Email Spam Detection and Classification

Anandu R

8/1/2020

Email Spam Detection and Classification

Loading necessary packages and data sources We load the dataset using the kernlab package which has the required dataset, as well as several other datasets that can be used for analysis.

```
if(!require("kernlab")){
  install.packages("kernlab")
}
```

```
## Loading required package: kernlab
```

```
library(kernlab)
```

Loading the data

```
data(spam)
```

Preliminary Analysis on data

```
str(spam[,1:5])
```

```
## 'data.frame':    4601 obs. of  5 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 ...
```

Subsampling the dataset Looking into the "type" variable we can know whether a mail is spam or not

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

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

we observe that there are 2788 nonspam mails and 1813 mails labelled as spam within the dataset.

```r
library(caTools)
set.seed(32)
split = sample.split(spam$type, SplitRatio = 0.7)
trainSpam = subset(spam, split == T)
testSpam = subset(spam, split ==F)
```

```r
names(spam)
```

Exploratory Analysis on the data

```
##  [1] "make"              "address"           "all"
##  [4] "num3d"             "our"               "over"
##  [7] "remove"            "internet"          "order"
## [10] "mail"              "receive"           "will"
## [13] "people"            "report"            "addresses"
## [16] "free"              "business"          "email"
## [19] "you"               "credit"            "your"
## [22] "font"              "num000"            "money"
## [25] "hp"                "hpl"               "george"
## [28] "num650"            "lab"               "labs"
## [31] "telnet"            "num857"            "data"
## [34] "num415"            "num85"             "technology"
## [37] "num1999"           "parts"             "pm"
## [40] "direct"            "cs"                "meeting"
## [43] "original"          "project"           "re"
## [46] "edu"               "table"             "conference"
## [49] "charSemicolon"     "charRoundbracket"  "charSquarebracket"
## [52] "charExclamation"   "charDollar"        "charHash"
## [55] "capitalAve"        "capitalLong"       "capitalTotal"
## [58] "type"
```

There are variables in the dataset that are named by common english words, lets take a look into what these fields store

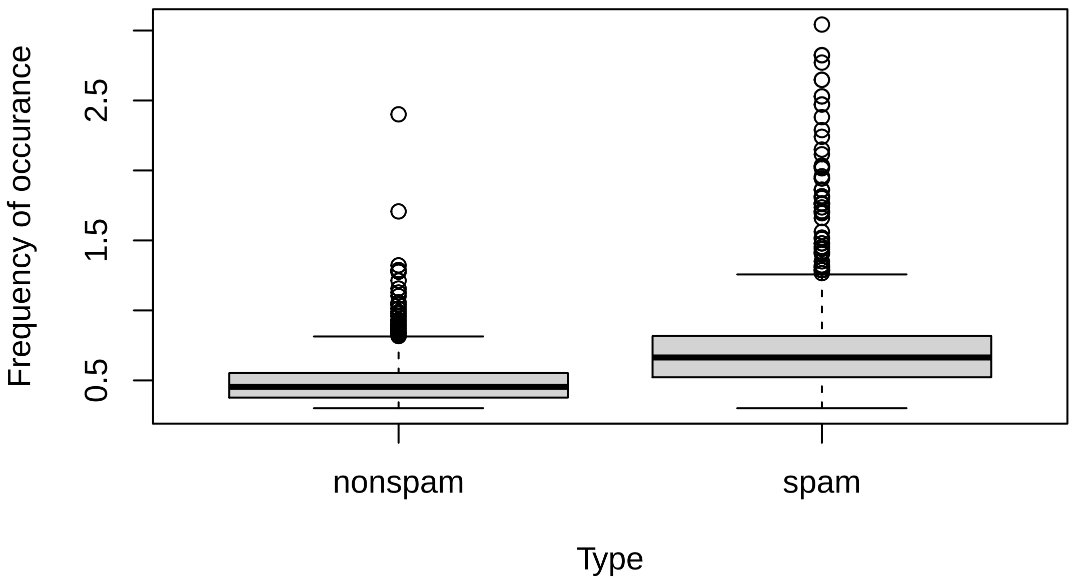
```r
head(spam)[,1:6]
```

```
##   make address  all num3d  our over
## 1 0.00    0.64 0.64     0 0.32 0.00
## 2 0.21    0.28 0.50     0 0.14 0.28
## 3 0.06    0.00 0.71     0 1.23 0.19
## 4 0.00    0.00 0.00     0 0.63 0.00
## 5 0.00    0.00 0.00     0 0.63 0.00
## 6 0.00    0.00 0.00     0 1.85 0.00
```

As we can see they represent the frequency of occurance of these terms within the mail(represented by a record in the dataset).

```
plot(
  log10(trainSpam$capitalAve+1) ~trainSpam$type,
  ylab = "Frequency of occurance",
  xlab = "Type")
```

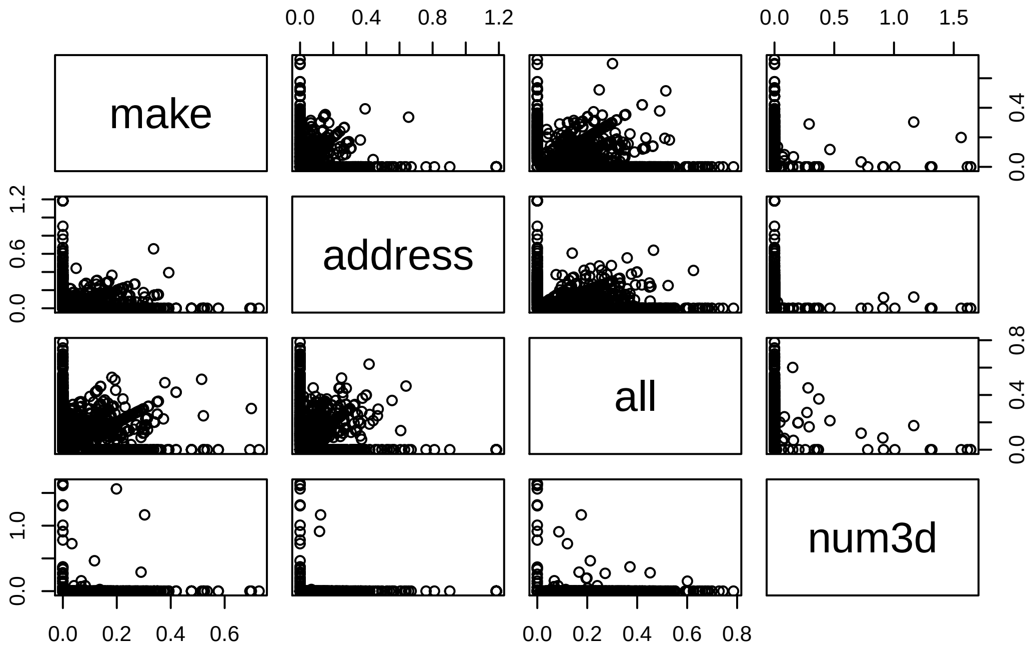
Comparing the values of data classified as spam vs nonspam



The spam data has higher median value for the occurance of 'capitalAve' ie. average usage of capital letter in the body of the mail for spam mails.

```
plot(log10(trainSpam[,1:4]+1))
```

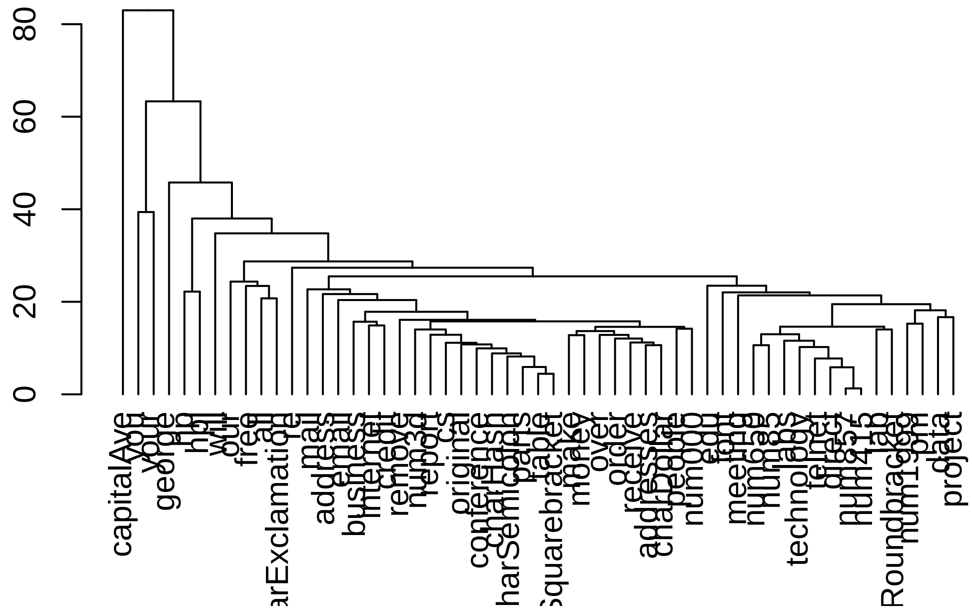
Analysing the relationships between various predictors



We do this to observe whether there is some correlation between the predictors, it is important that they be linearly independent for statistical reasons.

Performing hierarchical clustering To see which all predictors play a larger role in classifying the dataset

```
mdist = dist(t(log(trainSpam[,1:55]+1)))
hclustering = hclust(mdist)
plot(as.dendrogram(hclustering))
```



```r
trainSpam$numType = as.numeric(trainSpam$type)-1
```

Converting the label from character string to numeric type

```r
costFunc = function(x,y){
  sum(x != (y > 0.5))
}
```

Function to calculate cost function

Initialising a numeric vector to store the error The numeric vector is initialized with 'NULL' value, this numeric vector represents the cross validation matrix for linear models.

```r
cvError = rep(NULL, 55)
```

Fitting a linear model Fitting a linear model for each of the variable 1 through 55 and calculating the cost function error for each

```r
library(boot)
suppressWarnings(
```

```
 for(i in 1:55){
    lmFormula = reformulate(names(trainSpam)[i], response = "numType")
    glmFit = glm(lmFormula, family = "binomial", data = trainSpam)
    cvError[i] = cv.glm(trainSpam, glmFit, costFunc, 2)$delta[2]
 }
)
```

Getting names of top 5 predictors that have least cost function

```
## [1] "charDollar"      "charExclamation" "remove"          "money"
## [5] "free"
```

Getting a measure of uncertainity Fitting a linear model on the top 5 predictors

```
predModel = suppressWarnings(
  glm(
    numType ~ charDollar+charExclamation+remove+money+free,
    family = "binomial",
    data = trainSpam
    )
  )
```

Getting predictions on the test set

```
pred_y = as.character(
  ifelse(
    as.numeric(predict(predModel, testSpam))>0.5,
    "spam",
    "nonspam"
    )
  )
```

Comparing actual vs predicted

```
crossTab = table(pred_y,testSpam$type)
crossTab
```

```
##
## pred_y    nonspam spam
##   nonspam     810  190
##   spam         26  354
```

Accuracy

```
accuracy = sum(crossTab[-c(2:3)])/sum(crossTab[1:4])
accuracy
```

```
## [1] 0.8434783
```

Which means our model has an accuracy score of 84.35