
```

The above snap-shot shows the final top-10 rules of the assessment after using the Apriori method and the class attribute of the dataset. Notice the top rule with a lift greater than 2.0 and a confidence level of 90% that emphasizes “PriorDefault=t, “Employed=t, and “class=+” as the driving factors of the rule, capturing 204 of the observations, or applicants. These could result instrumental sifting through all the credit approval applications, prioritizing and expediting responses to the applicants.

Rules visualization. I decided to use a grouped matrix to illustrate the top-10 pruned rules of the model. This grouped matrix captures both, antecedents and consequents, provides an quick option to depict the strongest rules of the model. Per the below image, rule#1 [on the left side of the matrix] encompasses employment status, and prior default qualifiers. This rule, as shown by its size and hue, is the one with the greatest confidence level and lift ratio. Additionally, the subsequent 5 rules were also grouped as class "+". Conversely, the trailing 4 rules were categorized as class "-", and these also have a great confidence level for those within such class. Based on these findings, I believe class "+" refers to a sampled sector to receive positive credit approval, while class "-" a negative credit approval. However, additional background information is needed on the real meaning of these class symbols before confidently maintain such opinion. The only reason to believe the rule closest to the left-hand side is the strongest one, is by the combination of its confidence and lift characteristics.



Summary. In conclusion, by applying the Apriori algorithm unique association rules were generated, confirming that the application of a method such as the Apriori can greatly benefit a financial organization while reviewing or sifting through credit cases. Such practice can accelerate responses after a customer had submitted an application, and help organizations with their marketing strategies to target such exposed associations. The key for a method like the one here explored, is the combination of metrics, elevating the proportion value of the observations across both sides of the spectrum or rule support.

The lack of background information on the dataset or meaning of the altered values makes it hard to interpret the findings, thus, the observations are presented from a broad perspective. The greatest challenges while completing the exercise was getting familiar with the RStudio tools and options; making stackoverflow and google search engines a must.

References

Businesswire.com. (2020). Market basket Analysis Helped a Food Retailer to Increase Quarterly Sales by 50% | A Quantzig Success Story. https://www.businesswire.com/news/home/20200612005206/en/Market-basket-Analysis-Helped-Food-Retailer-Increase

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

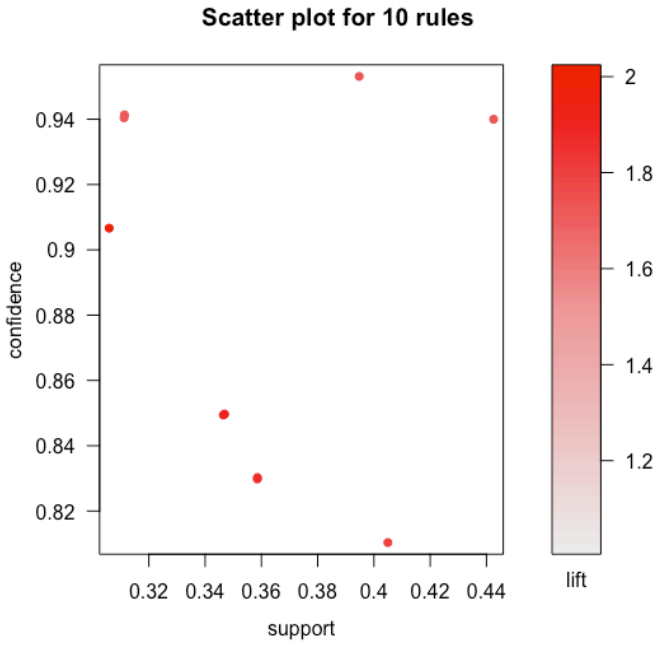
Ng, A. (2016). Association Rules and the Apriori Algorithm. https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html

Zhao, Y. (2015). R and Data Mining: Examples and Case Studies. http://www.RDataMining.com

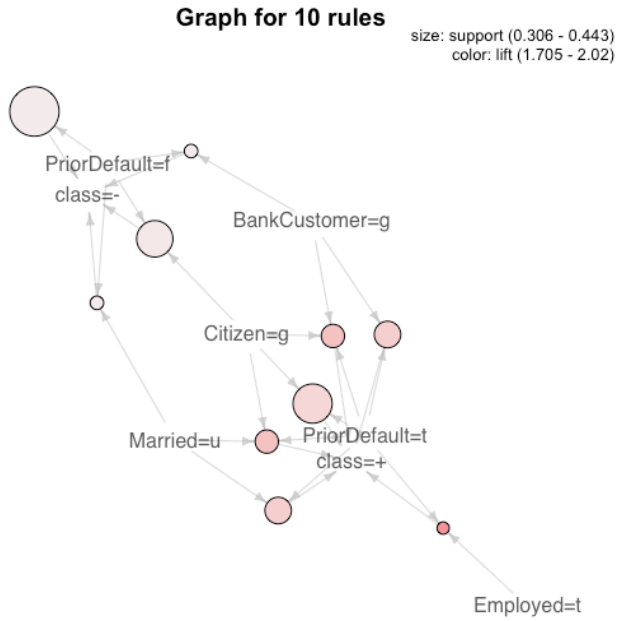
Appendix

Examples of alternative options when ploting the rules.

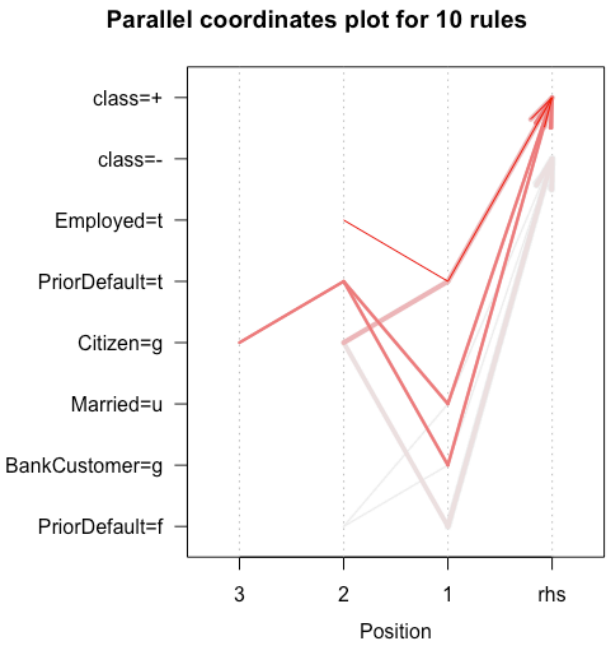
Command: plot(rules.pruned)



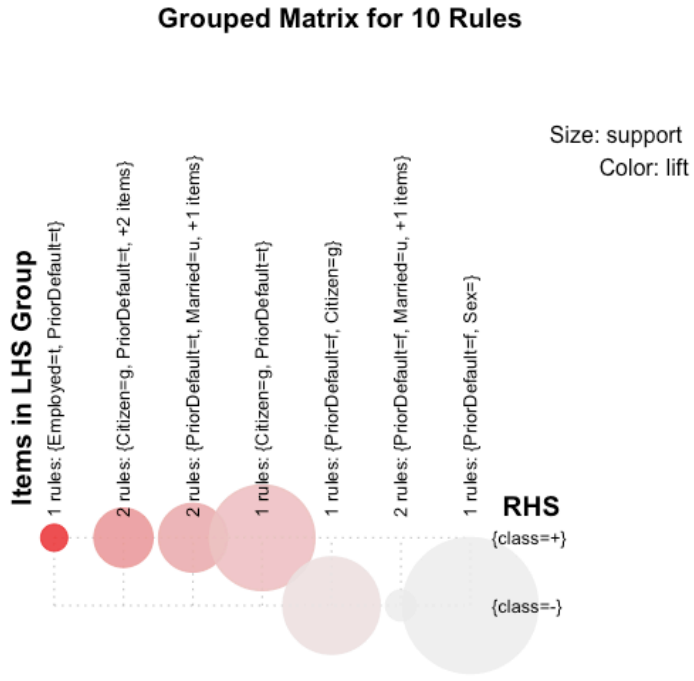
Command: plot(rules.pruned, method="graph", control=list(type="items"))



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



Command: plot(rules.pruned, method = "grouped")



Command: plot(rules.pruned, method="matrix", measure=c("lift", "confidence"))