Final Project

Isaiah Thompson Ocansey

5/13/2022

```
rm(list=ls(all=TRUE))

library(klaR)

## Loading required package: MASS

library(psych)
library(MASS)
#library(ggord)
library(devtools)

## Loading required package: usethis
```

Loading the Spam Data

```
library(kernlab);

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:psych':
##
##     alpha

data(spam)
dim(spam); head(spam)

## [1] 4601   58

##   make address  all num3d  our over remove internet order mail receive
will
## 1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00  0.00 0.00    0.00
0.64
## 2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07  0.00 0.94    0.21
0.79
## 3 0.06    0.00 0.71     0 1.23 0.19   0.19     0.12  0.64 0.25    0.38
0.45
## 4 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31
0.31
## 5 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31
0.31
## 6 0.00    0.00 0.00     0 1.85 0.00   0.00     1.85  0.00 0.00    0.00
0.00
##   people report addresses free business email  you credit your font num000
```

```
## 1   0.00   0.00      0.00 0.32     0.00  1.29 1.93   0.00 0.96    0   0.00
## 2   0.65   0.21      0.14 0.14     0.07  0.28 3.47   0.00 1.59    0   0.43
## 3   0.12   0.00      1.75 0.06     0.06  1.03 1.36   0.32 0.51    0   1.16
## 4   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0   0.00
## 5   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0   0.00
## 6   0.00   0.00      0.00 0.00     0.00  0.00 0.00   0.00 0.00    0   0.00
##   money hp hpl george num650 lab labs telnet num857 data num415 num85
## 1  0.00  0   0      0      0   0    0      0      0    0      0     0
## 2  0.43  0   0      0      0   0    0      0      0    0      0     0
## 3  0.06  0   0      0      0   0    0      0      0    0      0     0
## 4  0.00  0   0      0      0   0    0      0      0    0      0     0
## 5  0.00  0   0      0      0   0    0      0      0    0      0     0
## 6  0.00  0   0      0      0   0    0      0      0    0      0     0
##   technology num1999 parts pm direct cs meeting original project   re  edu
## 1          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 2          0    0.07     0  0   0.00  0       0     0.00       0 0.00 0.00
## 3          0    0.00     0  0   0.06  0       0     0.12       0 0.06 0.06
## 4          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 5          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 6          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
##   table conference charSemicolon charRoundbracket charSquarebracket
## 1     0          0          0.00            0.000                 0
## 2     0          0          0.00            0.132                 0
## 3     0          0          0.01            0.143                 0
## 4     0          0          0.00            0.137                 0
## 5     0          0          0.00            0.135                 0
## 6     0          0          0.00            0.223                 0
##   charExclamation charDollar charHash capitalAve capitalLong capitalTotal
type
## 1           0.778      0.000    0.000      3.756          61          278
spam
## 2           0.372      0.180    0.048      5.114         101         1028
spam
## 3           0.276      0.184    0.010      9.821         485         2259
spam
## 4           0.137      0.000    0.000      3.537          40          191
spam
## 5           0.135      0.000    0.000      3.537          40          191
spam
## 6           0.000      0.000    0.000      3.000          15           54
spam
```

The data has 4601 observations and 58 variables

1. (Data Preparation) Bring in the data and get familiar with the variable.

(a) Take a look at the dimension of the data. Inspect if there are missing values and, if so, impute them appropriately

```
anyNA(spam)

## [1] FALSE
```

From the above output, it can be observed that there are no missing values in the data set.

(b) Explore data using numerical and graphical EDA techniques. For example, what is the percentage of spam emails? What are types (categorical or continuous) of the inputs? Are there any peculiar features for any variable that we should pay attention to? Don't present any R output for this part unless really necessary. Instead, summarize your findings in concise language.

```
require(dplyr)

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

x <- spam %>%
  group_by(type) %>%
  summarise(cnt = n()) %>%
  mutate(freq = round(cnt / sum(cnt)*100, 3)) %>%
  arrange(desc(freq))
head(as.data.frame(x))

##       type  cnt   freq
## 1 nonspam 2788 60.596
## 2    spam 1813 39.404
```

The percentage of spam emails from the above output is 39.404% whiles non spam is 60.596%

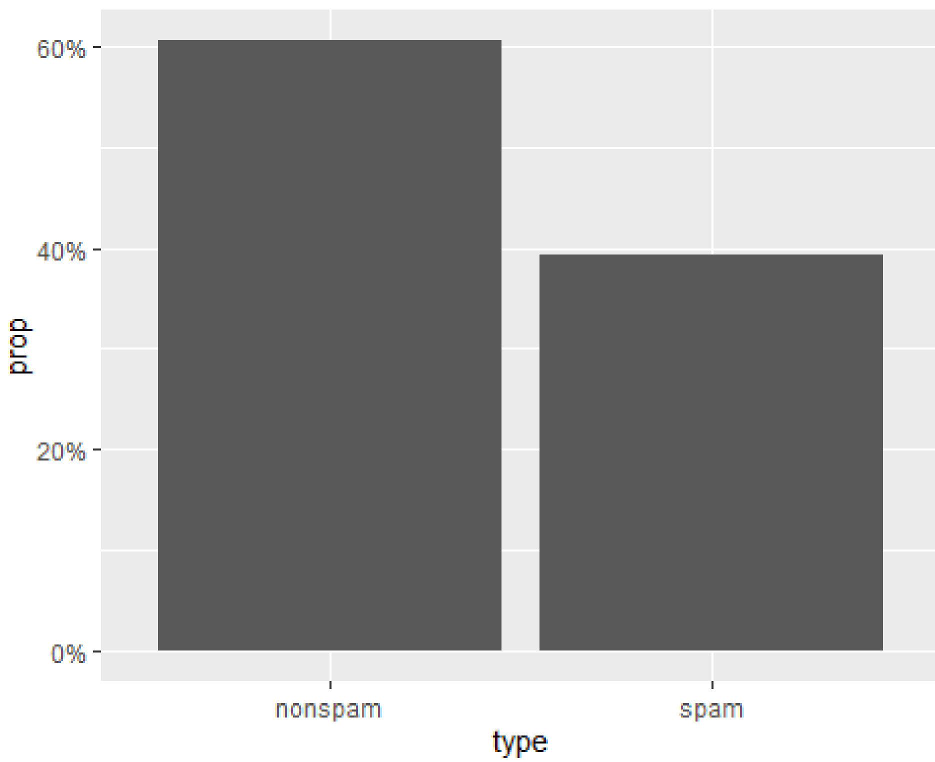
```
str(spam)

## 'data.frame':    4601 obs. of  58 variables:
##  $ make             : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ address          : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ all              : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ num3d            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ our              : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61
0.19 ...
##  $ over             : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
##  $ remove           : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.38 ...
##  $ internet         : num  0 0.07 0.12 0.63 0.63 1.85 0 1.88 0 0 ...
```

```
##  $ order           : num  0 0 0.64 0.31 0.31 0 0 0 0.92 0.06 ...
##  $ mail            : num  0 0.94 0.25 0.63 0.63 0 0.64 0 0.76 0 ...
##  $ receive         : num  0 0.21 0.38 0.31 0.31 0 0.96 0 0.76 0 ...
##  $ will            : num  0.64 0.79 0.45 0.31 0.31 0 1.28 0 0.92 0.64 ...
##  $ people          : num  0 0.65 0.12 0.31 0.31 0 0 0 0 0.25 ...
##  $ report          : num  0 0.21 0 0 0 0 0 0 0 0 ...
##  $ addresses       : num  0 0.14 1.75 0 0 0 0 0 0 0.12 ...
##  $ free            : num  0.32 0.14 0.06 0.31 0.31 0 0.96 0 0 0 ...
##  $ business        : num  0 0.07 0.06 0 0 0 0 0 0 0 ...
##  $ email           : num  1.29 0.28 1.03 0 0 0 0.32 0 0.15 0.12 ...
##  $ you             : num  1.93 3.47 1.36 3.18 3.18 0 3.85 0 1.23 1.67 ...
##  $ credit          : num  0 0 0.32 0 0 0 0 0 3.53 0.06 ...
##  $ your            : num  0.96 1.59 0.51 0.31 0.31 0 0.64 0 2 0.71 ...
##  $ font            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num000          : num  0 0.43 1.16 0 0 0 0 0 0 0.19 ...
##  $ money           : num  0 0.43 0.06 0 0 0 0 0 0.15 0 ...
##  $ hp              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ hpl             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ george          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num650          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ lab             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ labs            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ telnet          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num857          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ data            : num  0 0 0 0 0 0 0 0 0.15 0 ...
##  $ num415          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num85           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ technology      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num1999         : num  0 0.07 0 0 0 0 0 0 0 0 ...
##  $ parts           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ pm              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ direct          : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ cs              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ meeting         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ original        : num  0 0 0.12 0 0 0 0 0 0.3 0 ...
##  $ project         : num  0 0 0 0 0 0 0 0 0 0.06 ...
##  $ re              : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ edu             : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ table           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ conference      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ charSemicolon   : num  0 0 0.01 0 0 0 0 0 0 0.04 ...
##  $ charRoundbracket : num  0 0.132 0.143 0.137 0.135 0.223 0.054 0.206
0.271 0.03 ...
##  $ charSquarebracket: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ charExclamation  : num  0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181
0.244 ...
##  $ charDollar      : num  0 0.18 0.184 0 0 0 0.054 0 0.203 0.081 ...
##  $ charHash        : num  0 0.048 0.01 0 0 0 0 0 0.022 0 ...
##  $ capitalAve      : num  3.76 5.11 9.82 3.54 3.54 ...
##  $ capitalLong     : num  61 101 485 40 40 15 4 11 445 43 ...
```

```
##  $ capitalTotal     : num  278 1028 2259 191 191 ...
##  $ type             : Factor w/ 2 levels "nonspam","spam": 2 2 2 2 2 2 2 2
2 2 ...
```

From the above output the only categorical variable is type and the rears are all continuous variables.There were 391 duplicates samples in the data set, which will result in having a similarity/dissimilarity measure of *zero* in some cases. These duplicates were removed to enable further analysis.

```
# numerical and graphical EDA techniques
suppressPackageStartupMessages(library(ggplot2))
ggplot(data = spam) +
  geom_bar(mapping = aes(x = type, y = ..prop.., group = 1), stat = "count")
+
  scale_y_continuous(labels = scales::percent_format())
```



From the graph above, about 39-40% of the emails are spam whiles about 60% are non spam.

```
# code type as 0 and 1
print("Frequency Table")

## [1] "Frequency Table"

table(spam$type)
```

```
##
## nonspam    spam
##    2788    1813
```

```
spam$type <- ifelse(spam$type == "spam", 1, 0)
print("Frequency Table")
```

```
## [1] "Frequency Table"
```

```
table(spam$type)
```

```
##
##    0    1
## 2788 1813
```

We code the response variable type to 0 and 1 to aid further analysis, so we have 1813 of the spam emails coded as 1 and 2788 of the emails coded nonspam.

```
# duplicates
n.dup <- NROW(spam[duplicated(spam),])
n <- NROW(spam)
matrix(c("Sample Size", "Duplicate Sample Size", n, n.dup), nrow = 2, byrow =
T)
```

```
##      [,1]          [,2]
## [1,] "Sample Size" "Duplicate Sample Size"
## [2,] "4601"        "391"
```

```
# removes duplicates
suppressPackageStartupMessages(library(dplyr))
spam <- distinct(spam)
```

I observed duplicates in the data which brings problems in the further analysis of the data so the above codes removes the duplicates from the data set.

(c) Randomly divide the data into the training/learning sample and the test sample with a ratio of 2:1. We shall use the training sample to train a number of models and then use the test sample to compare them.

```
#splitting data into Train and Test Data in the ration 2:1
n <- NROW(spam)
set.seed(123)

id.test <- sample(1:n, size= trunc(n/3), replace=FALSE)
dat.training <- spam[-id.test, ];
dat.test <- spam[id.test, ]
head(dat.training)
```

```
##    make address  all num3d  our over remove internet order mail receive
will
## 3  0.06       0 0.71     0 1.23 0.19   0.19     0.12  0.64 0.25    0.38
0.45
## 5  0.00       0 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31
```

```
0.31
## 6  0.00       0 0.00     0 1.85 0.00   0.00     1.85  0.00 0.00    0.00
0.00
## 7  0.00       0 0.00     0 1.92 0.00   0.00     0.00  0.00 0.64    0.96
1.28
## 11 0.00       0 0.00     0 0.00 0.00   0.96     0.00  0.00 1.92    0.96
0.00
## 12 0.00       0 0.25     0 0.38 0.25   0.25     0.00  0.00 0.00    0.12
0.12
##    people report addresses free business email  you credit your font
num000
## 3    0.12      0      1.75 0.06     0.06  1.03 1.36   0.32 0.51    0
1.16
## 5    0.31      0      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0
0.00
## 6    0.00      0      0.00 0.00     0.00  0.00 0.00   0.00 0.00    0
0.00
## 7    0.00      0      0.00 0.96     0.00  0.32 3.85   0.00 0.64    0
0.00
## 11   0.00      0      0.00 0.00     0.00  0.96 3.84   0.00 0.96    0
0.00
## 12   0.12      0      0.00 0.00     0.00  0.00 1.16   0.00 0.77    0
0.00
##    money hp hpl george num650 lab labs telnet num857 data num415 num85
## 3   0.06  0   0      0      0   0    0      0      0    0      0     0
## 5   0.00  0   0      0      0   0    0      0      0    0      0     0
## 6   0.00  0   0      0      0   0    0      0      0    0      0     0
## 7   0.00  0   0      0      0   0    0      0      0    0      0     0
## 11  0.00  0   0      0      0   0    0      0      0    0      0     0
## 12  0.00  0   0      0      0   0    0      0      0    0      0     0
##    technology num1999 parts pm direct cs meeting original project   re
edu
## 3           0       0     0  0   0.06  0       0     0.12       0 0.06
0.06
## 5           0       0     0  0   0.00  0       0     0.00       0 0.00
0.00
## 6           0       0     0  0   0.00  0       0     0.00       0 0.00
0.00
## 7           0       0     0  0   0.00  0       0     0.00       0 0.00
0.00
## 11          0       0     0  0   0.96  0       0     0.00       0 0.00
0.00
## 12          0       0     0  0   0.00  0       0     0.00       0 0.00
0.00
##    table conference charSemicolon charRoundbracket charSquarebracket
## 3      0          0         0.010            0.143                 0
## 5      0          0         0.000            0.135                 0
## 6      0          0         0.000            0.223                 0
## 7      0          0         0.000            0.054                 0
## 11     0          0         0.000            0.000                 0
```

```
## 12     0           0          0.022             0.044                 0
##    charExclamation charDollar charHash capitalAve capitalLong capitalTotal
type
## 3              0.276      0.184     0.01      9.821         485         2259
1
## 5              0.135      0.000     0.00      3.537          40          191
1
## 6              0.000      0.000     0.00      3.000          15           54
1
## 7              0.164      0.054     0.00      1.671           4          112
1
## 11             0.462      0.000     0.00      1.312           6           21
1
## 12             0.663      0.000     0.00      1.243          11          184
1
```

From the above output, I split the data into training set and testing set in the ratio 2:1.

2. (Unsupervised Learning) Ignoring the target variable type for the time being, apply • one MDS technique of your choice • tSNE 1 to the training sample and plot the results by specifying the spam (red) or regular (green) email status with different colors. Interpret the results.

```
dat.training <- spam[!duplicated(dat.training),]
```

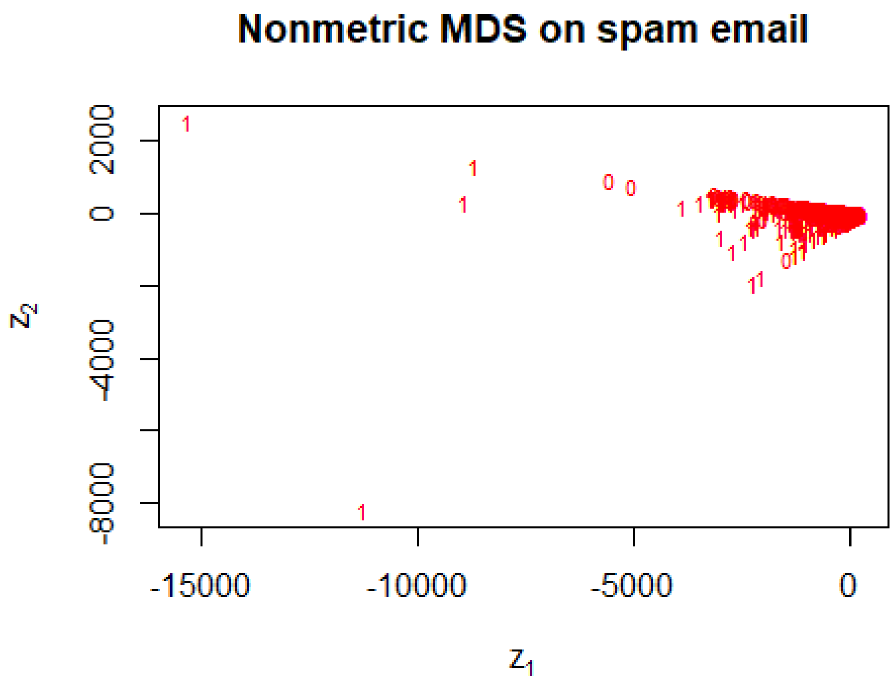
We now remove the target variable type

```
dat.training1<-dat.training[,-58]
dim(dat.training1)

## [1] 4210   57
```

The training data set now has 57 variables and 4201 observations.

```
# Nonmetric MDS
# ========================



mat<- unique.data.frame(dat.training1)

# Nonmetric MDS is performed using the isoMDS( ) function in the MASS
package.
library(MASS)
d <- dist(mat) # euclidean distances between the rows
fit <- isoMDS(d, k=2) # k is the number of dim

## initial  value 0.567372
## final  value 0.567367
## converged

colors = rainbow(length(unique(dat.training$type)))
names(colors) = unique(dat.training$type)
```

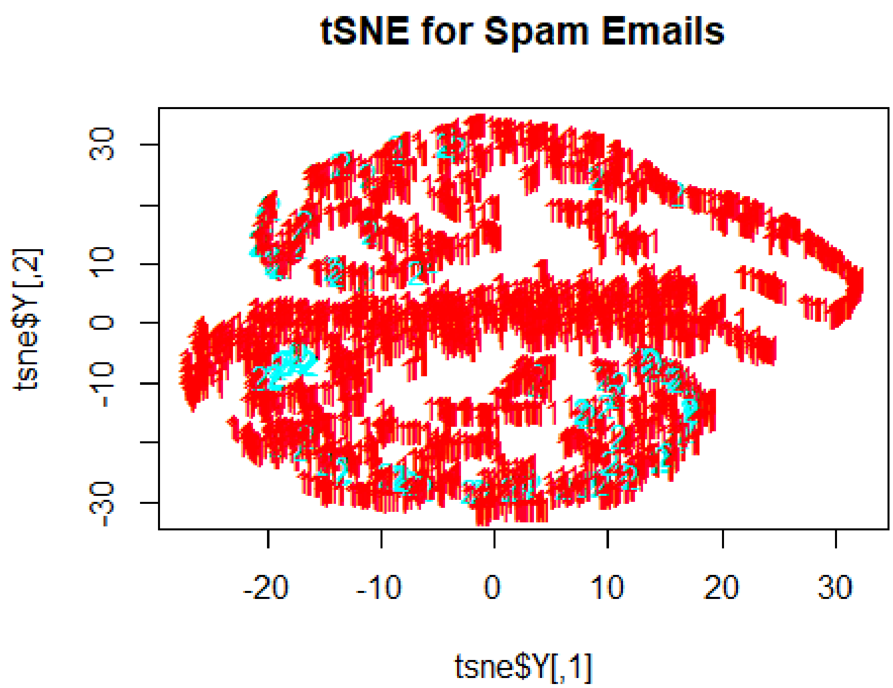
```r
# plot solution
x <- fit$points[,1]; y <- fit$points[,2]
plot(x, y, xlab=expression(z[1]), ylab=expression(z[2]),
  main="Nonmetric MDS on spam email", type="n")

text(x,y, labels = dat.training$type, cex=.7,col=colors[dat.training$type])
```



From the above I use k means as the iterative algorithm and plot using MDS.The number of clusters is chosen to be 2 for simpler interpretation in text of the response variable.

```r
library(dplyr)
dat.training1 <- distinct(dat.training1)
dat.training<- distinct(dat.training)
colors <- rainbow(length(unique(dat.training$type)))
names(colors) <- unique(dat.training$type)

library(Rtsne)
library(cluster)
dat<- unique.data.frame(dat.training1)
dismat <- daisy(dat, metric="gower", stand=TRUE)
## Warning in daisy(dat.training, metric = "gower", stand = TRUE): binary
## variable(s) 58 treated as interval scaled
fit.ward<-hclust(dismat,method = "ward.D2")
hclust.groups <- cutree(fit.ward, k=2)

colors = rainbow(length(unique(hclust.groups)))
```

```
names(colors) = unique(hclust.groups)
set.seed(5000)
tsne <- Rtsne(dat.training1, dims=2, perplexity=30, max_iter=500)
plot(tsne$Y, t="n", main = "tSNE for Spam Emails")
text(tsne$Y, labels = hclust.groups, col = colors[hclust.groups])
```



Tsne is used to reduce the dimension of the data and cluster the data as shown above.

(3)(Supervised Learning) Now we focus on predictive modeling of the binary target variable type.

(i)Linear Discriminant Analysis

```
library(lda)
set.seed(123)
fit.LDA <- lda(type ~ ., data=dat.training)
yhat.LDA <- predict(fit.LDA, newdata=dat.test, type="response")$x
yhat.LDA <- scale(yhat.LDA, center=min(yhat.LDA), scale = max(yhat.LDA)-
min(yhat.LDA))
yhat.LDA <- as.vector(as.numeric(yhat.LDA))
```

Variable Screening before Best Subset Selection (BSS)

```
formula0 <- as.formula(paste("type ~ ", paste(names(dat.training)[-c(58)],
collapse= "+")))
fit.log <- suppressWarnings(glm(formula = formula0, family = "binomial", data
= dat.training))
fit.log
```

```
##
## Call:  glm(formula = formula0, family = "binomial", data = dat.training)
##
## Coefficients:
##       (Intercept)              make            address                all
##        -1.351e+00        -9.250e-02          9.864e-02          6.928e-02
##             num3d               our               over             remove
##         2.477e+00         4.828e-01          7.879e-01          2.387e+00
##          internet             order               mail            receive
##         5.004e-01         6.312e-01          6.701e-02          6.450e-01
##              will            people             report          addresses
##        -1.656e-01        -1.614e-01          1.164e-01          1.190e+00
##              free          business              email                you
##         9.416e-01         9.463e-01          2.837e-02          4.732e-02
##            credit              your               font             num000
##         1.006e+00         2.760e-01          2.342e-01          2.244e+00
##             money                hp                hpl             george
##         2.635e-01        -1.917e+00         -9.547e-01         -1.108e+01
##            num650               lab               labs             telnet
##         4.981e-01        -2.594e+00         -3.221e-01         -1.810e-01
##            num857              data             num415              num85
##         2.042e+00        -6.663e-01          5.598e-01         -2.009e+00
##        technology           num1999              parts                 pm
##         7.951e-01        -3.411e-01         -6.150e-01         -8.077e-01
##            direct                cs            meeting           original
##        -3.492e-01        -4.148e+01         -2.910e+00         -1.472e+00
##           project                re                edu              table
##        -1.600e+00        -8.400e-01         -1.443e+00         -2.183e+00
##        conference      charSemicolon   charRoundbracket  charSquarebracket
##        -3.847e+00        -1.538e+00         -3.647e-01         -8.156e-01
##   charExclamation        charDollar           charHash         capitalAve
##         2.956e-01         5.351e+00          2.355e+00          1.517e-02
##       capitalLong       capitalTotal
##         7.904e-03         7.631e-04
##
## Degrees of Freedom: 4209 Total (i.e. Null);  4152 Residual
## Null Deviance:       5663
## Residual Deviance: 1672  AIC: 1788
```

Our aim is to fit a logistic regression via BSS, however, due to the fact that the errors of many predictors are large in running BSS using glmulti() function, I first performed some variable screening to include only the important variables for the method. I achieved this variable screening by performing a simple logistic regression involving all the variables and retaining the significant ones, that is (p-value < 0.10) for further analysis. A total of 27 variables were excluded from BSS analysis after variable screening.

(ii) Logistic Regression via Best Subset Selection (BSS)

```
set.seed(125)
suppressPackageStartupMessages(library(glmulti))
excludes <- c("charSquarebracket","charRoundbracket","receive",
```

```
             "table","original","cs","direct","parts","charHash",
             "num415","num857","telnet","labs","hpl","money","lab",
             "font","email","addresses","report","people","num3d",
             "all","address","make","pm","capitalLong")
cond <- names(dat.training[,-c(58)]) %in% excludes
xrs <- names(dat.training[,-c(58)])[!cond]
formula0 <- as.formula(paste("type ~ ", paste(xrs, collapse= "+")))
fitting <- suppressWarnings(glmulti(formula0, data = dat.training,
fitfunction = glm,
                   family=binomial, intercept = TRUE, crit = bic, level = 1,
                   method="g", confsetsize=1, plotty = FALSE, report =
FALSE))

## TASK: Genetic algorithm in the candidate set.
## Initialization...
## Algorithm started...
## Improvements in best and average IC have bebingo en below the specified
goals.
## Algorithm is declared to have converged.
## Completed.

fit.bss <- attributes(fitting)$objects[[1]]
yhat.bss <- suppressWarnings(predict(fit.bss, newdata=dat.test,
type="response"))
yhat.bss <- as.vector(as.numeric(yhat.bss))
summary(fit.bss)

##
## Call:
## fitfunc(formula = as.formula(x), family = ..1, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6075  -0.2411  -0.0001   0.1513   5.8037
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -1.404e+00  1.078e-01 -13.026  < 2e-16 ***
## our              5.068e-01  1.001e-01   5.061 4.18e-07 ***
## over             9.035e-01  2.533e-01   3.567 0.000361 ***
## remove           2.818e+00  3.669e-01   7.682 1.56e-14 ***
## internet         5.188e-01  1.556e-01   3.334 0.000856 ***
## free             8.977e-01  1.319e-01   6.803 1.02e-11 ***
## business         1.011e+00  2.183e-01   4.631 3.64e-06 ***
## credit           1.720e+00  5.570e-01   3.089 0.002011 **
## your             3.246e-01  5.324e-02   6.097 1.08e-09 ***
## num000           2.205e+00  4.800e-01   4.594 4.34e-06 ***
## hp              -2.507e+00  2.677e-01  -9.367  < 2e-16 ***
## george          -1.317e+01  2.268e+00  -5.807 6.34e-09 ***
## data            -1.000e+00  3.245e-01  -3.083 0.002050 **
```

```
## meeting          -3.018e+00  9.514e-01  -3.173 0.001511 **
## project          -2.087e+00  5.325e-01  -3.920 8.85e-05 ***
## re               -8.655e-01  1.545e-01  -5.601 2.14e-08 ***
## edu              -1.704e+00  2.735e-01  -6.229 4.68e-10 ***
## conference       -5.229e+00  1.854e+00  -2.820 0.004807 **
## charSemicolon    -7.242e-01  3.401e-01  -2.129 0.033214 *
## charExclamation   3.584e-01  8.380e-02   4.277 1.90e-05 ***
## charDollar        6.308e+00  7.412e-01   8.510  < 2e-16 ***
## capitalAve        7.750e-02  1.450e-02   5.344 9.07e-08 ***
## capitalTotal      9.556e-04  1.692e-04   5.647 1.64e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5662.7  on 4209  degrees of freedom
## Residual deviance: 1857.2  on 4187  degrees of freedom
## AIC: 1903.2
##
## Number of Fisher Scoring iterations: 11

print("BIC")

## [1] "BIC"

BIC(fit.bss)

## [1] 2049.131
```

We observe that all the variables including the model are significant.

(iii) One single Decision tree

```
set.seed(123)
library(rpart)
control.0 <- rpart.control(minsplit=10, minbucket=5, maxdepth=8, cp=0,
                           maxcompete=2,maxsurrogate=2, usesurrogate=2,
                           surrogatestyle=0,xval=10)

fit.tree <- rpart(type ~ ., data=dat.training, method="anova",
control=control.0)
yhat.tree <- predict(fit.tree, newdata=dat.test)
yhat.tree <- as.vector(as.numeric(yhat.tree))
fit.tree$variable.importance

##       charDollar          remove           money          num000
##     317.69678700    161.20350429    108.61854289    107.97222793
##  charExclamation              hp     capitalLong    capitalTotal
##      82.37583631     56.67776649     40.43512535     36.75471454
##             free             edu             hpl      capitalAve
##      35.14987237     29.25033244     29.23631597     26.98478087
##           george        business          telnet          num857
```

```
##      20.97246805       8.17194972       7.93229809       7.40722831
##              you           num415              our             your
##       7.10769180       6.73794726       5.69717025       5.38257505
##             over            email          meeting             labs
##       4.83252442       3.66305248       3.56025827       3.28012786
##             font               cs         charHash    charSemicolon
##       3.06000000       2.93150227       2.72000000       2.30423042
##             will charRoundbracket              all          receive
##       1.73399015       1.47049849       1.24411098       1.19226585
##            order          num1999         internet          address
##       0.77230931       0.68000000       0.54250000       0.20166667
##             make         original             mail            num3d
##       0.18387097       0.15221177       0.13833226       0.10277576
##           people               pm           report
##       0.07354839       0.06666667       0.03807148
```

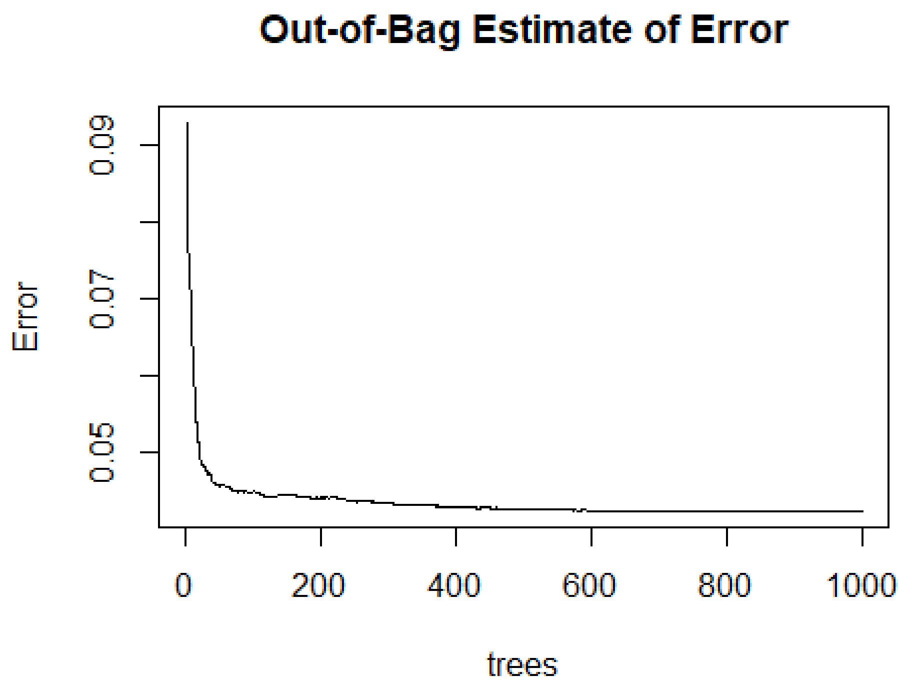
We observed that most of the variables that were found to be significant in the BSS model were also found to be important variables in the one single decision tree. However, there are a few of them that differs such as internet,credit,re,capitalTotal and conference.

(iv) Random Forest

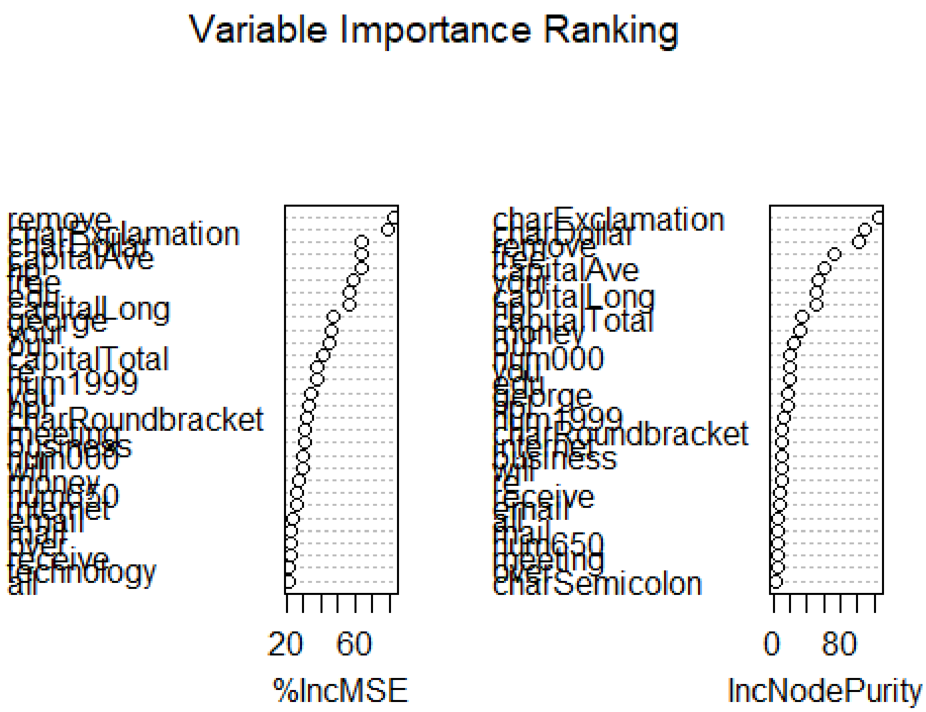
```
set.seed(123)
suppressPackageStartupMessages(library(randomForest))
# SEARCH THE BEST mtry PARAMETER, NUMBER OF VARIABLES RANDOMLY SAMPLED AT EACH SPLIT
m.try <- suppressWarnings(tuneRF(dat.training[ , -c(58)], dat.training[,58], ntreeTry=50, stepFactor=2,
                improve=0.05, trace=F, plot=F, dobest=FALSE))

## 0.02790792 0.05
## -0.02098307 0.05

best.m <- m.try[m.try[, 2] == min(m.try[, 2]), 1]
best.m

## [1] 10
```

the best number of variables randomly sample at each split is 10 and the out of bag error becomes low and stable at 1000 trees.

```
# RANDOM FOREST
fit.rf <- suppressWarnings(randomForest(type ~ ., data= dat.training, mtry=best.m, ntree=1000,
                        keep.forest=TRUE, importance=TRUE, proximity=TRUE, oob.prox=FALSE))
yhat.rf <- predict(fit.rf, newdata=dat.test, type="response")
yhat.rf <- as.vector(as.numeric(yhat.rf))
plot(fit.rf, main="Out-of-Bag Estimate of Error")
```




```
fit.rf; varImpPlot(fit.rf, main="Variable Importance Ranking")

##
## Call:
##  randomForest(formula = type ~ ., data = dat.training, mtry = best.m,
ntree = 1000, keep.forest = TRUE, importance = TRUE, proximity = TRUE,
oob.prox = FALSE)
##                Type of random forest: regression
##                      Number of trees: 1000
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 0.04248681
##                     % Var explained: 82.28
```



From the random forest model, the number of variables randomly sampled at each split is 10, the mean squared of residual were 0.0427 and the variation explained by the model were 82.16%. We also observe highly significant variables such as make, charEclamation, CharDollar,edu,george,etc, some of which were also highly significant in the single Decision tree and BSS.

(v) Boosting

```
set.seed(123)
suppressPackageStartupMessages(library(mboost))
dat <- dat.training
dat$type <- as.factor(dat$type)
control.boost <- boost_control(mstop = 1000, risk = "oobag")
fit.boost <- glmboost(type ~ ., data=dat.training, center = FALSE,
                      control=control.boost,
                      family = Binomial(type = c("glm")))
yhat.boost <- predict(fit.boost, newdata=dat.test, type="response")
yhat.boost <- as.vector(as.numeric(yhat.boost))
summary(fit.boost)

##
##   Generalized Linear Models Fitted via Gradient Boosting
##
## Call:
## glmboost.formula(formula = type ~ ., data = dat.training, family =
Binomial(type = c("glm")),     center = FALSE, control = control.boost)
##
```

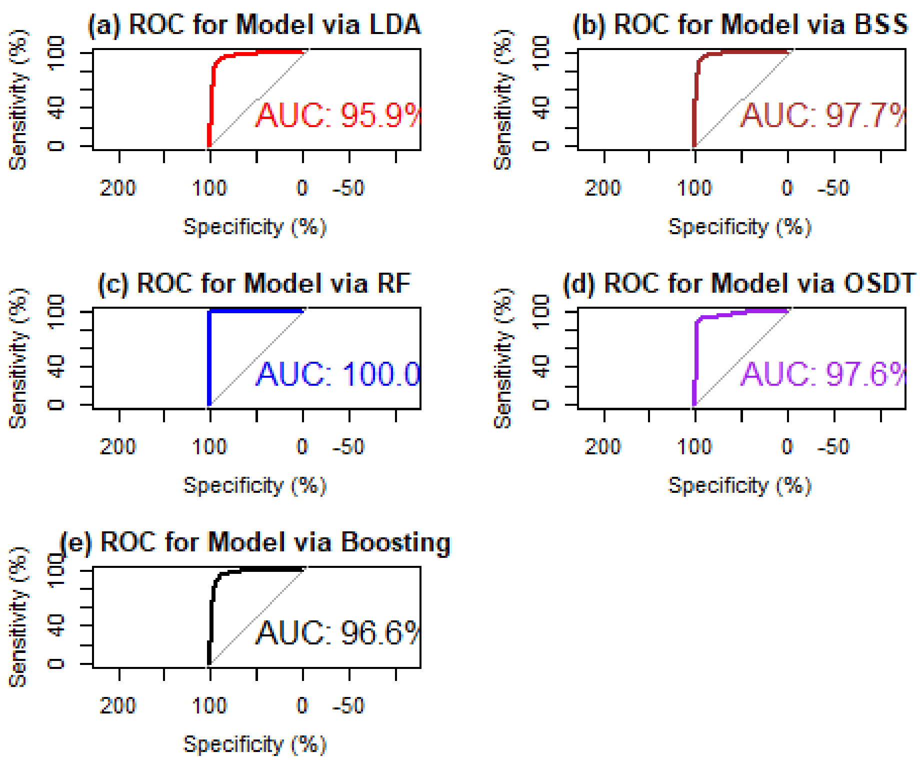
```
##
##   Binomial Distribution (similar to glm)
##
## Loss function: {
##      ntrials <- rowSums(y)
##      y <- y[, 1]
##      p <- link$linkinv(f)
##      -dbinom(x = y, size = ntrials, prob = p, log = log)
##  }
##
##
## Number of boosting iterations: mstop = 1000
## Step size:  0.1
## Offset:  -0.2030703
##
## Coefficients:
##      (Intercept)               our              over            remove
##    -0.5365927241      0.1086049640      0.0920213234      1.3363515610
##         internet             order           receive              will
##     0.1577336351      0.0860287852      0.5228684905     -0.0028110089
##             free          business            credit              your
##     0.3354587873      0.2091771347      0.1307921910      0.1812298960
##             font            num000             money                hp
##     0.0086035936      1.0216773507      0.1580386857     -0.3758580484
##              hpl            george              data           num1999
##    -0.1766131639     -0.1058070086     -0.1021556735     -0.3801541425
##          meeting           project                re               edu
##    -0.1794698209     -0.0246799691     -0.1732198957     -0.1896584781
## charRoundbracket  charExclamation        charDollar       capitalLong
##    -0.0566143520      0.1465558205      2.1279065928      0.0002448152
##     capitalTotal
##     0.0001117738
## attr(,"offset")
## [1] -0.2030703
##
## Selection frequencies:
##      (Intercept)                hp        charDollar            remove
##            0.120             0.111             0.094             0.092
##           num000              free              your            george
##            0.059             0.054             0.043             0.040
##               re               edu           meeting   charExclamation
##            0.038             0.038             0.034             0.031
##              hpl           num1999               our          business
##            0.027             0.026             0.023             0.022
##     capitalTotal           receive          internet             money
##            0.020             0.019             0.017             0.017
##           credit              data       capitalLong              over
##            0.016             0.015             0.015             0.008
##            order           project  charRoundbracket              font
##            0.007             0.005             0.005             0.003
```

```
##              will
##             0.001
```

We observed that Boosting have similar variables as the others, however, it also have variables that differ from the others. The initial number of boosting iterations, step size and other hyper-parameters for boosting algorithms was determined by using boost.control()

Model Comparison: ROC Curves and AUC Values

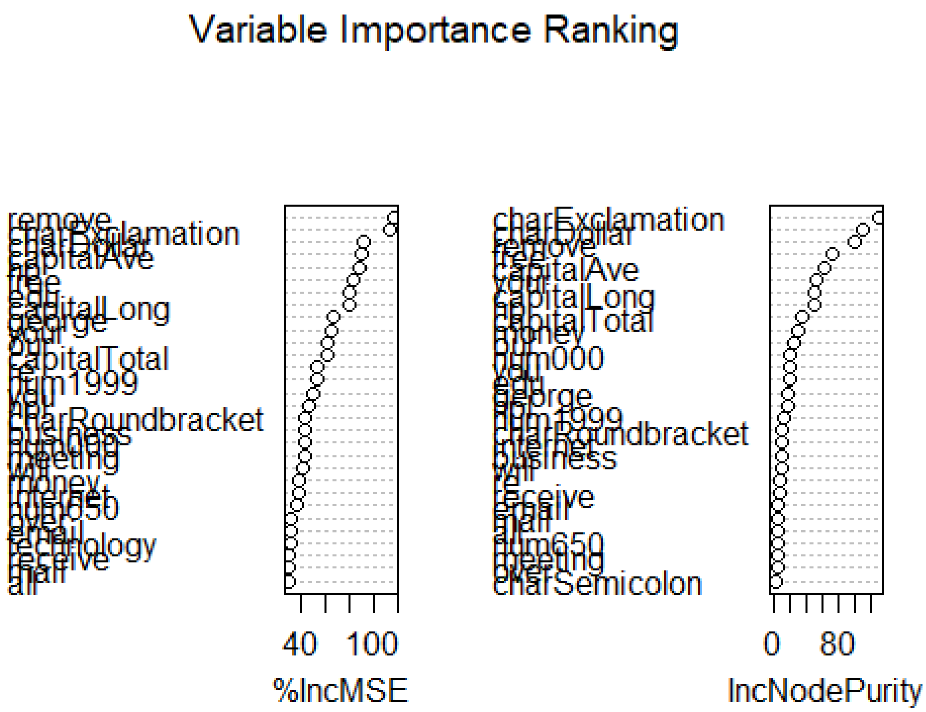
```
set.seed(200)
suppressPackageStartupMessages(library(pROC))
yobs <- as.vector(as.numeric(dat.test$type))
par(mfrow=c(3, 2), mar=rep(4,4))
roc.LDA <- suppressMessages(plot.roc(yobs, yhat.LDA,  ylim=c(0, 100),
    main="(a) ROC for Model via LDA", percent=TRUE,
    print.auc=TRUE, print.auc.cex=1.5, col="red"))
roc.bss <- suppressMessages(plot.roc(yobs, yhat.bss, ylim=c(0, 100),
    main="(b) ROC for Model via BSS", percent=TRUE,
    print.auc=TRUE, print.auc.cex=1.5, col="brown"))
roc.rf <- suppressMessages(plot.roc(yobs, yhat.rf,  ylim=c(0, 100),
    main="(c) ROC for Model via RF", percent=TRUE,
    print.auc=TRUE, print.auc.cex=1.5, col="blue"))
roc.tree <- suppressMessages(plot.roc(yobs, yhat.tree, ylim=c(0, 100),
    main="(d) ROC for Model via OSDT", percent=TRUE,
    print.auc=TRUE, print.auc.cex=1.5, col="purple"))
roc.boost <- suppressMessages(plot.roc(yobs, yhat.boost, ylim=c(0, 100),
    main="(e) ROC for Model via Boosting", percent=TRUE,
    print.auc=TRUE, print.auc.cex=1.5, col="black"))
```



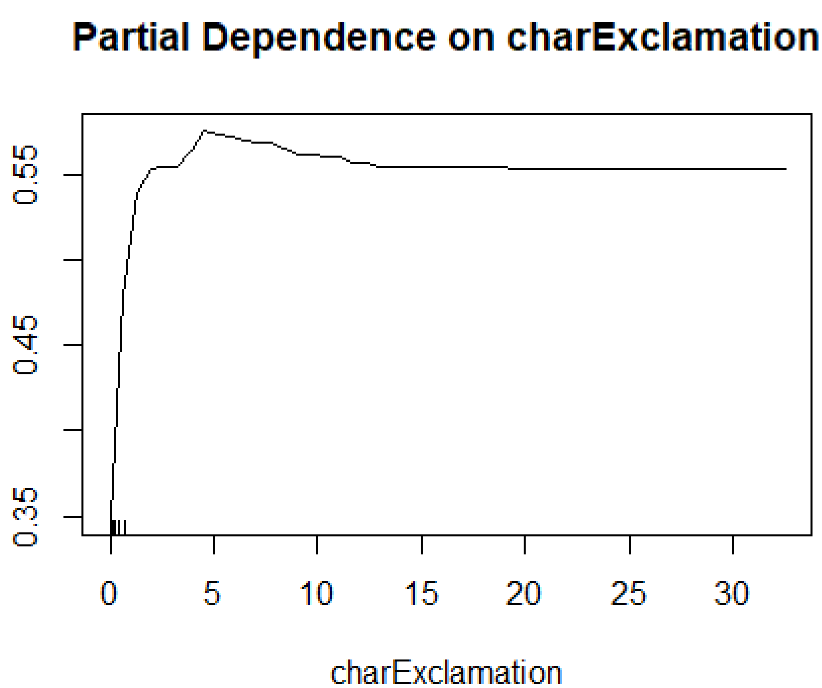
From the above ROC plots it can be observed that the model with the highest area under the curve (AUC) is the Random Forest, followed by BSS,One Single Decision Tree, Boosting and then Linear Descriminant Analysis. Per AUC, the random Forest model outperformed others on this data set.

Additional Features from RF

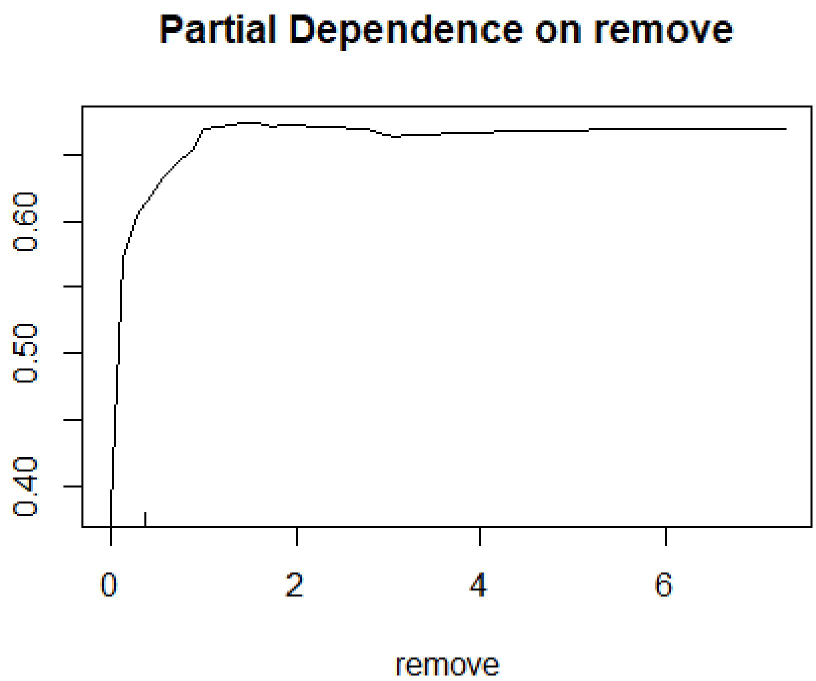
```
set.seed(123)
# SEARCH THE BEST mtry PARAMETER, NUMBER OF VARIABLES RANDOMLY SAMPLED AT
EACH SPLIT
m.try <- suppressWarnings(tuneRF(spam[ , -c(58)], spam[,58], ntreeTry=50,
stepFactor=2,
                improve=0.05, trace=F, plot=F, dobest=FALSE))

## 0.02790792 0.05
## -0.02098307 0.05

best.m <- m.try[m.try[, 2] == min(m.try[, 2]), 1]
# RANDOM FOREST
fit.rffd <- suppressWarnings(randomForest(type ~ ., data= spam,  mtry=best.m,
ntree=2000,
                       keep.forest=TRUE, importance=TRUE, proximity=TRUE,
oob.prox=FALSE))
# Variable Importance plot
varImpPlot(fit.rffd, main="Variable Importance Ranking")
```




```
partialPlot(fit.rffd, pred.data=spam, x.var=charExclamation, rug=TRUE)
```



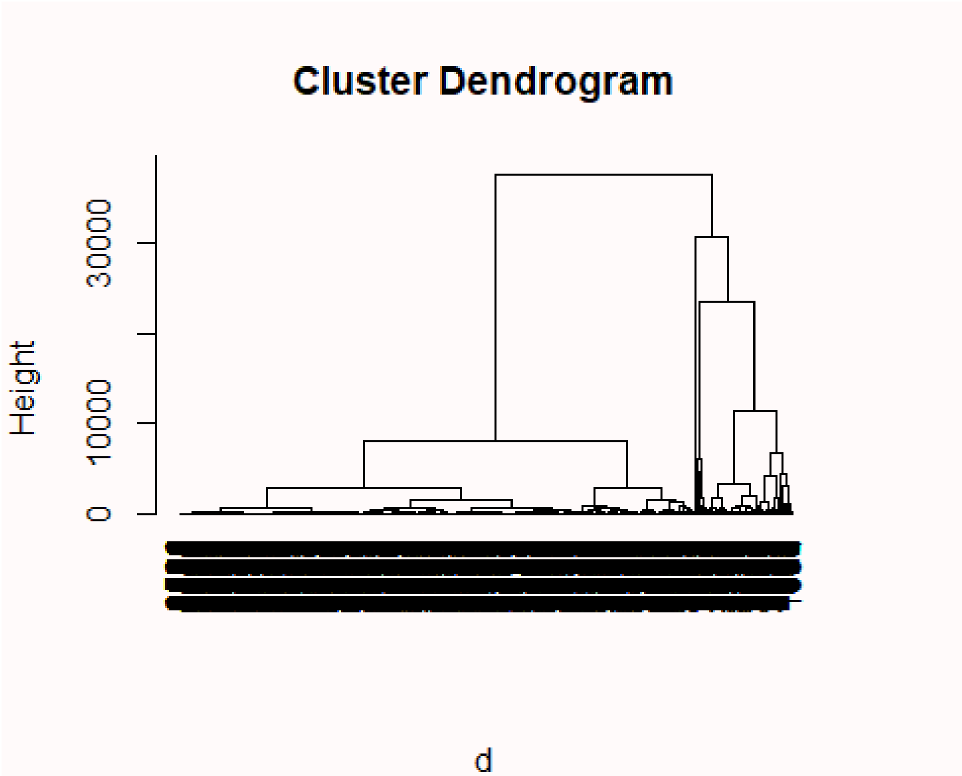

```
partialPlot(fit.rffd, pred.data=spam, x.var=remove, rug=TRUE)
```



In this subsection, we perform Random Forest on the entire data set, and choose the number of tree to be 2000. We observed all the variables, with the exception of credit, data, project, conference, and charSemicolon, that were selected by BSS were also selected as important variables by RF. Also, variables that were ranked highly significant by RF were also found to be highly significant by BSS e.g., $remove$, $charExclamation$, $charDollar$, hp, $free$, etc.

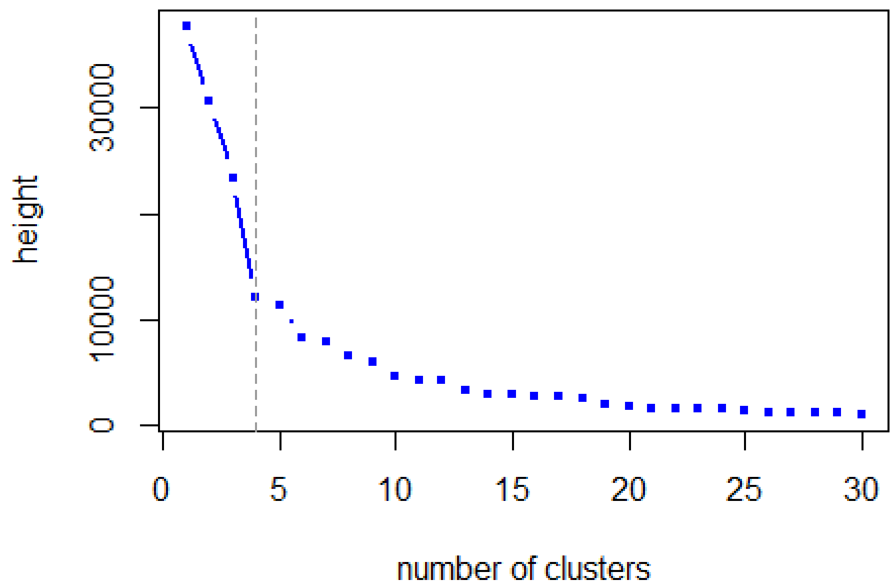
(4)(a) Use one clustering method of your choice to cluster training sample into K groups and add a new column cluster to the training sample.

```
# Hierarchical Clustering
d <- dist(dat.training, method = "euclidean") # distance matrix

fit.ward.D2 <- hclust(d, method="ward.D2")
fit.ward <- hclust(d, method="ward.D")
#fit.complete <- hclust(d, method="complete")

fit <- fit.ward.D2
par(mfrow=c(1,1), bg="snow", mar=rep(4,4))
plot(fit, hang = -0.5)
```

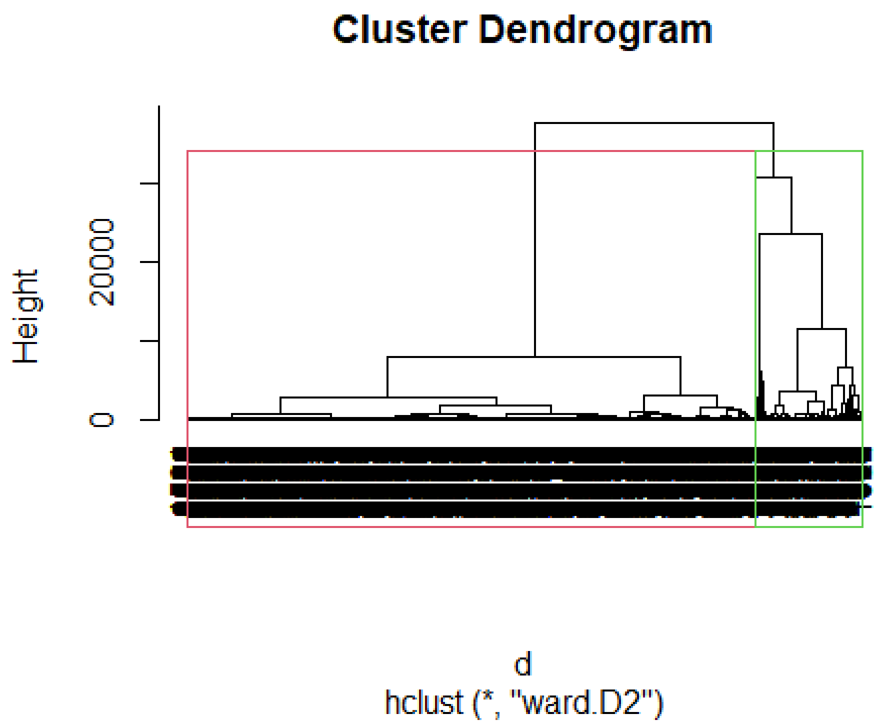


From the above output, I used hierarchical clustering to cluster the training sample into two groups.

```
# SCREE PLOT OF HEIGHT IN HIERARCHICAL CLUSTERING
K.max <- 30
height <- tail(fit$height, n=K.max)
n.cluster <- tail((nrow(dat.training)-1):1, n=K.max)
plot(n.cluster, height,  type="b", pch=19, cex=.5, xlab="number of clusters",
    ylab="height", col="blue", lwd=2)
abline(v=4, col="gray60", lty=2)
```



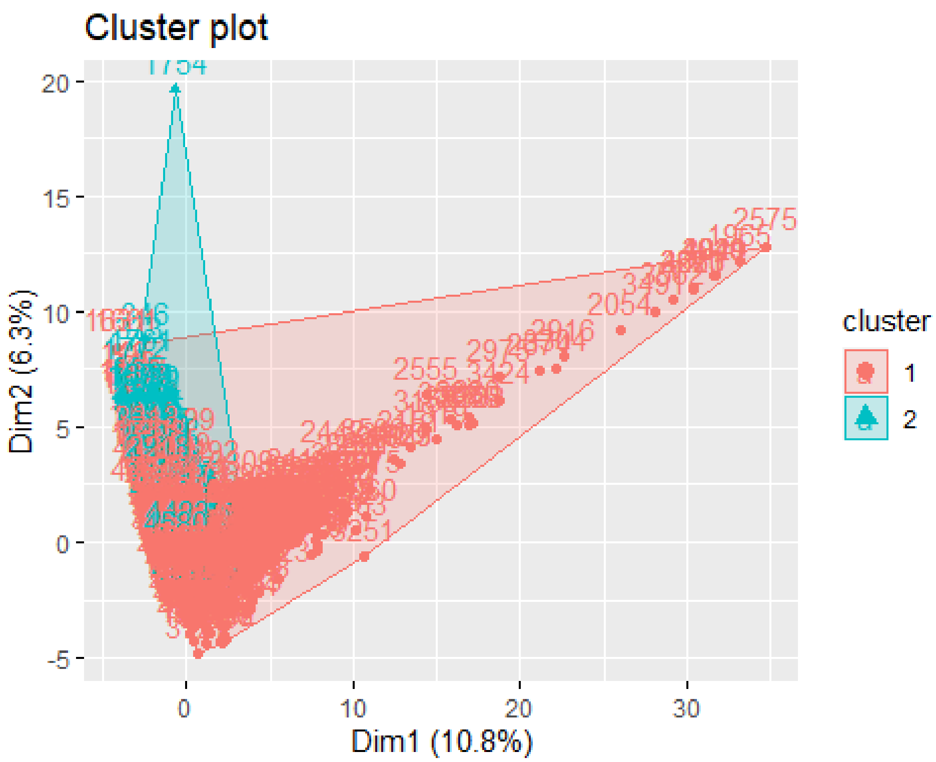

```
groups <- cutree(fit, k=2) # cut tree into 2 clusters
dat0 <- data.frame(dat.training, h.cluster=groups) # COLLECT THE CLUSTER
MEMBERSHIP
# draw dendogram with red borders around the 4 clusters
plot(fit, hang = -0.5)
rect.hclust(fit, k=2, border=2:5)
```



From the scree plot, we choose k to be 2

Another way to view hierarchical clustering.

```
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa

fviz_cluster(list(data = dat0, cluster = groups))
```




```
dat1<-rbind(dat0)
dat1<-dat1[,-58]
head(dat1);dim(dat1)

##   make address  all num3d  our over remove internet order mail receive
will
## 1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00  0.00 0.00    0.00
0.64
## 2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07  0.00 0.94    0.21
0.79
## 3 0.06    0.00 0.71     0 1.23 0.19   0.19     0.12  0.64 0.25    0.38
0.45
## 4 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31
0.31
## 5 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31
0.31
## 6 0.00    0.00 0.00     0 1.85 0.00   0.00     1.85  0.00 0.00    0.00
0.00
##   people report addresses free business email  you credit your font num000
## 1   0.00   0.00      0.00 0.32     0.00  1.29 1.93   0.00 0.96    0   0.00
## 2   0.65   0.21      0.14 0.14     0.07  0.28 3.47   0.00 1.59    0   0.43
## 3   0.12   0.00      1.75 0.06     0.06  1.03 1.36   0.32 0.51    0   1.16
## 4   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0   0.00
## 5   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0   0.00
## 6   0.00   0.00      0.00 0.00     0.00  0.00 0.00   0.00 0.00    0   0.00
##   money hp hpl george num650 lab labs telnet num857 data num415 num85
## 1  0.00  0   0      0      0   0    0      0      0    0      0     0
```

```
## 2  0.43  0   0      0      0   0    0      0      0    0      0     0
## 3  0.06  0   0      0      0   0    0      0      0    0      0     0
## 4  0.00  0   0      0      0   0    0      0      0    0      0     0
## 5  0.00  0   0      0      0   0    0      0      0    0      0     0
## 6  0.00  0   0      0      0   0    0      0      0    0      0     0
##   technology num1999 parts pm direct cs meeting original project   re  edu
## 1          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 2          0    0.07     0  0   0.00  0       0     0.00       0 0.00 0.00
## 3          0    0.00     0  0   0.06  0       0     0.12       0 0.06 0.06
## 4          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 5          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 6          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
##   table conference charSemicolon charRoundbracket charSquarebracket
## 1     0          0          0.00            0.000                 0
## 2     0          0          0.00            0.132                 0
## 3     0          0          0.01            0.143                 0
## 4     0          0          0.00            0.137                 0
## 5     0          0          0.00            0.135                 0
## 6     0          0          0.00            0.223                 0
##   charExclamation charDollar charHash capitalAve capitalLong capitalTotal
## 1           0.778      0.000    0.000      3.756          61          278
## 2           0.372      0.180    0.048      5.114         101         1028
## 3           0.276      0.184    0.010      9.821         485         2259
## 4           0.137      0.000    0.000      3.537          40          191
## 5           0.135      0.000    0.000      3.537          40          191
## 6           0.000      0.000    0.000      3.000          15           54
##   h.cluster
## 1         1
## 2         2
## 3         2
## 4         1
## 5         1
## 6         1

## [1] 4210   58
```

I removed the response variable type from the data set and we have 58 variables because we added a new column cluster to the data set.

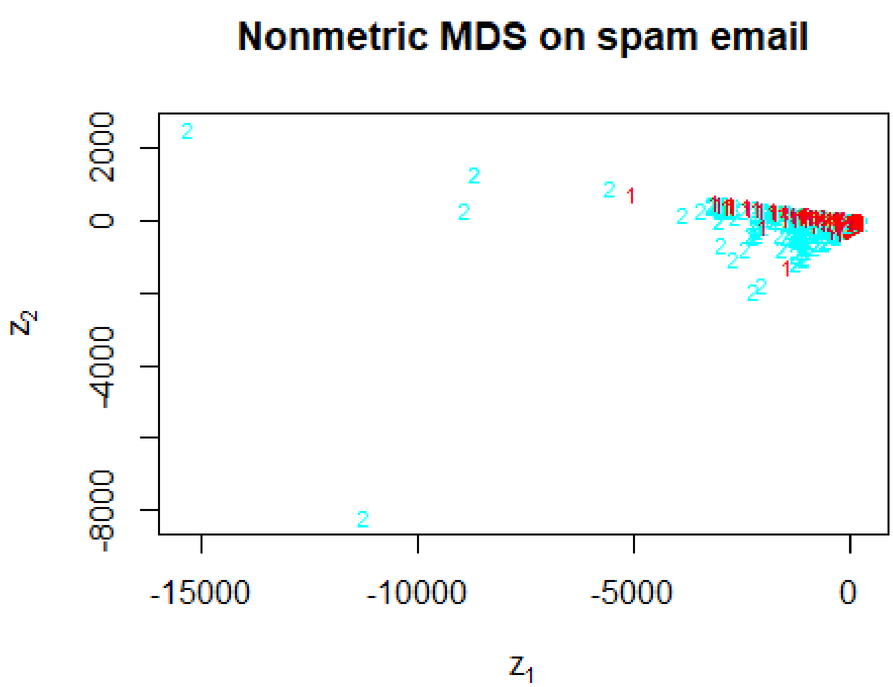
(4)(b) Re-plot the training data via MDS or tSNE results in Part 2, highlighting cluster membership of each point with different colors and the target type with different symbols.

```
# Nonmetric MDS
# ========================

set.seed(200)
mat<- unique.data.frame(dat1)

# Nonmetric MDS is performed using the isoMDS( ) function in the MASS
package.
```

```
library(MASS)
d <- dist(mat) # euclidean distances between the rows
fit <- isoMDS(d, k=2) # k is the number of dim
```

```
## initial  value 0.567372
## final  value 0.567367
## converged
```

```
colors = rainbow(length(unique(dat.training$type)))
names(colors) = unique(dat.training$type)

# plot solution
x <- fit$points[,1]; y <- fit$points[,2]
plot(x, y, xlab=expression(z[1]), ylab=expression(z[2]),
  main="Nonmetric MDS on spam email", type="n")

text(x,y, labels = dat1$h.cluster, cex=.7,col=colors[dat1$h.cluster])
```



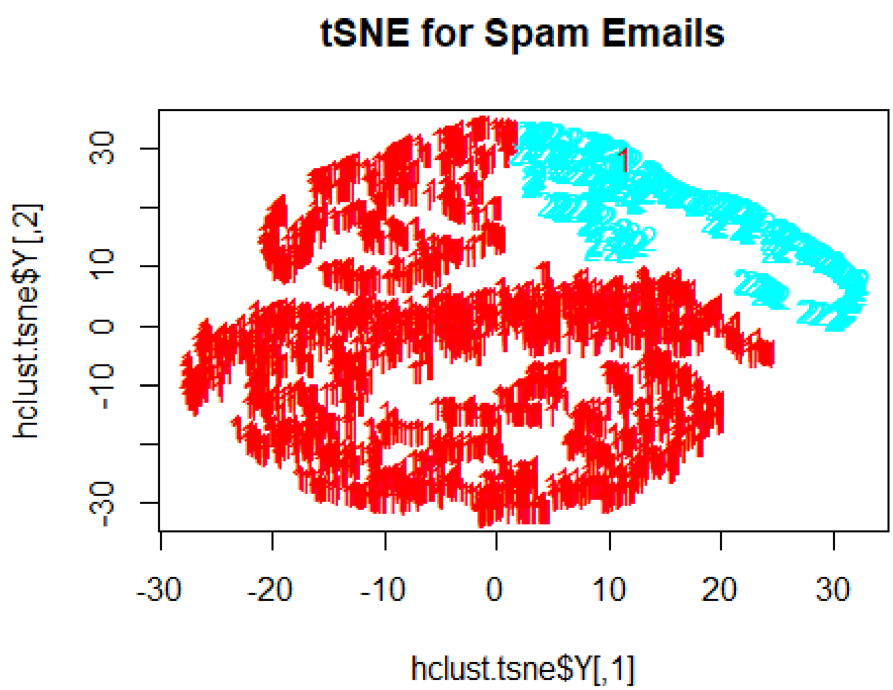

```
# dev.off()
```

The above output shows a non metric MDS of the new data with their cluster membership.

```
library(Rtsne)
library(cluster)
set.seed(200)
df<- unique.data.frame(dat1)
dis <- daisy(df, metric="gower", stand=TRUE)
```

```
## Warning in daisy(df, metric = "gower", stand = TRUE): binary variable(s)
58
## treated as interval scaled
```

```r
fit.ward1<-hclust(dis,method = "ward.D2")
hclust.groups <- cutree(fit.ward1, k=2)

colors = rainbow(length(unique(hclust.groups)))
names(colors) = unique(hclust.groups)
set.seed(5000)
hclust.tsne <- Rtsne(df, dims=2, perplexity=30, max_iter=500)
plot(hclust.tsne$Y, t="n", main = "tSNE for Spam Emails")
text(hclust.tsne$Y, labels = hclust.groups, col = colors[hclust.groups])
```



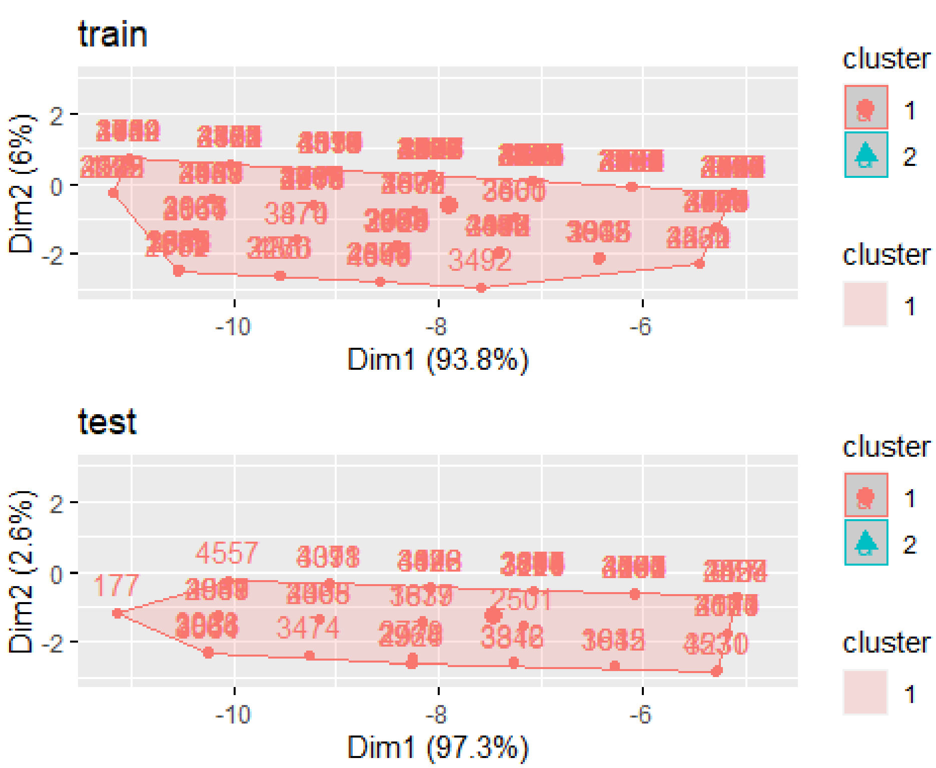
The above output shows clustering the data into two different clusters using tsne.

(4)(c) Accordingly predict the cluster membership for each observation in the test sample.

```r
library(scorecard)
```

```
##
## Attaching package: 'scorecard'

## The following object is masked from 'package:psych':
##
##     describe
```

```
library(factoextra)
library(class)

knnClust <- knn(train = dat1[,-58], test = dat.test[,-58] , k = 2, cl =
groups)


p1 <- fviz_cluster(list(data = dat1[,-58], cluster = groups), stand = F) +
xlim(-11.2,-4.8) + ylim(-3,3) + ggtitle("train")
p2 <- fviz_cluster(list(data = dat.test[,-58], cluster = knnClust),stand = F)
+ xlim(-11.2,-4.8) + ylim(-3,3) + ggtitle("test")
gridExtra::grid.arrange(p1,p2,nrow = 2)

## Warning: Removed 4005 rows containing non-finite values (stat_chull).

## Warning: Removed 4005 rows containing non-finite values (stat_mean).

## Warning: Removed 4005 rows containing missing values (geom_point).

## Warning: Removed 4005 rows containing missing values (geom_text).

## Warning: Removed 1340 rows containing non-finite values (stat_chull).

## Warning: Removed 1340 rows containing non-finite values (stat_mean).

## Warning: Removed 1340 rows containing missing values (geom_point).

## Warning: Removed 1340 rows containing missing values (geom_text).
```



```
pca1 <- data.frame(prcomp(dat1[,-58], scale. = T)$x[,1:2], cluster =
as.factor(groups), factor = "train")
pca2 <- data.frame(prcomp(dat.test[,-58], scale. = T)$x[,1:2], cluster =
as.factor(knnClust), factor = "test")
pca <- as.data.frame(rbind(pca1,pca2))
```

I used KNN for this part where the cluster membership is used as the target variable and others predictors.

(4d) Fit a 'best' logistic regression model by including interactions term between cluster and all other predictors and applying regularization. We may call this model a 'clustered logistic regression model'.

```
formula1 <- cluster ~ PC1+factor
fit.fulla <- glm(formula1, family=binomial, data=pca)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# ?glm
summary(fit.fulla); # names(summary(fit.full))

##
## Call:
## glm(formula = formula1, family = binomial, data = pca)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4733  -0.6446  -0.5450  -0.2611   3.2220
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.06189    0.08491 -24.284   <2e-16 ***
## PC1         -0.81590    0.05433 -15.018   <2e-16 ***
## factortrain -0.02231    0.08561  -0.261    0.794
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4958.6  on 5612  degrees of freedom
## Residual deviance: 4599.3  on 5610  degrees of freedom
## AIC: 4605.3
##
## Number of Fisher Scoring iterations: 6

BIC(fit.fulla)

## [1] 4625.159

formula2 <- cluster ~ PC2+factor
fit.full <- glm(formula2, family=binomial, data=pca)
```

```
# ?glm
summary(fit.full); # names(summary(fit.full))
```

```
##
## Call:
## glm(formula = formula2, family = binomial, data = pca)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3814  -0.5574  -0.4454  -0.3663   2.6748
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.82173    0.07893 -23.081   <2e-16 ***
## PC2          0.46642    0.02037  22.895   <2e-16 ***
## factortrain -0.04659    0.08948  -0.521    0.603
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4958.6  on 5612  degrees of freedom
## Residual deviance: 4361.4  on 5610  degrees of freedom
## AIC: 4367.4
##
## Number of Fisher Scoring iterations: 5
```

```
BIC(fit.full)
```

```
## [1] 4387.313
```

From the above output, I fitted a logistic regression within each cluster.

(e) Apply the 'clustered logistic regression model' in (d) to the test sample. Obtain the ROC curve based on the test sample predictions. Compare the resultant AUC with the 'best' logistic regression model in Part 3.

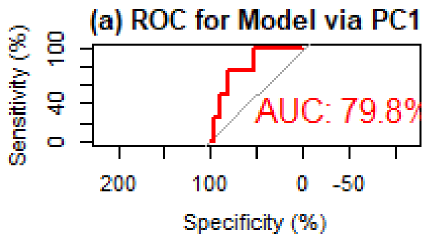
```
set.seed(1000)
dat.test0 <- pca2[sample(1:NROW(pca), 100, replace=FALSE), ]
pred <- predict(fit.fulla, newdata=dat.test0, type="response", se.fit=TRUE);
pred
```

```
## $fit
##           NA          3496          NA.1          3197          NA.2
NA.3
##           NA  0.1321113151            NA  0.1642018281            NA
NA
##         1331          2088          NA.4          3576          NA.5
3972
## 0.2232512177 0.0042099001            NA  0.1608367104            NA
0.1062281491
##         NA.6          NA.7          NA.8          NA.9         NA.10
```

```
NA.11
##          NA          NA          NA          NA          NA
NA
##       NA.12       NA.13       NA.14       NA.15       NA.16
NA.17
##          NA          NA          NA          NA          NA
NA
##       NA.18        2144       NA.19         702       NA.20
NA.21
##          NA 0.0154720895          NA 0.1693176071          NA
NA
##       NA.22       NA.23        1507       NA.24       NA.25
NA.26
##          NA          NA 0.1542171187          NA          NA
NA
##         491         302       NA.27       NA.28        1870
1805
## 0.3201284339 0.4628140812          NA          NA 0.1286840288
0.1357733390
##       NA.29       NA.30         971       NA.31       NA.32
719
##          NA          NA 0.2270430895          NA          NA
0.2327554130
##       NA.33       NA.34       NA.35       NA.36       NA.37
NA.38
##          NA          NA          NA          NA          NA
NA
##       NA.39       NA.40        2064       NA.41       NA.42
NA.43
##          NA          NA 0.1217641688          NA          NA
NA
##       NA.44       NA.45       NA.46       NA.47         416
NA.48
##          NA          NA          NA          NA 0.2653856720
NA
##       NA.49       NA.50       NA.51       NA.52        3386
NA.53
##          NA          NA          NA          NA 0.1874504977
NA
##        1355       NA.54         810       NA.55       NA.56
NA.57
## 0.2108782702          NA 0.3501205208          NA          NA
NA
##        1837       NA.58       NA.59       NA.60       NA.61
NA.62
## 0.1673125255          NA          NA          NA          NA
NA
##       NA.63       NA.64       NA.65       NA.66       NA.67
NA.68
##          NA          NA          NA          NA          NA
```

```
NA
##            2         2427         4372        NA.69          628
NA.70
## 0.2810011312 0.0001922236 0.1904994775           NA 0.1493006759
NA
##        NA.71        NA.72        NA.73        NA.74
##           NA           NA           NA           NA
##
## $se.fit
##           NA         3496         NA.1         3197         NA.2
NA.3
##           NA 9.135261e-03           NA 1.028401e-02           NA
NA
##         1331         2088         NA.4         3576         NA.5
3972
## 1.297058e-02 1.168054e-03           NA 1.015493e-02           NA
8.282432e-03
##         NA.6         NA.7         NA.8         NA.9        NA.10
NA.11
##           NA           NA           NA           NA           NA
NA
##        NA.12        NA.13        NA.14        NA.15        NA.16
NA.17
##           NA           NA           NA           NA           NA
NA
##        NA.18         2144        NA.19          702        NA.20
NA.21
##           NA 2.981016e-03           NA 1.048484e-02           NA
NA
##        NA.22        NA.23         1507        NA.24        NA.25
NA.26
##           NA           NA 9.907661e-03           NA           NA
NA
##          491          302        NA.27        NA.28         1870
1805
## 1.882475e-02 2.805099e-02           NA           NA 9.020972e-03
9.258623e-03
##        NA.29        NA.30          971        NA.31        NA.32
719
##           NA           NA 1.317051e-02           NA           NA
1.347745e-02
##        NA.33        NA.34        NA.35        NA.36        NA.37
NA.38
##           NA           NA           NA           NA           NA
NA
##        NA.39        NA.40         2064        NA.41        NA.42
NA.43
##           NA           NA 8.792708e-03           NA           NA
NA
##        NA.44        NA.45        NA.46        NA.47          416
```

```
NA.48
##          NA          NA          NA          NA 1.535055e-02
NA
##       NA.49       NA.50       NA.51       NA.52         3386
NA.53
##          NA          NA          NA          NA 1.124431e-02
NA
##        1355       NA.54         810       NA.55        NA.56
NA.57
## 1.234033e-02          NA 2.081764e-02          NA           NA
NA
##        1837       NA.58       NA.59       NA.60        NA.61
NA.62
## 1.040545e-02          NA          NA          NA           NA
NA
##       NA.63       NA.64       NA.65       NA.66        NA.67
NA.68
##          NA          NA          NA          NA           NA
NA
##           2        2427        4372       NA.69          628
NA.70
## 1.630786e-02 9.228174e-05 1.137958e-02          NA 9.729342e-03
NA
##       NA.71       NA.72       NA.73       NA.74
##          NA          NA          NA          NA
##
## $residual.scale
## [1] 1

yhat <- pred$fit

suppressPackageStartupMessages(library(pROC))
yobs <- as.vector(as.numeric(dat.test0$cluster))
par(mfrow=c(3, 2), mar=rep(4,4))
roc.pca1 <- suppressMessages(plot.roc(yobs, yhat,  ylim=c(0, 100),
    main="(a) ROC for Model via PC1", percent=TRUE,
    print.auc=TRUE, print.auc.cex=1.5, col="red"))
```



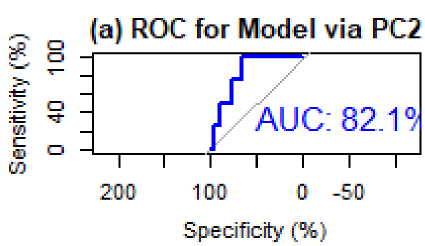

```
set.seed(1000)
dat.test0 <- pca2[sample(1:NROW(pca), 100, replace=FALSE), ]
pred1 <- predict(fit.full, newdata=dat.test0, type="response", se.fit=TRUE);
pred

## $fit
##            NA         3496           NA.1         3197           NA.2
NA.3
##            NA 0.1321113151             NA 0.1642018281             NA
NA
##          1331         2088           NA.4         3576           NA.5
3972
## 0.2232512177 0.0042099001             NA 0.1608367104             NA
0.1062281491
##          NA.6         NA.7           NA.8         NA.9          NA.10
NA.11
##            NA           NA             NA           NA             NA
NA
##         NA.12        NA.13          NA.14        NA.15          NA.16
NA.17
##            NA           NA             NA           NA             NA
NA
##         NA.18         2144          NA.19          702          NA.20
NA.21
##            NA 0.0154720895             NA 0.1693176071             NA
NA
##         NA.22        NA.23           1507        NA.24          NA.25
```

```
NA.26
##          NA          NA 0.1542171187          NA          NA
NA
##         491         302         NA.27        NA.28        1870
1805
## 0.3201284339 0.4628140812          NA          NA 0.1286840288
0.1357733390
##       NA.29       NA.30          971        NA.31       NA.32
719
##          NA          NA 0.2270430895          NA          NA
0.2327554130
##       NA.33       NA.34        NA.35        NA.36       NA.37
NA.38
##          NA          NA          NA          NA          NA
NA
##       NA.39       NA.40         2064        NA.41       NA.42
NA.43
##          NA          NA 0.1217641688          NA          NA
NA
##       NA.44       NA.45        NA.46        NA.47         416
NA.48
##          NA          NA          NA          NA 0.2653856720
NA
##       NA.49       NA.50        NA.51        NA.52        3386
NA.53
##          NA          NA          NA          NA 0.1874504977
NA
##        1355       NA.54          810        NA.55       NA.56
NA.57
## 0.2108782702          NA 0.3501205208          NA          NA
NA
##        1837       NA.58        NA.59        NA.60       NA.61
NA.62
## 0.1673125255          NA          NA          NA          NA
NA
##       NA.63       NA.64        NA.65        NA.66       NA.67
NA.68
##          NA          NA          NA          NA          NA
NA
##           2        2427         4372        NA.69         628
NA.70
## 0.2810011312 0.0001922236 0.1904994775          NA 0.1493006759
NA
##       NA.71       NA.72        NA.73        NA.74
##          NA          NA          NA          NA
##
## $se.fit
##          NA        3496         NA.1         3197        NA.2
NA.3
##          NA 9.135261e-03          NA 1.028401e-02          NA
```

```
NA
##         1331         2088         NA.4         3576         NA.5
3972
## 1.297058e-02 1.168054e-03           NA 1.015493e-02           NA
8.282432e-03
##         NA.6         NA.7         NA.8         NA.9        NA.10
NA.11
##           NA           NA           NA           NA           NA
NA
##        NA.12        NA.13        NA.14        NA.15        NA.16
NA.17
##           NA           NA           NA           NA           NA
NA
##        NA.18         2144        NA.19          702        NA.20
NA.21
##           NA 2.981016e-03           NA 1.048484e-02           NA
NA
##        NA.22        NA.23         1507        NA.24        NA.25
NA.26
##           NA           NA 9.907661e-03           NA           NA
NA
##          491          302        NA.27        NA.28         1870
1805
## 1.882475e-02 2.805099e-02           NA           NA 9.020972e-03
9.258623e-03
##        NA.29        NA.30          971        NA.31        NA.32
719
##           NA           NA 1.317051e-02           NA           NA
1.347745e-02
##        NA.33        NA.34        NA.35        NA.36        NA.37
NA.38
##           NA           NA           NA           NA           NA
NA
##        NA.39        NA.40         2064        NA.41        NA.42
NA.43
##           NA           NA 8.792708e-03           NA           NA
NA
##        NA.44        NA.45        NA.46        NA.47          416
NA.48
##           NA           NA           NA           NA 1.535055e-02
NA
##        NA.49        NA.50        NA.51        NA.52         3386
NA.53
##           NA           NA           NA           NA 1.124431e-02
NA
##         1355        NA.54          810        NA.55        NA.56
NA.57
## 1.234033e-02           NA 2.081764e-02           NA           NA
NA
##         1837        NA.58        NA.59        NA.60        NA.61
```

```
NA.62
## 1.040545e-02            NA            NA            NA            NA
NA
##         NA.63         NA.64         NA.65         NA.66         NA.67
NA.68
##            NA            NA            NA            NA            NA
NA
##             2          2427          4372         NA.69           628
NA.70
## 1.630786e-02 9.228174e-05 1.137958e-02            NA 9.729342e-03
NA
##         NA.71         NA.72         NA.73         NA.74
##            NA            NA            NA            NA
##
## $residual.scale
## [1] 1
```

```
yhat1 <- pred1$fit
suppressPackageStartupMessages(library(pROC))
yobs <- as.vector(as.numeric(dat.test0$cluster))
par(mfrow=c(3, 2), mar=rep(4,4))
roc.pca2 <- suppressMessages(plot.roc(yobs, yhat1,  ylim=c(0, 100),
    main="(a) ROC for Model via PC2", percent=TRUE,
    print.auc=TRUE, print.auc.cex=1.5, col="blue"))
```



From the above output, the logistic regression for the PC1(AUC=82.1%) outperformed the logistic regression for PC2(AUC=79.8%).