Task 2

# Introduction

For task 2, we are going to deal with a dataset containing information about webmails. We have variables describing the frequency of some specific words like *“remove”* in each observation, variables describing the frequency of some specific chars like *“$”* in one observation, and three variables, *capital\_run\_length\_longest*, *capital\_run\_length\_average* and *capital\_run\_length\_total*, describing the length of the longest uninterrupted sequence of capital letters, the average length of uninterrupted sequences of capital letters, and the total number of capital letters in each observation respectively. We also have a variable called *spam*, which indicates whether this webmail is a spam with and , where for spam, and for not spam. Here all our variables are numeric type. Our task is to use these attribute variables to classify whether a webmail is spam.

# Methodology

In order to validate the accuracy of our methods, we firstly divide our dataset into a train set, which contains 2500 observations, and a test set, which contains 2101 observations. We use the train set to train our models, and then apply it to the test set to validate its accuracy.

# Results

## Classification Trees

In this part, we are going to discuss the results obtained by complex tree model and pruned tree model. We begin with construct a complex tree model by dividing our observation into small non-overlapping regions according to some numerical criteria. Here we split our dataset until each leaf of our classification tree contains only less then 2 observations. The method used here is recursive binary splitting.

#grow complex tree using deviance as criterion  
tree.mod = tree(factor(spam) ~ . ,data.train,   
 control = tree.control(nobs = 2500, minsize = 2, mincut = 1),  
 split = "deviance")  
  
#tree.mod = rpart(spam ~ . ,data.train,   
# control = tree.control(nobs = 2500, minsize = 2, mincut = 1))  
  
summary(tree.mod)

Classification tree:  
tree(formula = factor(spam) ~ ., data = data.train, control = tree.control(nobs = 2500,   
 minsize = 2, mincut = 1), split = "deviance")  
Variables actually used in tree construction:  
[1] "char\_freq\_dollar" "word\_freq\_remove"   
[3] "char\_freq\_exclmark" "word\_freq\_hp"   
[5] "word\_freq\_our" "capital\_run\_length\_longest"  
[7] "word\_freq\_free" "word\_freq\_edu"   
Number of terminal nodes: 12   
Residual mean deviance: 0.4883 = 1215 / 2488   
Misclassification error rate: 0.084 = 210 / 2500