# Homework2

Julissa Duenas

2/4/2021

```r
spam <- read_table2("spambase.tab", guess_max=2000)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```r
spam <- spam %>%
mutate(y = factor(y, levels=c(0,1), labels=c("good", "spam"))) %>% # label as factors
mutate_at(.vars=vars(-y), .funs=scale) # scale others

calc_error_rate <- function(predicted.value, true.value){
return(mean(true.value!=predicted.value))
}

records = matrix(NA, nrow=3, ncol=2)
colnames(records) <- c("train.error","test.error")
rownames(records) <- c("knn","tree","logistic")

set.seed(1)
test.indices = sample(1:nrow(spam), 1000)
spam.train=spam[-test.indices,]
spam.test=spam[test.indices,]

nfold = 10
set.seed(1)
folds = seq.int(nrow(spam.train)) %>% ## sequential obs ids
cut(breaks = nfold, labels=FALSE) %>% ## sequential fold ids
sample ## random fold ids
```

## Problem 1

```r
do.chunk <- function(chunkid, folddef, Xdat, Ydat, k){
  train = (folddef!=chunkid)
  Xtr = Xdat[train,]
  Ytr = Ydat[train]
  Xvl = Xdat[!train,]
  Yvl = Ydat[!train]
  ## get classifications for current training chunks
  predYtr = knn(train = Xtr, test = Xtr, cl = Ytr, k = k)
  ## get classifications for current test chunk
  predYvl = knn(train = Xtr, test = Xvl, cl = Ytr, k = k)
  data.frame(train.error = calc_error_rate(predYtr, Ytr),
             val.error = calc_error_rate(predYvl, Yvl))
}
```

```
set.seed(1)
kvec <- c(1,seq(10,50,length.out=5))

Xtrain <- spam.train%>%select(-y)%>%scale(center=TRUE,scale=TRUE)
Ytrain <- spam.train$y
Ytest=spam.test$y
Xtest <- spam.test%>%select(-y)%>%scale(center=TRUE,scale=TRUE)
error.folds=NULL
for(j in kvec){
  tmp <- ldply(1:nfold,do.chunk,folddef=folds,Xdat=Xtrain,Ydat=Ytrain,k=j)
  tmp$neighbors <- j
  error.folds <- rbind(error.folds,tmp)
}
errors <- melt(error.folds,id.vars = c('neighbors'),value.name = 'error')
val.error.means=errors%>%
  filter(variable=='val.error')%>%
  group_by(neighbors,variable)%>%
  summarise_each(funs(mean),error)%>%
  ungroup()%>%
  filter(error==min(error))
```

```
## Warning: `summarise_each_()` is deprecated as of dplyr 0.7.0.
## Please use `across()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
best.kfold <- max(val.error.means$neighbors)
best.kfold
```

```
## [1] 10
```

Problem 2

```
set.seed(1)
pred.Ytest <- knn(train=Xtrain,test=Xtest,cl=Ytrain,k=best.kfold)
test.error <- calc_error_rate(pred.Ytest,Ytest)
train.error <- mean((error.folds%>%filter(neighbors==best.kfold))$train.error)
records[1,1] <- train.error
```

```r
records[1,2] <- test.error
```

Problem 3

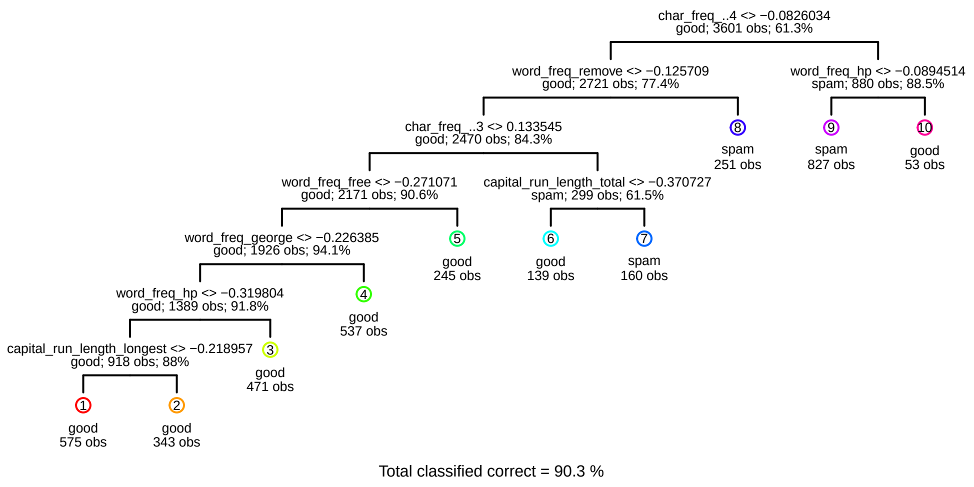
```r
spamtree <- tree(y~.,data=spam.train,control = tree.control(nobs = nrow(spam.train),minsize=5,mindev=1e
summary(spamtree)
```

```
##
## Classification tree:
## tree(formula = y ~ ., data = spam.train, control = tree.control(nobs = nrow(spam.train),
##     minsize = 5, mindev = 1e-05))
## Variables actually used in tree construction:
##  [1] "char_freq_..4"              "word_freq_remove"
##  [3] "char_freq_..3"              "word_freq_free"
##  [5] "word_freq_george"           "word_freq_hp"
##  [7] "capital_run_length_longest" "word_freq_receive"
##  [9] "word_freq_credit"           "capital_run_length_average"
## [11] "word_freq_your"             "word_freq_mail"
## [13] "word_freq_re"               "word_freq_our"
## [15] "word_freq_you"              "capital_run_length_total"
## [17] "word_freq_make"             "word_freq_all"
## [19] "word_freq_internet"         "word_freq_email"
## [21] "word_freq_project"          "word_freq_money"
## [23] "word_freq_1999"             "word_freq_will"
## [25] "char_freq_..1"              "word_freq_order"
## [27] "char_freq_."                "word_freq_data"
## [29] "word_freq_over"             "word_freq_meeting"
## [31] "word_freq_650"              "word_freq_edu"
## [33] "word_freq_address"          "word_freq_business"
## Number of terminal nodes:  149
## Residual mean deviance:  0.04568 = 157.7 / 3452
## Misclassification error rate: 0.01361 = 49 / 3601
```

There are 149 leaf nodes. 49 training observations were misclassified

Problem 4

```r
pruned_tree <- prune.tree(spamtree,best=10)
draw.tree(pruned_tree,nodeinfo = TRUE,cex=.4)
```

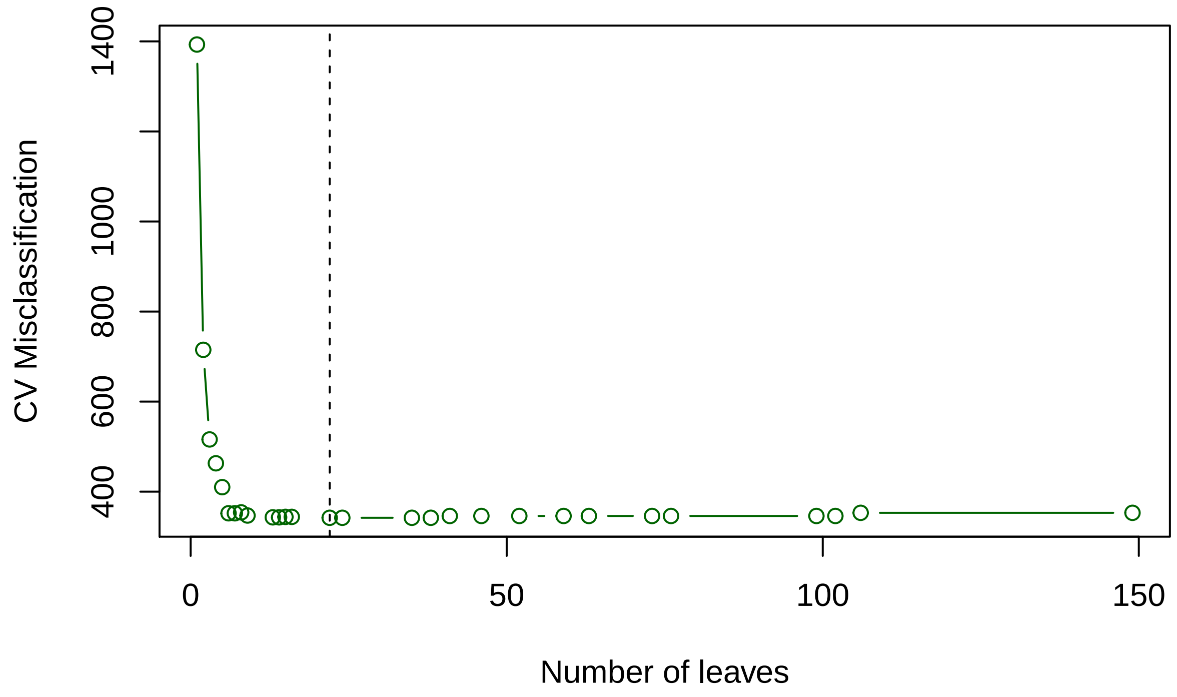


Problem 5

```
cv.spamtree <- cv.tree(spamtree,rand = folds,FUN=prune.tree,method='misclass')
plot(cv.spamtree$size,cv.spamtree$dev,xlab='Number of leaves',ylab='CV Misclassification',type='b',col=
best.size.cv <- cv.spamtree$size[max(which(cv.spamtree$dev==min(cv.spamtree$dev)))]
best.size.cv
```

```
## [1] 22
```

```
abline(v=best.size.cv,lty=2)
```



Optimal tree size is 22

Problem 6

```
spamtree.pruned <- prune.misclass(spamtree,best=best.size.cv)
pred.spamtree.pruned <- predict(spamtree.pruned,spam.test,type='class')
prune.train.error <- mean(predict(spamtree.pruned,spam.train,type='class')!=spam.train$y)
prune.test.error <- calc_error_rate(pred.spamtree.pruned,Ytest)
records[2,1] <- prune.train.error
records[2,2] <- prune.test.error
```

Problem 7 a)

$$p(z) = \frac{e^z}{1+e^z}$$

$$p(z) = x, then x = \frac{e^z}{1+e^z}$$
$$x + xe^z = e^z$$
$$x = e^z - xe^z$$
$$x = e^z(1-x)$$
$$e^z = \frac{x}{1-x}$$
$$z(p) = ln(\frac{p}{1-p})$$

b) increasing $x_1$ by 2 would change the term $e^{\beta_1 x_1}$ into $e^{\beta_1(x_1+2)}$ increasing the odds of the outcome by $e^{2\beta_1}$

$p(z) = \frac{e^z}{(1+e^z)}$ and $z = \beta_0 + \beta_1 x_1$ therefore, $p(x) = \frac{e^{\beta_0+\beta_1 x_1}}{(1+e^{\beta_0+\beta_1 x_1})}$

we know $\beta_1$ is negative so $p(x) = \frac{e^{\beta_0} * e^{-\beta_1 x_1}}{(1+e^{\beta_0} * e^{-\beta_1 x_1})}$ now, when $x_1$ approaches $\infty$, $e^{-\beta_1 x_1}$ approaches 0 so $p(z) = \frac{e^{\beta_0} * 0}{(1+e^{\beta_0} * 0)}$ therefore, when $x_1$ approaches $\infty$, $p$ aproaches 0

similarly, when $x_1$ aproaches $-\infty$, $e^{-\beta_1 x_1}$ approaches $\infty$ so $p(z) = \frac{e^{\beta_0} * \infty}{(1+e^{\beta_0} * \infty)}$ therefore, when $x_1$ approaches $-\infty$, $p$ aproaches 1

Problem 8

```r
glm.spam <- glm(y~.,data=spam,family=binomial)
```

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

```r
glm.spam.train <- predict(glm.spam,spam.train,type='response')
fit.train <- spam.train%>%mutate(predSPAM=as.factor(ifelse(glm.spam.train<=.5,'good','spam')))
glm.spam.test <- predict(glm.spam,spam.test,type='response')
fit.test <- spam.test%>%mutate(predSPAM=as.factor(ifelse(glm.spam.test<=0.5,'good','spam')))

glm.train.error <- table(pred=fit.train$predSPAM,true=fit.train$y)
glm.train.error <- 1-sum(diag(glm.train.error))/sum(glm.train.error)
glm.test.error <- table(pred=fit.test$predSPAM,true=fit.test$y)
glm.test.error <- 1-sum(diag(glm.test.error))/sum(glm.test.error)

records[3,1] <- glm.train.error
records[3,2] <- glm.test.error
records
```

```
##          train.error test.error
## knn       0.07911393      0.114
## tree      0.06053874      0.091
## logistic  0.06775896      0.072
```

The logistic methos had the lowest missclassification error on the test set with 0.072

Problem 9

I would be more concerned about false positive rates that are too large. large rates of false positives could lead to emails that are not spam being classified as spam. important information can be lost this way. Having the true positive rate being too small would lead to having a lot of spam emails in your inbox that one would have to sort through. This isn't as big a consequence as missing an important email.