**Report 5**

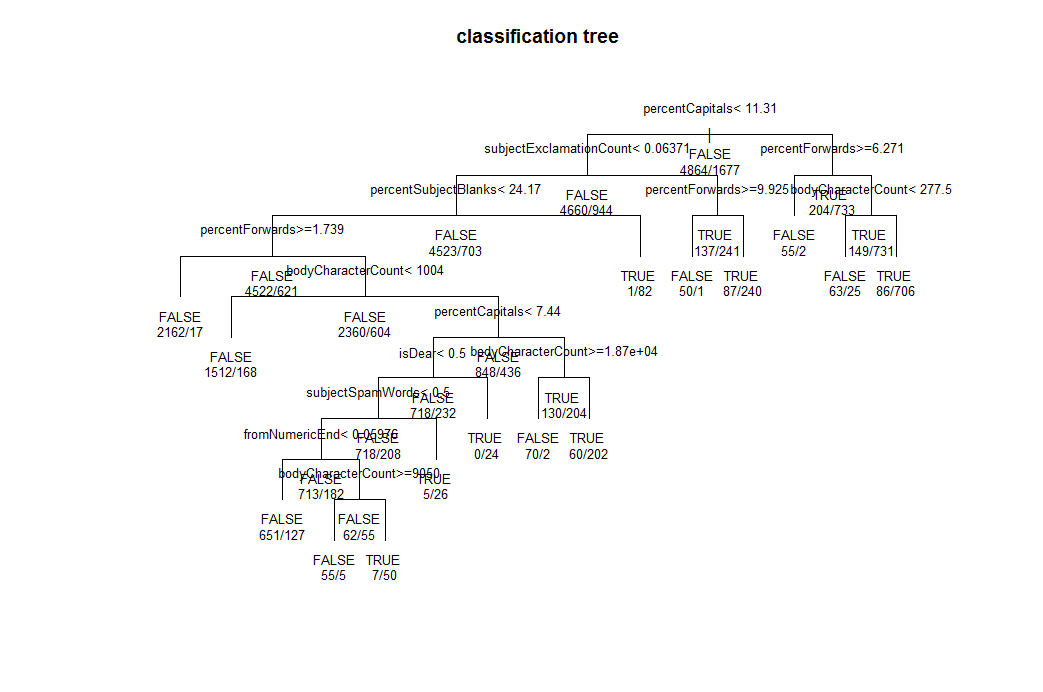
**Yibing Luo**

*" I certify that I have acknowledged any code that I used from any other person in the class, from Piazza or any Web site or book or other source. Any other work is my own. "*

1. **Training Data**

*We use cross validation this part and set n = 5.*

1. **Classification tree model**



Below is the table of each fold:

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 268 | 47 |
| ham | 68 | 926 |

Prediction err: 0.088 Type1: 0.051 Type2:0.0036

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 252 | 50 |
| ham | 83 | 923 |

Prediction err: 0.101 Type1:0.063 Type2:0.0038

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 268 | 58 |
| ham | 67 | 915 |

Prediction err: 0.096 Type1: 0.051 Type2:0.0044

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 255 | 41 |
| ham | 80 | 932 |

Prediction err: 0.0925 Type1: 0.061 Type2:0.0031

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 244 | 52 |
| ham | 92 | 920 |

Prediction err: 0.0975 Type1: 0.0703 Type2:0.00398

The mean of prediction error in classification tree is **0.0975.**

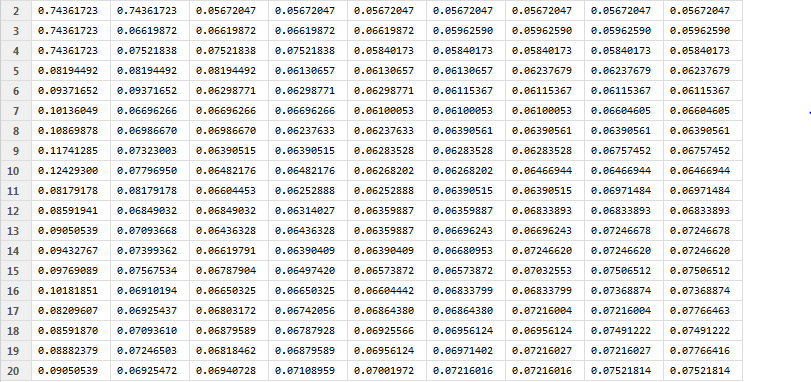
1. **kNN model**

First the optimized parameters are wanted. The parameters are above:

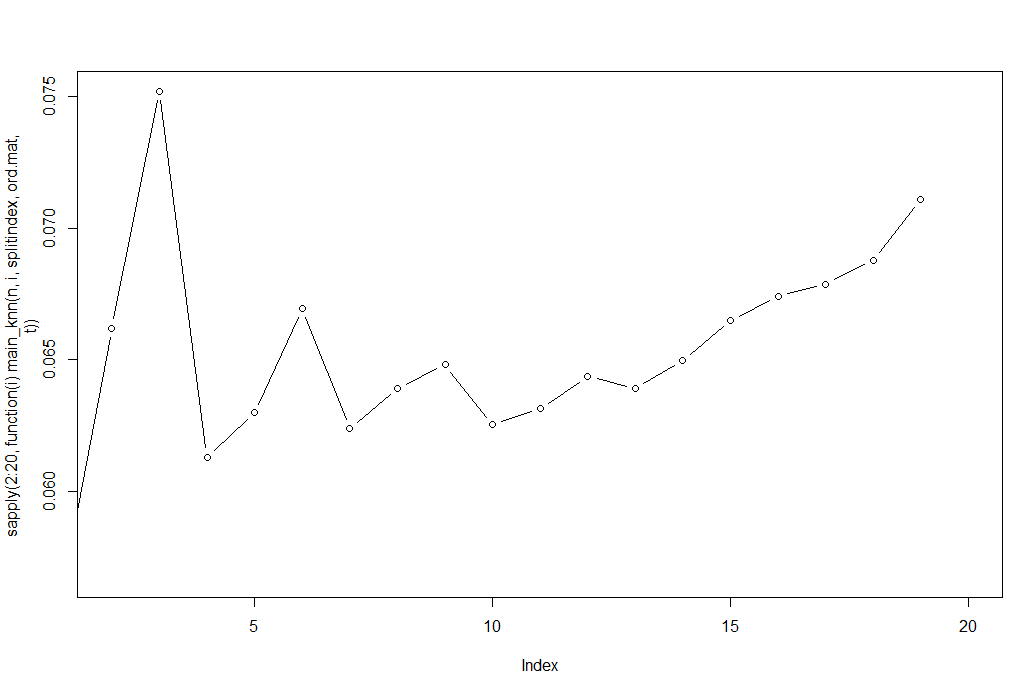
**k**: the number of the nearest point, which we are used to classify the specific point.

**p**: the probability in voting mechanism to classify the point.

**distance**: there are some ways to calculate the distance, we want to find which one is the best.



The table shows that when k =2, a better result can be gotten. The columns correspond to p (p is from 0.2 to 1). It can be found that when p<0.3, the result is relatively bad. And if p>0.3, the result seems to be stable and a lit bit variance. Further if k = 2, p>0.3 seems have no effect to result. In my model, I choose p =0.5 .



This plot provides evidence to this conclusion. From this plot, the prediction error is increasing when k>10. And when k<10, k = 2 is the best one.

When considered about ways to calculate the distance, ‘manhattan’ is chosen based on calculation.

As a result, the parameter is set, which is: **k=2, p=.5, dist(…’manhattan’).**

This is the result:

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 312 | 39 |
| ham | 24 | 934 |

Prediction err: 0.048 Type1: 0.018 Type2:0.029

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 303 | 44 |
| ham | 32 | 929 |

Prediction err: 0.0581 Type1:0.0244 Type2:0.0336

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 308 | 57 |
| ham | 27 | 916 |

Prediction err: 0.064 Type1: 0.02 Type2:0.043

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 309 | 38 |
| ham | 26 | 935 |

Prediction err: 0.049 Type1: 0.02 Type2:0.029

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 298 | 46 |
| ham | 38 | 926 |

Prediction err: 0.064 Type1: 0.029 Type2:0.035

The mean prediction error of this set of folds is **0.0567**

**So compare with two results , kNN model has lower mis-classification rate. Thus kNN model can create better result.**

1. **Test data**
2. **Classification tree model**

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 325 | 81 |
| ham | 164 | 1430 |

Prediction err: 0.1225 Type1: 0.082 Type2:0.0405

1. **kNN model**

|  |  |  |
| --- | --- | --- |
| true pred | spam | ham |
| spam | 451 | 82 |
| ham | 38 | 1429 |

Prediction err: 0.06 Type1: 0.019 Type2:0.041

**Obviously, the misclassification rate is lower in kNN model. From above, I think kNN model works better than classification tree model in classification e-mails.**

The number of both classifiers misclassifying the same test messages is **27+22.**

And number of both classifiers correctly classifying the same test messages is **1370+313**.

(the format of number means: number of ham + number of spam)

1. **Blind data**

*The model classification tree and model kNN are applied to predict.*

1. **Classification tree model**

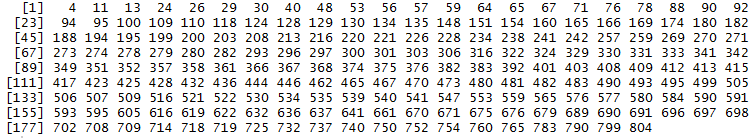
In this part, ct model is used to classify the e-mails:

In total 808 mails：

**Spam: 195**

**Ham: 613**

Index of ham:



1. **kNN model**

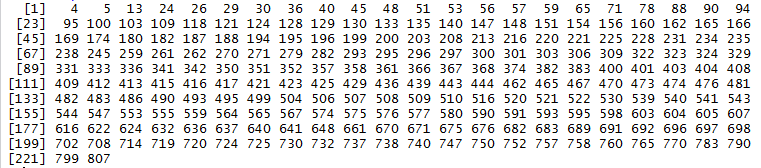
In this part, kNN model is used to classify the e-mails:

In total 808 mails:

**Spam: 586**

**Ham: 222**

Index of ham:



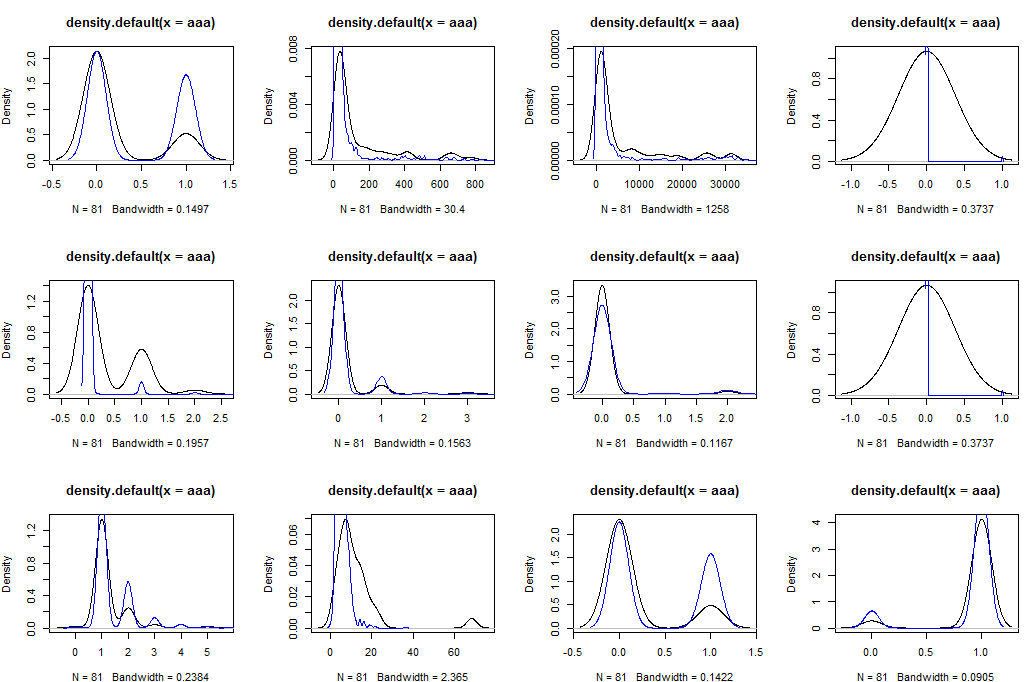
1. **Explore the misclassified observation**

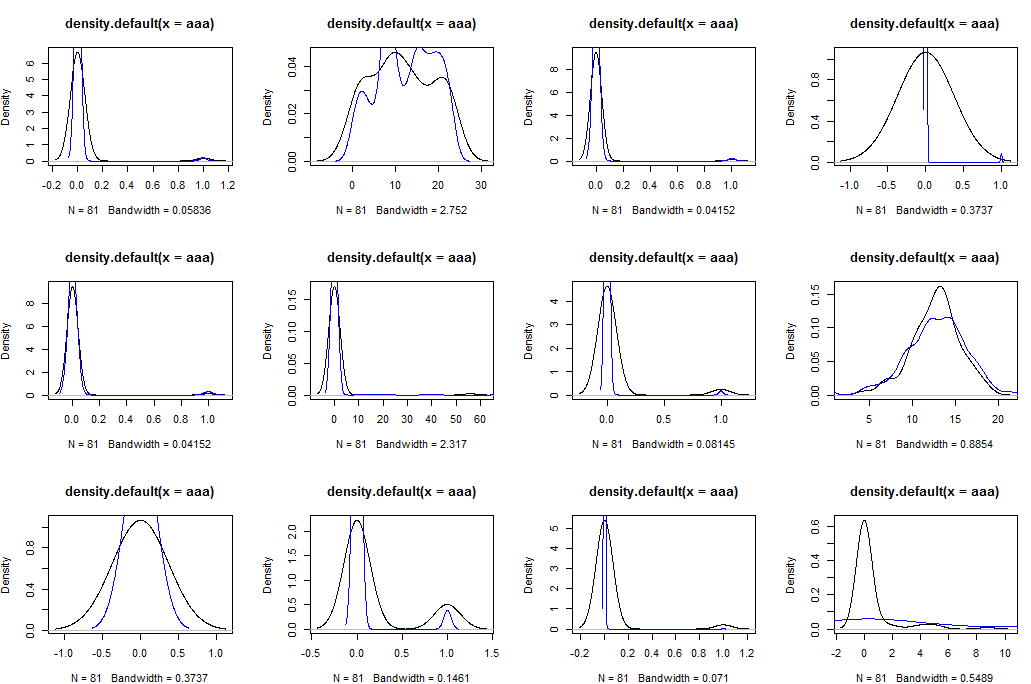
*In this part, we consider about the result from* ***test data***

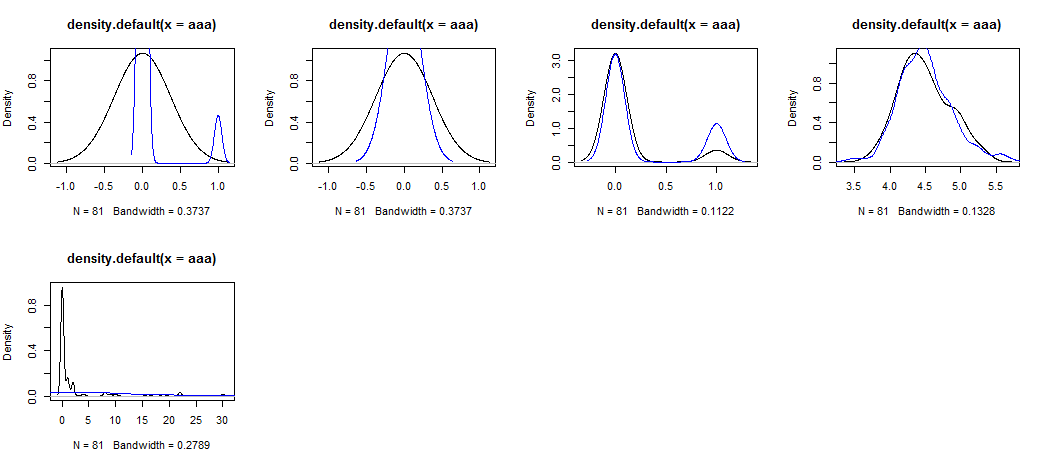
1. **classification tree model**

When ct model is used to predict, there is 81 spam misclassified as ham and 164 ham misclassified as spam

Below is the exploration of **spam misclassification**:



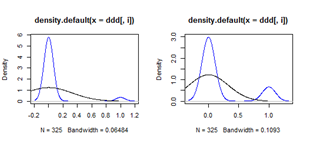
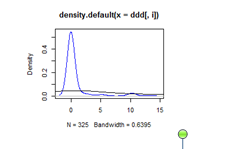




Variables of **hourSent, percentForwards, numDollarSigns** have a quite different density plot.

Looking at exploration of **ham misclassification** which is approached by the same way, and find that:

**subJectpunctuationCheck, subjectspamwords, percentForward** have a quite different density plot.

‘percentForward’ is abnormal in both situation. This might suggests that we need to abandon this variable.

1. **kNN model**

Using density plot cannot find obviously abnormal variables in this model.(The plot takes too much room, so it is ignored in this place)

1. **Conclusion and further step**

In test data, the mean prediction error for classification tree is 0.1225, and the mean prediction error for kNN model is 0.06. This is strong evidence which supposes kNN model works well, in my opinion.

When the misclassified observations are explored, by my method, no interesting characteristics are found. However, the plots show something for ct model. They suggest some variables need to be deleted and some variables need to be transformed in order to decrease the bad effect for model building.

Further, based on last homework’s report, I found different variables have different levels of effect to the model. Thus, we may want to set some weights to these variables in the further steps. That’s may make classification more precise.

I think why classification tree didn’t work well is because I didn’t deal with the dataset well. I did some transformations though, but didn’t work well.

1. **Appendix (code)**

library(rpart)

load("E:/STA141/ASSIGNMENT/5/trainVariables hehe.rda")

load("E:/STA141/ASSIGNMENT/5/testData.rda")

load("E:/STA141/ASSIGNMENT/5/blindTestData.rda")

t= trainVariables

tes = testVariables

bl = blindTestVariables

#replace na by mean of corresponding column

for(i in 1: 29){

t[which(is.na(t[,i])),i] = mean(t[,i] , na.rm = TRUE)

}

for(i in 1: 29){

tes[which(is.na(tes[,i])),i] = mean(tes[,i] , na.rm = TRUE)

}

for(i in 1: 29){

bl[which(is.na(bl[,i])),i] = mean(bl[,i] , na.rm = TRUE)

}

split\_index =

#

#split the trainDate

#n：the number of data set in cross validation

#trainVariables: the trainVariables which need to be splitted

#

function(n,trainVariables){

index=0

groups = rep(1:n, length=length(trainVariables[[1]]))

for(i in 1:n){

index[i] =list(which(groups==i))

}

index

}

result =

#

#the result and prediction err of the model

#testVariables: the test data set

#pred: the prediction value of each model

#

function(testVariables , pred){

l1=l2=l3=l4=0

for(i in 1:length(testVariables$isSpam)){

if(testVariables$isSpam[i]==TRUE & pred[i]==TRUE) l1 = l1+1

if(testVariables$isSpam[i]==TRUE & pred[i]==FALSE) l2 = l2+1

if(testVariables$isSpam[i]==FALSE & pred[i]==TRUE) l3 = l3+1

if(testVariables$isSpam[i]==FALSE & pred[i]==FALSE) l4 = l4+1

}

err = (l2+l3)/length(testVariables$isSpam)

result = data.frame(spam = c(l1,l2), ham = c(l3,l4))

rownames(result) = c("spam","ham")

list(err,result)

}

#model of classification tree

classification =

#

#function for creating classification tree model to classify

#trainVariables:variables in training dataset

#testVariables: variables in testing dataset

#first create the tree, then prune the tree, finally do prediction

#

function(trainVariables , testVariables){

model = rpart(isSpam~. , method = "class" ,data = trainVariables)

pmodel = prune(model,cp =

model$cptable[which.min(model$cptable[,"xerror"]),"CP"])

pred = as.logical(predict(pmodel , testVariables,type = "class"))

pred

}

classification\_test =

#

#function to test model classification tree

#use classification() to test testdataset

#

function(testVariables , trainVariables){

pred = classification(trainVariables,testVariables)

result = result(testVariables,pred)

result

}

main\_ct =

#

#cross validation for classification tree model

#t: the trainVariables

#splitindex: the index of the trainVriables

#n:the number to split

#

function(t , splitindex,　n){

r=0

sum\_c = 0

for(i in 1:n){

tt = t[unlist(splitindex[i]),]

ttr = t[-unlist(splitindex[i]),]

r[i] = list(classification\_test(testVariables = tt , ttr))

print(r[i])

}

for(i in 1:n){

sum\_c = r[[i]][[1]]+sum\_c

}

sum\_c/n

}

#model of k-NN(**this code is from dicussion7 and i understand this code**)

knn =

#

#k: the number of the nearest neighbors

#test.index: the indext of the cases which need to be test

#distmatrix: the matrix of distence which we calculate before

#

function(k, test.index , distmatrix){

ord = distmatrix[ ,test.index]

nn = apply(ord , 2 , drop.first , drop = test.index)

nn = nn[seq(k),]

}

drop.first =

function(x,drop)

{

x[-match(drop,x)]

}

knn\_test =

#

#test\_index: the index of test cases

#ord.mat: pre\_computed matrix

#k: the number of the nearest neighbors

#data: dataset(trainVariables or testVariables)

function(test\_index,ord.mat,k,data){

pred = 0

knn\_num = knn(k,test\_index,ord.mat )

for(i in 1:length(knn\_num[1 , ])){

xx = table(c( data[knn\_num[ , i] , ]$isSpam , FALSE , TRUE ))

if(xx['TRUE'] / xx['FALSE'] >= .5) pred[i] = TRUE

else pred[i] = FALSE

}

result(data[test\_index , ] , pred)

}

main\_knn =

#

#n: the number of splitted datasets

#k: the number of the nearest nighbors

#ord.mat: the matrix of distance

#data: dataset

#

function(n, k , splitindex , ord.mat,data){

sum\_k = 0

r\_knn=0

for(i in 1:n){

tt = unlist(splitindex[i])

r\_knn[i] =list(knn\_test(tt,ord.mat,k,data))

}

for(i in 1:n){

sum\_k = r\_knn[[i]][[1]]+sum\_k

}

sum\_k/n

}

plotmix =

#

#function to plot

#

function(a,b)

{

plot(density(a))

lines(density(b),col = "blue")

}

#training data: cross validation

#data using :trainingVariables

n = 5

splitindex = split\_index(n,t)

#model classification

main\_ct(t, splitindex,n)

#model k-nn

t1 = t[-30]

scale\_t = scale(t1)

dists\_t = dist(scale\_t,"manhattan")

dists\_t = as.matrix(dists\_t)

ord.mat = apply(dists\_t , 2, order)

k=2

main\_knn(n,k,splitindex,ord.mat,t)

#test which k is best

eff = 0

for(k in 2:20){

eff[k]=main\_knn(n,k,splitindex,ord.mat,t)

}

#testData

#model1

classification\_test(tes,t)

#model2

a = rbind(tes,t)

scale\_a = scale(a[-30])

dists\_a = dist(scale\_a,"manhattan")

dists\_a = as.matrix(dists\_a)

ord.mat\_a = apply(dists\_a , 2, order)

k =2

knn\_test(1:2000,ord.mat\_a , k , a)

################################################

knn\_test2 =

#

#test\_index: the index of test cases

#ord.mat: pre\_computed matrix

#k: the number of the nearest neighbors

#data: dataset(trainVariables or testVariables)

function(test\_index,ord.mat,k,data){

pred = 0

knn\_num = knn(k,test\_index,ord.mat )

for(i in 1:length(knn\_num[1 , ])){

xx = table(c( data[knn\_num[ , i] , ]$isSpam , FALSE , TRUE ))

if(xx['TRUE'] / xx['FALSE'] >= .5) pred[i] = TRUE

else pred[i] = FALSE

}

pred

}

################################################

#explore the misclassified observations

table(tes$isSpam)

resct = classification(t,tes)

ctmis\_f = which(resct[1:1511]==TRUE)

ctmis\_t = which(resct[1512:2000]==FALSE)

#subset the table

#aaa is the mis-classify of ham

#bbb is correct mail of ham

#ccc is the mis-classify of spam

#ddd is correct mail of spam

aaa = data.frame(tes[ctmis\_f,])

bbb = data.frame(tes[1:1511,][-ctmis\_f,])

ccc = data.frame(tes[ctmis\_t,])

ddd = data.frame(tes[1512:2000,][-ctmis\_t,])

par(mfrow = c(3,4))

sapply(1:12 , function(i) plotmix(ccc[,i],ddd[,i]))

sapply(13:24 , function(i) plotmix(ccc[,i],ddd[,i]))

sapply(25:29 , function(i) plotmix(ccc[,i],ddd[,i]))

#kNN

resknn = knn\_test2(1:2000 , ord.mat\_a , k , a)

knnmis\_f = which(resknn[1:1511]==TRUE)

knnmis\_t = which(resknn[1512:2000]==FALSE)

aaa1 = data.frame(tes[knnmis\_f,])

bbb1 = data.frame(tes[1:1511,][-knnmis\_f,])

ccc1 = data.frame(tes[ctmis\_t,])

ddd1 = data.frame(tes[1512:2000,][-ctmis\_t,])

par(mfrow = c(3,4))

sapply(1:12 , function(i) plotmix(aaa1[,i],bbb[,i]))

sapply(13:24 , function(i) plotmix(aaa1[,i],bbb[,i]))

sapply(25:29 , function(i) plotmix(aaa1[,i],bbb[,i]))

#find both classifiers' mis-classify

which(resct[1:1511] + resknn[1:1511]==2)

which(resct[1511:2000] + resknn[1511:2000]==0)

###################################

knn\_test1 =

#

#test\_index: the index of test cases

#ord.mat: pre\_computed matrix

#k: the number of the nearest neighbors

#data: dataset(trainVariables or testVariables)

function(test\_index,ord.mat,k,data){

pred = 0

knn\_num = knn(k,test\_index,ord.mat )

for(i in 1:length(knn\_num[1 , ])){

xx = table(c( data[knn\_num[ , i] , ]$isSpam , FALSE , TRUE ))

if(xx['1'] / xx['0'] >= .5) pred[i] = TRUE

else pred[i] = FALSE

}

pred

}

######################################

#test blind

#knn

load("E:/STA141/ASSIGNMENT/5/blindTestData.rda")

bl = blindTestVariables

temp = data.frame(c(1:808))

colnames(temp) = "isSpam"

bl = cbind(bl,temp)

b = rbind(bl,t)

scale\_b = scale(b[-30])

dists\_b = dist(scale\_b,"manhattan")

dists\_b = as.matrix(dists\_b)

ord.mat\_b = apply(dists\_b , 2, order)

k = 2

p = knn\_test1(1:808,ord.mat\_b , k , b)

table(p)

#classification tree

table(classification(trainVariables = t,testVariables = bl))