Advanced Stats Assignment 1

2023-01-21

Data Exploration Exercise

1. Source and file format

Question(a)

Download the data source from the Machine Learning Repository at http://archive.ics.uci.edu/ml/datasets/Credit+Approval

```
### Download the data to local, no R code needed ###
```

Question(b)

Convert the crx.data file to a CSV file.

```
# Convert crx.data to csv
data <- read_csv("crx.data")
```

```
## New names:
## Rows: 689 Columns: 16
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (10): b, 30.83, u, g...5, w, v, 01, g...13, 00202, + dbl (3): 0...3, 1.25,
## 0...15 lgl (3): t...9, t...10, f
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `0` -> `0...3`
## * `g` -> `g...5`
## * `t` -> `t...9`
## * `t` -> `t...10`
## * `g` -> `g...13`
## * `0` -> `0...15`
```

```
write.csv(data, file = "data.csv")
```

Question(c)

Define a variable called source path. Assign to it the full path of the source file, which the data are to be read from (as saved in your local environment).

```
# Set source path
source_path <- "~/Desktop/UC Davis/Winter/Advanced Stats/data.csv"
```

2. Load data into R, indexing and printing

Question(a)

Write a command that import the CSV file. Save into a dataframe called data.

```
# Import csv file
data <- read.csv("data.csv", header=TRUE, na.strings=c("?"))
data <- data[,-1]
```

Question(b)

Write a command that shows only the first 10 observations.

```
# Show first 10 observations
head(data, 10)
```

```
##    b X30.83 X0...3 u g...5  w v X1.25 t...9 t...10 X01     f g...13 X00202
## 1  a  58.67  4.460 u     g  q h 3.040  TRUE   TRUE   6 FALSE      g     43
## 2  a  24.50  0.500 u     g  q h 1.500  TRUE  FALSE   0 FALSE      g    280
## 3  b  27.83  1.540 u     g  w v 3.750  TRUE   TRUE   5  TRUE      g    100
## 4  b  20.17  5.625 u     g  w v 1.710  TRUE  FALSE   0 FALSE      s    120
## 5  b  32.08  4.000 u     g  m v 2.500  TRUE  FALSE   0  TRUE      g    360
## 6  b  33.17  1.040 u     g  r h 6.500  TRUE  FALSE   0  TRUE      g    164
## 7  a  22.92 11.585 u     g cc v 0.040  TRUE  FALSE   0 FALSE      g     80
## 8  b  54.42  0.500 y     p  k h 3.960  TRUE  FALSE   0 FALSE      g    180
## 9  b  42.50  4.915 y     p  w v 3.165  TRUE  FALSE   0  TRUE      g     52
## 10 b  22.08  0.830 u     g  c h 2.165 FALSE  FALSE   0  TRUE      g    128
##    X0...15 X.
## 1       560  +
## 2       824  +
## 3         3  +
## 4         0  +
## 5         0  +
## 6     31285  +
## 7      1349  +
## 8       314  +
## 9      1442  +
## 10        0  +
```

Question(c)

Write a command that shows only the target attribute (the true class).

```
# Rename the columns
colnames(data) <- c("V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9", "V10", "V11", "V12", "V13", "

# Show only the target attribute
which(data$V16 == "+")
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69 117 118 119
```

```
##  [73] 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137
##  [91] 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155
## [109] 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173
## [127] 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191
## [145] 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209
## [163] 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227
## [181] 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245
## [199] 246 247 248 249 250 251 252 253 268 269 270 317 318 319 320 321 322 323
## [217] 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507
## [235] 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 547 548 549
## [253] 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567
## [271] 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585
## [289] 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 606 621 622
```

3. Data Exploration

Question(a)

Write a command that show features data type.

```
# Show data type
str(data)
```

```
## 'data.frame':    689 obs. of  16 variables:
##  $ V1 : chr  "a" "a" "b" "b" ...
##  $ V2 : num  58.7 24.5 27.8 20.2 32.1 ...
##  $ V3 : num  4.46 0.5 1.54 5.62 4 ...
##  $ V4 : chr  "u" "u" "u" "u" ...
##  $ V5 : chr  "g" "g" "g" "g" ...
##  $ V6 : chr  "q" "q" "w" "w" ...
##  $ V7 : chr  "h" "h" "v" "v" ...
##  $ V8 : num  3.04 1.5 3.75 1.71 2.5 ...
##  $ V9 : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ V10: logi  TRUE FALSE TRUE FALSE FALSE FALSE ...
##  $ V11: int  6 0 5 0 0 0 0 0 0 0 ...
##  $ V12: logi  FALSE FALSE TRUE FALSE TRUE TRUE ...
##  $ V13: chr  "g" "g" "g" "s" ...
##  $ V14: int  43 280 100 120 360 164 80 180 52 128 ...
##  $ V15: num  560 824 3 0 0 ...
##  $ V16: chr  "+" "+" "+" "+" ...
```

```
sapply(data, class)
```

```
##          V1          V2          V3          V4          V5          V6
## "character"   "numeric"   "numeric" "character" "character" "character"
##          V7          V8          V9         V10         V11         V12
## "character"   "numeric"   "logical"   "logical"   "integer"   "logical"
##         V13         V14         V15         V16
## "character"   "integer"   "numeric" "character"
```

Question(b)

Create a summary statistics; mean, standard deviation, variance, min, max, median, range, and quantile – for each feature in the data.

```
# Create summary statistics
summary(data)
```

```
##       V1                  V2              V3              V4
##  Length:689         Min.   :13.75   Min.   : 0.000   Length:689
##  Class :character   1st Qu.:22.58   1st Qu.: 1.000   Class :character
##  Mode  :character   Median :28.42   Median : 2.750   Mode  :character
##                     Mean   :31.57   Mean   : 4.766
##                     3rd Qu.:38.25   3rd Qu.: 7.250
##                     Max.   :80.25   Max.   :28.000
##                     NA's   :12
##       V5                  V6                  V7                  V8
##  Length:689         Length:689         Length:689         Min.   : 0.000
##  Class :character   Class :character   Class :character   1st Qu.: 0.165
##  Mode  :character   Mode  :character   Mode  :character   Median : 1.000
##                                                           Mean   : 2.225
##                                                           3rd Qu.: 2.625
##                                                           Max.   :28.500
##
##      V9              V10               V11             V12
##  Mode :logical   Mode :logical   Min.   : 0.000   Mode :logical
##  FALSE:329       FALSE:395       1st Qu.: 0.000   FALSE:373
##  TRUE :360       TRUE :294       Median : 0.000   TRUE :316
##                                  Mean   : 2.402
##                                  3rd Qu.: 3.000
##                                  Max.   :67.000
##
##      V13                 V14               V15             V16
##  Length:689         Min.   :   0.0   Min.   :     0   Length:689
##  Class :character   1st Qu.:  74.5   1st Qu.:     0   Class :character
##  Mode  :character   Median : 160.0   Median :     5   Mode  :character
##                     Mean   : 184.0   Mean   :  1019
##                     3rd Qu.: 277.0   3rd Qu.:   396
##                     Max.   :2000.0   Max.   :100000
##                     NA's   :13
```

Question(c)

Write a command that shows features unique (distinct) values.

```
# Show unique values for categorical features
unique(data$V1)
```

```
## [1] "a" "b" NA
```

```
unique(data$V4)
```

```
## [1] "u" "y" NA  "l"
```

```
unique(data$V5)
```

```
## [1] "g"  "p"  NA   "gg"
```

```
unique(data$V6)
```

```
##  [1] "q"  "w"  "m"  "r"  "cc" "k"  "c"  "d"  "x"  "i"  "e"  "aa" "ff" "j"  NA
```

```
unique(data$V9)
```

```
## [1]  TRUE FALSE
```

```
unique(data$V10)
```

```
## [1]  TRUE FALSE
```

```
unique(data$V12)
```

```
## [1] FALSE  TRUE
```

```
unique(data$V13)
```

```
## [1] "g" "s" "p"
```

```
unique(data$V16)
```

```
## [1] "+" "-"
```

Question(d)

Compute the "Skewness" measure of the target variable. Based on this value, can we apply a balanced classification technique?

Based on the value, the V16 data is imbalanced but right skewed (skewness > 0), therefore a balanced classification technique can not be applied here.

```
# Transform the data
data$V16 <- ifelse(data$V16 == "+", 1, 0)

# Function to compute skewness
skew <- function(x){
  m3 <- sum((x-mean(x))^3)/length(x)
  s3 <- sqrt(var(x))^3
  m3/s3
}

# See how many 1 are there in V16
length(which(data$V16 == "1"))
```

```
## [1] 306
```

```r
# Compute skewness
skew(data$V16)
```

```
## [1] 0.2244318
```

Question(e)

Compute the "Skewness" measure of the variable V1. What does it tell us about V1 variable?

Based on the positive value of skewness, the V1 data is right skewed.

```r
# Transform the data
data$V1 <- ifelse(data$V1 == "a", 1, 0)

# See how many 1 are there in V1
length(which(data$V1 == "1"))
```

```
## [1] 210
```

```r
# Omit NA and compute skewness
skew(na.omit(data$V1))
```

```
## [1] 0.8188457
```

Question(f)

Compute the "Kurtosis" measure of the variable V8. Is it from a Gaussian distribution?

No, V8 is not from a Gaussian distribution, since its kurtosis is not 0.

```r
# Function to compute kurtosis
kurtosis <- function(x){
  m4 <- sum((x-mean(x))^4)/length(x)
  s4 <- var(x)^2
  m4/s4 - 3
}

# Kurtosis of V8
kurtosis(data$V8)
```

```
## [1] 11.04778
```

Question(g)

Assume V10 variable indicates if the person has a first degree. Compute the frequency table, marginal frequency and row- wise proportions of V10 and the target variable.

```r
# Frequency table
crosstab <- table(data$V10, data$V16)
crosstab
```

```
##
##           0   1
##   FALSE 297  98
##   TRUE   86 208
```

```r
# Marginal Frequency
margin.table(crosstab, 1)
```

```
##
## FALSE  TRUE
##   395   294
```

```r
margin.table(crosstab, 2)
```

```
##
##   0   1
## 383 306
```

```r
# Proportion
prop.table(crosstab, 1)
```

```
##
##                 0         1
##   FALSE 0.7518987 0.2481013
##   TRUE  0.2925170 0.7074830
```

```r
# Accuracy
(208) / (208 + 86)
```

```
## [1] 0.707483
```

4. Missing Values

Question(a)

Write a command that finds all missing values

```r
# Find missing values
which(is.na(data) == TRUE)
```

```
##  [1]  248  327  346  374  453  479  489  520  598  601  641  673  772  775  781
## [16]  786  943  975 1018 1134 1139 1189 1204 1297 2273 2337 2397 2523 2659 2689
## [31] 2962 3026 3086 3212 3348 3378 3651 3715 3775 3901 3924 3984 4037 4046 4067
## [46] 4340 4404 4464 4590 4613 4673 4726 4735 4756 9028 9159 9163 9200 9227 9235
## [61] 9287 9363 9402 9413 9549 9579 9583
```

Question(b)

Categorical variables – replace all missing values for a variable with the most frequent value

```r
# Replace NA with mode
mode <- function(x){
  distinct_values <- unique(x)
  distinct_tabulate <- tabulate(match(x, distinct_values))
  distinct_values[which.max(distinct_tabulate)]
```

```
}

for(i in 1:ncol(data)){
  data[is.na(data[,i]), i] <- mode(data[,i])
}
```

5. Data Normalization and Transformation

Question(a)

Replace the target attribute to a binary attribute (logical data type).

```
# Replace target attribute to a binary attribute
data$V16 <- as.logical(data$V16)
```

Question(b)

Discretization - discretize variable V2 using the equal frequencies or equal width binning algorithm.

```
# Equal width binning
classIntervals(data$V2, 5, style = 'equal')
```

```
## Warning in classIntervals(data$V2, 5, style = "equal"): var has missing values,
## omitted in finding classes
```

```
## style: equal
## [13.75,27.05) [27.05,40.35) [40.35,53.65) [53.65,66.95) [66.95,80.25]
##           305           233            96            34             9
```

Question(c)

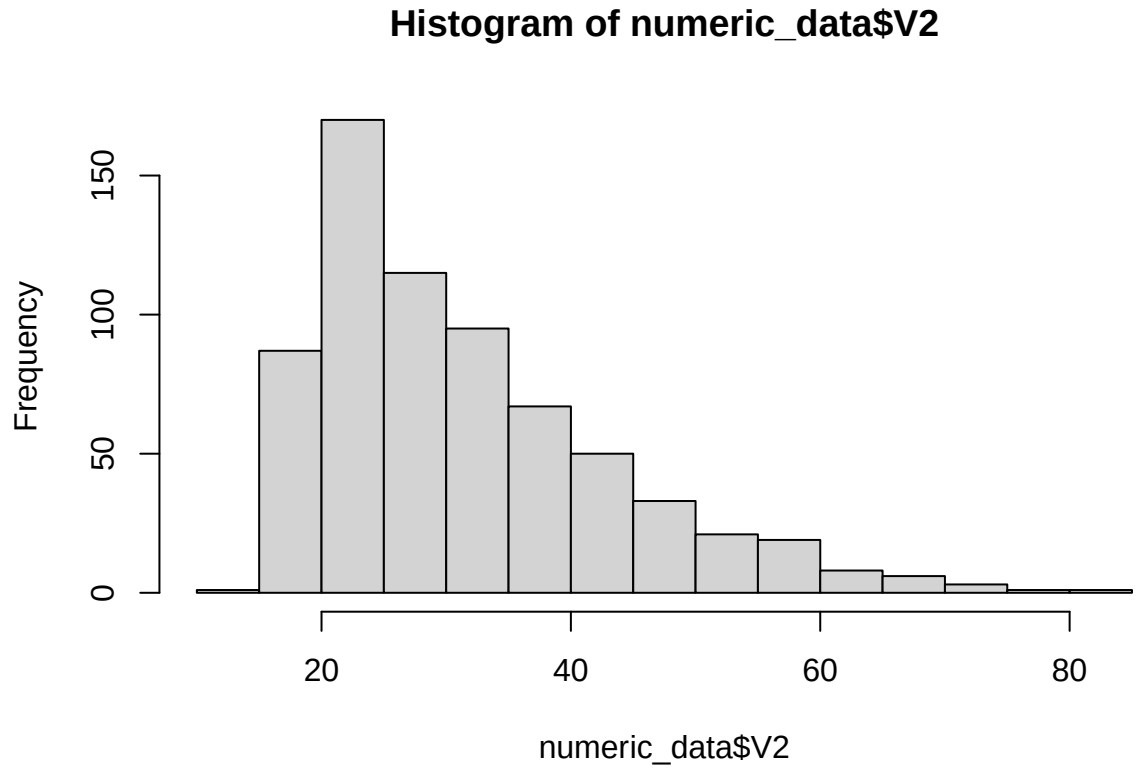
Scaling - after examining the range and distribution of each feature (explain how you did that), apply on each numerical feature a relevant scaling. Use at least 2 different scaling methods and explain why each method was chosen.

According to the summary statistics of numeric data, the distribution of different variables varies dramatically, we definitely need to scale them to eliminate units of measurements. Two scaling methods chosen here are standardization and normalization. Standardization is the process of transforming data so that the new data will have a mean of 0 and standard deviation of 1, whereas normalization transforms the data to a range of 0 to 1. Based on the histograms, it's pretty obvious that the data does not come from a normal distribution, therefore normalization scaling method can be chosen. Also, since standardization has no bounding range, making it more flexible at handling outliers, it is chosen as the second scaling method.

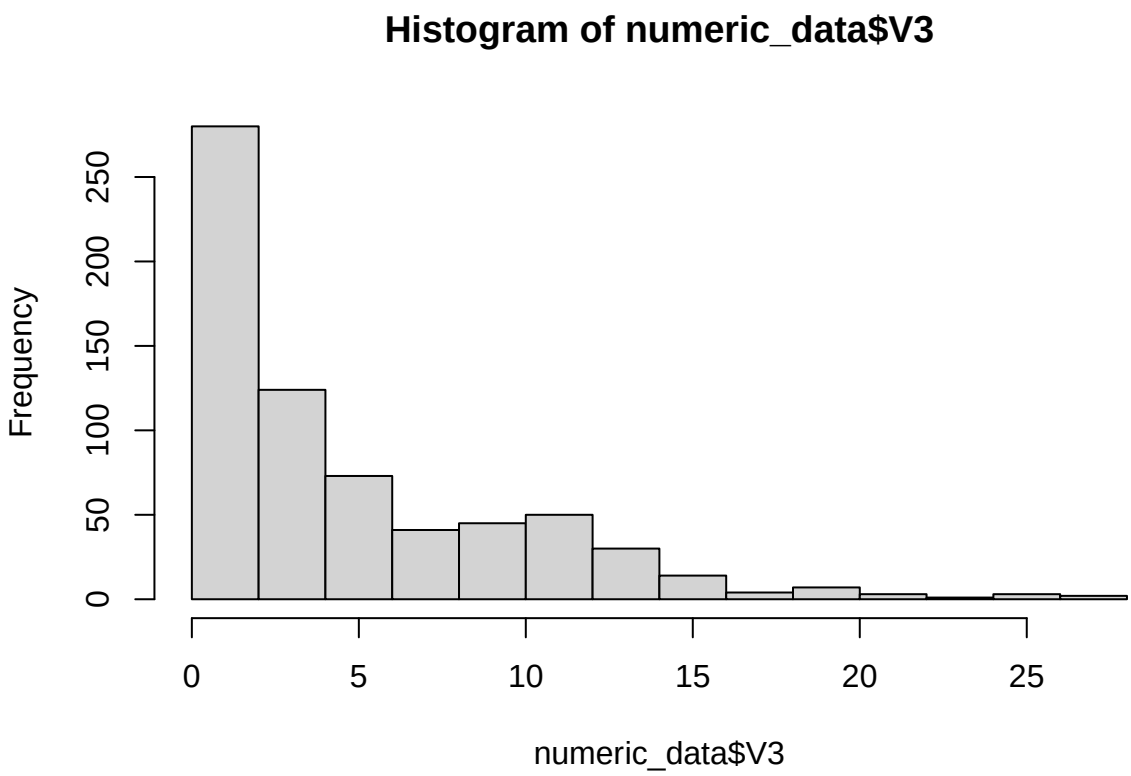
```
# Subset numeric columns
numeric <- c("V2", "V3", "V8", "V11", "V14", "V15")
numeric_data <- data[numeric]
numeric_data <- na.omit(numeric_data)

# Examine the range and distribution of each feature
summary(numeric_data)
```

```
##        V2              V3               V8              V11
##  Min.   :13.75   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:22.58   1st Qu.: 1.000   1st Qu.: 0.165   1st Qu.: 0.000
##  Median :28.42   Median : 2.750   Median : 1.000   Median : 0.000
##  Mean   :31.57   Mean   : 4.785   Mean   : 2.211   Mean   : 2.437
##  3rd Qu.:38.25   3rd Qu.: 7.500   3rd Qu.: 2.585   3rd Qu.: 3.000
##  Max.   :80.25   Max.   :28.000   Max.   :28.500   Max.   :67.000
##       V14             V15
##  Min.   :   0.0   Min.   :     0
##  1st Qu.:  60.0   1st Qu.:     0
##  Median : 154.0   Median :     5
##  Mean   : 178.9   Mean   :  1023
##  3rd Qu.: 260.0   3rd Qu.:   396
##  Max.   :2000.0   Max.   :100000
```

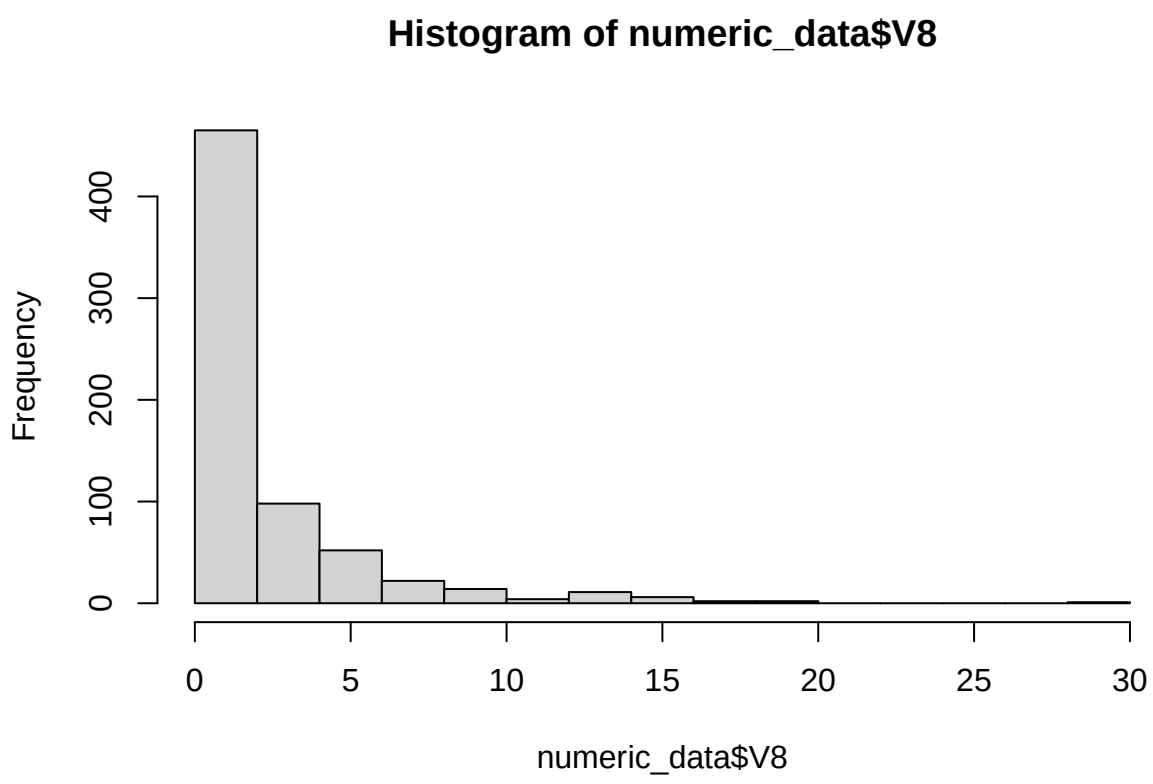
```
# Plot histogram to see if data come from a normal distribution
hist(numeric_data$V2)
```



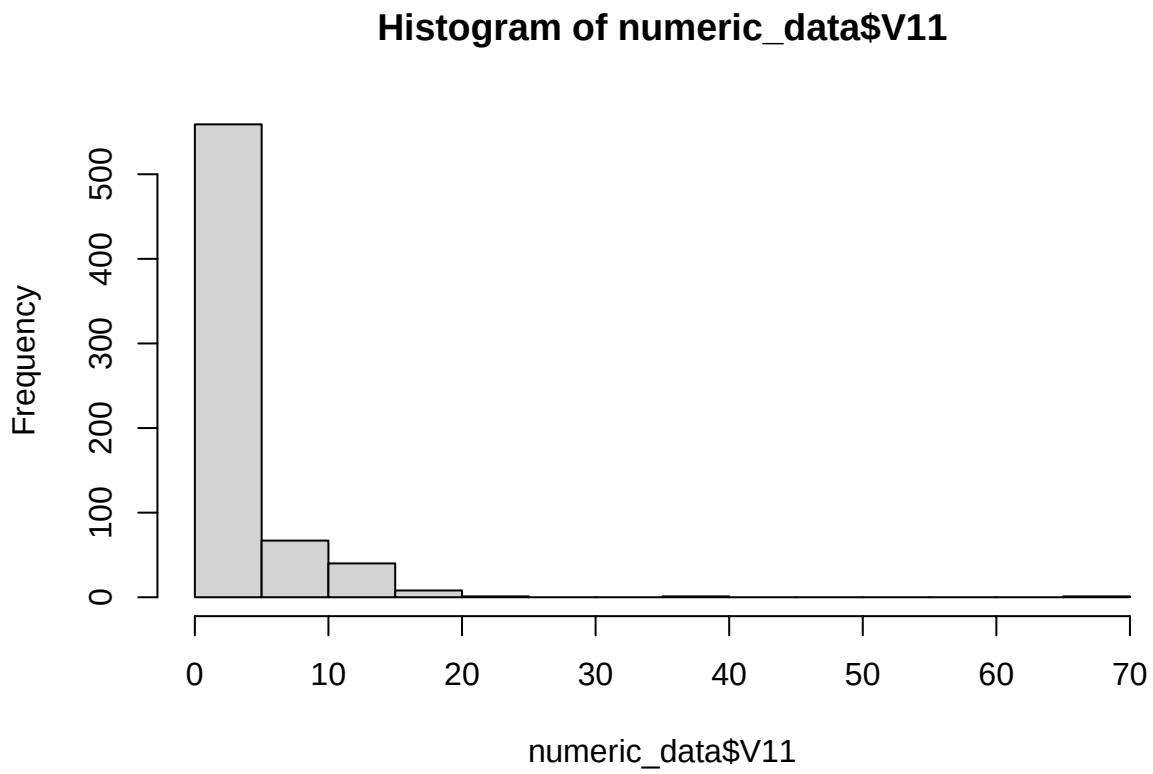

```
hist(numeric_data$V3)
```



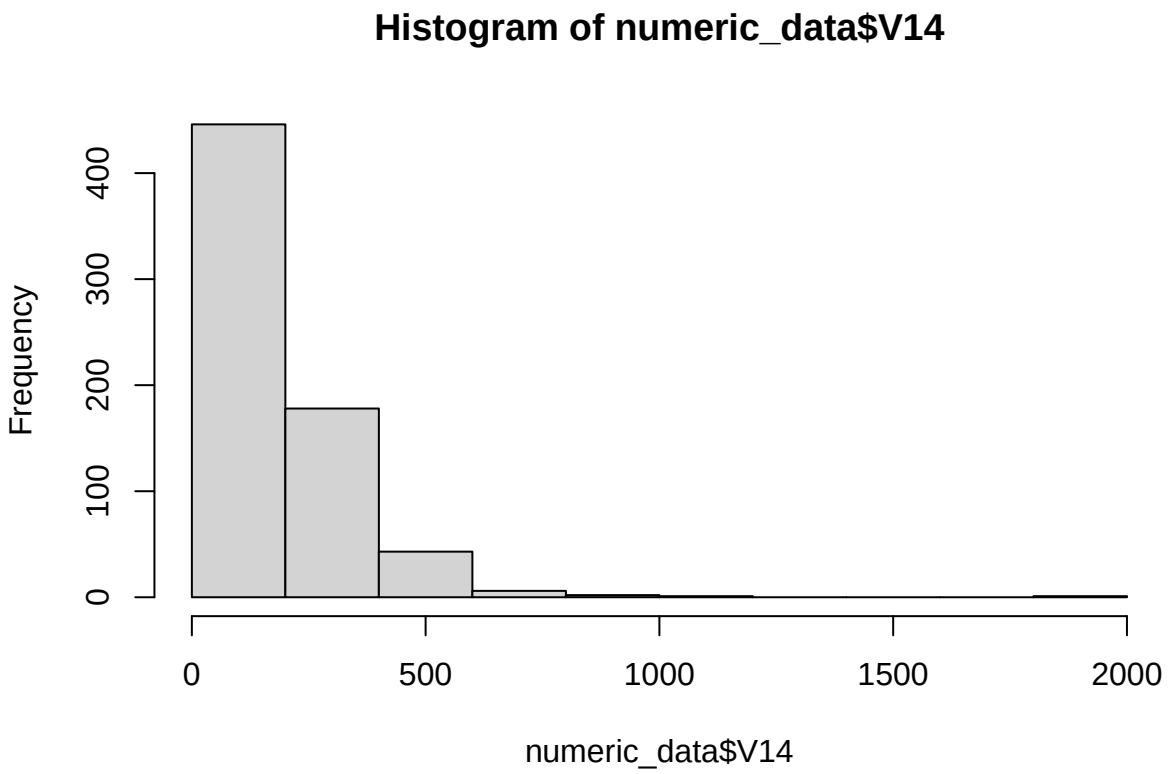

```
hist(numeric_data$V8)
```



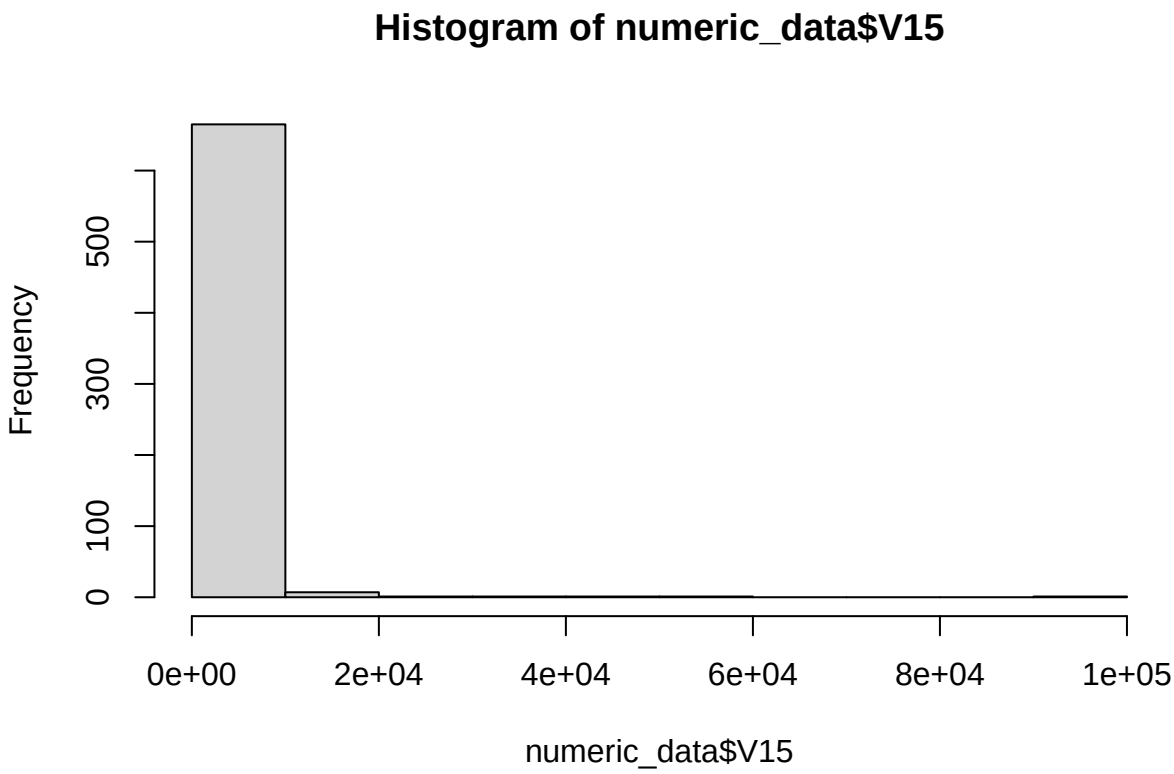

```
hist(numeric_data$V11)
```



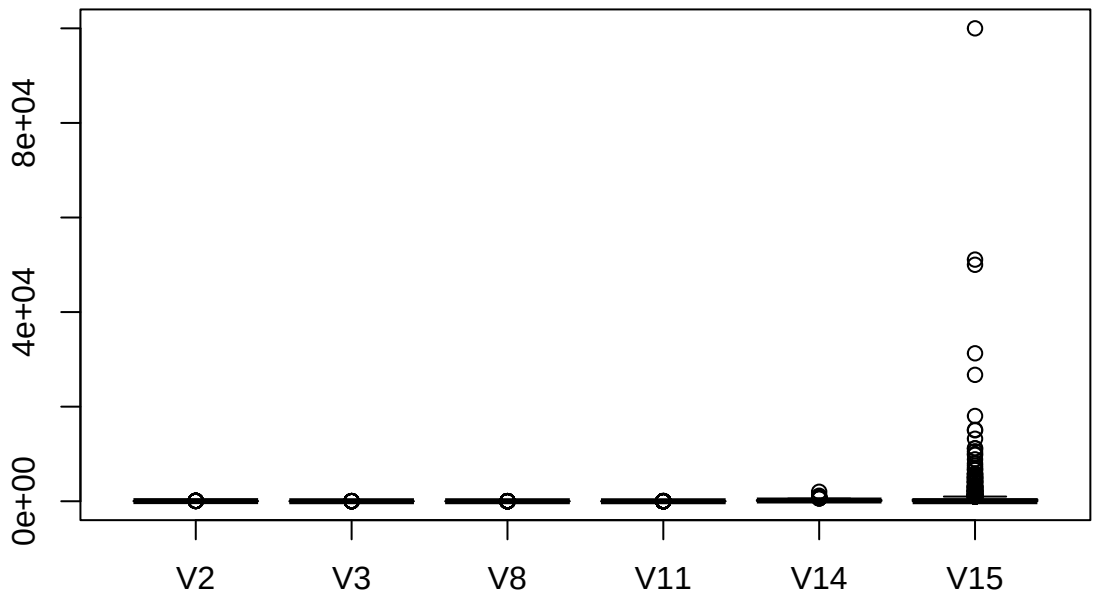

```
hist(numeric_data$V14)
```




```
hist(numeric_data$V15)
```




```
# Plot boxplot to see outliers
boxplot(numeric_data)
```




```
# First Method: Standardization
numeric_data <- as.data.frame(scale(numeric_data))

# Second Method: Normalization
numeric_data <- as.data.frame(sapply(numeric_data, function(x) (x-min(x))/(max(x)-min(x))))
```

6. Outliers and Visualizations

Question(a)

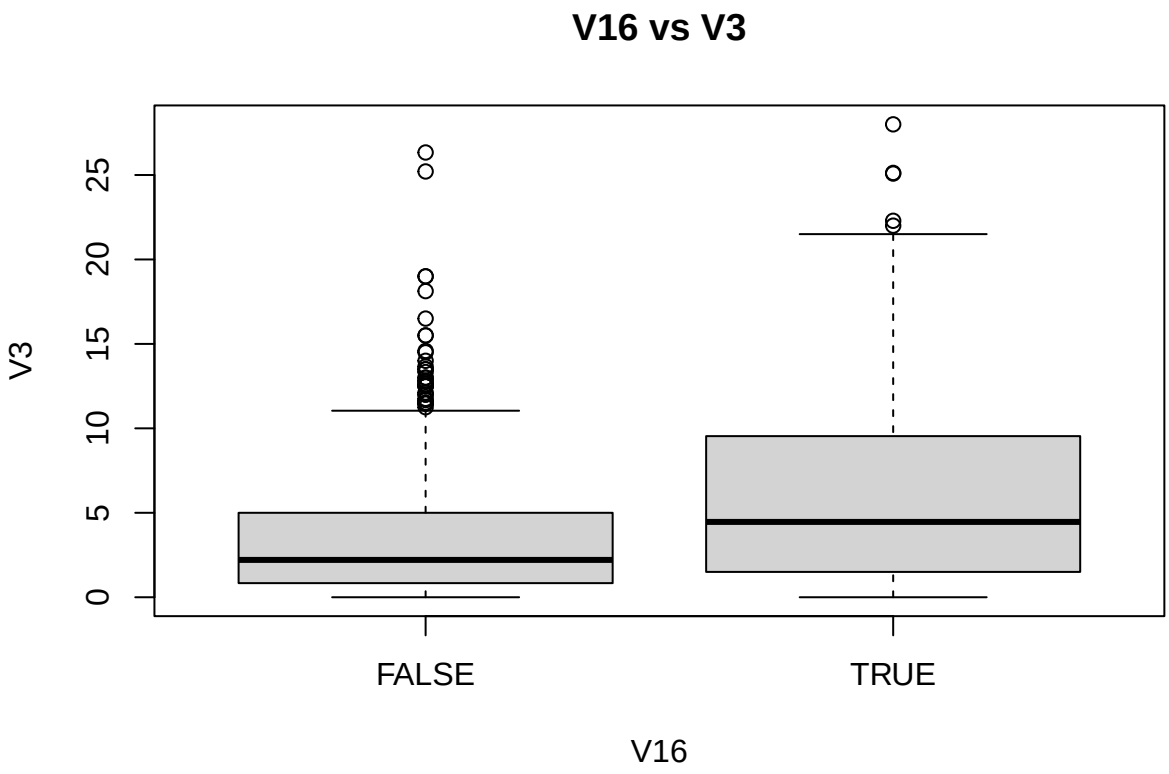
Create a boxplot of the target variable by V3.

i. What can be inferred about V3 by this graphical presentation?

ii. Find which samples are outliers (the output should be row numbers)

The value of V3 tends to be higher when V16=TRUE, suggesting that there may be an association between V3 and V16.

```
boxplot(V3~V16,data=data, main="V16 vs V3",
   xlab="V16", ylab="V3")
```




```
outliers <- boxplot(V3~V16,data=data, main="V16 vs V3",
   xlab="V16", ylab="V3", plot = FALSE)$out
which(data$V3 %in% outliers)
```

```
##  [1]  20  23  26  37  44  69  80  90  91 101 103 105 112 121 122 139 155 164 172
## [20] 176 205 212 213 217 241 248 278 284 299 304 305 317 333 352 368 371 376 384
## [39] 388 392 398 445 455 470 478 485 487 508 529 549 550 554 566 573 617 625 628
## [58] 653 666 671 687
```

Question(b)

Select 2 numerical features to plot against each other and add a regression line. Explain the plot.

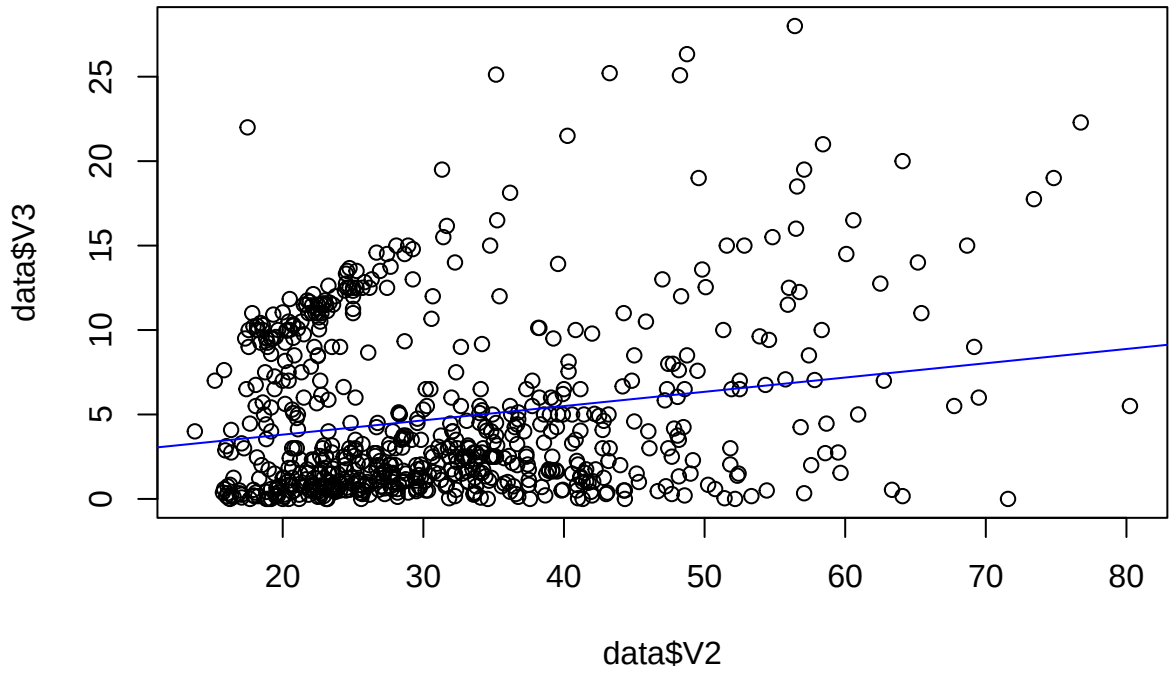
Based on the shape of scatter plot, V2 and V3 have slight positive correlation. However, we can observe the values of V2 concentrate between 20 to 30, suggesting some transformation (such as log transformation) should be performed to choose the suitable functional form of regression model.

```
# Linear model
model <- lm(V3 ~ V2, data = data)
summary(model)
```

```
##
## Call:
## lm(formula = V3 ~ V2, data = data)
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -8.166 -3.432 -1.991  2.580 21.115
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.11667    0.53141   3.983 7.54e-05 ***
## V2           0.08451    0.01574   5.369 1.09e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.898 on 675 degrees of freedom
##   (12 observations deleted due to missingness)
## Multiple R-squared:  0.04095,    Adjusted R-squared:  0.03953
## F-statistic: 28.82 on 1 and 675 DF,  p-value: 1.092e-07
```

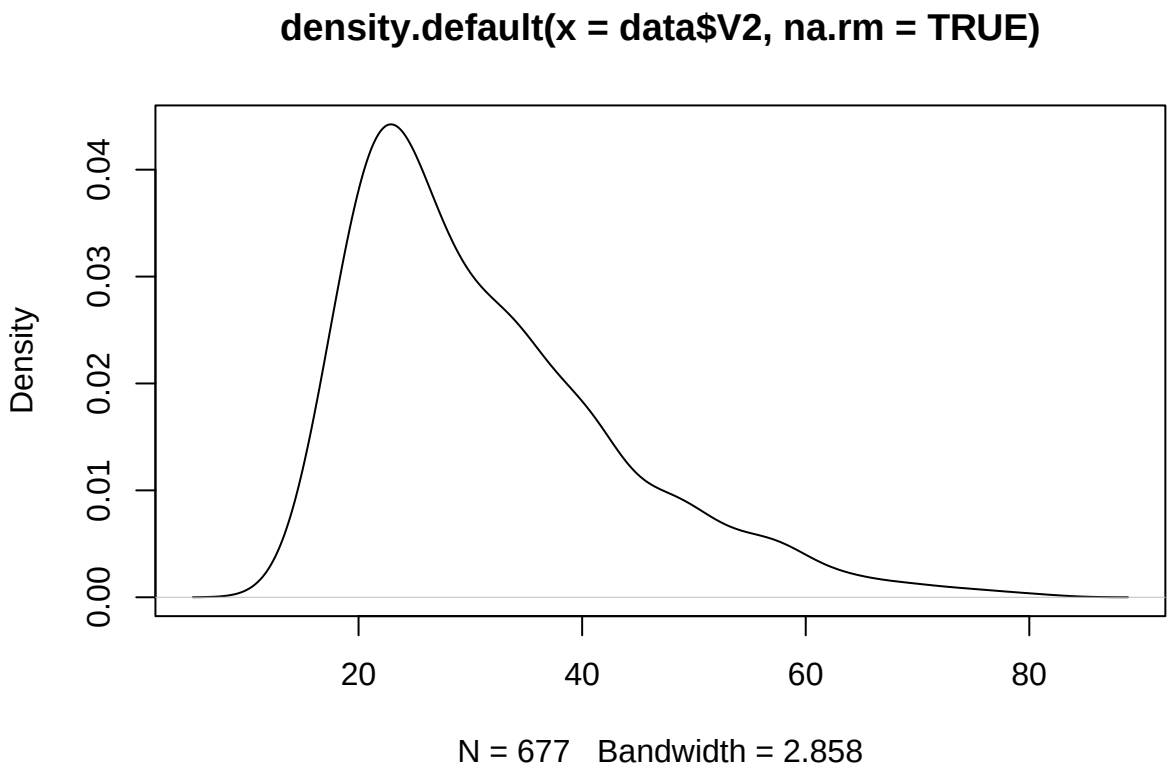
```r
# Plot between V2 and V3
plot(data$V2, data$V3)
abline(model , col = "blue")
```



Question(c)

Plot the density of V2.

```r
# Plot the density plot
plot(density(data$V2, na.rm=TRUE))
```



7. Correlation Analysis

Question(a)

Correlation test (Pearson) - calculate the "degree of association" between V3 and V15.

```
# Calculate the correlation between V3 and V15
cor(data$V3, data$V15)
```

```
## [1] 0.1229349
```

Question(b)

Plot the correlation matrix of all variables. Which variables are redundant? Explain.

An attribute (column or feature of data set) is called redundant if it can be derived from any other attribute or set of attributes. From the correlation matrix, we can see V8 has a relatively high correlation (although it's still pretty mild relationship) in comparison to other, making it a redundant variable.

```
# Subset numeric columns
numeric <- c("V2", "V3", "V8", "V11", "V14", "V15")
numeric_data <- data[numeric]

# Drop NA to aviod 0 in correlation matrix
numeric_data <- na.omit(numeric_data)
```

```r
# Convert data type
numeric_data$V2 <- as.numeric(numeric_data$V2)

# Correlation matrix for all variables
cor(numeric_data, method = "pearson", use = "pairwise.complete.obs")
```

```
##              V2         V3          V8         V11         V14        V15
## V2   1.00000000  0.2023670  0.39575039  0.18589715 -0.07798716 0.01853611
## V3   0.20236700  1.0000000  0.30077302  0.27148338 -0.20933321 0.12158871
## V8   0.39575039  0.3007730  1.00000000  0.32702575 -0.06353694 0.05331075
## V11  0.18589715  0.2714834  0.32702575  1.00000000 -0.11135675 0.06330123
## V14 -0.07798716 -0.2093332 -0.06353694 -0.11135675  1.00000000 0.06300410
## V15  0.01853611  0.1215887  0.05331075  0.06330123  0.06300410 1.00000000
```

Question(c)

What can be done with these redundant features to improve our algorithms?

We should remove the redundant features from our algorithm since they not only lead to higher computational complexity but also reduce the accuracy and efficiency of classification methods.

‘‘‘