APPIAH PRINCE HW7 STAT 5474..

Prince Appiah

12/2/2021

Data Preparation

Read in the Data

```r
# install.packages("kernlab")
library(kernlab)
data(spam)
dim(spam)
```

```
## [1] 4601   58
```

```r
head(spam)
```

```
##   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
```

Comment The dimension of the spam data is 4601 observations by 58 variables.

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

```r
library(questionr)
freq.na(spam)
```

```
##                   missing %
## make                    0 0
## address                 0 0
## all                     0 0
## num3d                   0 0
## our                     0 0
## over                    0 0
## remove                  0 0
## internet                0 0
## order                   0 0
## mail                    0 0
## receive                 0 0
## will                    0 0
## people                  0 0
## report                  0 0
## addresses               0 0
## free                    0 0
## business                0 0
## email                   0 0
## you                     0 0
## credit                  0 0
## your                    0 0
## font                    0 0
## num000                  0 0
## money                   0 0
## hp                      0 0
## hpl                     0 0
## george                  0 0
## num650                  0 0
## lab                     0 0
```

```
## labs                     0 0
## telnet                   0 0
## num857                   0 0
## data                     0 0
## num415                   0 0
## num85                    0 0
## technology               0 0
## num1999                  0 0
## parts                    0 0
## pm                       0 0
## direct                   0 0
## cs                       0 0
## meeting                  0 0
## original                 0 0
## project                  0 0
## re                       0 0
## edu                      0 0
## table                    0 0
## conference               0 0
## charSemicolon            0 0
## charRoundbracket         0 0
## charSquarebracket        0 0
## charExclamation          0 0
## charDollar               0 0
## charHash                 0 0
## capitalAve               0 0
## capitalLong              0 0
## capitalTotal             0 0
## type                     0 0
```

Comment There are no missing values in the data

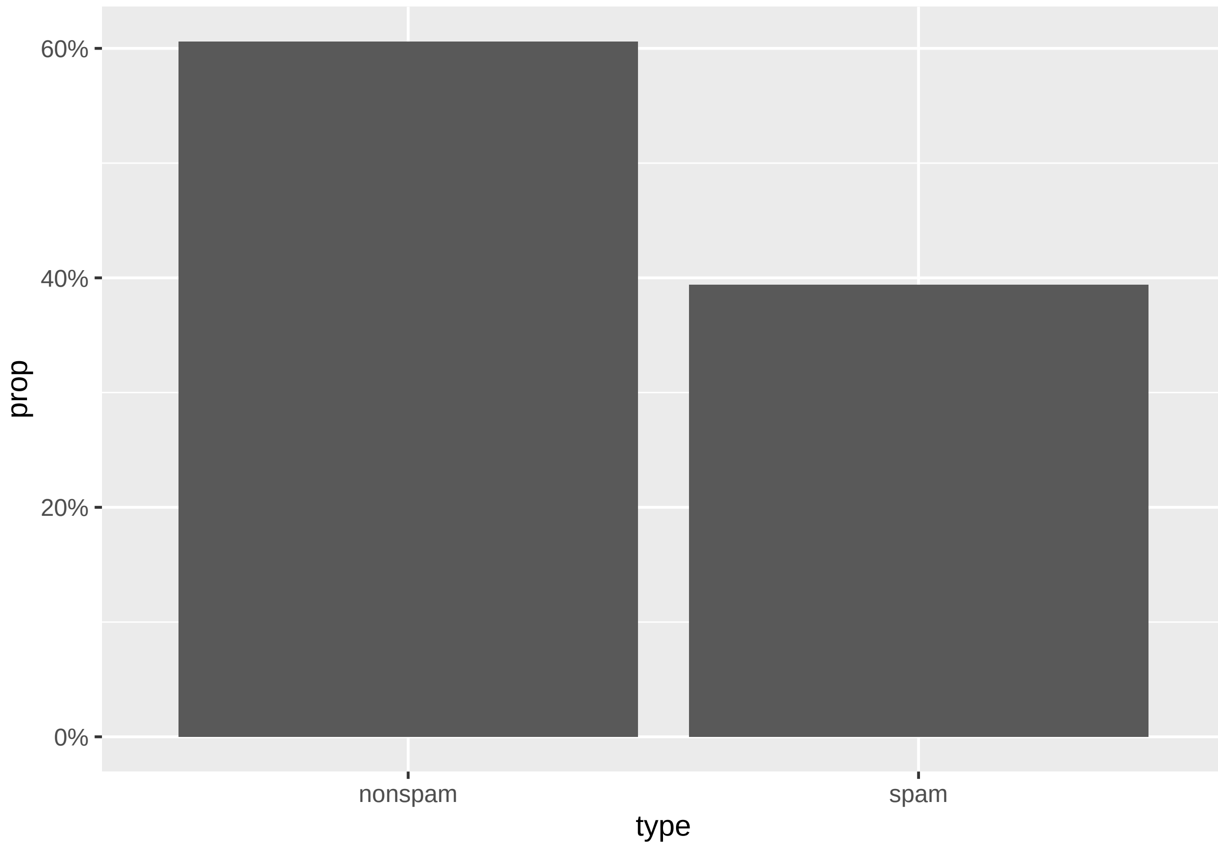
Numerical and graphical EDA techniques

Percentage of spam emails

```
set.seed(123)
table(spam$type)/4601*100
```

```
##
##  nonspam     spam
## 60.59552 39.40448
```

```
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())
```



Comment We see from the plot and the table that the percentage of nonspam is 60.596% and spam is 39.404%

What are the types (categorical or continuous) of the inputs?

```
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 ...
```

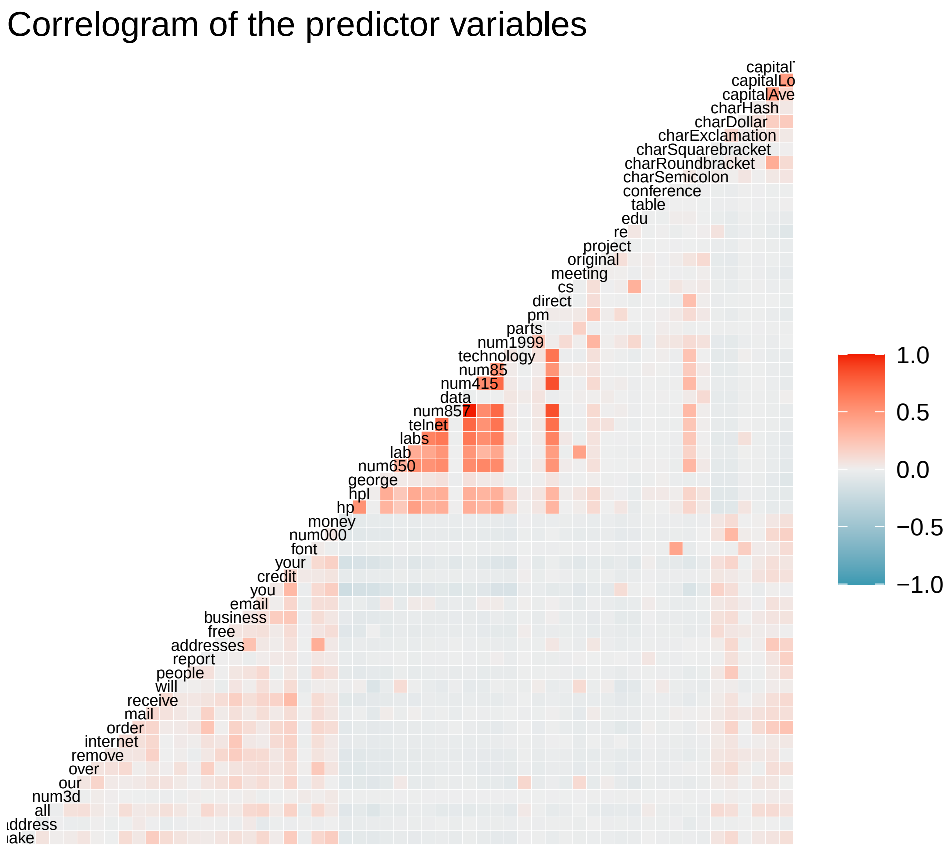
Comment We see from the output that there are 57 numeric(continuous) variables and 1 categorical variable

Correlation Plot

```r
set.seed(123)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
ggcorr(spam[,-58], size = 2) +
ggplot2::labs(title = "Correlogram of the predictor variables")
```



Comment From the plot we see that generally most of our predictors are mostly weakly correlated.Some of the variables are moderately negatively correlated and some are strongly correlated.We can assume that multicollinearity might be present in this dataset.

PARTITION OF THE DATA

(c) Randomly divide your datasets into the training sample and for the test sample with a ratio of 2:1. We will use the training sample to train a number of models and then use the test sample to compare them.

```
n <- NROW(spam); ratio <- 2/3
set.seed(123)
id.training <- sample(1:n, size=trunc(n*ratio), replace=FALSE)
train <- spam[id.training, ]
test <- spam[-id.training, ]
dim(train); dim(test)
```

```
## [1] 3067   58
```

```
## [1] 1534   58
```

```
yobs <- test$type
```

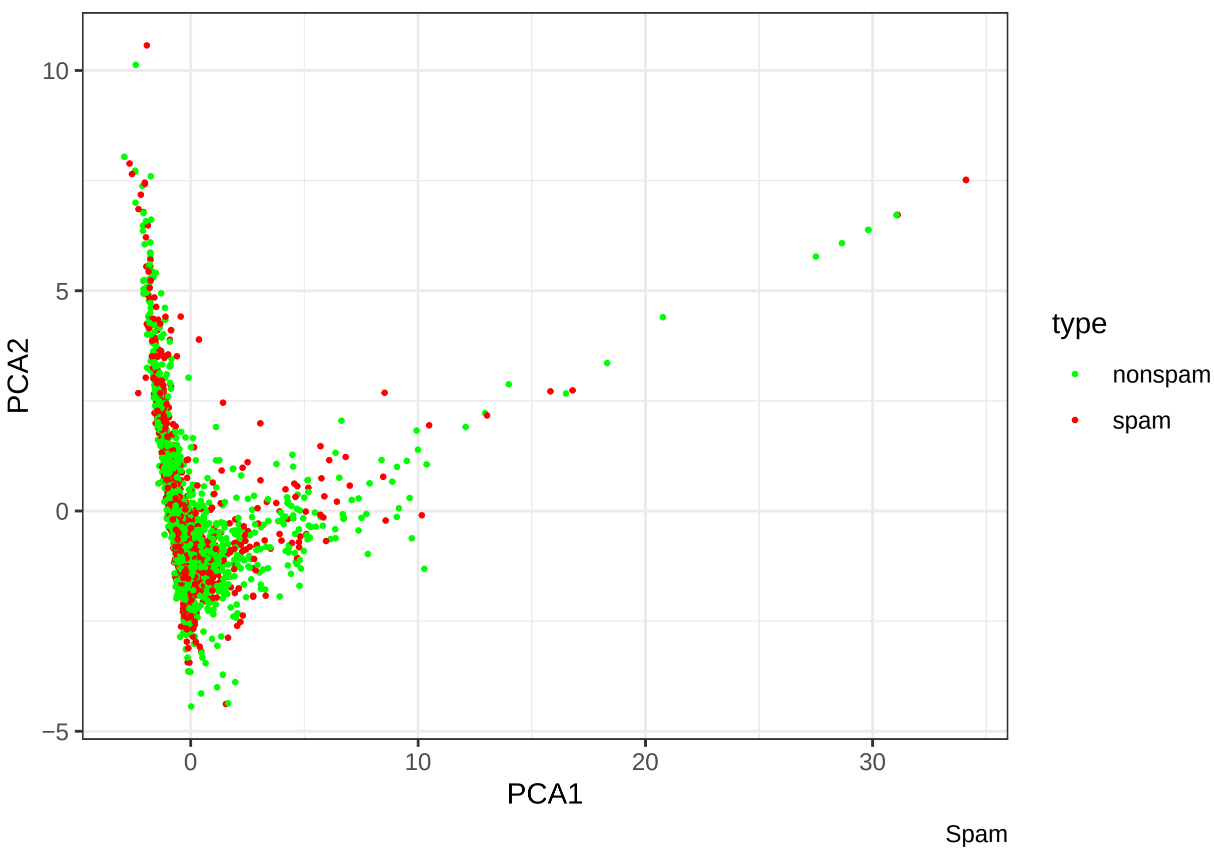
Comment The dimension of the training set is 3067 observations by 58 variables while the dimension of the test set is 1534 observations by 58 variables.

Unsupervised Learning

Ignoring the target variable type for the time being, apply to the training sample and plot the results by specifying the spam (red) or regular (green) email status with different colors. Interpret the results.

PCA

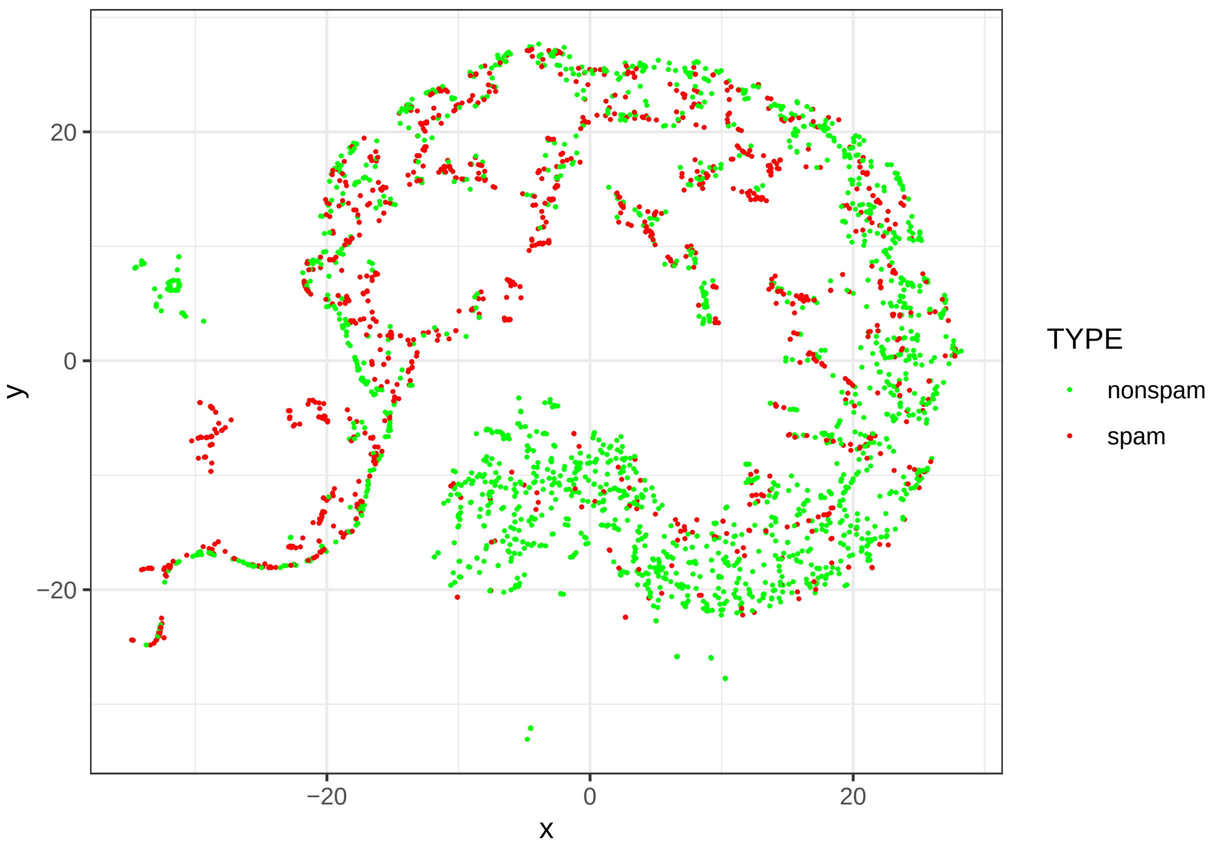
```
dat0 <- as.matrix(train[order(train$type),-58])
pca.res <- prcomp(dat0, center=TRUE, scale=TRUE, retx=TRUE); #pca.res
PC.directions <- pca.res$rotation
a1.a2 <- pca.res$rotation[,1:2]
ggplot(train, aes(x=pca.res$x[,1],y=pca.res$x[,2], color=type)) +
  geom_point(size=.9, shape=20) +
  theme_bw() +
  labs(caption = 'Spam', x = 'PCA1', y = 'PCA2')+
  scale_color_manual(breaks = c("nonspam", "spam"),
  values=c("green", "red"))
```



Comment PCA helps us to explain as much of the variation and information in the data as possible using only a small number of axes. We see from the plot there is no specific pattern. However, we see that there has been some clustering done.Spam and nonspam observations are all clustered in one group.

tSNE

```r
library(Rtsne)
#tsne <- Rtsne(train, dims = 2, perplexity=30, verbose=TRUE, max_iter = 500, check_duplicates = FALSE)
set.seed(123)
tsne<-Rtsne(train, dims = 2, perplexity =30, verbose = FALSE, check_duplicates = FALSE, max_iter = 500)
tsne_plot <- data.frame(x = tsne$Y[,1], y = tsne$Y[,2], col = as.factor(train[, 58]))
TYPE<-train$type
ggplot(tsne_plot) +
geom_point(aes(x=x, y=y, color=TYPE), size=0.5, shape=20) +
scale_color_manual(breaks = c("nonspam", "spam"),
values=c("green", "red"))+
theme_bw()
```



Comment We see from the plot that there are no specific significant patterns in the data just like we saw in the plot for PCA.

Supervised Learning

Try out the following predictive modeling tools. For each method, use the training set to identify the best model and apply the model to the test set. Then plot the ROC curve and compute the C statistic or C index (area under the ROC curve), all based on the test set performance. It would be best, but not required, to have the ROC curves plotted on one figure and compared. Which method gives the highest C index?

Linear discriminant analysis (LDA)

```
library(MASS)
library(verification)
library(cvAUC)
library(MASS)
fit.LDA <- lda(type~., data=train)
yhat.LDA <- predict(fit.LDA, newdata=test[,-58])$x
yobs<-test$type
n <- NROW(test)
AUC.LDA <- ci.cvAUC(predictions=yhat.LDA,
                    labels=yobs, folds=1:n, confidence=0.95)
```

Comment LDA creates new axes that maximize class separation, so that we can classify new data using these new axes.

LASSO

Train a 'best' logistic regression model. Depending on the situation, you might want to use a regularized logistic regression

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:spam':
##
##     det
```

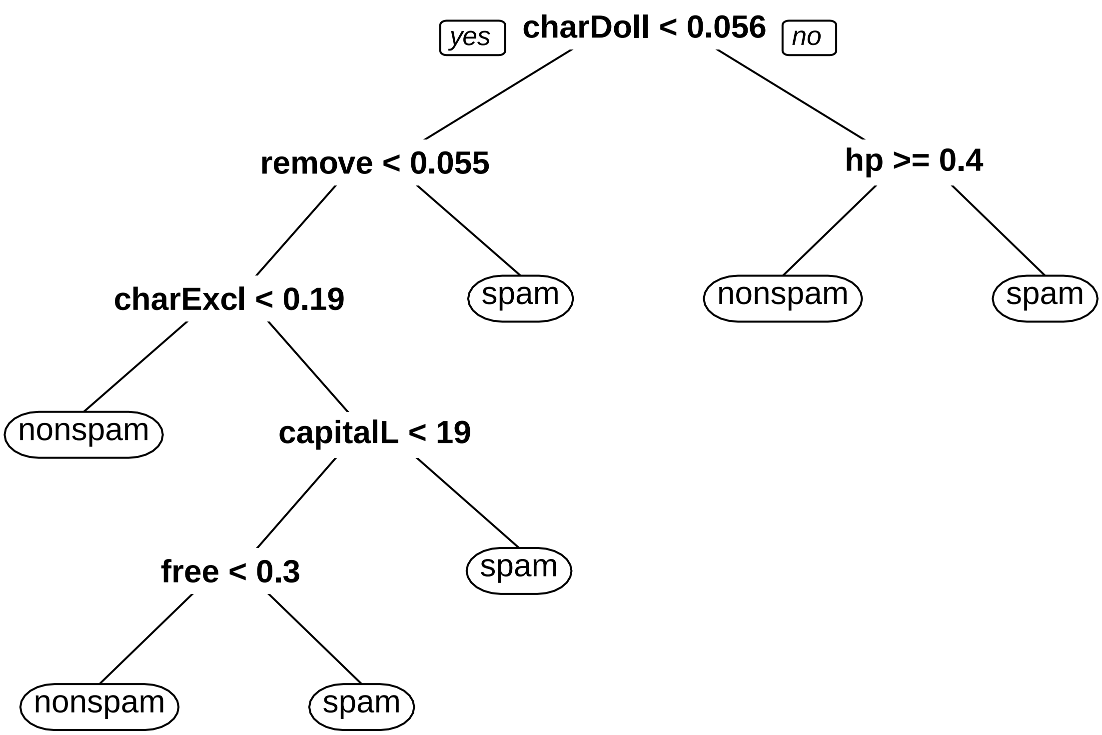
```
## Loaded glmnet 4.1-3
```

```
lambda <- seq(0, 10.0, 0.01)
X <- model.matrix(type~., data=train)
cv.LA <- cv.glmnet(x=X, y=train$type, family="binomial", alpha = 1, lambda = lambda, nfolds=10)
#plot(cv.LA)
lmbd0 <- cv.LA$lambda.min; #lmbd0 # MINIMUM CV ERROR
fit.logitLA <- cv.LA$glmnet.fit
X.test <- model.matrix(type~., data=test);
yhat.LA <- predict(fit.logitLA, s=lmbd0, newx=X.test, type="response")
```

Comment We used a regularization technique LASSO mainly due to the large number of variables in the dataset.

One single decision tree

```r
library(rpart)
library(RColorBrewer)
library(party)
library(rJava)
library(partykit)
library(rpart.plot)
control0 <- rpart.control(minsplit=10, minbucket=3, maxdepth=10,
        cp=0.01, maxcompete=4,
        maxsurrogate=3, usesurrogate=2, surrogatestyle=0,
        xval=10)

tre0 <- rpart(type~., data=as.data.frame(train),  method="class", control=control0,
         parms=list(split="information"), model = T)
prp(tre0)
```

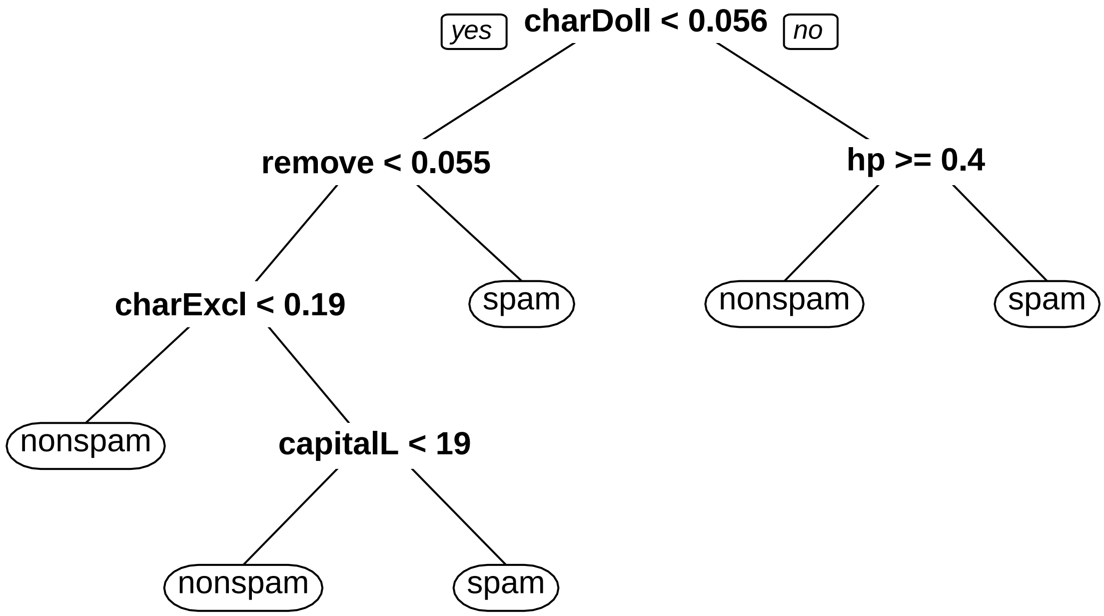



```r
cv.error <- (tre0$cptable)[,4]
a0 <- 1      # IF a0=0, THEN OSE
SE1 <- min(cv.error) + a0*((tre0$cptable)[,5])[which.min(cv.error)]       # 1SE
position <- min((1:length(cv.error))[cv.error <= SE1])
n.size  <- (tre0$cptable)[,2] + 1  # TREE SIZE IS ONE PLUS NUMBER OF SPLITS.
best.size <- n.size[position]; best.size
```

```
## 5
## 6
```

```r
best.cp <-  sqrt(tre0$cptable[position,1] *  tre0$cptable[(position-1),1])
#best.cp
best.tree <- prune(tre0, cp=best.cp)
```

```r
#best.tree
prp(best.tree)
```




```r
yhat.tree <- predict(best.tree, newdata =test, type="class",
        na.action = na.pass) %>% as.numeric()
```

Comment We see that the first tree has free as terminal node while in the second tree, free was not included.

Bagging

```r
set.seed(123)
library(ipred)
library(mlbench)
fit.bagging <- bagging(type~., data=train, nbagg=58, coob=TRUE)
print(fit.bagging)
```

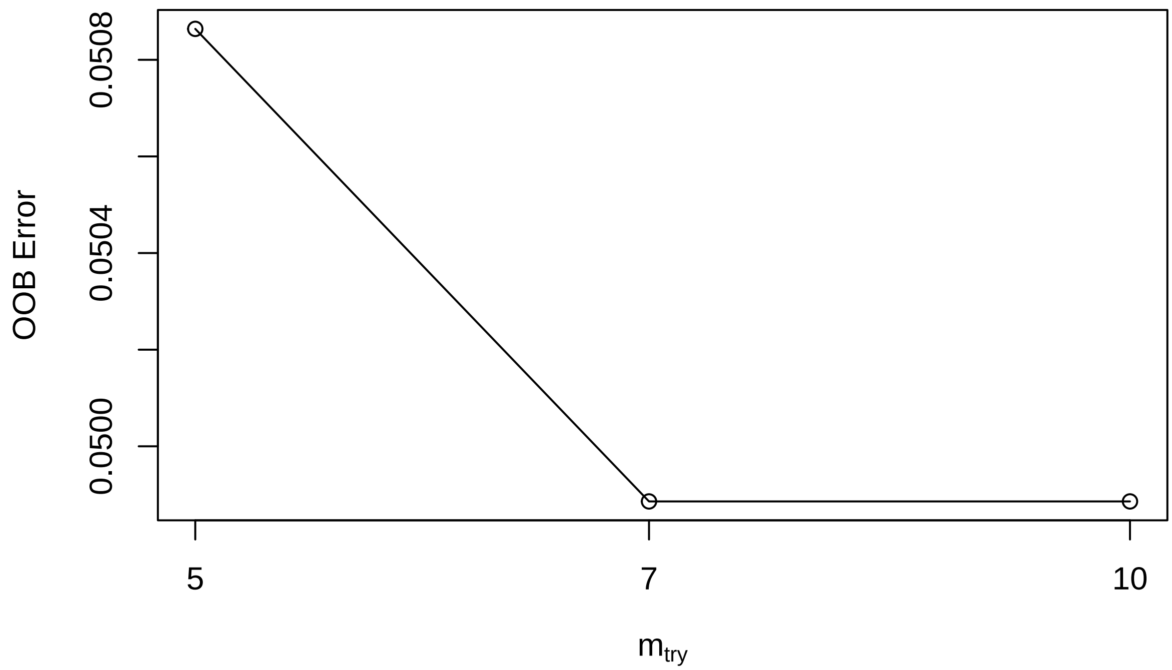
```
##
## Bagging classification trees with 58 bootstrap replications
##
## Call: bagging.data.frame(formula = type ~ ., data = train, nbagg = 58,
##     coob = TRUE)
##
## Out-of-bag estimate of misclassification error:  0.0639
```

```r
#summary(fit.bagging)
yhat.bag <- predict(fit.bagging, newdata=test, type="prob")
```

Random Forest

```r
library(randomForest)
set.seed(123)
x <- train[,1:57]
y <- train[,58]
best.m <- tuneRF(x, y, stepFactor=1.5, improve=1e-5, ntree=500)
```

```
## mtry = 7  OOB error = 4.99%
## Searching left ...
## mtry = 5     OOB error = 5.09%
## -0.01960784 1e-05
## Searching right ...
## mtry = 10    OOB error = 4.99%
## 0 1e-05
```

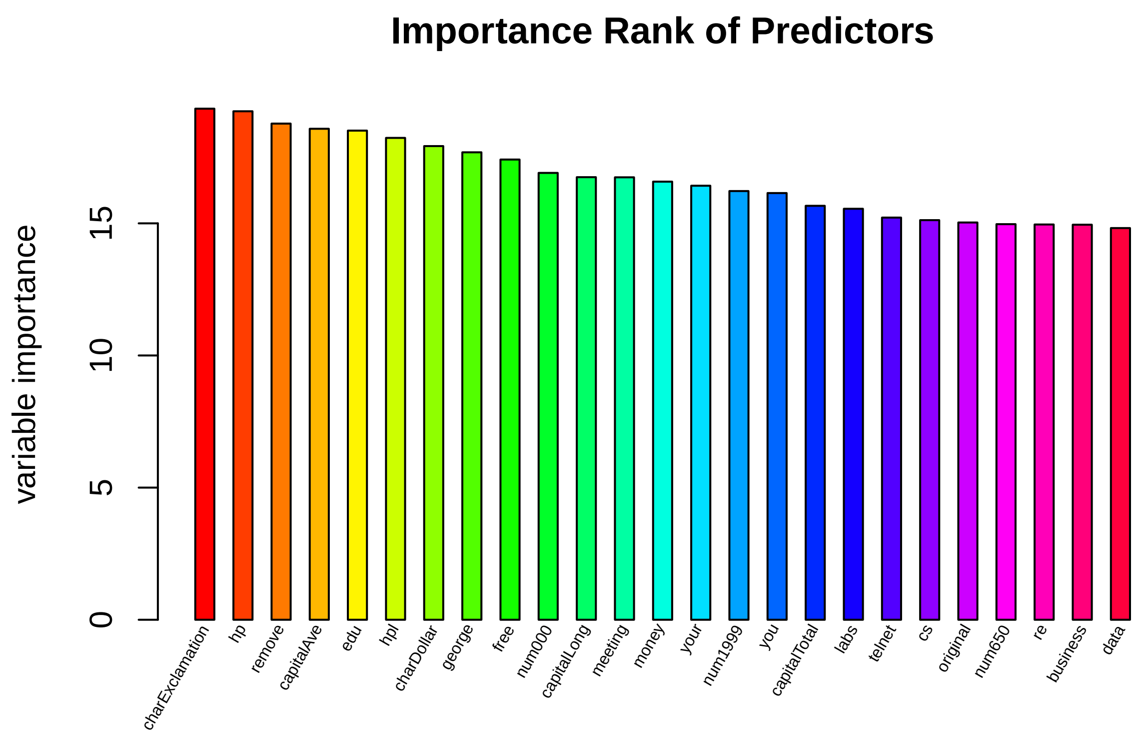



```r
print(best.m)
```

```
##        mtry   OOBError
## 5.OOB     5 0.05086404
## 7.OOB     7 0.04988588
## 10.OOB   10 0.04988588
```

```r
train.rf <- randomForest(type ~ .,
        data=train,  mtry=best.m,
        ntree=1000, keep.forest=TRUE,
        importance=TRUE,
        proximity=TRUE, oob.prox=FALSE)
#print(train.rf)
imp.tmp <- importance(train.rf, type = 1);
p0 <- NROW(imp.tmp)
```

```
rf.imp <- imp.tmp[order(imp.tmp[, 1], decreasing = TRUE),][1:25]
p0 <- length(rf.imp)
barplot(rf.imp, col = rainbow(p0),
main="Importance Rank of Predictors", names.arg ="",
ylab="variable importance", ylim=c(0, max(rf.imp)),
xlab="", space=1, axes=TRUE)
end_point <- 0.5 + p0 + p0 -1
#rotate 60 degrees, srt=60
text(seq(1.5, end_point,by=2), par("usr")[3]-0.25,
srt = 60, adj= 1, xpd = TRUE,
labels = names(rf.imp), cex=0.5)
```



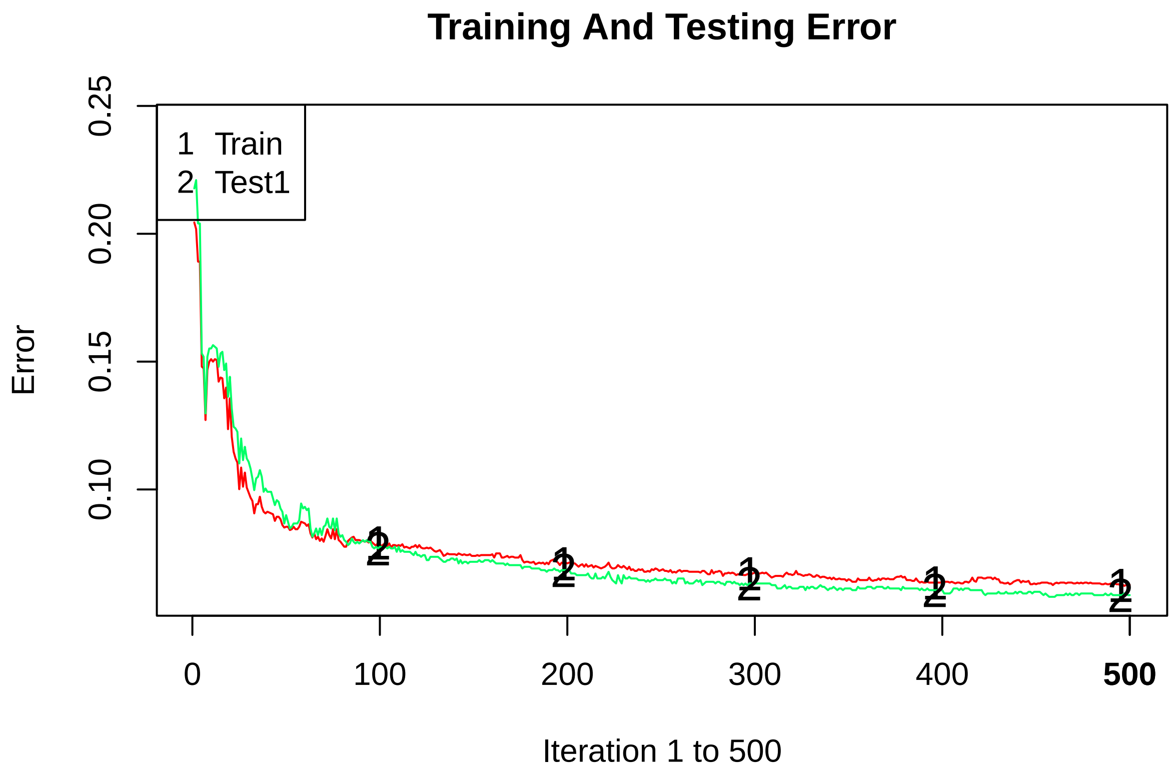

```
yhat.rf <- predict(train.rf, newdata=test[,-58], type="class") %>% as.numeric()
```

Comment We see from the plot that optimal mtry is 7 or the best.m is 7. We also see from the display of the top 25 predictors by importance that our top 2 are charExclamation and hp.

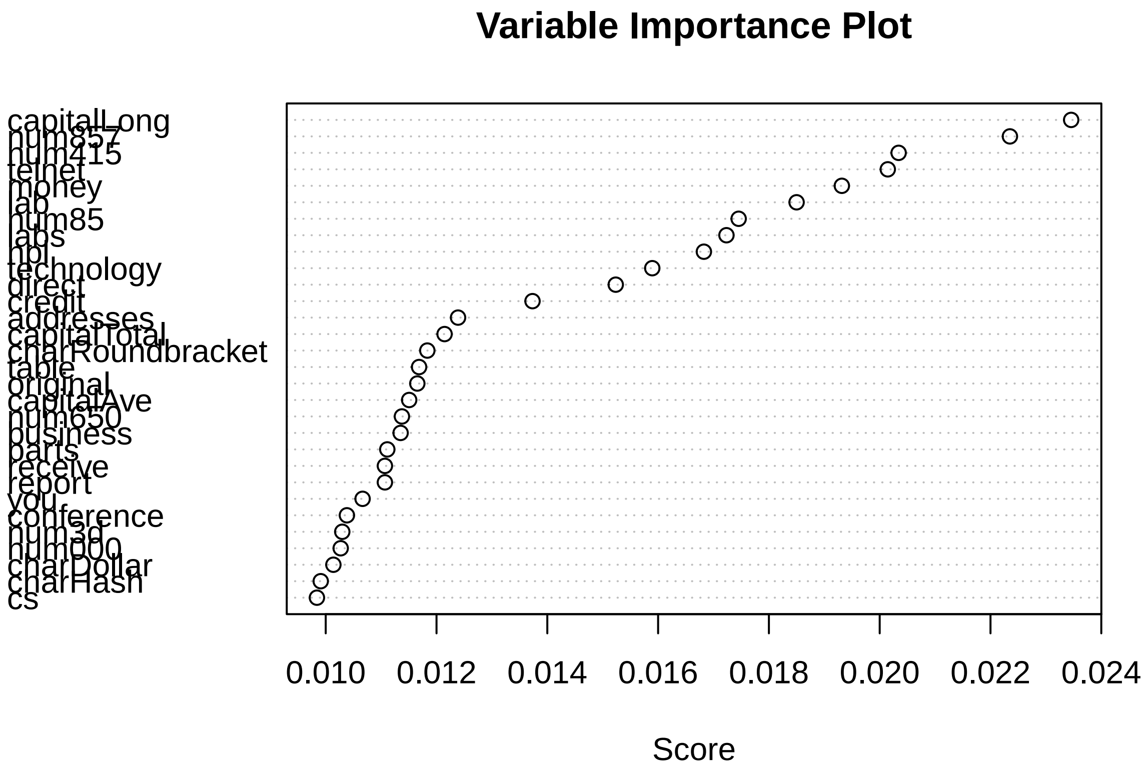
Boosting

```
library(ada)
stump <- rpart.control(cp=-1, maxdepth=1, minsplit=0)
# DISCRETE ADABOOST
fit.stump <- ada(type~., data=train, iter=500,
    loss="e", type="discrete",
    control=stump);
#fit.stump

# SOME EXPLORATION
```

```r
#summary(fit.stump)
#plot(fit.stump, kappa =FALSE, test=FALSE)
#plot(fit.stump, kappa =TRUE, test=FALSE)  # KAPPA AGREEMENT
# IF YOU WANT TO BASE THE MODEL ASSESSMENT ON THE TEST DATA
fit1.stump <- addtest(x=fit.stump, test.x=test[, -58], test.y=test[,58])
plot(fit1.stump, kappa =FALSE, test=TRUE)
```




```r
# VARIABLE IMPORTANCE
varplot(fit.stump, plot.it=TRUE,type="scores")
```




```
##      capitalLong           num857           num415           telnet
##      0.023456211      0.022350533      0.020341591      0.020145946
##            money              lab            num85             labs
##      0.019316589      0.018499596      0.017453934      0.017233749
##              hpl       technology           direct           credit
##      0.016826157      0.015894207      0.015235190      0.013732443
##        addresses     capitalTotal charRoundbracket            table
##      0.012388757      0.012145226      0.011833940      0.011685212
##         original       capitalAve           num650         business
##      0.011653156      0.011506074      0.011375130      0.011351873
##            parts          receive           report              you
##      0.011111616      0.011068138      0.011067813      0.010665365
##       conference            num3d           num000       charDollar
##      0.010382111      0.010298616      0.010270086      0.010137926
##         charHash               cs
##      0.009909814      0.009840939
```

```
vip <- varplot(fit.stump, plot.it=FALSE, type="scores")
#round(vip,4)

# PREDICTION
yhat.boost <- predict(fit.stump, newdata=test[,-58],
 type="class") %>% as.numeric()
```

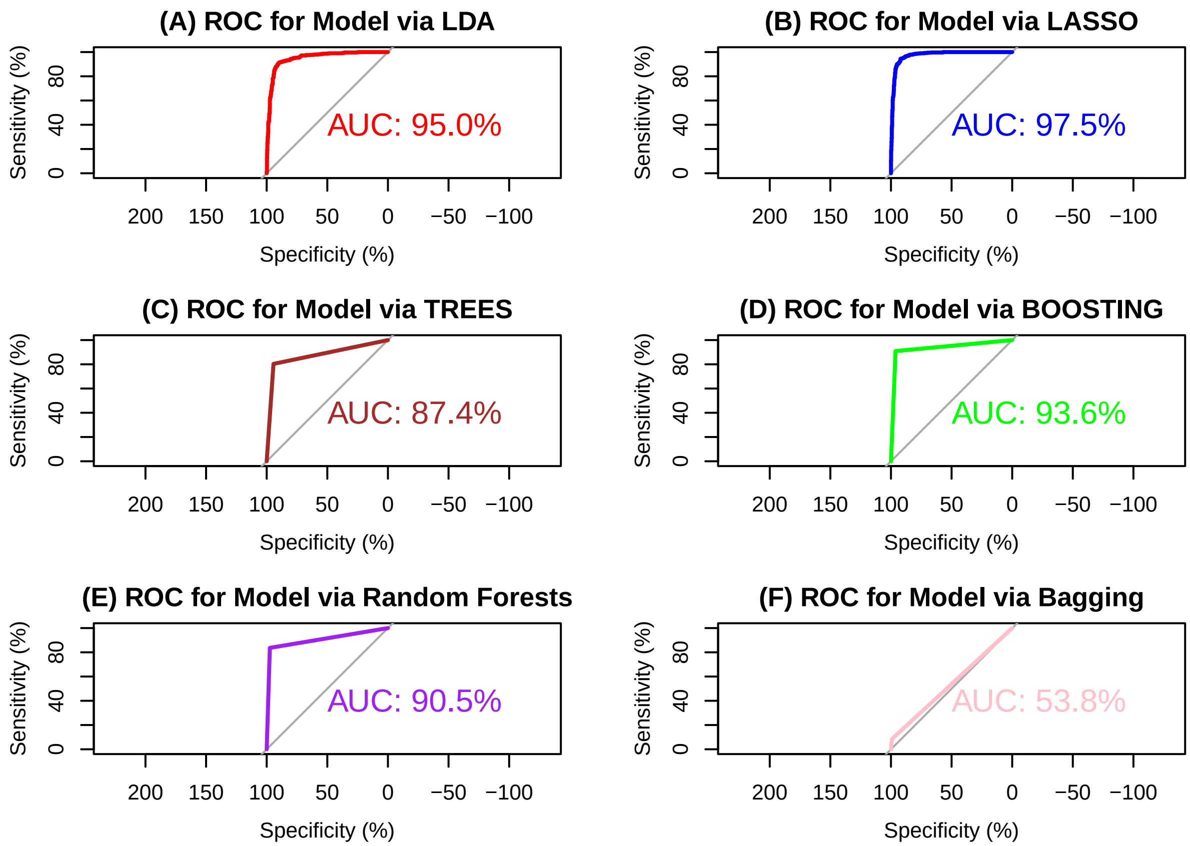
Comment We see from the Training and Testing Error plot that the curve begins to decrease at iteration 30. Also, the curves of spam and nonspam are intertwined on the Training and Testing Error plot. Moreover, on the Variable Importance Plot, we see that capitalLong and num857 are the top two important variables.

ROC Curves

```r
library(pROC)
par(mfrow = c(3, 2))
# ROC Curve for LDA

roc.LDA <-plot.roc(yobs, yhat.LDA,
                    ylim=c(0, 100), xlim = c(100, 0),
                    main="(A) ROC for Model via LDA",
                    percent=TRUE,print.auc=TRUE,
                    print.auc.cex=1.5, col="red")

#ROC Plot for LASSO

roc.LASSO <-plot.roc(yobs, yhat.LA,
                    ylim=c(0, 100), xlim = c(100, 0),
                    main="(B) ROC for Model via LASSO",
                    percent=TRUE,print.auc=TRUE,
                    print.auc.cex=1.5, col="blue")

#ROC Plot for Tree
roc.TREE <-plot.roc(yobs, yhat.tree,
                    ylim=c(0, 100), xlim = c(100, 0),
                    main="(C) ROC for Model via TREES",
                    percent=TRUE,print.auc=TRUE,
                    print.auc.cex=1.5, col="brown") #must be numeric or ordered go up

#ROC Plot for BOOSTING
roc.BOOST <-plot.roc(yobs, yhat.boost,
                    ylim=c(0, 100), xlim = c(100, 0),
                    main="(D) ROC for Model via BOOSTING",
                    percent=TRUE,print.auc=TRUE,
                    print.auc.cex=1.5, col="green") #same fix

#ROC Plot for RF
roc.RF <-plot.roc(yobs, yhat.rf,
                    ylim=c(0, 100), xlim = c(100, 0),
                    main="(E) ROC for Model via Random Forests",
                    percent=TRUE,print.auc=TRUE,
                    print.auc.cex=1.5, col="purple") #same fix

# ROC Plot for Bagging
yhat.bag <- !duplicated(yhat.bag)
dim(yhat.bag)
```

```
## [1] 1534
```

```r
roc.bag <-plot.roc(yobs, as.numeric(yhat.bag),
                   ylim=c(0, 100), xlim = c(100, 0),
                   main="(F) ROC for Model via Bagging",
                   percent=TRUE,print.auc=TRUE,
                   print.auc.cex=1.5, col="pink")
```



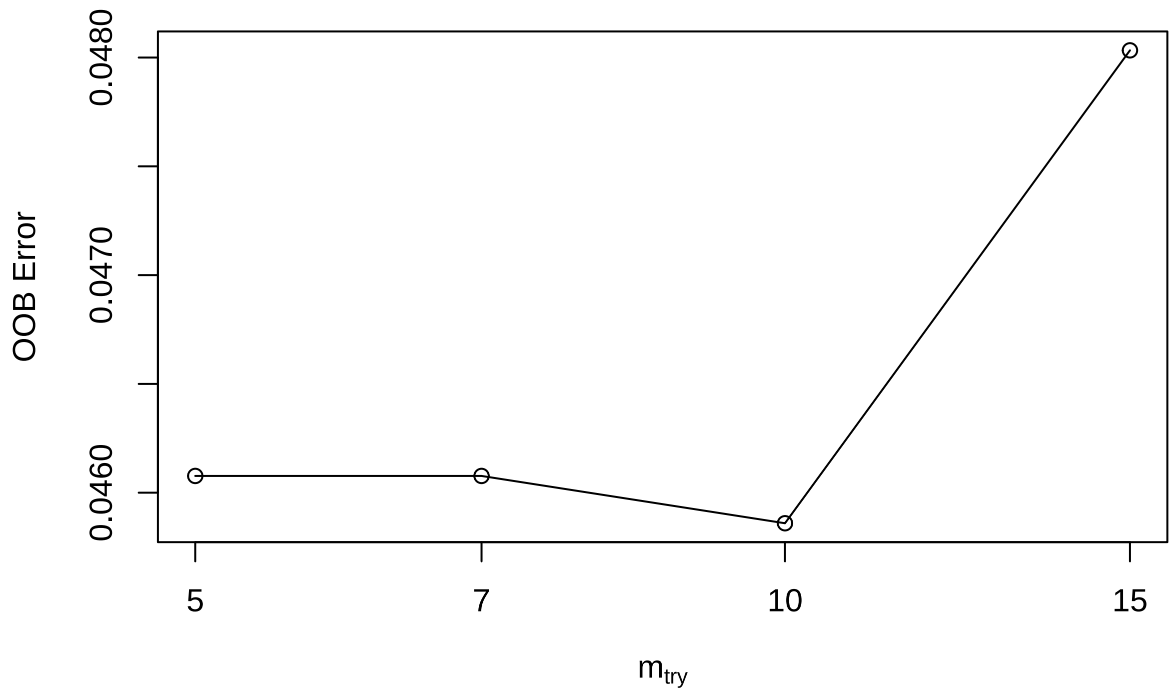
Comment The model with the best AUC/ C statistic score is the Regularization Technique LASSO. Thus, this is our best model.

Additional Features from RF

Train an RF model with B = 2,000 trees using the entire data set. Make sure that you set these two options: importance = TRUE and proximity = TRUE.

```
set.seed(123)
x0 <- spam[,1:57]
y0 <- spam[,58]
best.m0 <- tuneRF(x0, y0, stepFactor=1.5, improve=1e-5, ntree=500)
```

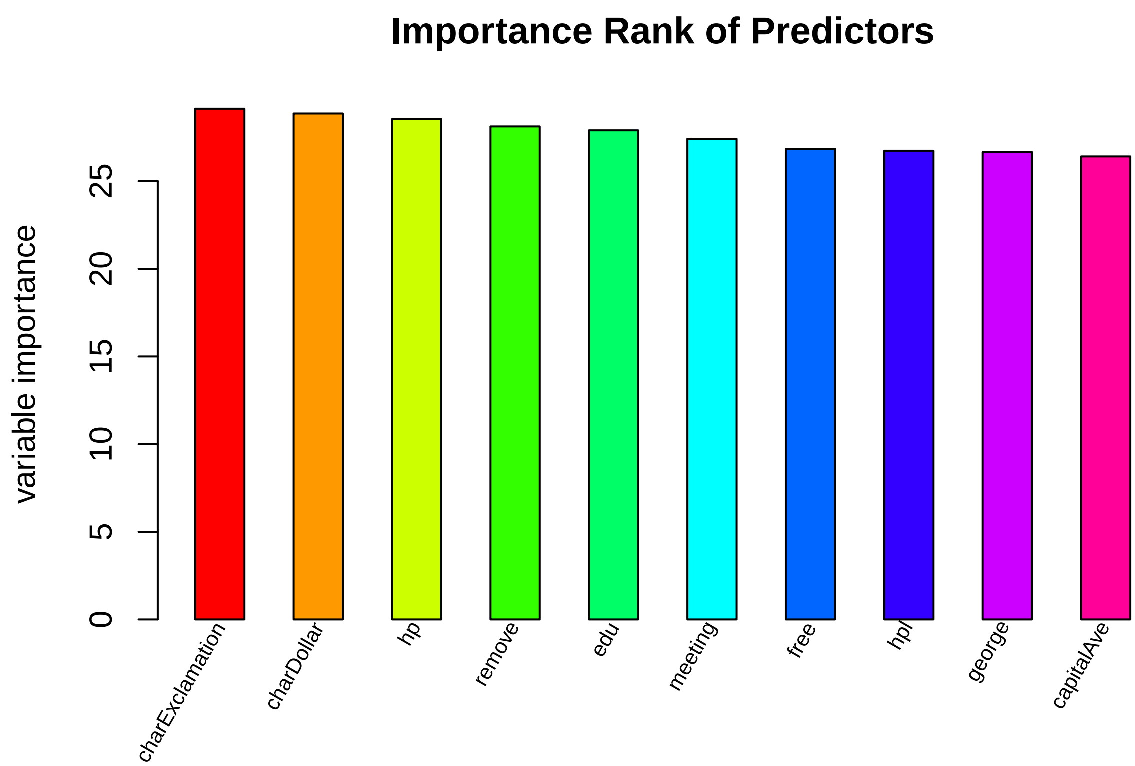
```
## mtry = 7  OOB error = 4.61%
## Searching left ...
## mtry = 5     OOB error = 4.61%
## 0 1e-05
## Searching right ...
## mtry = 10    OOB error = 4.59%
## 0.004716981 1e-05
## mtry = 15    OOB error = 4.8%
## -0.04739336 1e-05
```



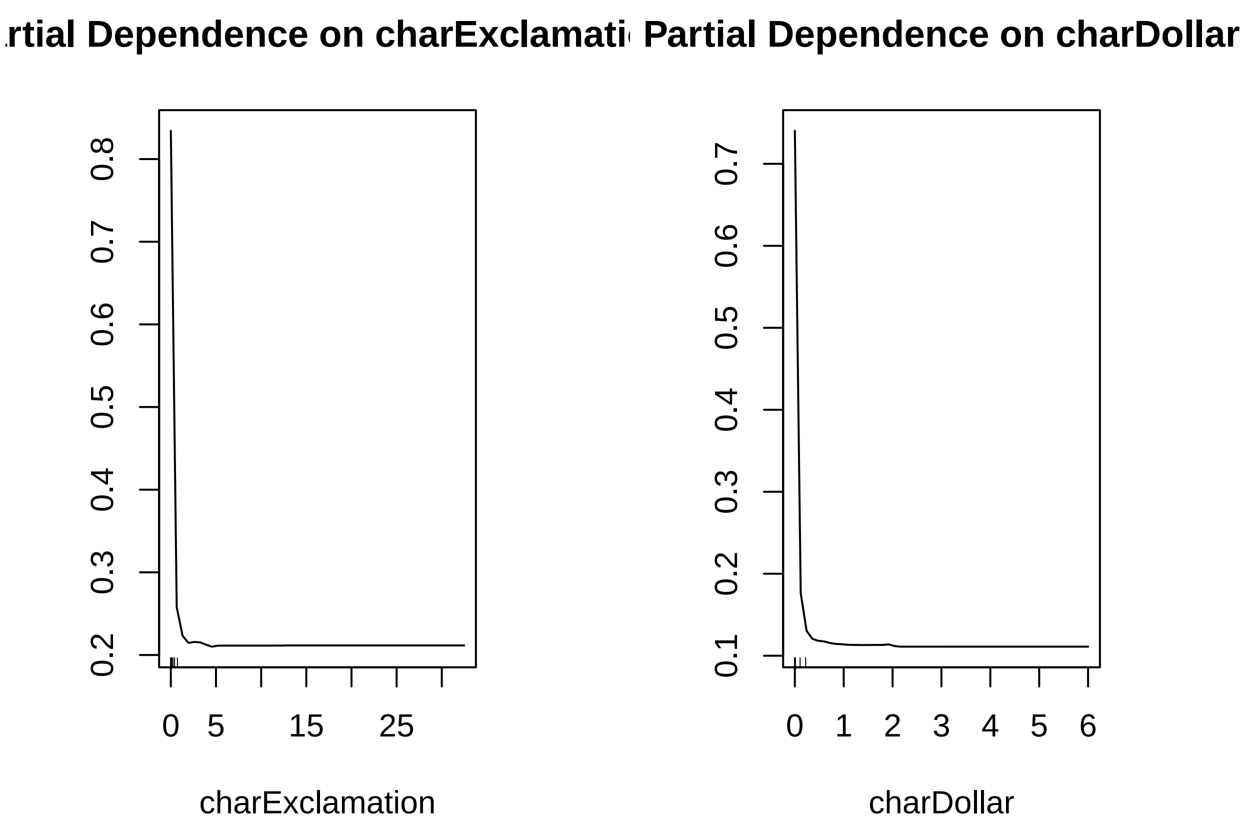

```
#print(best.m0) #best=10
fit.rf2 <- randomForest(type~.,data=spam, mtry=best.m0, ntree=2000, keep.forest=TRUE,
        importance=TRUE, proximity=TRUE, oob.prox=FALSE)
print(fit.rf2)
```

```
##
## Call:
##  randomForest(formula = type ~ ., data = spam, mtry = best.m0,      ntree = 2000, keep.forest = TRUE
##                Type of random forest: classification
##                      Number of trees: 2000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 7.72%
## Confusion matrix:
##         nonspam spam class.error
## nonspam    2726   62  0.02223816
## spam        293 1520  0.16161059
```

```
imp.tmp <- importance(fit.rf2, type = 1);
p0 <- NROW(imp.tmp)
rf.imp <- imp.tmp[order(imp.tmp[, 1], decreasing = TRUE),][1:10]
p0 <- length(rf.imp)
barplot(rf.imp, col = rainbow(p0),
        main="Importance Rank of Predictors", names.arg ="",
        ylab="variable importance", ylim=c(0, max(rf.imp)),
        xlab="", space=1, axes=TRUE)
end_point <- 0.5 + p0 + p0 -1
#rotate 60 degrees, srt=60
text(seq(1.5, end_point,by=2), par("usr")[3]-0.25,
     srt = 60, adj= 1, xpd = TRUE,
     labels = names(rf.imp), cex=0.7)
```




```r
par(mfrow=c(1, 2), mar=rep(4,4));
partialPlot(fit.rf2, pred.data=spam[, -58], x.var=charExclamation, rug=TRUE)
partialPlot(fit.rf2, pred.data=spam[, -58], x.var=charDollar, rug=TRUE)
```



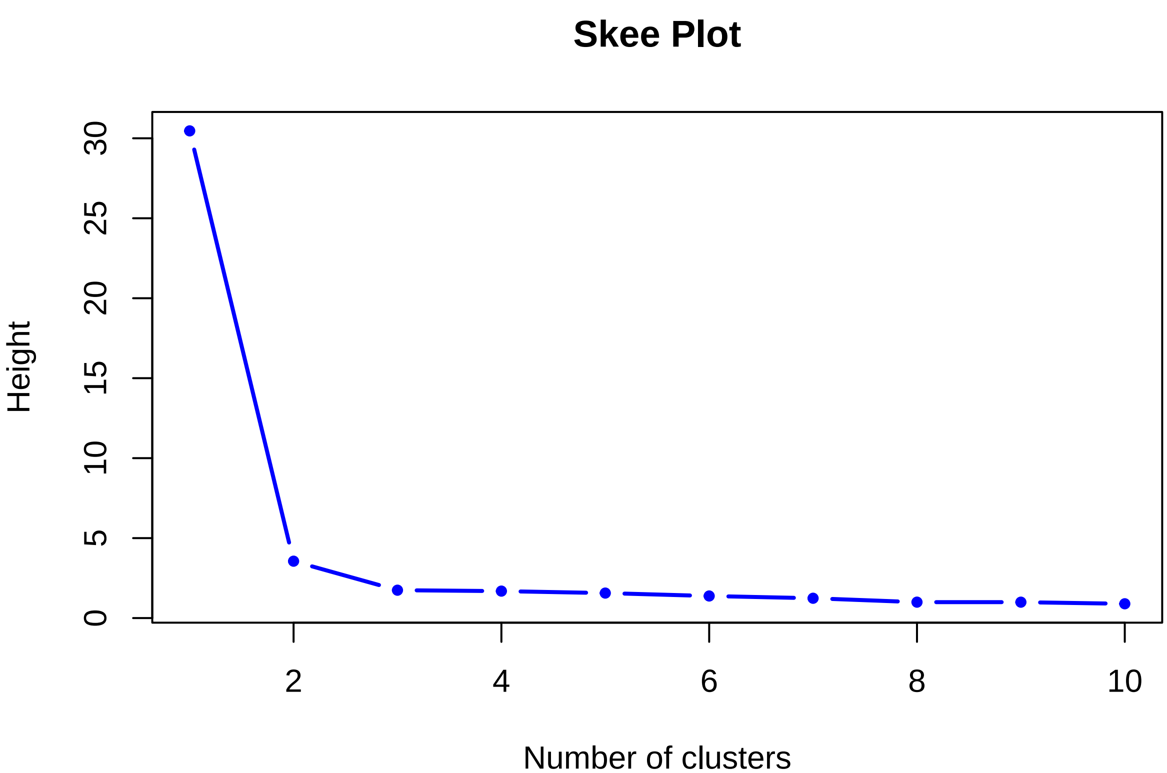
Comment We see that the optimal mtry or best.m is 10. The top 2 variables are charExclamation and charDollar out of the 10 variables we considered. Moreover,we printed out the Partial Dependence Plots for our top 2 variables. We see from these plots that the curve for charExclamation start decreasing at 2 while the curve for charDollar starts decreasing at 0.4.

Combing Unsupervised/Supervised Learning

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

PAM

```
library(cluster)
response<-train[,58]
train.scale<-cbind(response, scale(train[,-58]))
gower_dist <- daisy(train.scale[,-58], metric = "gower")
pfit <- hclust(gower_dist, method="ward.D")
K.max <- 10
height <- tail(pfit$height, n=K.max)
n.cluster <- tail((nrow(train.scale)-1):1, n=K.max)
plot(n.cluster, height,  type="b", pch=19, cex=.5,main="Skee Plot", xlab="Number of clusters",
     ylab="Height", col="blue", lwd=2) # 2 Clusters
```




```
pam_fit <- pam(gower_dist, diss = TRUE, k = 2)
table(pam_fit$cluster)
```

```
##
##    1    2
## 1860 1207
```

```
train[pam_fit$medoids, ]
```

```
##      make address  all num3d our over remove internet order mail receive will
## 3335    0    0.00 0.00     0   0    0   0.00        0     0    0       0    0
## 1415    0    0.19 0.39     0   0    0   0.19        0     0    0       0    0
##      people report addresses free business email  you credit your font num000
## 3335      0      0         0    0        0  0.00 0.00      0 0.00    0      0
## 1415      0      0         0    0        0  0.19 2.36      0 1.18    0      0
##      money hp hpl george num650 lab labs telnet num857 data num415 num85
## 3335  0.00  0   0      0      0   0    0      0      0    0      0     0
## 1415  0.19  0   0      0      0   0    0      0      0    0      0     0
##      technology num1999 parts pm direct cs meeting original project re edu
## 3335          0       0     0  0      0  0       0        0       0  0   0
## 1415          0       0     0  0      0  0       0        0       0  0   0
##      table conference charSemicolon charRoundbracket charSquarebracket
## 3335     0          0             0             0.00                 0
## 1415     0          0             0             0.03                 0
##      charExclamation charDollar charHash capitalAve capitalLong capitalTotal
## 3335           0.000          0        0      1.625           9           26
## 1415           0.152          0        0      1.357          19          148
##         type
## 3335 nonspam
## 1415    spam
```

```
ct.pam=table(train$type, pam_fit$clustering);ct.pam
```

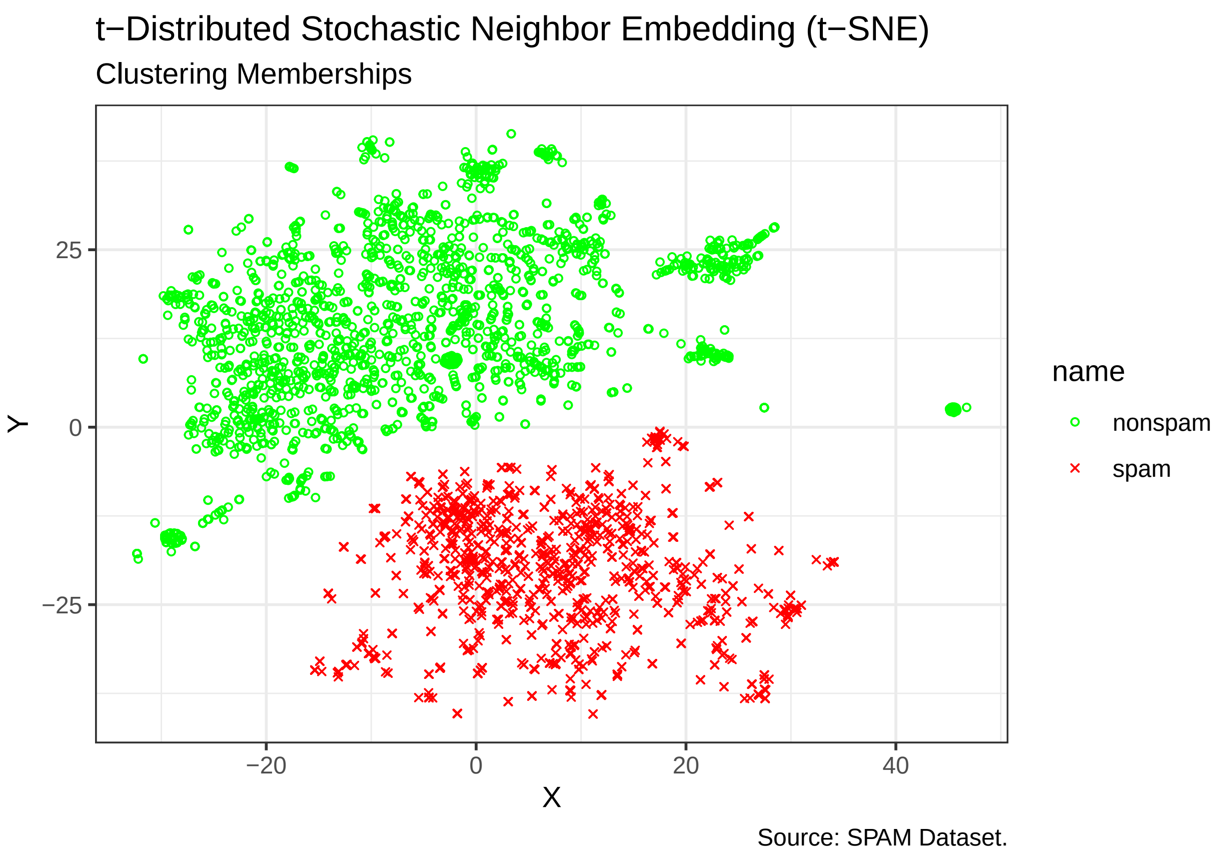
```
##
##              1    2
##   nonspam 1860    0
##   spam       0 1207
```

Comment We see from the output that the optimal number of clusters is 2 as shown by skee plot, whereas 1860 nonspam observations are in cluster 1 and 1207 spam observations are in cluster 2.

tSNE Replot

```
library(gower)
library(dplyr)
library(Rtsne)
library(ggplot2)
set.seed(6969)
tsne_obj <- Rtsne(gower_dist, is_distance = TRUE)
tsne_data <- tsne_obj$Y %>%
  data.frame() %>%
  setNames(c("X", "Y")) %>%
  mutate(cluster = factor(pam_fit$clustering),
         name = train$type)
```

```r
ggplot(tsne_data, aes(x = X, y = Y, shape = name)) +
  geom_point(size = 1, aes(color=name)) +
  scale_shape_manual(values = c(1, 4)) +
  scale_color_manual(breaks = c("nonspam", "spam"),
  values=c("green", "red"))+
  labs(title = "t-Distributed Stochastic Neighbor Embedding (t-SNE)",
         subtitle = "Clustering Memberships",
         caption = "Source: SPAM Dataset.") +
  theme_bw()
```



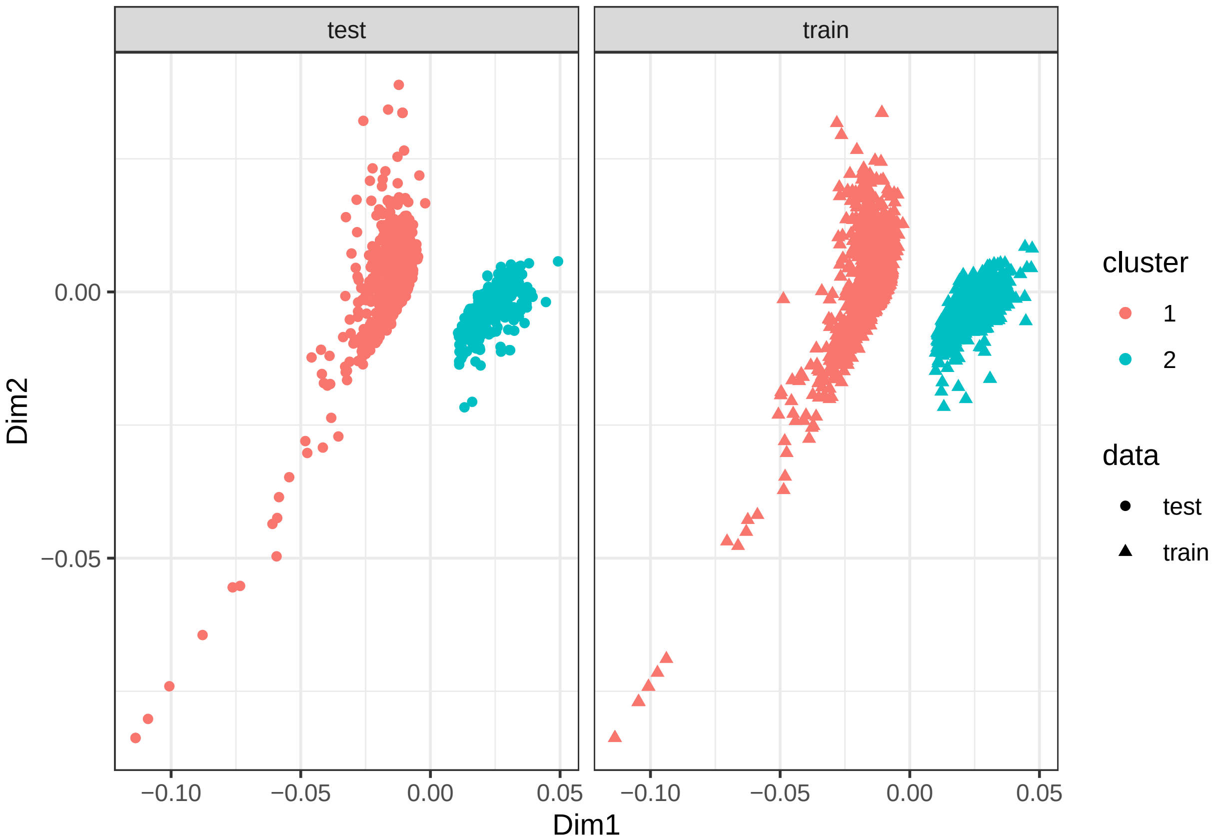
Comment We see clearly from this tsne plot that it far better than the previous one. We see a clear distinction between groups whereas the green cluster is the nonspam and red cluster is the spam.

Predict Cluster Membership

Accordingly predict the cluster membership for each observation in the test sample. This part is not previously illustrated in my R code. Do some search and research on your own.

```r
library(cluster)
predict_pam<-function(model,traindata,newdata){
                        nclus = length(model$medoids)
                        DM = daisy(rbind(traindata[model$medoids,],newdata),metric="gower")
                        max.col(-as.matrix(DM)[-c(1:nclus),1:nclus])}

df<-data.frame(cmdscale(daisy(rbind(train,test),metric="gower")),
```

```r
rep(c("train","test"),c(nrow(train),nrow(test))))
colnames(df) = c("Dim1","Dim2","data")

df$cluster<-c(pam_fit$clustering,predict_pam(pam_fit,train,test))
df$cluster<-factor(df$cluster)

ggplot(df,aes(x=Dim1,y=Dim2,shape=data, col=cluster)) +  #col=cluster
        geom_point() +
        facet_wrap(~data) +
        theme_bw()
```



Comment We see from the plot that it is apparent that the training set holds up extremely well when compared to the test set.

Fit Clustered Logistic Regression Model

Fit a **best** clustered logistic regression model by including interactions term between cluster and all other predictors and applying regularization.

```r
library(glmnet)
trainclust<-df[-c(3068:4601),]$cluster
#dim(trainclust)
X <- model.matrix(type~.*trainclust, data=as.data.frame(train))[,-1] #remove intercept
```

```
## Warning in terms.formula(object, data = data): 'varlist' has changed (from
## nvar=58) to new 59 after EncodeVars() -- should no longer happen!
```

```
## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=58) to new 59 after EncodeVars() -- should no longer happen!
```

```
y <- train$type
par(mfrow = c(2, 1), mar = c(4.5, 4.5, 4, 1))
CV <- cv.glmnet(x=X, y=y, family="binomial", alpha = 1,
    lambda.min = 1e-4, nlambda = 200, standardize = T, thresh = 1e-07,
    maxit=1000)
CV
```

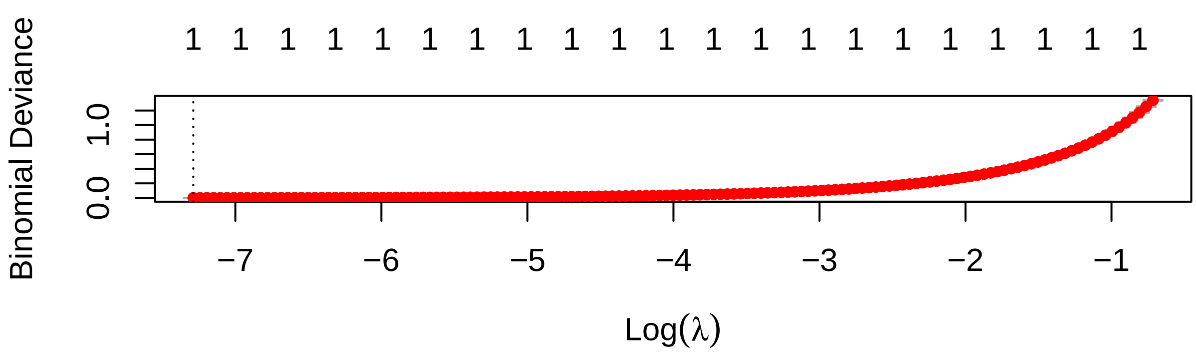
```
##
## Call:  cv.glmnet(x = X, y = y, family = "binomial", alpha = 1, lambda.min = 1e-04,      nlambda = 20
##
## Measure: Binomial Deviance
##
##         Lambda Index  Measure        SE Nonzero
## min 0.0006833   143 0.001337 3.508e-06       1
## 1se 0.0006833   143 0.001337 3.508e-06       1
```

```
plot(CV)

# TWO WAYS OF SELECTING THE BEST TUNING PARAMETER
b.lambda <- CV$lambda.1se; b.lambda  # THE BEST lamdba WITH 1SE RULE
```

```
## [1] 0.0006833214
```

```
# b.lambda <- CV$lambda.min; b.lambda
fit.lasso <- glmnet(x=X, y=y, family="binomial", alpha = 1,
    lambda=b.lambda, standardize = T, intercept=TRUE, thresh = 1e-07,
    maxit=1000)
#coef(fit.lasso) that is cool
str(df)
```

```
## 'data.frame':    4601 obs. of  4 variables:
##  $ Dim1   : num  -0.0134 -0.0101 -0.0087 0.03 -0.0113 ...
##  $ Dim2   : num  -0.0024 0.0137 0.00438 -0.0052 0.00497 ...
##  $ data   : chr  "train" "train" "train" "train" ...
##  $ cluster: Factor w/ 2 levels "1","2": 1 1 1 2 1 1 1 2 1 1 ...
```



Comment Here we used trainclust as our interaction between all the predictors, and we also using the regularization technique LASSO. We also see from the plot that the curve starts increasing from where log(lambda) is -3.5

Apply on Test Set

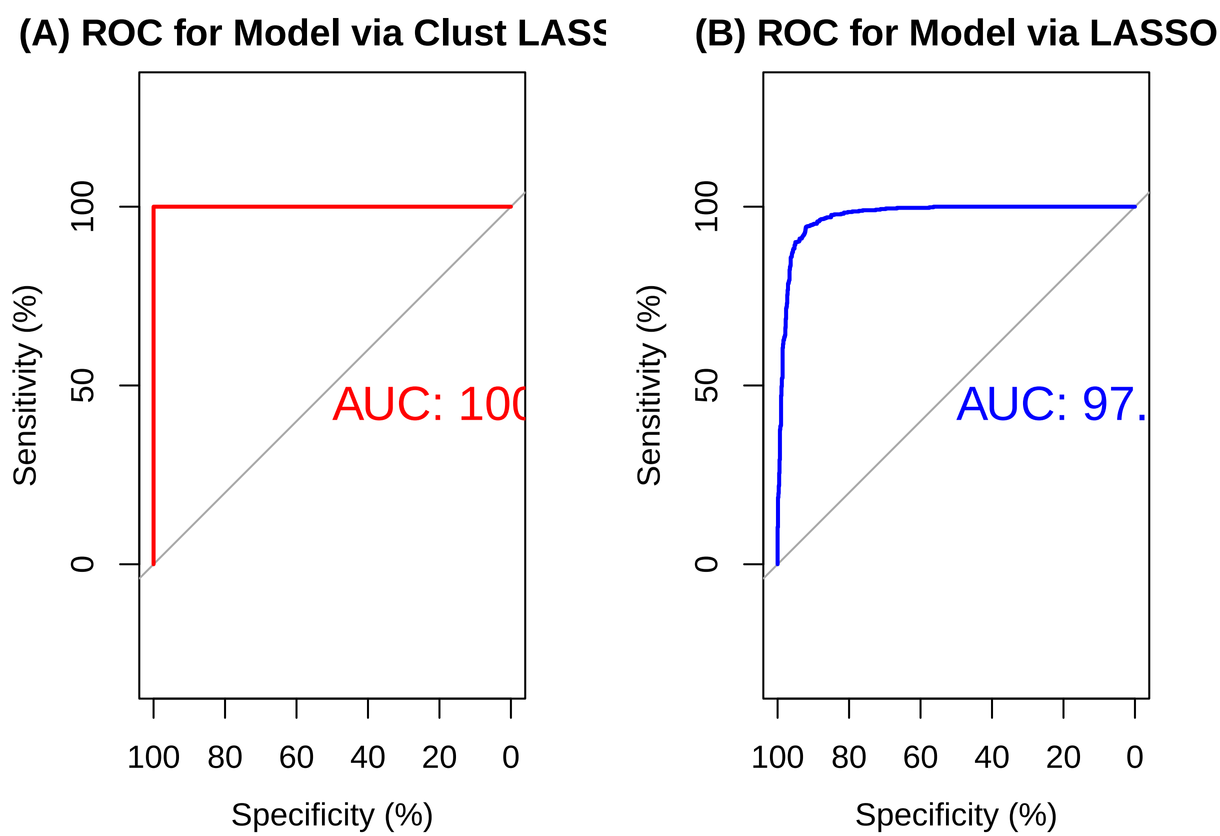
Apply the 'best' clustered logistic regression model to the test sample.

```
testclust<-df[-c(1:3067),]$cluster
X0.test <- model.matrix(type~.*testclust, data=test)[,-1]
yHat.clust<-predict(fit.lasso, newx = X0.test,
                    s = lmbd0, type = "response")
```

Comment We used our best clustered logistic regression model fit.LASSO to our test sample.

Compare

```
library(pROC)
par(mfrow = c(1, 2))
roc.Clutered.LASSO <-plot.roc(yobs, yHat.clust,
                    ylim=c(0, 100), xlim = c(100, 0),
                    main="(A) ROC for Model via Clust LASSO",
                    percent=TRUE,print.auc=TRUE,
                    print.auc.cex=1.5, col="red")
roc.LASSO <-plot.roc(yobs, yhat.LA,
                    ylim=c(0, 100), xlim = c(100, 0),
                    main="(B) ROC for Model via LASSO",
                    percent=TRUE,print.auc=TRUE,
                    print.auc.cex=1.5, col="blue")
```



Comment Our new logistic regression model using clusters as our interaction term outperformed our other model with an AUC score of 100.