Student Name: Brandon Russell

Submission Date: 6/3/18

DBST 667 – Data Mining

Dr. Irene Tsapara

Week 2 Individual Exercise

Deliverables: Two Files: (1) Submit this lab report with answers to all questions including output screenshots into the 'Individual Exercises Week 2' 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 CreditApproval.csv data set, 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:**

i. **Read the dataset description at UCI Machine Learning: Credit Approval. In your own words, describe your understanding of the dataset, what the attributes (columns) mean, and what each observation (row) represents.**

This is a small dataset with 16 attributes for 690 rows of data. While it is identified as credit approval data, it goes on to say the attribute names and values have been changed for confidentiality purposes, meaning the dataset would be strictly for education purposes. I'm saying educational because of the diverse mix

ii. **Run the read.csv() command to load the data into a variable named 'credit'. Then, run the command to preview the first 10 data rows in 'credit'. Include the command and output screenshot.** ***Note: Ensure that you use the utils::read.csv() command and not any other similar commands from other packages.***

Command: > credit <- utils::read.csv("CreditApproval.csv") / head(credit, n=

Output:

```
> head(credit, n=10)
   A1    A2     A3 A4 A5 A6 A7    A8 A9 A10 A11 A12 A13 A14   A15 class
1   b 30.83  0.000  u  g  w  v 1.250  t   t   1   f   g 202     0     +
2   a 58.67  4.460  u  g  q  h 3.040  t   t   6   f   g  43   560     +
3   a 24.50  0.500  u  g  q  h 1.500  t   f   0   f   g 280   824     +
4   b 27.83  1.540  u  g  w  v 3.750  t   t   5   t   g 100     3     +
5   b 20.17  5.625  u  g  w  v 1.710  t   f   0   f   s 120     0     +
6   b 32.08  4.000  u  g  m  v 2.500  t   f   0   t   g 360     0     +
7   b 33.17  1.040  u  g  r  h 6.500  t   f   0   t   g 164 31285     +
8   a 22.92 11.585  u  g cc  v 0.040  t   f   0   f   g  80  1349     +
9   b 54.42  0.500  y  p  k  h 3.960  t   f   0   f   g 180   314     +
10  b 42.50  4.915  y  p  w  v 3.165  t   f   0   t   g  52  1442     +
```

iii. Run the str() command on the 'credit' dataset. Include the command, output screenshot, and a brief description of how the structure is presented.

Command: > str(credit)

Output:

```
> str(credit)
'data.frame':   690 obs. of  16 variables:
 $ A1   : Factor w/ 3 levels "","a","b": 3 2 2 3 3 3 3 2 3 3 ...
 $ A2   : num  30.8 58.7 24.5 27.8 20.2 ...
 $ A3   : num  0 4.46 0.5 1.54 5.62 ...
 $ A4   : Factor w/ 4 levels "","l","u","y": 3 3 3 3 3 3 3 3 4 4 ...
 $ A5   : Factor w/ 4 levels "","g","gg","p": 2 2 2 2 2 2 2 2 4 4 ...
 $ A6   : Factor w/ 15 levels "","aa","c","cc",..: 14 12 12 14 14 11 13 4 10 14 ...
 $ A7   : Factor w/ 10 levels "","bb","dd","ff",..: 9 5 5 9 9 9 5 9 5 9 ...
 $ A8   : num  1.25 3.04 1.5 3.75 1.71 ...
 $ A9   : Factor w/ 2 levels "f","t": 2 2 2 2 2 2 2 2 2 2 ...
 $ A10  : Factor w/ 2 levels "f","t": 2 2 1 2 1 1 1 1 1 1 ...
 $ A11  : int  1 6 0 5 0 0 0 0 0 0 ...
 $ A12  : Factor w/ 2 levels "f","t": 1 1 1 2 1 2 2 1 1 2 ...
 $ A13  : Factor w/ 3 levels "g","p","s": 1 1 1 1 3 1 1 1 1 1 ...
 $ A14  : int  202 43 280 100 120 360 164 80 180 52 ...
 $ A15  : 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 ...
```

Description:

The str command is a useful function that shows a breakdown of each attribute, the type of value represented, and the different values and labels found within each attribute. For example, here we see that A1 consist of a factor with 3 different levels: "", "a", and "b". This is followed by a label for these factors for the first 10 rows in the dataset. In this case, "a" has a label of 2 and "b" has a label of "3". In the case of A2, it shows a type of num followed by the first 5 numbers rounded up.

b. Descriptive Statistics:

i. Run the summary() command on the 'credit' dataset to display the descriptive statistics for all variables. Include the command and output screenshot.

Command: > summary(credit)

Output:

```
> summary(credit)
 A1           A2               A3          A4        A5             A6          A7
  : 12   Min.   :13.75   Min.   : 0.000    :  6     :  6   c      :137   v      :399
a:210   1st Qu.:22.60   1st Qu.: 1.000   l:  2   g :519   q      : 78   h      :138
b:468   Median :28.46   Median : 2.750   u:519   gg:  2   w      : 64   bb     : 59
        Mean   :31.57   Mean   : 4.759   y:163   p :163   i      : 59   ff     : 57
        3rd Qu.:38.23   3rd Qu.: 7.207                    aa     : 54   dd     :  9
        Max.   :80.25   Max.   :28.000                    ff     : 53   j      :  8
        NA's   :12                                        (Other):245   (Other): 20
       A8          A9      A10          A11          A12     A13          A14
Min.   : 0.000   f:329   f:395   Min.   : 0.0   f:374   g:625   Min.   :   0
1st Qu.: 0.165   t:361   t:295   1st Qu.: 0.0   t:316   p:  8   1st Qu.:  75
Median : 1.000                   Median : 0.0           s: 57   Median : 160
Mean   : 2.223                   Mean   : 2.4                   Mean   : 184
3rd Qu.: 2.625                   3rd Qu.: 3.0                   3rd Qu.: 276
Max.   :28.500                   Max.   :67.0                   Max.   :2000
                                                                NA's   :13

      A15             class
Min.   :     0.0   -:383
1st Qu.:     0.0   +:307
Median :     5.0
Mean   :  1017.4
3rd Qu.:   395.5
Max.   :100000.0
```

ii. Choose two numeric attributes from 'credit', run the summary() command on both, and provide your interpretation of each of the six descriptive statistics.

Command: > summary(credit$A14)

Output:

```
> summary(credit$A14)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      0      75     160     184     276    2000      13
>
```

Command: > summary(credit$A2)

Output:

```
> summary(credit$A2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  13.75   22.60   28.46   31.57   38.23   80.25      12
>
```

Interpretation: The "Min" value tells us the lowest value for this attribute in the dataset. The "1st Qu.", or first quartile, marks the upper end of the lower 25% of values for this attribute. "Median" is simply the middle point of the data in this attribute. The "Mean" is the calculated average of values in this attribute. The "3rd Qu.", or third quartile, marks the lower end of the top 25% of values of this attribute. "Max" tells us the highest

iii. Choose two factor attributes from 'credit', run the summary() command on both, and provide your interpretation of each of the six descriptive statistics.

Command: > summary(credit$A1)

Output:

```
> summary(credit$A1)
      a   b
 12 210 468
```

Command: > summary(credit$A4)

Output:

```
> summary(credit$A4)
      l   u   y
 6    2 519 163
```

Interpretation: When we run the summary command on a factor it does not return the six descriptive statistics. Rather, it returns a list of the values found in this set and the number of instances of each. For example, in credit$A1 there are 12 occurrences of "", 210 of "a", and 468 of "b". In credit$A4 there are 6 occurrences of "", 2 of "l", 519 of "u", and 163 of "y".

iv. What differences did you observe between the output of the str() and summary() commands?

The str() command provided a general view of what each attribute was comprised of. It showed the type and a preview of the values within that attribute. The summary() command, on the other hand, provided descriptive statistics of each attribute. For example, when dealing with a number attribute it provided the min, 1st quartile, median, mean, 3rd quartile, max, and in some cases the NA values. When dealing with factors it showed the count of occurrence for each value.

c. Variable Filters – Discretization and Removing Variables:

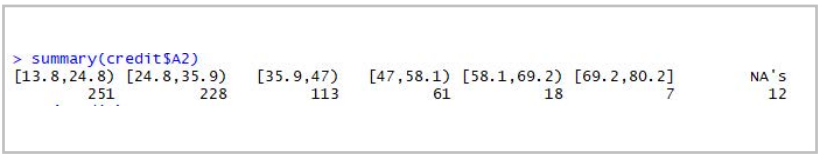
i. What is discretization? Provide a one-paragraph, masters-level response in your own words.

Discretization is the process of processing continuous numeric attributes into nominal categorical attributes algorithmically. This process can be done with various methods, both unsupervised and supervised. The overall purpose is to create discrete variables for other algorithms to use which cannot work with continuous variables. A simple example of this process would be dividing

ii. Run the three different discretization methods discussed in the tutorial (equal interval, equal frequency, k-means clustering). For each method, include the command and output screenshot. For all commands, provide a one-paragraph discussion of the input parameters used, the number of bins, and your interpretation of the output.

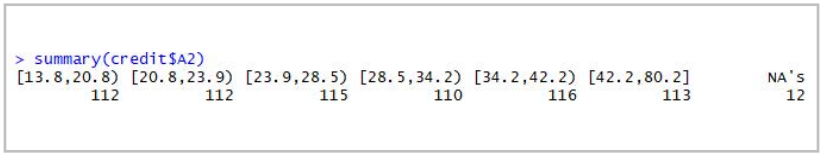
Command: > credit$A2<-discretize(credit$A2, "interval", breaks=6)

Output:



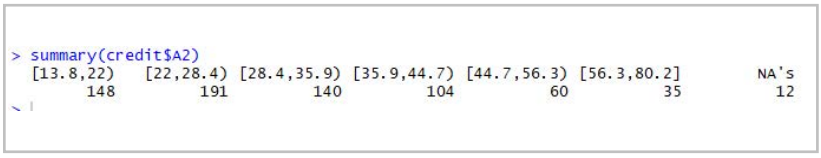
Command: > credit$A2<-discretize(credit$A2, "frequency", breaks=6)

Output:



Command: > credit$A2<-discretize(credit$A2, "cluster", breaks=6)

Output:



Discussion: In each command I used the credit$A2 data for purposes of comparing the results. I also specified 6 breaks in each command for the same reason. The first method used is the equal interval width. This method takes the number of breaks specified and splits the data into intervals of equal size. If we look at the command above we can see each bin is approximately a width of 11. This method could be affected by

iii. Compare and contrast the discretization methods above providing at least one example of when you would use each one.

The equal interval width method is an unsupervised parametric algorithm that splits data into a specified number of equal breaks. While it is susceptible to extreme outliers, it would be useful if we had a large set of continuous variables and we need to convert this into discrete nominal variables. If we take a look at the car data provided in this week's tutorial, we could apply this method to the cars$city_mpg field. In this case we could break our intervals into each widths of 6 to more easily visual how many cars have an acceptable mpg rating per government guidelines. In this case

iv. Run a command to remove one of the attributes from the 'credit' dataset. Run another command to demonstrate that the attribute was successfully removed. Include both commands and output screenshots as well as a discussion of when and why variables should be removed from a dataset.

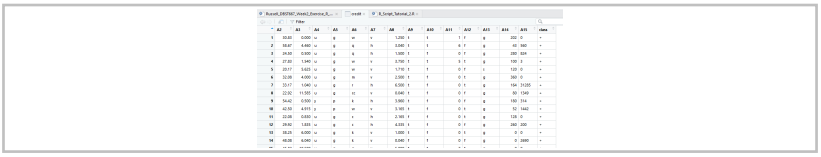
Command: > credit$A1 <- NULL

Output:

```
> credit$A1 <- NULL
```

Command: > View(credit)

Output:



Discussion: As we can see from the exercise above, removing an attribute is trivial, we only need to assign it to a "NULL" value. There are a number of reasons we might want to remove a variable from a dataset. One example, if we are getting ready to merge two different datasets and both have an ID column, there might be conflicting values and removing the whole column and starting new might be the best move. Additionally, we

d. Row Filters – Handling Missing Values and Sorting:

i. Run a command to check if the 'credit' dataset has any missing values. Your command and output should show all attributes along with how many observations total have missing values. Include the command and output screenshot.

Command: > apply(credit, 2, function (credit) sum(is.na(credit)))

Output:

```
> apply(credit, 2, function (credit) sum(is.na(credit)))
   A1    A2    A3    A4    A5    A6    A7    A8    A9   A10   A11   A12   A13   A14   A15 class
    0    12     0     0     0     0     0     0     0     0     0     0     0    13     0     0
```

ii. Choose one of the numeric attributes with missing values and run the command to replace the missing values with the attribute mean. Then run the command to verify that the variable no longer has missing values. Include both commands and output screenshots.

Command: > credit$A2[is.na(credit$A2)]<-mean(credit$A2, na.rm=TRUE)

Output:

```
> credit$A2[is.na(credit$A2)]<-mean(credit$A2, na.rm=TRUE)
```

Command: > apply(credit, 2, function (credit) sum(is.na(credit)))

Output:

```
> apply(credit, 2, function (credit) sum(is.na(credit)))
   A1    A2    A3    A4    A5    A6    A7    A8    A9   A10   A11   A12   A13   A14   A15 class
    0     0     0     0     0     0     0     0     0     0     0     0     0    13     0     0
```

iii. Why is it important to handle missing values in your dataset prior to beginning your primary data analysis? Provide a one-paragraph, masters-level response in your own words.

Managing missing values in your dataset is a critical part of data preprocessing. If data is missing it could disrupt the results of the algorithm we are attempting to run. While some missing values could be acceptable, it is important for us to assess this and make an educated decision on whether to leave the values as NULL, replace with the attribute mean value, or fill with some other predetermined value to meet the requirements of the analysis we are conducting.

iv. Describe at least one alternative approach for handling missing values other than replacing the values with the attribute mean. Provide a one-paragraph, masters-level response in your own words.

One method I have used in previous database migration when there are a large volume of missing values is to identify a universal placeholder, such as "-1" or "NA depending on the dataset. I would run a script to place this value in all fields missing a value and trained my reporting tools to look for and ignore these fields in the calculations. Another option would be to fill in each missing value manually. This is arguably the best method to ensure data accuracy, however it will usually be the most time consuming and likely not practiced due to the size of most datasets that are useful for analysis

v. Run the command to sort the 'credit' dataset by one of the attributes. Then run the command to validate the sorting. Include both commands and output screenshots as well as a discussion where you provide at least two reasons why data should be sorted.

Command: > credit_sorted<-credit[order(credit$A6),]

Output:

```
> credit_sorted<-credit[order(credit$A6), ]
```

Command: > head(credit_sorted$A6)

Output:

```
> head(credit_sorted$A6)
[1]
Levels:  aa c cc d e ff i j k m q r w x
>
```

Discussion:

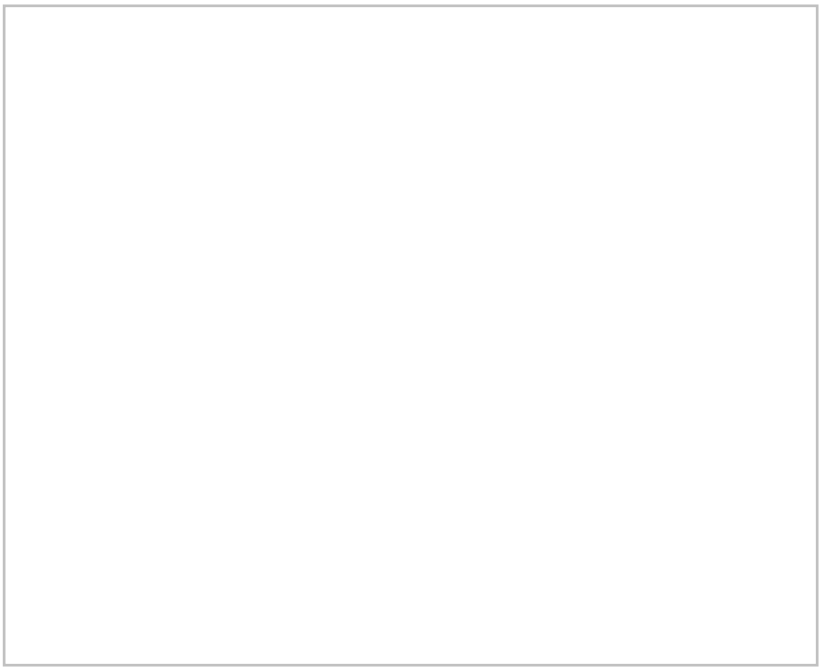
Two important reasons we want to sort our data is readability for the user and algorithm efficiency. The first of these seems obvious enough, it is much easier for a person to read a sorted list, alphabetical or numeric, versus an unsorted list. In terms of algorithm efficiency, let's discuss the difference between a merge sort and a bubble sort algorithm, and then how a binary search works. Merge sort takes a divide-and-conquer approach to sorting elements. This method is O (n log n), a combination of linear and logarithmic complexity. Bubble sort, on the other hand, is O (n^2). This quadratic complexity is since a bubble sort runs through list for each element, checking to see if it is greater than or less than the next element. As we can see

e. Data Visualization:

i. Run the plot() function for one of the variables in the 'credit' dataset. Include the command, output screenshot, and a one-paragraph, masters-level interpretation of what the plot shows.

Command: >

Output:



Discussion:

f. Summary:

i. Why is data pre-processing important? Describe at least two advantages that pre-processing results in as well as two disadvantages of not pre-processing. Provide a one-paragraph, masters-level response in your own words.

ii. What differences did you observe between variable filters and row filters? Provide at least one scenario for each filter type where implementing the filter would benefit your data analysis. Provide a one-paragraph, masters-level response in your own words.

iii. (Not Graded) Which section of this exercise did you find the most challenging? What approach did you take to resolve this challenge? Were there any sections from this report or the tutorial which could use additional focus?

References