Final Project

Ruben Solis

Due: 29 April 2020

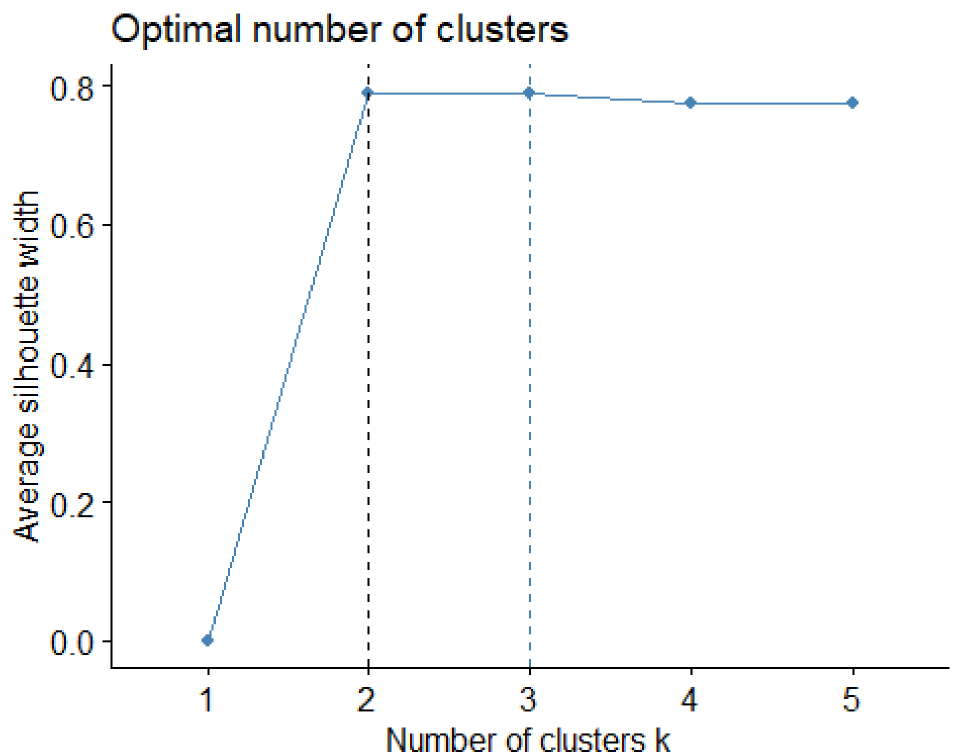
We begin this projecy by bringing the data into `R`. The data comes from the `kernlab` package, and we seek to use several classification methods to determine if an email is a regualr email or spam email.

```
library(kernlab)
data(spam)
dat <- spam
```

The data set contains 4601 emails in total and of which 1813 were spam emails, so approximately 40% of the emails in the data set are spam. Looking at the data we see that the type of each variable, except for our target variable, is continuous. Our target variable `type` is a character variable that is recorded as either spam or nonspam. Moreover, we note that there are no missing values in our dataset.

We next split the data into a training and test set with the following code

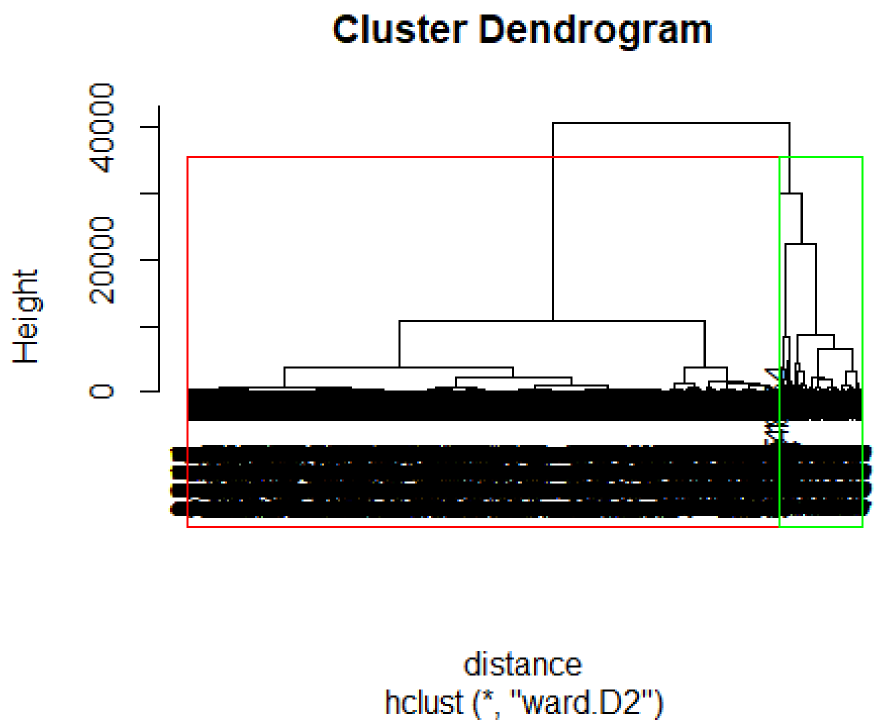
```
set.seed(5474)
training.data.index <- sample(1:nrow(dat), 0.667*nrow(dat))
training <- dat[training.data.index, ]
test <- dat[-training.data.index, ]

library(cluster)
library(factoextra)
distance <- get_dist(dat[,-58], method = "euclidean")
fviz_nbclust(dat[,-58], hcut, k.max=5) + geom_vline(xintercept = 2, linetype
= 2)
```



The figure above shows that the optimal number of clusters neeeded is 2, as we suspected.

We will use the distance matrix to build the dendogram. Futhermore, we will use Ward's method.

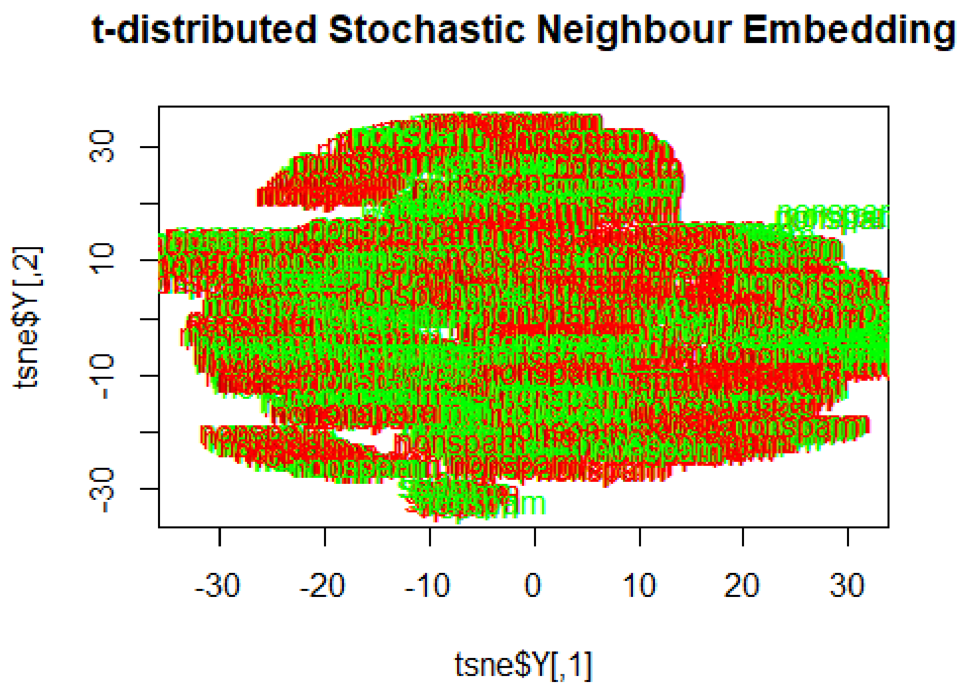
```
hc.out <- hclust(distance, method = "ward.D2")
hc.clusters <- cutree(hc.out, k=2)
plot(hc.out)
rect.hclust(hc.out, k=2, border=c("red", "green"))
```




```
table(hc.clusters)/4601 * 100

## hc.clusters
##       1       2
## 87.7418 12.2582
```

We see that the clustering did not produce great results. The original data set had approximately 40% spam emails whereas here we have about only 12%. We now move on to tSNE.

```
library(Rtsne)
labs <- dat[, 58]
tsne <- Rtsne(dat[, -58], dims=2, perplexity=30, check_duplicates = FALSE,
max_iter=500)
plot(tsne$Y, t='n', main = "t-distributed Stochastic Neighbour Embedding")
text(tsne$Y, labels = labs, col = c("green", "red"))
```



We see that the tsne did not really perform that well.

We next fit models using three different statistical learning methods to classify an email as either spam or nonspam; these methods are listed in each of the subsection below.

Our first method is linear discriminate analysis (LDA). The code below is used to fit said model.

```
library(MASS)
spam.lda <- lda(type ~., data = training, CV = FALSE)
spam.lda.pred <- as.vector(predict(spam.lda, test)$x)
```

Our second method is a logistic regession model selected via backward elimination based on BIC. The code below is used to fit the model.

```
fit.full <- glm(type~., data = training, family = binomial)
spam.back <- step(fit.full, direction = "backward",
                  k = log(nrow(dat)), trace = FALSE)
spam.back.pred <- predict(spam.back, newdata = test, type = "response")

library(rpart)
library(RColorBrewer)
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 = training,  method="class", control=control0,
              parms=list(split="information"), model = T)
```

Next, we need to prune the tree and find the optimal one. The code below will accomplish this task.

```
cv.error <- (tre0$cptable)[,4]
a0 <- 1     # IF a0=0, THEN 0SE
SE1 <- min(cv.error) + a0*((tre0$cptable)[,5])[which.min(cv.error)]
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

## 7
## 9

best.cp <-  sqrt(tre0$cptable[position,1] *  tre0$cptable[(position-1),1])
best.tree <- prune(tre0, cp=best.cp)
pred <- predict(best.tree, newdata =test, type="prob", na.action = na.pass)
spam.tree.pred <- pred[,1]
```

Our next classification model is a random forest. We have elected to use the default values as arguments/parameters when fitting this model. The code for this is below.

```
library(randomForest, quietly = TRUE)
set.seed(5474)
spam.rf <- randomForest(type~., data = training)
spam.rf.pred <- as.data.frame(predict(spam.rf, test, type = "prob"))$spam
```

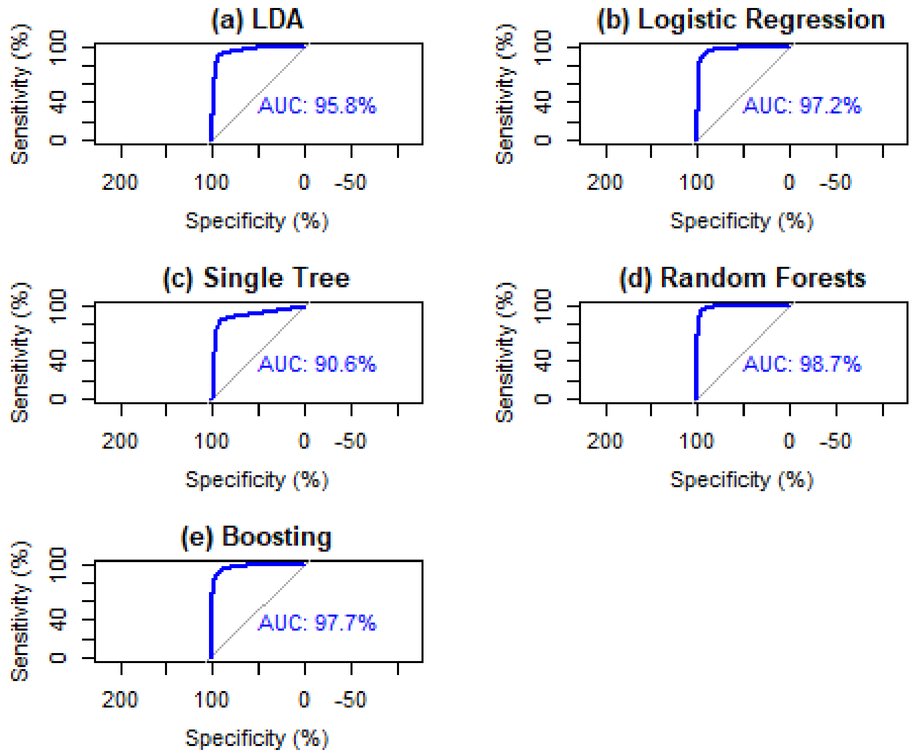
The goal is to make a weaker leaner into stronger learner.

```
library(ada)
stump <- rpart.control(cp=-1, maxdepth=1, minsplit=0)
fit.stump <- ada(type ~., data=training, iter=500, loss="e",
                 type="discrete", control=stump);
fit1.stump <- addtest(x=fit.stump, test.x = test[,-58], test.y = test[,58])
spam.boost.pred <- predict(fit.stump, newdata=test, type="probs")[, 2]
```

Finally, we evaluate the performance of these models by comparing their area under the reciever operating characteristic curve (AUC) values.

```
library(pROC, quietly = TRUE)
par(mfrow=c(3,2), mar=rep(4,4))
plot.roc(test$type, spam.lda.pred, main="(a) LDA", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue")
plot.roc(test$type, spam.back.pred, main="(b) Logistic Regression",
percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue")
plot.roc(test$type, spam.tree.pred, main="(c) Single Tree", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue")
plot.roc(test$type, spam.rf.pred, main="(d) Random Forests", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue")
```

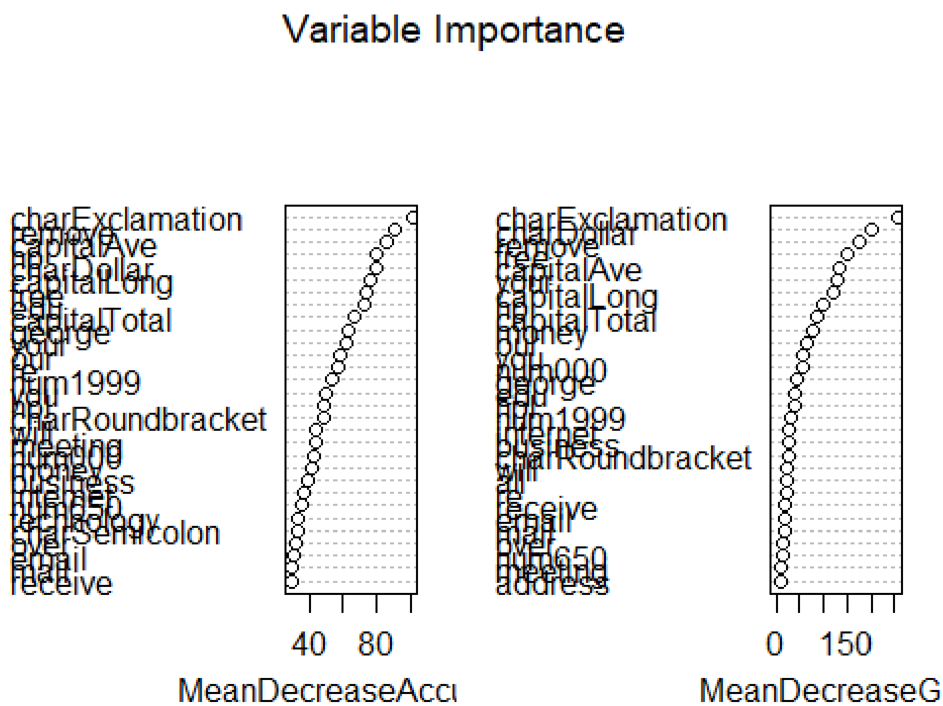
```r
plot.roc(test$type, spam.boost.pred, main = "(e) Boosting", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue")
```



We see that random forests method out performs the other methods with an AUC of 98.7%

We train a random forest model with 2000 trees using the entire data set.

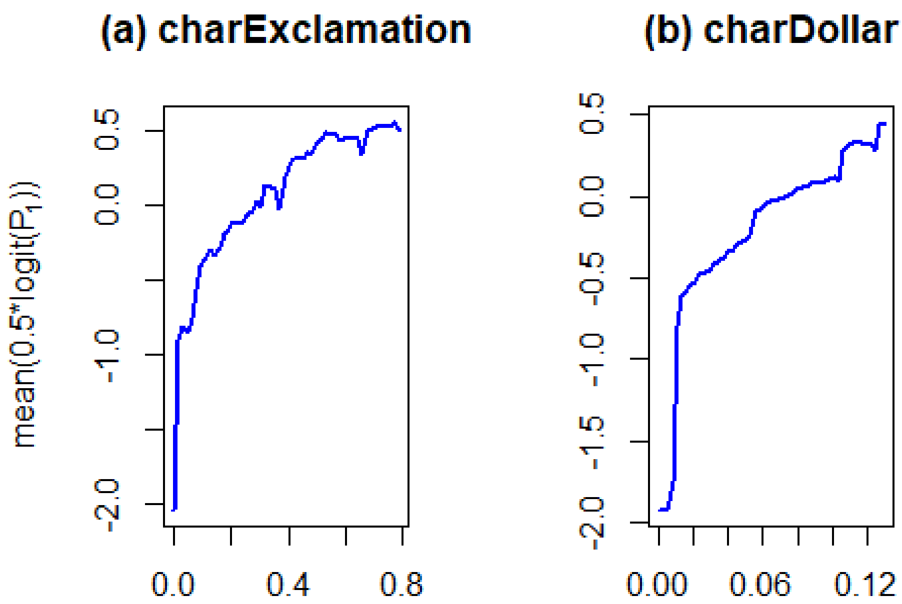
```r
set.seed(5474)
spam.rf.full <- randomForest(type~., data = dat, importance = TRUE, proximity
= TRUE,
                                  ntree = 2000)
varImpPlot(spam.rf.full, main = "Variable Importance")
```



Variable importance plot for the random forest fit with the full data set

From the variable importance plots we see that `charExclamation` and `charDollar` are the top two most important variables according to the mean decrease in Gini index. We next plot the partial dependence plots for these two variables.

```r
library(interpretR, quietly = TRUE)
par(mfrow=c(1, 2))
parDepPlot(x.name = "charExclamation", spam.rf.full, data = dat,
           main = "(a) charExclamation", col = "blue", lwd = 2, xlab = "",
           yalb = "Partial Dependence")
parDepPlot(x.name = "charDollar", spam.rf.full, data = dat,
           main = "(b) charDollar", col = "blue", lwd = 2, xlab="", ylab="")
```

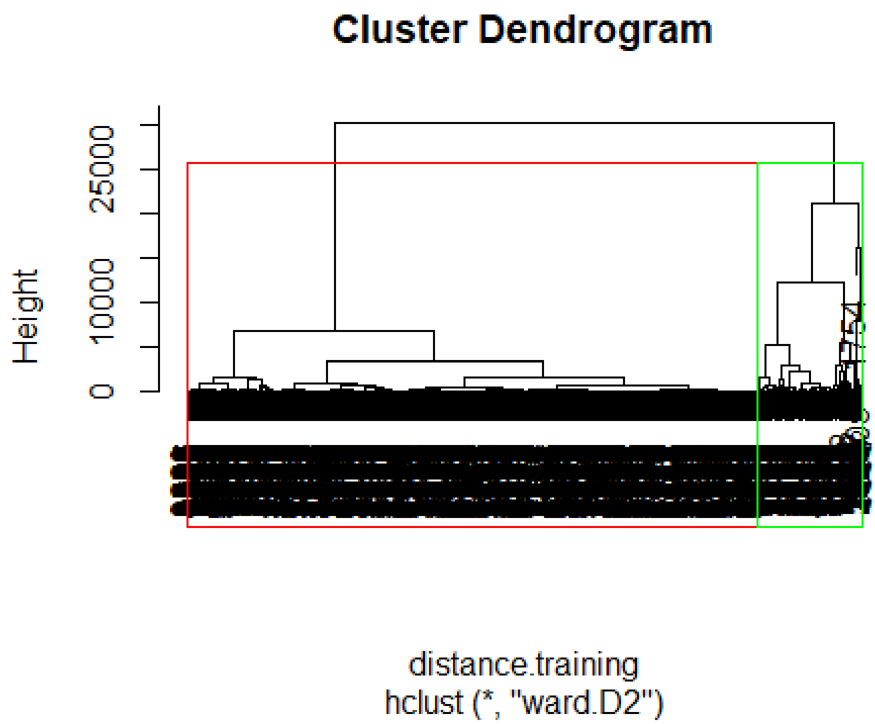


Partial dependence plots for top two important variables.

In the partial dependence plot, we see that as the number of exclamation marks (a) and dollar signs (b) increase, the more likely an email will be classified as a spam email.

We will be running hierarchical clustering technique, but this time with only the training data. We will also plot the tSNE denoting the new response variable and as well as the original.

```
distance.training <- get_dist(training[, -58], method = "euclidean")
hc.out.training <- hclust(distance.training, method = "ward.D2")
hc.clusters.training <- cutree(hc.out.training, k=2)
plot(hc.out.training)
rect.hclust(hc.out.training, k=2, border = c("red","green"))
```

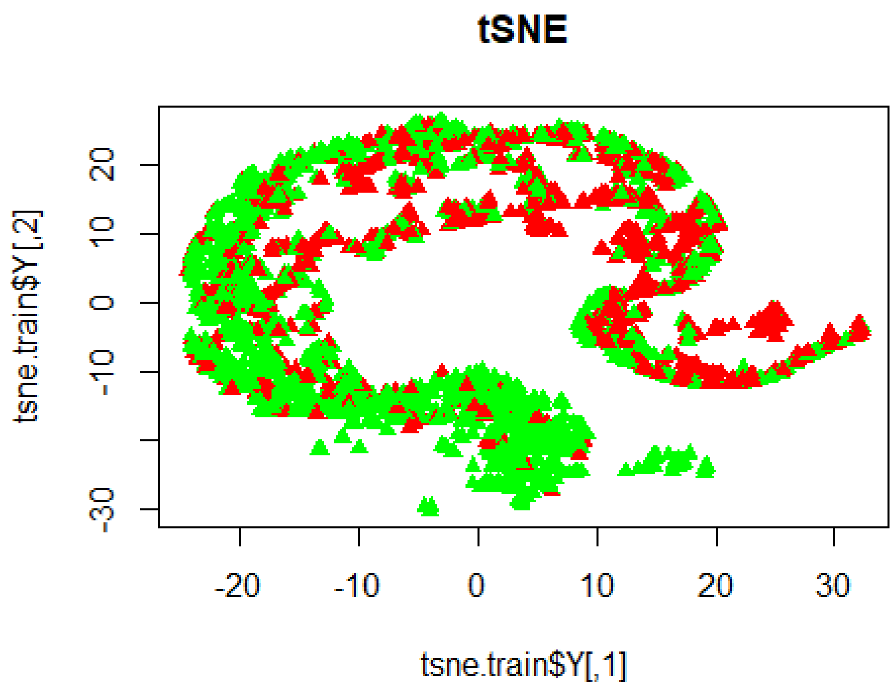



```
table(hc.clusters.training)/length(hc.clusters.training) * 100

## hc.clusters.training
##        1        2
## 15.41721 84.58279
```

Again, hierarchical clustering didn't perform well.

now for tSNE

```
new.train <- cbind(training, hc.clusters.training)
tsne.train <- Rtsne(new.train[, -c(58,59)], dims = 2,
                    perplexity = 43, verbose = FALSE,
                    check_duplicates = FALSE, max_iter = 500)
plot(tsne.train$Y, main = "tSNE", col = c("green", "red")[new.train[, 58]],
     pch = tsne.train$Y[new.train[, 59]])
```



The plot still hasn't improved much.

```
n <- nrow(dat)
out <- NULL
for (k in 1:ncol(dat)){
  vname <- colnames(dat)[k]
  x <- as.vector(dat[,k])
  n1 <- sum(is.na(x), na.rm = TRUE)
  n2 <- sum(x=="NA", na.rm = TRUE)
  n3 <- sum(x=='', na.rm = TRUE)
  n4 <- sum(x=="?", na.rm = TRUE)
  n5 <- sum(x=="*", na.rm = TRUE)
  n6 <- sum(x==".", na.rm = TRUE)
  nmiss <- n1 + n2 + n3 + n4 + n5 + n6
  ncomplete <- n - nmiss
  var.type <- typeof(x)
  if (var.type == "integer"){
    if (length(unique(x)) == 2){
      out <- rbind(out, c(col.number=k, vname=vname, mode="binary",
                   n.levels=length(unique(x)), ncomplete = ncomplete,
                   miss.prop=round(nmiss/n, digits = 4)))
    } else {
      out <- rbind(out, c(col.number=k, vname=vname, mode=typeof(x),
                          n.levels=length(unique(x)), ncomplete=ncomplete,
                          miss.prop=round(nmiss/n, digits = 4)))
    }
  } else {
```

```r
      out <- rbind(out, c(col.number=k, vname=vname, mode=typeof(x),
                          n.levels=length(unique(x)), ncomplete=ncomplete,
                          miss.prop=round(nmiss/n, digits = 4)))

  }
}
out <- as.data.frame(out)
row.names(out) <- NULL
out
```

```
##    col.number          vname    mode n.levels ncomplete miss.prop
## 1           1           make  double      142      4601         0
## 2           2        address  double      171      4601         0
## 3           3            all  double      214      4601         0
## 4           4          num3d  double       43      4601         0
## 5           5            our  double      255      4601         0
## 6           6           over  double      141      4601         0
## 7           7         remove  double      173      4601         0
## 8           8       internet  double      170      4601         0
## 9           9          order  double      144      4601         0
## 10         10           mail  double      245      4601         0
## 11         11        receive  double      113      4601         0
## 12         12           will  double      316      4601         0
## 13         13         people  double      158      4601         0
## 14         14         report  double      133      4601         0
## 15         15      addresses  double      118      4601         0
## 16         16           free  double      253      4601         0
## 17         17       business  double      197      4601         0
## 18         18          email  double      229      4601         0
## 19         19            you  double      575      4601         0
## 20         20         credit  double      148      4601         0
## 21         21           your  double      401      4601         0
## 22         22           font  double       99      4601         0
## 23         23         num000  double      164      4601         0
## 24         24          money  double      143      4601         0
## 25         25             hp  double      395      4601         0
## 26         26            hpl  double      281      4601         0
## 27         27         george  double      240      4601         0
## 28         28         num650  double      200      4601         0
## 29         29            lab  double      156      4601         0
## 30         30           labs  double      179      4601         0
## 31         31         telnet  double      128      4601         0
## 32         32         num857  double      106      4601         0
## 33         33           data  double      184      4601         0
## 34         34         num415  double      110      4601         0
## 35         35          num85  double      177      4601         0
## 36         36     technology  double      159      4601         0
## 37         37        num1999  double      188      4601         0
## 38         38          parts  double       53      4601         0
## 39         39             pm  double      163      4601         0
```

```
## 40          40              direct    double      125       4601           0
## 41          41                  cs    double      108       4601           0
## 42          42             meeting    double      186       4601           0
## 43          43            original    double      136       4601           0
## 44          44             project    double      160       4601           0
## 45          45                  re    double      230       4601           0
## 46          46                 edu    double      227       4601           0
## 47          47               table    double       38       4601           0
## 48          48          conference    double      106       4601           0
## 49          49       charSemicolon    double      313       4601           0
## 50          50    charRoundbracket    double      641       4601           0
## 51          51 charSquarebracket    double      225       4601           0
## 52          52     charExclamation    double      964       4601           0
## 53          53          charDollar    double      504       4601           0
## 54          54            charHash    double      316       4601           0
## 55          55          capitalAve    double     2161       4601           0
## 56          56         capitalLong    double      271       4601           0
## 57          57        capitalTotal    double      919       4601           0
## 58          58                type character        2       4601           0
```