BUAN6356 - HW 3

("Chitresh Kumar","Chaitanya Narella","Disha Punjabi","Mai Han Tran","Nidaa Tamkeen")

11/03/2019

R Markdown

```
if(!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
pacman::p_load(tidyverse, reshape, gplots, ggmap, MASS,
               mlbench, data.table,leaps, pivottabler, forecast, dplyr,caret)
```

```
Spam_Data <- fread ("spambase.data")
Spam_DF <- setDF (Spam_Data)
str(Spam_DF)
```

```
## 'data.frame':    4601 obs. of  58 variables:
##  $ V1 : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ V2 : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ V3 : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ V4 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V5 : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
##  $ V6 : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
##  $ V7 : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.38 ...
##  $ V8 : num  0 0.07 0.12 0.63 0.63 1.85 0 1.88 0 0 ...
##  $ V9 : num  0 0 0.64 0.31 0.31 0 0 0 0.92 0.06 ...
##  $ V10: num  0 0.94 0.25 0.63 0.63 0 0.64 0 0.76 0 ...
##  $ V11: num  0 0.21 0.38 0.31 0.31 0 0.96 0 0.76 0 ...
##  $ V12: num  0.64 0.79 0.45 0.31 0.31 0 1.28 0 0.92 0.64 ...
##  $ V13: num  0 0.65 0.12 0.31 0.31 0 0 0 0 0.25 ...
##  $ V14: num  0 0.21 0 0 0 0 0 0 0 0 ...
##  $ V15: num  0 0.14 1.75 0 0 0 0 0 0 0.12 ...
##  $ V16: num  0.32 0.14 0.06 0.31 0.31 0 0.96 0 0 0 ...
##  $ V17: num  0 0.07 0.06 0 0 0 0 0 0 0 ...
##  $ V18: num  1.29 0.28 1.03 0 0 0 0.32 0 0.15 0.12 ...
##  $ V19: num  1.93 3.47 1.36 3.18 3.18 0 3.85 0 1.23 1.67 ...
##  $ V20: num  0 0 0.32 0 0 0 0 0 3.53 0.06 ...
##  $ V21: num  0.96 1.59 0.51 0.31 0.31 0 0.64 0 2 0.71 ...
##  $ V22: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V23: num  0 0.43 1.16 0 0 0 0 0 0 0.19 ...
##  $ V24: num  0 0.43 0.06 0 0 0 0 0 0.15 0 ...
##  $ V25: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V26: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V27: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V28: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V29: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V30: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V31: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V32: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V33: num  0 0 0 0 0 0 0 0 0.15 0 ...
```

```
##  $ V34: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V35: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V36: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V37: num  0 0.07 0 0 0 0 0 0 0 0 ...
##  $ V38: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V39: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V40: num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ V41: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V42: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V43: num  0 0 0.12 0 0 0 0 0 0.3 0 ...
##  $ V44: num  0 0 0 0 0 0 0 0 0 0.06 ...
##  $ V45: num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ V46: num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ V47: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V48: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V49: num  0 0 0.01 0 0 0 0 0 0 0.04 ...
##  $ V50: num  0 0.132 0.143 0.137 0.135 0.223 0.054 0.206 0.271 0.03 ...
##  $ V51: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V52: num  0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181 0.244 ...
##  $ V53: num  0 0.18 0.184 0 0 0 0.054 0 0.203 0.081 ...
##  $ V54: num  0 0.048 0.01 0 0 0 0 0 0.022 0 ...
##  $ V55: num  3.76 5.11 9.82 3.54 3.54 ...
##  $ V56: int  61 101 485 40 40 15 4 11 445 43 ...
##  $ V57: int  278 1028 2259 191 191 54 112 49 1257 749 ...
##  $ V58: int  1 1 1 1 1 1 1 1 1 1 ...
```

```
glimpse(Spam_DF)
```

```
## Observations: 4,601
## Variables: 58
## $ V1  <dbl> 0.00, 0.21, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.15, 0.06...
## $ V2  <dbl> 0.64, 0.28, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.12...
## $ V3  <dbl> 0.64, 0.50, 0.71, 0.00, 0.00, 0.00, 0.00, 0.00, 0.46, 0.77...
## $ V4  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V5  <dbl> 0.32, 0.14, 1.23, 0.63, 0.63, 1.85, 1.92, 1.88, 0.61, 0.19...
## $ V6  <dbl> 0.00, 0.28, 0.19, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.32...
## $ V7  <dbl> 0.00, 0.21, 0.19, 0.31, 0.31, 0.00, 0.00, 0.00, 0.30, 0.38...
## $ V8  <dbl> 0.00, 0.07, 0.12, 0.63, 0.63, 1.85, 0.00, 1.88, 0.00, 0.00...
## $ V9  <dbl> 0.00, 0.00, 0.64, 0.31, 0.31, 0.00, 0.00, 0.00, 0.92, 0.06...
## $ V10 <dbl> 0.00, 0.94, 0.25, 0.63, 0.63, 0.00, 0.64, 0.00, 0.76, 0.00...
## $ V11 <dbl> 0.00, 0.21, 0.38, 0.31, 0.31, 0.00, 0.96, 0.00, 0.76, 0.00...
## $ V12 <dbl> 0.64, 0.79, 0.45, 0.31, 0.31, 0.00, 1.28, 0.00, 0.92, 0.64...
## $ V13 <dbl> 0.00, 0.65, 0.12, 0.31, 0.31, 0.00, 0.00, 0.00, 0.00, 0.25...
## $ V14 <dbl> 0.00, 0.21, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V15 <dbl> 0.00, 0.14, 1.75, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.12...
## $ V16 <dbl> 0.32, 0.14, 0.06, 0.31, 0.31, 0.00, 0.96, 0.00, 0.00, 0.00...
## $ V17 <dbl> 0.00, 0.07, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V18 <dbl> 1.29, 0.28, 1.03, 0.00, 0.00, 0.00, 0.32, 0.00, 0.15, 0.12...
## $ V19 <dbl> 1.93, 3.47, 1.36, 3.18, 3.18, 0.00, 3.85, 0.00, 1.23, 1.67...
## $ V20 <dbl> 0.00, 0.00, 0.32, 0.00, 0.00, 0.00, 0.00, 0.00, 3.53, 0.06...
## $ V21 <dbl> 0.96, 1.59, 0.51, 0.31, 0.31, 0.00, 0.64, 0.00, 2.00, 0.71...
## $ V22 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V23 <dbl> 0.00, 0.43, 1.16, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.19...
## $ V24 <dbl> 0.00, 0.43, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.15, 0.00...
## $ V25 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
```

```
## $ V26 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V27 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V28 <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V29 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V30 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V31 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V32 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V33 <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.15, 0.00...
## $ V34 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V35 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V36 <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V37 <dbl> 0.00, 0.07, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V38 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V39 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V40 <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V41 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V42 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V43 <dbl> 0.00, 0.00, 0.12, 0.00, 0.00, 0.00, 0.00, 0.00, 0.30, 0.00...
## $ V44 <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.06...
## $ V45 <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V46 <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V47 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V48 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V49 <dbl> 0.000, 0.000, 0.010, 0.000, 0.000, 0.000, 0.000, 0.000, 0....
## $ V50 <dbl> 0.000, 0.132, 0.143, 0.137, 0.135, 0.223, 0.054, 0.206, 0....
## $ V51 <dbl> 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0....
## $ V52 <dbl> 0.778, 0.372, 0.276, 0.137, 0.135, 0.000, 0.164, 0.000, 0....
## $ V53 <dbl> 0.000, 0.180, 0.184, 0.000, 0.000, 0.000, 0.054, 0.000, 0....
## $ V54 <dbl> 0.000, 0.048, 0.010, 0.000, 0.000, 0.000, 0.000, 0.000, 0....
## $ V55 <dbl> 3.756, 5.114, 9.821, 3.537, 3.537, 3.000, 1.671, 2.450, 9....
## $ V56 <int> 61, 101, 485, 40, 40, 15, 4, 11, 445, 43, 6, 11, 61, 7, 24...
## $ V57 <int> 278, 1028, 2259, 191, 191, 54, 112, 49, 1257, 749, 21, 184...
## $ V58 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
```

Question:1a)Examine how each predictor is different between spam & non-sapm email by comparing class averages.

```
Pivot <- Spam_DF %>%
  group_by (V58)%>%
  summarise_all(funs(mean))
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```
head(Pivot)
```

```
## # A tibble: 2 x 58
##     V58     V1    V2    V3      V4    V5     V6      V7     V8     V9   V10
##   <int>  <dbl> <dbl> <dbl>   <dbl> <dbl>  <dbl>   <dbl>  <dbl>  <dbl> <dbl>
## 1     0 0.0735 0.244 0.201 8.86e-4 0.181 0.0445 0.00938 0.0384 0.0380 0.167
## 2     1 0.152  0.165 0.404 1.65e-1 0.514 0.175  0.275   0.208  0.170  0.351
## # ... with 47 more variables: V11 <dbl>, V12 <dbl>, V13 <dbl>, V14 <dbl>,
## #   V15 <dbl>, V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>, V20 <dbl>,
## #   V21 <dbl>, V22 <dbl>, V23 <dbl>, V24 <dbl>, V25 <dbl>, V26 <dbl>,
## #   V27 <dbl>, V28 <dbl>, V29 <dbl>, V30 <dbl>, V31 <dbl>, V32 <dbl>,
## #   V33 <dbl>, V34 <dbl>, V35 <dbl>, V36 <dbl>, V37 <dbl>, V38 <dbl>,
## #   V39 <dbl>, V40 <dbl>, V41 <dbl>, V42 <dbl>, V43 <dbl>, V44 <dbl>,
## #   V45 <dbl>, V46 <dbl>, V47 <dbl>, V48 <dbl>, V49 <dbl>, V50 <dbl>,
## #   V51 <dbl>, V52 <dbl>, V53 <dbl>, V54 <dbl>, V55 <dbl>, V56 <dbl>,
## #   V57 <dbl>
```

The centroid of all the variables are calculated as shown above in the summary of the pivot.Centroid of classifying non-spam from V1 is 0.0735 and that of spam is 0.1523.Similarly we can observe other predictor variables.

Question:1b)Identify the 10 variables with highest difference between class average

```
Var_Table <-data.frame(r1=names(Pivot), t(Pivot))
Var_Table["Difference"] <- NA
Var_Table$Difference <- round(abs(Var_Table$X1 - Var_Table$X2),4)

# testing slice of top 10 values
Var_Order_table <-Var_Table[-c(1),]
Ordered_Table <- Var_Order_table %>%
    arrange(desc(Difference)) %>%
    slice(1:10)
Ordered_Table
```

```
##     r1          X1           X2 Difference
## 1  V57 161.4709469 4.706194e+02   309.1485
## 2  V56  18.2144907 1.043933e+02    86.1788
## 3  V55   2.3773009 9.519165e+00     7.1419
## 4  V27   1.2652654 1.549917e-03     1.2637
## 5  V19   1.2703407 2.264539e+00     0.9942
## 6  V21   0.4387016 1.380370e+00     0.9417
## 7  V25   0.8954735 1.747932e-02     0.8780
## 8  V16   0.0735868 5.183618e-01     0.4448
## 9  V26   0.4319943 9.172642e-03     0.4228
## 10 V52   0.1099835 5.137126e-01     0.4037
```

The difference of the mean of spam and non-spam are calculated.Then we have sorted the data to get top 10 values for spam and non-spam values.

Check the variables names(r1) for the values given in the above table
capital_run_length_average(v55)
capital_run_length_longest(v56)
capital_run_length_total(v57)
word_freq_hp(V25)
word_freq_hpl(V26)
word_freq_george(V27)
word_freq_free(v16)

```
word_freq_you(v19)
word_freq_your(v21)
char_freq_!(v52)
```

#Question:2) Perform a linear discriminant analysis using the training dataset.Using training dataset with only 10 predictors with higest difference - Data Parition - Normalize the data

```
#Creating table for Data Analysis
Final_Table <- Var_Table[-c(1),]
Final_Table[order(-Final_Table$Difference),]
```

```
##      r1           X1           X2 Difference
## V57 V57 1.614709e+02 4.706194e+02   309.1485
## V56 V56 1.821449e+01 1.043933e+02    86.1788
## V55 V55 2.377301e+00 9.519165e+00     7.1419
## V27 V27 1.265265e+00 1.549917e-03     1.2637
## V19 V19 1.270341e+00 2.264539e+00     0.9942
## V21 V21 4.387016e-01 1.380370e+00     0.9417
## V25 V25 8.954735e-01 1.747932e-02     0.8780
## V16 V16 7.358680e-02 5.183618e-01     0.4448
## V26 V26 4.319943e-01 9.172642e-03     0.4228
## V52 V52 1.099835e-01 5.137126e-01     0.4037
## V5   V5 1.810402e-01 5.139548e-01     0.3329
## V45 V45 4.157604e-01 1.250910e-01     0.2907
## V46 V46 2.871844e-01 1.472697e-02     0.2725
## V7   V7 9.383070e-03 2.754054e-01     0.2660
## V23 V23 7.087518e-03 2.470546e-01     0.2400
## V17 V17 4.834648e-02 2.875069e-01     0.2392
## V18 V18 9.729197e-02 3.192278e-01     0.2219
## V42 V42 2.168077e-01 2.443464e-03     0.2144
## V3   V3 2.005811e-01 4.037948e-01     0.2032
## V20 V20 7.578910e-03 2.055212e-01     0.1979
## V24 V24 1.713773e-02 2.128792e-01     0.1957
## V22 V22 4.522597e-02 2.380364e-01     0.1928
## V10 V10 1.671700e-01 3.505074e-01     0.1833
## V28 V28 1.938056e-01 1.879757e-02     0.1750
## V8   V8 3.841463e-02 2.081412e-01     0.1697
## V4   V4 8.859397e-04 1.646718e-01     0.1638
## V53 V53 1.164849e-02 1.744782e-01     0.1628
## V35 V35 1.694548e-01 6.927744e-03     0.1625
## V29 V29 1.627941e-01 6.839493e-04     0.1621
## V30 V30 1.658537e-01 5.968009e-03     0.1599
## V37 V37 1.977439e-01 4.346939e-02     0.1543
## V33 V33 1.509864e-01 1.456150e-02     0.1364
## V9   V9 3.804878e-02 1.700607e-01     0.1320
## V6   V6 4.454448e-02 1.748759e-01     0.1303
## V44 V44 1.266356e-01 6.243795e-03     0.1204
## V36 V36 1.416714e-01 2.951462e-02     0.1122
## V39 V39 1.216786e-01 1.242692e-02     0.1093
## V31 V31 1.060330e-01 1.274131e-03     0.1048
## V15 V15 8.317791e-03 1.120794e-01     0.1038
## V11 V11 2.171090e-02 1.184335e-01     0.0967
## V13 V13 6.166428e-02 1.435466e-01     0.0819
## V2   V2 2.444656e-01 1.646498e-01     0.0798
## V1   V1 7.347920e-02 1.523387e-01     0.0789
```

```
## V32 V32 7.730631e-02 5.184777e-04     0.0768
## V34 V34 7.778694e-02 1.776062e-03     0.0760
## V41 V41 7.202654e-02 5.515720e-05     0.0720
## V43 V43 7.058106e-02 8.450083e-03     0.0621
## V54 V54 2.171306e-02 7.887700e-02     0.0572
## V50 V50 1.585782e-01 1.089702e-01     0.0496
## V48 V48 5.122669e-02 2.101489e-03     0.0491
## V40 V40 8.311693e-02 3.671815e-02     0.0464
## V14 V14 4.240316e-02 8.357419e-02     0.0412
## V49 V49 5.028085e-02 2.057308e-02     0.0297
## V51 V51 2.268364e-02 8.198566e-03     0.0145
## V38 V38 1.872310e-02 4.710425e-03     0.0140
## V12 V12 5.363235e-01 5.499724e-01     0.0136
## V47 V47 8.192253e-03 1.218974e-03     0.0070
```

```
#creating working directory
Working_Table <- Spam_DF[,c(57,56,55,27,19,21,25,16,26,52,58)]

Working_Data <-transform(Working_Table, V58 = as.character(V58))
str(Working_Data)
```

```
## 'data.frame':    4601 obs. of  11 variables:
##  $ V57: int  278 1028 2259 191 191 54 112 49 1257 749 ...
##  $ V56: int  61 101 485 40 40 15 4 11 445 43 ...
##  $ V55: num  3.76 5.11 9.82 3.54 3.54 ...
##  $ V27: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V19: num  1.93 3.47 1.36 3.18 3.18 0 3.85 0 1.23 1.67 ...
##  $ V21: num  0.96 1.59 0.51 0.31 0.31 0 0.64 0 2 0.71 ...
##  $ V25: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V16: num  0.32 0.14 0.06 0.31 0.31 0 0.96 0 0 0 ...
##  $ V26: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V52: num  0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181 0.244 ...
##  $ V58: chr  "1" "1" "1" "1" ...
```

```
set.seed(42)
levels(Working_Data$V58)<-c(0,1)
trainings.index <- createDataPartition(Working_Data$V58, p = 0.8, list = FALSE)
predictors.train <- Working_Data[trainings.index, ]
predictors.valid <- Working_Data[-trainings.index, ]

#normailizing the data
norm.values <- preProcess(predictors.train, method = c("center", "scale"))
# Transform the data using the estimated parameters
norm.train.df <- predict(norm.values, predictors.train)
norm.valid.df <- predict(norm.values, predictors.valid)
#### code tests
str(norm.valid.df)
```

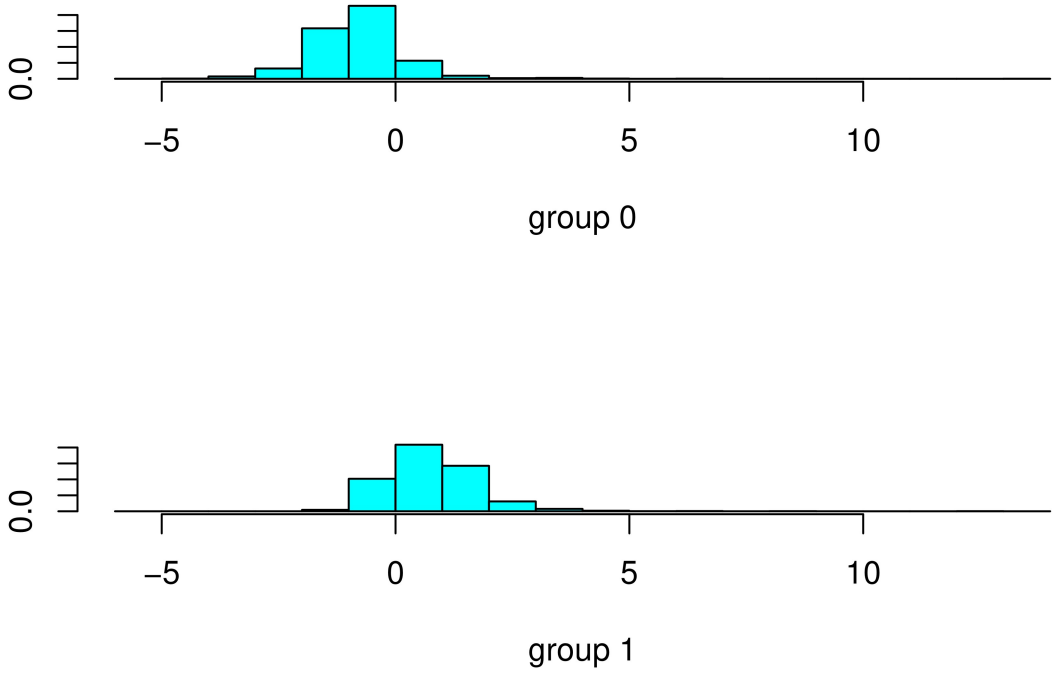
```
## 'data.frame':    919 obs. of  11 variables:
##  $ V57: num  -0.386 -0.137 -0.167 -0.427 -0.37 ...
##  $ V56: num  -0.205 -0.144 0.175 -0.215 -0.21 ...
##  $ V55: num  -0.0842 -0.0983 -0.0531 -0.0435 -0.0985 ...
##  $ V27: num  -0.225 -0.225 -0.225 -0.225 -0.225 ...
##  $ V19: num  -0.933 0.86 0.307 -0.933 1.034 ...
##  $ V21: num  -0.681 1.738 0.94 -0.681 -0.681 ...
##  $ V25: num  -0.328 -0.328 -0.328 -0.328 -0.328 ...
```

```
##  $ V16: num  -0.292 6.064 0.456 -0.292 -0.292 ...
##  $ V26: num  -0.293 -0.293 -0.293 -0.293 -0.293 ...
##  $ V52: num  -0.33 0.116 -0.261 0.58 0.776 ...
##  $ V58: chr  "1" "1" "1" "1" ...
```

Run LDA & run prediction using Training Data set & plot

```
lda2 <- lda(V58~., data = norm.train.df)

pred2.train <- predict(lda2, norm.train.df)
pred2.valid <- predict(lda2, norm.train.df)
plot(lda2)
```



The prior probability of getting a spam is 39.4% and that of non-spam is 60.6%.As we are classifying in
From graph we can see there are few misclassified values which have been predicted as 'spam'(Group-1,ne

Question:3) What are the prior probabilities?

```
lda2$prior
```

```
##         0         1
## 0.6059207 0.3940793
```

As per the Dataset Description the spam and non-spam are depicted as 1 and 0 respectively.
The prior probabilities give information about the distribution of spam and non-spam in

the entire data set.Here the non-spams are having the 0.606 and spam of 0.394 probabilities.The number of non-spam cases are more compared to the spam.

Question:4) What are the coefficients of linear discriminants? Explain.

```
lda2$scaling
```

```
##             LD1
## V57  0.37245497
## V56  0.12966688
## V55  0.05268179
## V27 -0.21043836
## V19  0.24656232
## V21  0.57149367
## V25 -0.23541887
## V16  0.38749847
## V26 -0.15061430
## V52  0.32675618
```

The coefficients of linear discriminants are V55,V56,V57- measure the length of sequences of consecutive capital letters V27,V19,V21

#Question:5) Generate linear discriminants using your analysis. How are they used in classifying spams and non-spams?

```
lda2$scaling
```

```
##             LD1
## V57  0.37245497
## V56  0.12966688
## V55  0.05268179
## V27 -0.21043836
## V19  0.24656232
## V21  0.57149367
## V25 -0.23541887
## V16  0.38749847
## V26 -0.15061430
## V52  0.32675618
```

We can infer from the above data that the predictors V27,V25,V26 are mainly classified non-spam.Whereas the other Predictors(V57,V56,V55,V19,V21,V16,V52) are classified as spam.The discriminant values obtained negative are considered as non-spams and the values which are positive are considered spam.

#Question:6) How many linear discriminants are in the model? Why?

```
lda2$scaling
```

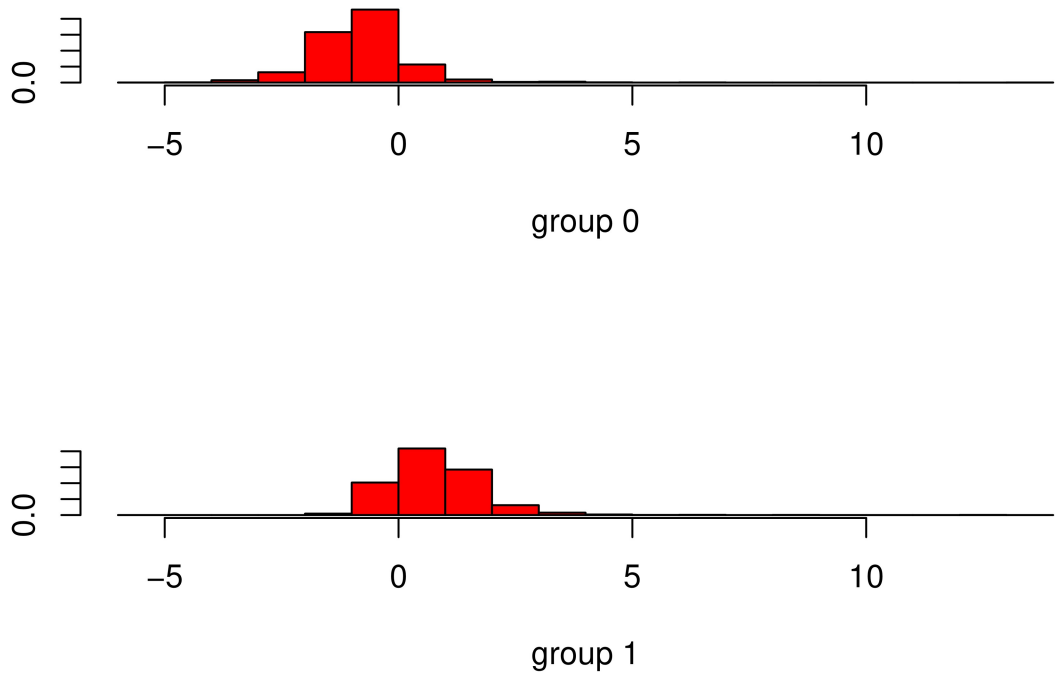
```
##             LD1
## V57  0.37245497
## V56  0.12966688
## V55  0.05268179
## V27 -0.21043836
## V19  0.24656232
## V21  0.57149367
```

```
## V25 -0.23541887
## V16  0.38749847
## V26 -0.15061430
## V52  0.32675618
```

/* There is only 1 linear discriminant in the model which is LD1 ,because there are two classes which are specifically spam and non-spam. */

#Question:7) Generate LDA plot using the training and validation data. What information is presented in these plots? How are they different?

```
plot(lda2,col="red",main="Training DataSet")
```

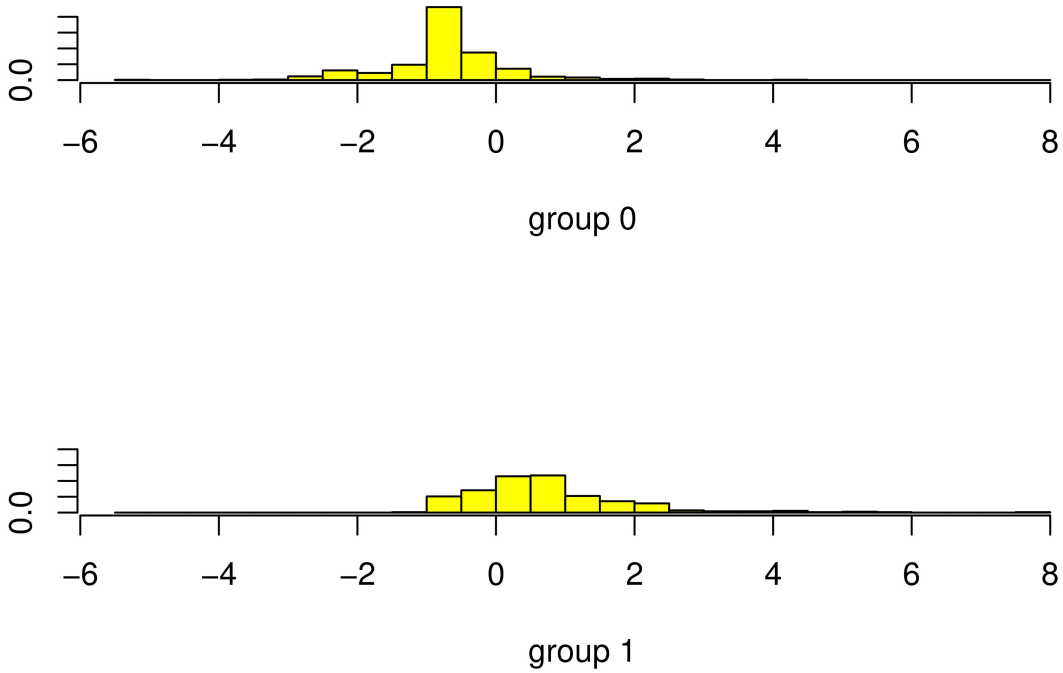



```
lda3 <- lda(V58~., data = norm.valid.df)
lda3
```

```
## Call:
## lda(V58 ~ ., data = norm.valid.df)
##
## Prior probabilities of groups:
##         0         1
## 0.6060936 0.3939064
##
## Group means:
##          V57        V56         V55        V27        V19        V21
## 0 -0.2290651 -0.1806284 -0.09078748  0.2160170 -0.1928716 -0.2691568
## 1  0.2051261  0.1561984  0.09419330 -0.2244976  0.2404940  0.4946953
##          V25        V16        V26        V52
```

```
## 0  0.1609472 -0.2131732  0.195465 -0.1588808
## 1 -0.3198471  0.3706067 -0.281216  0.3185921
##
## Coefficients of linear discriminants:
##              LD1
## V57  0.329956112
## V56  0.577102578
## V55 -0.007415411
## V27 -0.206260282
## V19  0.128502922
## V21  0.409383178
## V25 -0.266968220
## V16  0.545364738
## V26 -0.247278099
## V52  0.205369048
```

```
plot(lda3,col="yellow")
```



The red chart shows the training data in which we can see there are few misclassified values which have been predicted as 'spam'(Group-1,negative values which are classified as spam), as well as for non-spam(Group-0,positive values which are classified as non-spam). The yellow chart shows lda based on validation data.We have much less values which are misclassified as compared to the training LDA.We can check the misclassified from spam >0 and non-spam <0.

#Question:8) Generate the relevant confusion matrix. What are the sensitivity and specificity?

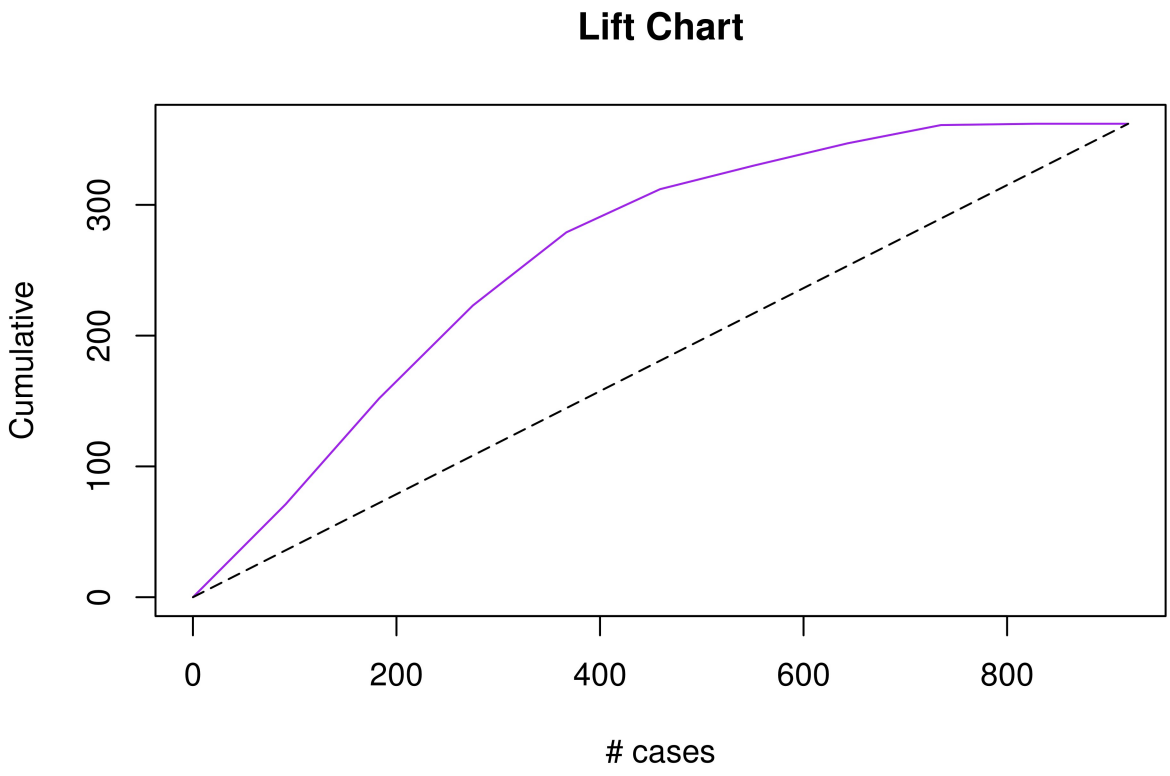
```r
#CMat <- table(pred2.valid$class, norm.valid.df$V58)  # pred v actual
#confusionMatrix(CMat)
pred1 <- predict(lda2, norm.valid.df)
acc <- table(pred1$class, norm.valid.df$V58)
confusionMatrix(acc)
```

```
## Confusion Matrix and Statistics
##
##
##       0   1
##   0 502 118
##   1  55 244
##
##                Accuracy : 0.8118
##                  95% CI : (0.7849, 0.8365)
##     No Information Rate : 0.6061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5934
##
##  Mcnemar's Test P-Value : 2.432e-06
##
##             Sensitivity : 0.9013
##             Specificity : 0.6740
##          Pos Pred Value : 0.8097
##          Neg Pred Value : 0.8161
##              Prevalence : 0.6061
##          Detection Rate : 0.5462
##    Detection Prevalence : 0.6746
##       Balanced Accuracy : 0.7876
##
##        'Positive' Class : 0
##
```

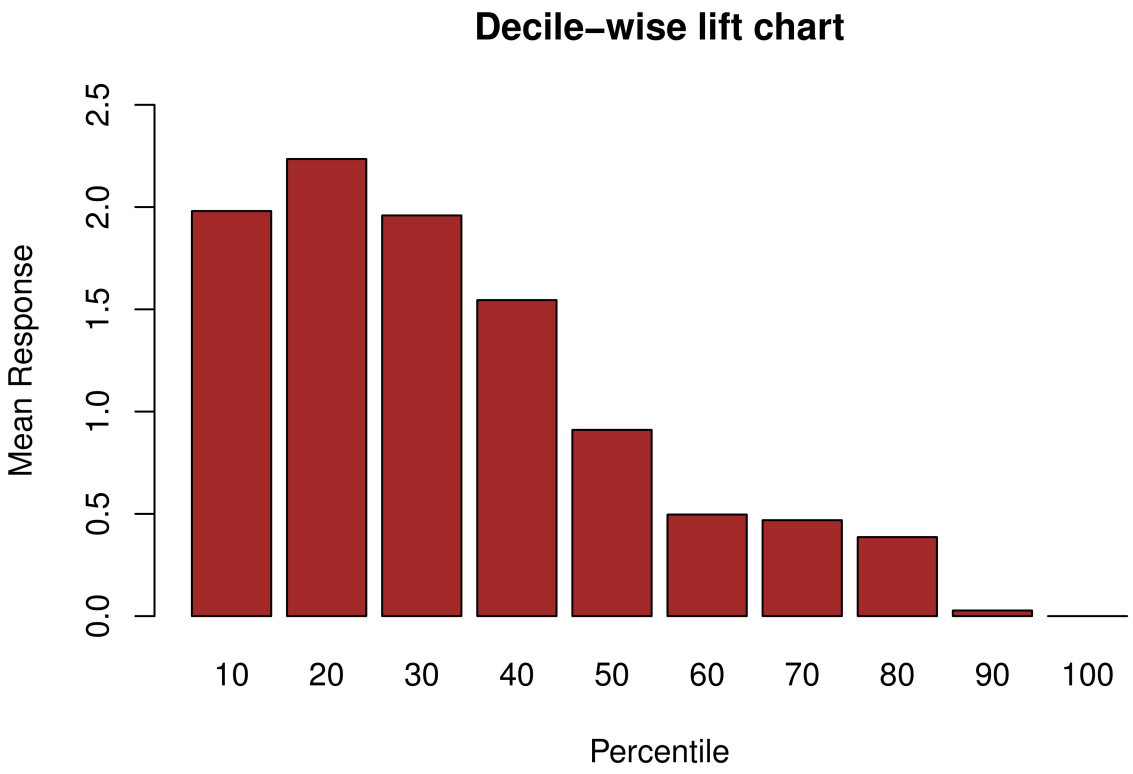
/* Sensitivity is 92.5% and the Specificity is 63.0% */

```r
library(gains)
gain <- gains(as.numeric(norm.valid.df$V58), pred1$x[,1], groups = 10)
options(scipen=999)

### Computing gains relative to price
spam<- as.numeric(norm.valid.df$V58)
plot3<- plot(c(0,gain$cume.pct.of.total*sum(spam))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative", main="Lift Chart",
     col = "purple", type="l")
### inserting a baseline to the lift chart
lines(c(0,sum(spam))~c(0, dim(predictors.valid)[1]), lty = 5)
```




```
### Decile-wise chart lift chart
heights <- gain$mean.resp/mean(spam)
midpoints <- barplot(heights, names.arg = gain$depth,  ylim = c(0,2.5), col = "brown",
                     xlab = "Percentile", ylab = "Mean Response",
                     main = "Decile-wise lift chart")
```



From the lift chart,we can see that the lift from the naive prediction shows that our model is able to perform well than naive condition.From decile chart we will take the values till 40 percent as it is more than 1 and would be better to predict our spam properly.

```
options(scipen=999)
table(pred1$class, norm.valid.df$V58)  # pred v actual
```

```
##
##       0   1
##   0 502 118
##   1  55 244
```

```
mean(pred1$class ==  norm.valid.df$V58)  # percent accurate
```

```
## [1] 0.8117519
```

```
sum(pred1$posterior[, 1] >=.5)
```

```
## [1] 620
```

```
sum(pred1$posterior[, 1] >=.2)  # decrease the cut-off from .5 to .2
```

```
## [1] 803
```

The mean of correctly predicted values are 80.8 percent.As we decrease the cutoff values for spam from 0.5 to 0.2,the number of spam values increases from 649 to 823 because 823 linear discriminant posterior values are more than 0.2 while in 0.5 cutoff there are 649 above it.