Project part 3 Report

Yilun Yang

1. Applying KNN algorithm

With our code, we can give a set of k and a set of distance to get the best combination with biggest correct rate. Although not specified here, we can still explore how number of cross-validation groups and more distance types affect the correct rate. In the following part, we use 5 as the number of CV groups. The combination table is showed below:

	1	2	3	4	5
manhattan	0.9483255	0.9341074	0.9409893	0.9353302	0.9353320
euclidean	0.9423634	0.9270760	0.9344128	0.9298266	0.9302859
maximum	0.9350258	0.9241702	0.9278382	0.9206549	0.9230993
minkowski	0.9446550	0.9322727	0.9333422	0.9287561	0.9287587
minkowski	0.9409881	0.9238665	0.9299803	0.9260053	0.9293687

From the table above, k=1 and distance="Manhattan" is the best pair of all with correct rate near 95%. In the following part, we will use this combination to analyze.

Training data validation:

1. Confusion matrix:

   We may get the confusion matrix in each cross-validation and sum up as the total confusion matrix.

orderk \ trueValue	FALSE	TRUE
FALSE	4731	203
TRUE	133	1474

The correct rate is 0.948326. Type I and II errors are 0.0310 and 0.0203, they are nearly equal and not very much.

2. Exploration of misclassified observations.

   There are 336 misclassified emails. Here I'd like to use two approaches to explore the misclassified observations. First way is to draw density plot for numeric variables and frequency table for logical variable and observe if there is significant difference between misclassified emails and correctly classified emails. Misclassification may also be caused by incorrectly substituting NA values in our code, this will be discussed as our second approach.

   First approach:

   By comparing 13 numerical plots, it seems no significant difference between misclassified and correctly classified observations within each variable. Now we look into logical variables.

```
facind: FALSE
              isRe          replyUnderline                priority
        0.31845238              0.01190476              0.00297619
       isInReplyTo        sortedRecipients subjectPunctuationCheck
        0.27678571              0.91071429              0.03571429
     multipartText          containsImages             isPGPsigned
        0.02380952              0.00000000              0.02380952
  subjectSpamWords  messageIdHasNoHostname          fromNumericEnd
        0.04464286              0.00297619              0.11309524
         isYelling       isOriginalMessage                  isDear
        0.01785714              0.03273810              0.00297619
           isWrote                  isSpam
        0.18452381              0.24107143
------------------------------------------------------------
facind: TRUE
              isRe          replyUnderline                priority
       0.319419823             0.013215149             0.006446414
       isInReplyTo        sortedRecipients subjectPunctuationCheck
       0.299597099             0.900725222             0.028364222
     multipartText          containsImages             isPGPsigned
       0.021112006             0.001450443             0.019016922
  subjectSpamWords  messageIdHasNoHostname          fromNumericEnd
       0.053988719             0.003223207             0.119742143
         isYelling       isOriginalMessage                  isDear
       0.023045931             0.039000806             0.008380338
           isWrote                  isSpam
       0.196615633             0.257211926
```

In the table, numbers represent the ratio of TRUE values, we denote “significant” if the ratio in one class is three times of the other class.

For the logical variables, containImages, isDear seem to be significant different in two groups. We can get the conclusion if an email doesn’t contain any image, doesn’t use word “dear”, it tends to be misclassified. We also observe isSpam are nearly the same in both class which means HAM and SPAM are nearly equiprobable to be misclassified to the opposite side in KNN.

Second Approach:
Misclassified emails that have at least one missing value in their original training data takes about 30 percent of total misclassified number. That is not a really big number so that it’s not safe to say missing value has big influence on misclassification.

Test data prediction:

1. Prediction confusion matrix is

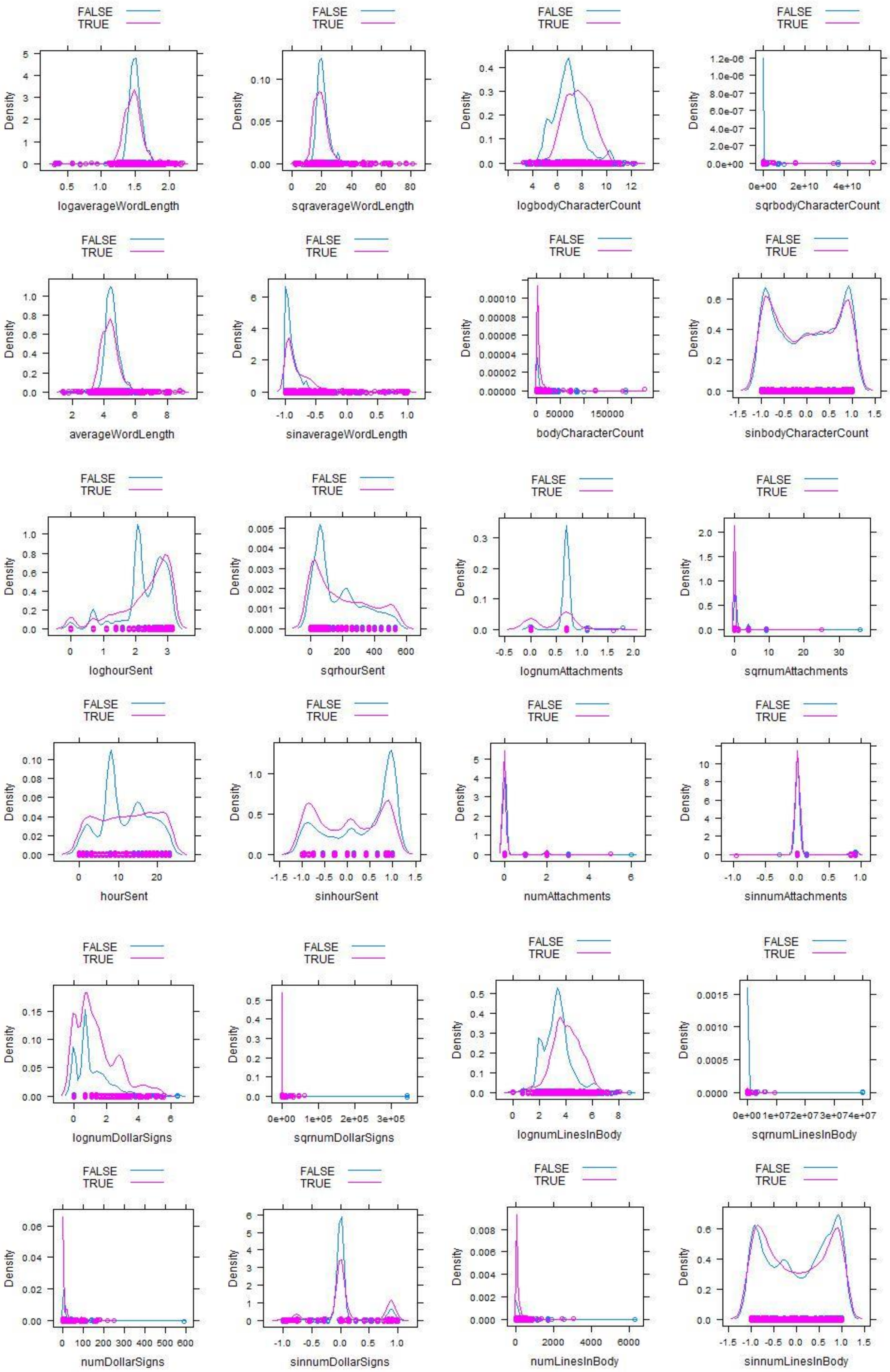
```
       trueValue
orderk  FALSE TRUE
  FALSE  1472   54
  TRUE     39  435
```

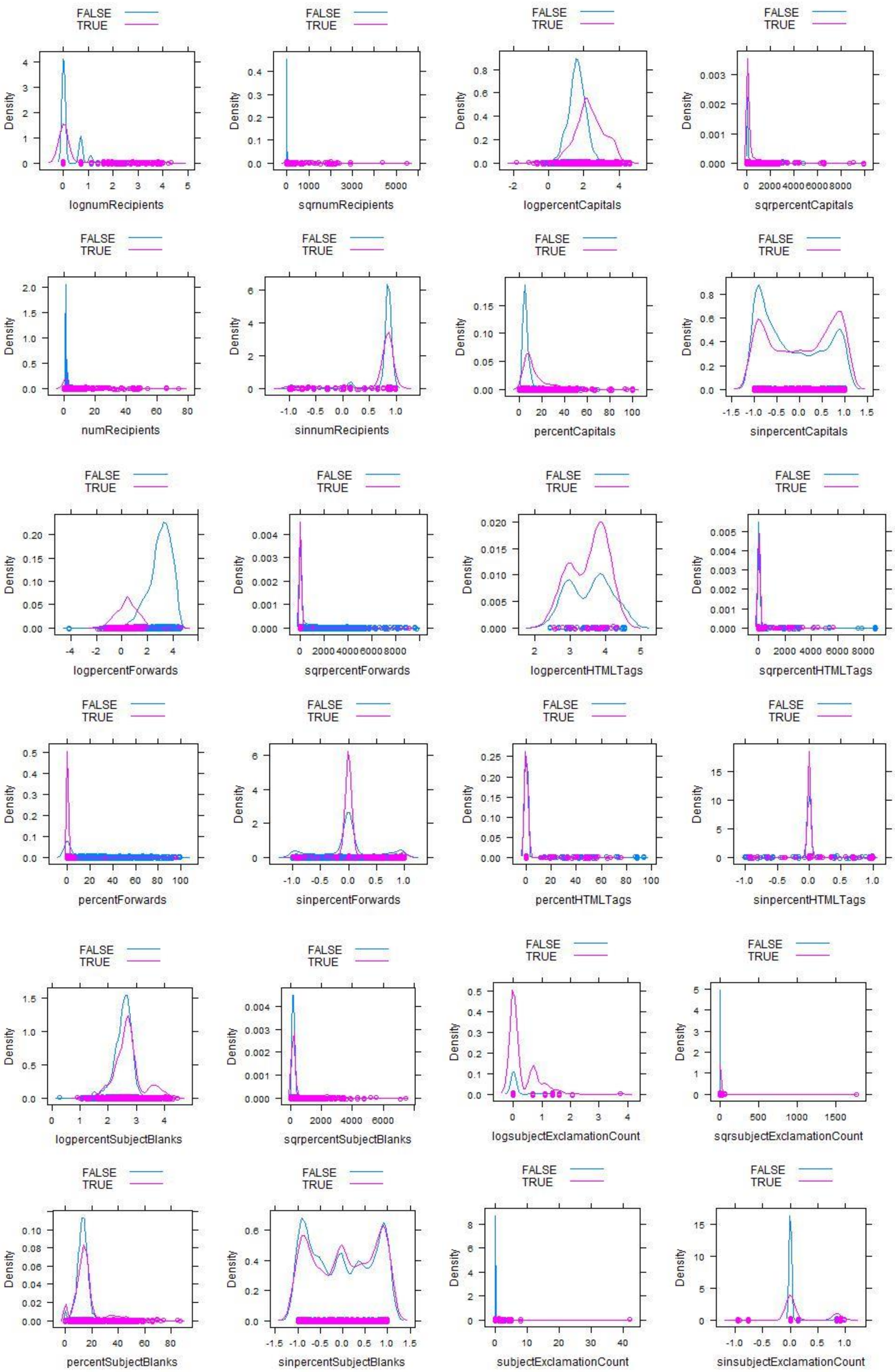
The correct rate is 0.9535. Type I and II errors are 0.027 and 0.0195. They are nearly equal and not very big numbers which indicate it is a good model for prediction.

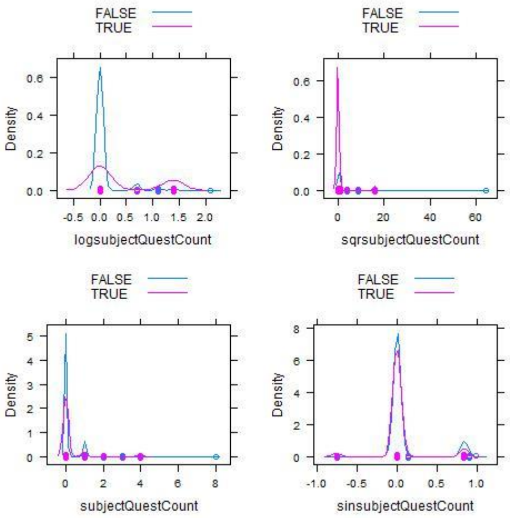
2. Prediction correct rate is quite similar with the training data correct rate.
3. Comment: This similarity gives us a strong confidence they have similar characteristics and seem to come from the same population. Besides, the high correct rate in prediction new observations shows us this is a good model.

2. Applying Classification Tree Algorithm

There are 13 numeric variables, I applied three transformations—log, square and sin to each of them. Density plots of all variables are given above.

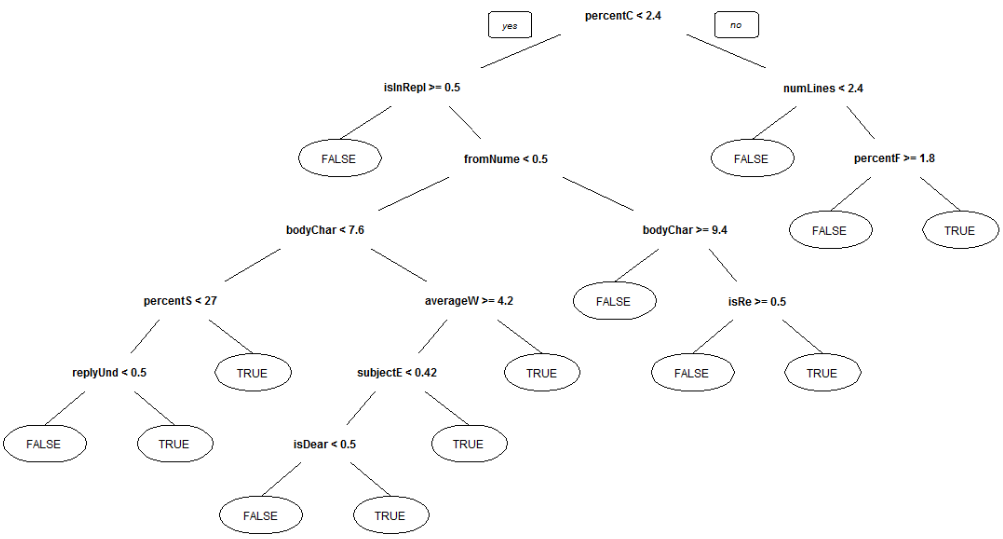



FALSE
TRUE
Density
lognumRecipients
sqrnumRecipients
logpercentCapitals
sqrpercentCapitals
numRecipients
sinnumRecipients
percentCapitals
sinpercentCapitals
logpercentForwards
sqrpercentForwards
logpercentHTMLTags
sqrpercentHTMLTags
percentForwards
sinpercentForwards
percentHTMLTags
sinpercentHTMLTags
logpercentSubjectBlanks
sqrpercentSubjectBlanks
logsubjectExclamationCount
sqrsubjectExclamationCount
percentSubjectBlanks
sinpercentSubjectBlanks
subjectExclamationCount
sinsubjectExclamationCount



For each of them, we want to pick out the best form of transformation that TRUE FLASE density plots overlap least. By comparing, these variables are selected to be averageWordLength, log (bodyCharacterCount), hourSent, log(numAttachments), log(numLinesInBody), numRecipients, log(percentCapitals), log(percentForwards), log(percentHTMLTags), percentSubjectBlanks, sin(subjectExclamationCount), log(subjectQuestCount), numDollarSigns，although transformation for averageWordLength, percentHTMLTags, percentSubjectBlanks, numDollarSigns are not significant at all.

We then fit the classification tree with the transformed variables above and plot the tree. This tree has 13 variables including 8 numeric variables and 5 logical variables.



Training data validation:

1. Confusion matrix:

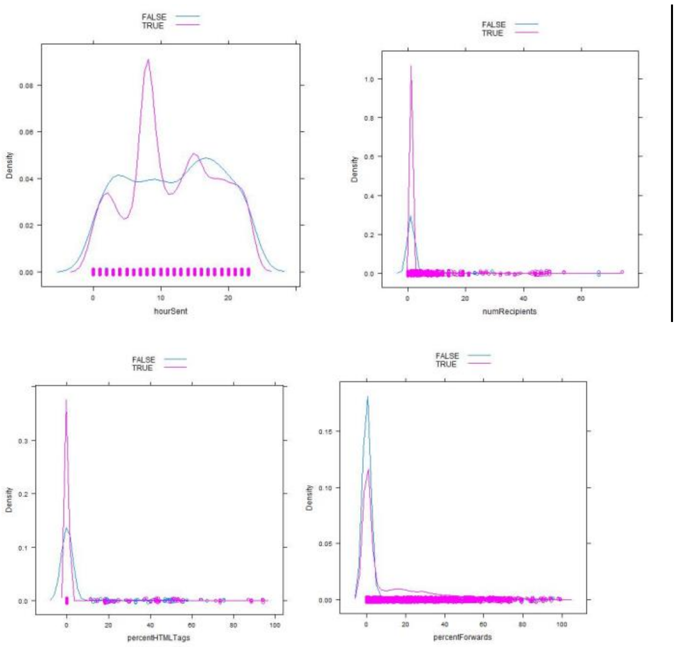
```
         truevalue
predicted FALSE TRUE
    FALSE  4703  455
```

```
    TRUE      161 1222
```

Correct rate is 0.9058248, though it may not be as good as that in KNN algorithm, it is still a very high correction rate. Type I and II errors are 0.0696 and 0.0246. They are not equal and Type I error is the second biggest among all error rates in confusion matrix. It indicates that we are not much confident to classify an email predicted to be HAM as really HAM as KNN algorithm.

2. Exploration of the misclassified observations:
   In classification tree algorithm, there are 616 misclassified observations. I will apply the same approaches as KNN to analysis.

   First Approach:



For numerical variables, we have four interesting variables to analyze. If an email's hourSent is more convergent to 8-9 (all the numbers are estimated), numRecipients to 1-2, percentHTMLtags to 0, it is more likely to be correctly classified. On the contrary, if an email's percentForwards is more convergent to 0, it tends to be mis-classified!

```
facind: FALSE
                isRe          replyUnderline                priority
         0.061688312             0.009740260             0.017857143
         isInReplyTo        sortedRecipients subjectPunctuationCheck
         0.014610390             0.909090909             0.051948052
       multipartText          containsImages             isPGPsigned
         0.073051948             0.009740260             0.003246753
    subjectSpamWords  messageIdHasNoHostname          fromNumericEnd
         0.113636364             0.006493506             0.142857143
           isYelling       isOriginalMessage                  isDear
         0.058441558             0.022727273             0.024350649
             isWrote                  isSpam
         0.032467532             0.738636364
--------------------------------------------------------------
facind: TRUE
                isRe          replyUnderline                priority
        0.3461603376            0.0135021097            0.0050632911
         isInReplyTo        sortedRecipients subjectPunctuationCheck
        0.3279324895            0.9004219409            0.0263291139
       multipartText          containsImages             isPGPsigned
        0.0158649789            0.0005063291            0.0209282700
    subjectSpamWords  messageIdHasNoHostname          fromNumericEnd
        0.0472573840            0.0028691983            0.1169620253
           isYelling       isOriginalMessage                  isDear
        0.0190717300            0.0403375527            0.0064135021
             isWrote                  isSpam
        0.2129957806            0.2062447257
```

For logical variables, if an email doesn't have "Re" in subject, has a high priority, has multipart-text, contains images, is not PGP signed, doesn't have spam words, its subject is in capital letters, has "dear", doesn't have word wrote, it tends to be mis-classified. isSpam in the two classes are very different which means SPAM are more likely to be misclassified than HAM with this algorithm.

Second Approach:
Classification Tree missing value ratio is 0.224026. For the same reason as mentioned in KNN algorithm, missing value also doesn't have big influence on misclassification.

Test data prediction:
1. Prediction confusion matrix is

```
         truevalue
predicted FALSE TRUE
    FALSE  1449  181
    TRUE     62  308
```

The correct rate is 0.8785 which is a little bit different from training data correct rate. Type I and II errors are 0.0905 and 0.031. They are not equal and Type I error is the biggest among all error rates in confusion matrix. It has proved the conclusion we got in training data confusion matrix. It may not be as good as the KNN model for prediction.

2. Prediction correct rate is a little bit different with the training data correct rate.
3. Comment: Although with some difference between training data correct rate and test data correct rate, this evidence is not strong enough to reject the assumption that the two data set come from the same population. Besides, although the prediction result is not as good as KNN algorithm, it is still a good way.

Comparison:
On the whole, from correct rate, both of the algorithms are pretty good, although KNN is better. However, from other perspectives, things may be different. From the analysis above, we have little knowledge why some observations in KNN method are misclassified thus it is really difficult to improve the model. There may still be other influential variables not considered in our model. In classification tree, although we have some information in what situations some observations tend to be misclassified, the real reason of misclassification would be hard to get because there are so many potential variables to explore. Improvement of this model would be costly. Besides, the imbalance of misclassification in HAM and SPAM in this method increases our doubt on the validity of this model. I would say both of the model are not satisfying enough.

Compare the behavior of the classifiers:
1.
By simple revision of code and calculation, we find there are 93 misclassification in the test data with KNN algorithm and 243 misclassification in the test data with classification tree algorithm. Among them, 58 observations are both misclassified, they are:

```
[1]   61   70  285  326  386  581  582  584  615  629  854  9
10 1133 1167 1216
```

```
[16] 1233 1347 1449 1503 1532 1536 1567 1572 1586 1660 1676
 1684 1685 1687 1718
[31] 1728 1733 1734 1741 1742 1743 1766 1772 1815 1825 1828
 1854 1872 1877 1881
[46] 1885 1894 1918 1922 1937 1947 1957 1959 1981 1986 1993
 1994 2000
```

Only KNN misclassified:

```
[1]  244  365  426  469  593  692  717  743  791  831  961 11
30 1187 1213 1252
[16] 1255 1298 1318 1319 1328 1702 1717 1725 1762 1764 1818
 1835 1845 1847 1871
[31] 1886 1910 1920 1923 1929
```

Only Classification misclassified:

```
[1]   24   43   53   58  194  197  250  300  364  437  462  53
4  574  616  708
 [16]  709  756  796  812  861  885  913 1086 1108 1116 1170
 1248 1249 1309 1322
 [31] 1404 1417 1446 1457 1460 1466 1476 1482 1483 1488 149
7 1507 1510 1513 1518
 [46] 1524 1525 1526 1527 1528 1535 1537 1538 1540 1542 155
0 1551 1554 1556 1557
 [61] 1558 1560 1562 1563 1565 1566 1569 1571 1575 1577 157
8 1580 1583 1584 1585
 [76] 1587 1592 1595 1596 1598 1600 1602 1604 1608 1610 161
2 1618 1620 1621 1627
 [91] 1629 1631 1632 1633 1634 1639 1643 1645 1648 1658 165
9 1661 1664 1666 1667
[106] 1668 1671 1672 1673 1686 1689 1690 1698 1700 1701 170
8 1711 1713 1716 1730
[121] 1735 1744 1745 1747 1749 1750 1752 1756 1758 1761 176
7 1774 1776 1777 1778
[136] 1785 1790 1791 1792 1794 1796 1797 1799 1806 1813 181
7 1821 1823 1824 1826
[151] 1838 1840 1841 1844 1848 1852 1857 1862 1869 1874 188
3 1884 1887 1890 1901
[166] 1905 1914 1916 1919 1924 1928 1938 1944 1945 1946 194
9 1953 1960 1963 1967
[181] 1969 1970 1974 1976 1996.
```

Apart from these emails, all are correctly classified by both of the algorithms.

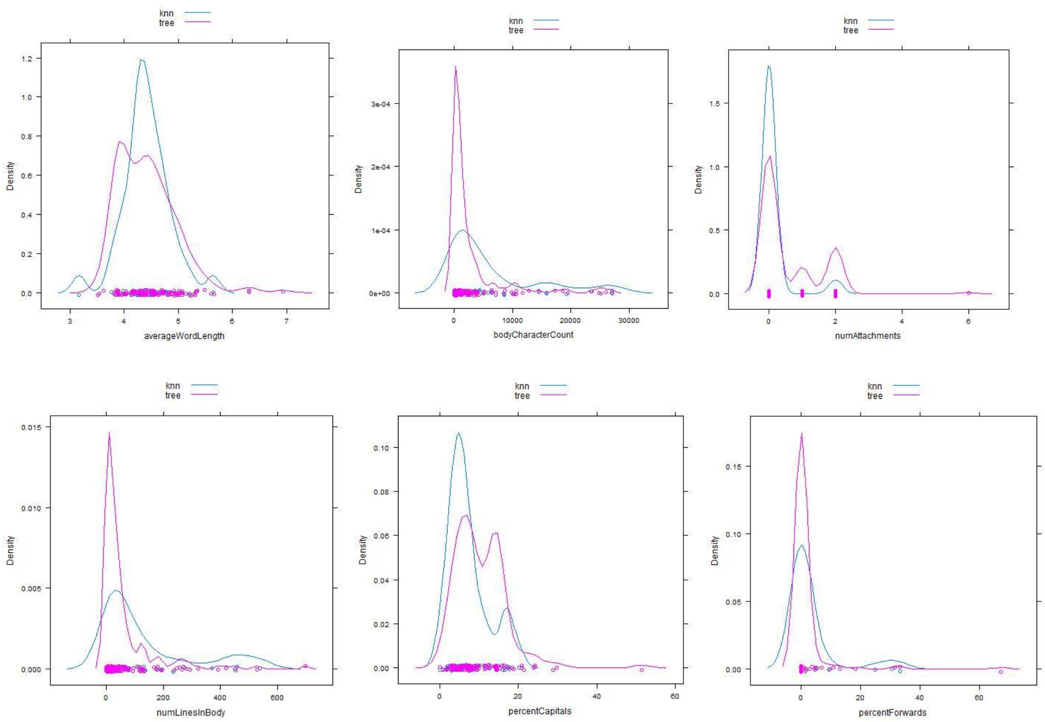
2.

We also apply the approaches to two types of misclassification for different variables.

Logical variables:

```
factest: knn
                isRe            replyUnderline                  priority
          0.02857143                0.00000000                0.02857143
          isInReplyTo         sortedRecipients subjectPunctuationCheck
          0.02857143                0.88571429                0.02857143
        multipartText            containsImages               isPGPsigned
          0.05714286                0.00000000                0.00000000
     subjectSpamWords   messageIdHasNoHostname           fromNumericEnd
          0.00000000                0.00000000                0.14285714
            isYelling         isOriginalMessage                    isDear
          0.05714286                0.00000000                0.00000000
              isWrote                    isSpam
          0.02857143                0.42857143
--------------------------------------------------------------
factest: tree
                isRe            replyUnderline                  priority
         0.075675676               0.010810811               0.005405405
          isInReplyTo         sortedRecipients subjectPunctuationCheck
         0.021621622               0.913513514               0.043243243
        multipartText            containsImages               isPGPsigned
         0.286486486               0.021621622               0.000000000
     subjectSpamWords   messageIdHasNoHostname           fromNumericEnd
         0.178378378               0.016216216               0.183783784
            isYelling         isOriginalMessage                    isDear
         0.048648649               0.005405405               0.048648649
              isWrote                    isSpam
         0.032432432               0.767567568
```

In knn method, many variables' ratio equal to zero. Lacking of enough observation number (only 35) may be a good explanation. However, it may also prove that knn is weak in classifying emails with such variables to be FALSE. In tree method there is also a variable has ratio 0—isPGPsigned which means classification tree lacks the ability to handle emails without PGP signing. In other words, when we meet the emails with such variables as FALSE, we should use the other algorithm such that we should use classification tree to classify emails with no images (for emails without PGP signing, we probably need to find the third way). For other variables, priority, multiparttext seem to have big difference between two methods. Thus we get the conclusion if an email has high priority, we should apply classification tree and if it has multipart text, we should use knn respectively.



For numeric variables, six variables seem to be significantly different. By comparing, we may get the conclusion if an email has average word length converging to 4-5, number of attachments to 0, percent capitals to 0-10, the mis-classification density for knn is bigger thus we should apply classification tree. If an email has body characters converging to 0-2000, body lines number to 0-

100, percent forwards to 0-5, we should use knn respectively.

Appendix:

```
#This is the code for KNN algorithm
kNNclassify <- function (trainingdata, testdata, k, dist = "euclidean", p = 2, idx){
  #trainingdata                                    trainingdata
  #testdata                                        testdata
  #k                                               number of the nearest neighbors
  #dist                                            distance method to choose
  #p                                               The power of the Minkowski distance
  #idx                                             The observation index of testdata, used to find
misclassified index
  #apply majority voting method for each observation

  #Separate True values from training and test data
  isSpamtrain <- trainingdata$isSpam
  trainingdata <- trainingdata[,-which(names(trainingdata) == "isSpam")]
  isSpamtest <- testdata$isSpam
  testdata <- testdata[,-which(names(testdata) == "isSpam")]

  #Handle missing values
  trainingdata <- as.data.frame(sapply(trainingdata, nasub))
  testdata <- as.data.frame(sapply(testdata, nasub))

  #Rescale all variables
  trainingdata <- scale(trainingdata)
  testdata <- scale(testdata)
```

```
    trainrow <- nrow(trainingdata)
    testrow <- nrow(testdata)

    distance <- dist(rbind(trainingdata, testdata), method = dist, p = p)
    distmat <- as.matrix(distance)[(trainrow + 1):(trainrow + testrow), 1:trainrow]
    orderk <- apply(distmat, 1, order)[1:k,]

    if(k == 1){

    #Only the nearest 1 points
    predspam <- sapply(as.data.frame(orderk), function(x) isSpamtrain[x])
    }

    else{

    #K nearest points
    Spamlist <- sapply(as.data.frame(orderk), function(x) table(isSpamtrain[x]))
    predspam <- sapply(Spamlist, function(x) names(x)[which.is.max(x)])
    predspam <- as.logical(predspam)
    }

    correctrate <- mean(predspam == isSpamtest)
    confusionmat <- table(data.frame(predicted=predspam,trueValue=isSpamtest))

    print(idx[which(predspam != isSpamtest)])
    print(confusionmat)
    correctrate

}

#Cross-Validation code
CVclassify<-function(trainVariables, grp, k, dist = "euclidean", p = 2){
    #trainVariables          training dataset
    #grp                     number of groups in the cross validation
    #k                       number of the nearest neighbors
    #dist                    distance method to choose
    #p                       The power of the Minkowski distance

    #Permute observations
    sampleind <- sample(nrow(trainVariables), nrow(trainVariables), replace = FALSE)
    trainVariables <- trainVariables[sampleind,]

    #divide training dataset into given groups, remainder obervations will be allocated to former
```

groups randomly.

```
    grpnum <- nrow(trainVariables)%/%grp
    leftnum <- nrow(trainVariables)%%grp
    grpind <- sample(grp, leftnum, replace = FALSE)
    factor <- as.factor(c(gl(grp, grpnum), grpind))
    CVlist <- split(trainVariables, factor)
    idx <- split(sampleind, factor)

    #For each group, apply KNN algorithm
    correctvec <- vector(length = grp)
    for(i in 1:grp){
        testdata <- CVlist[[i]]
        trainingdata <- do.call(rbind, CVlist[-i])
        correctvec[i] <- kNNclassify(trainingdata, testdata, k = k, dist = dist, p = p, idx = idx[[i]])

    }

    #Get the mean correctrate as final correctrate
    meancorrect <- mean(correctvec)
    meancorrect
}


#Test for different k and distance method to get the best cv correcrrate
testkdist <- function(trainVariables, grp, k, dist){

    #trainVariables                          training dataset
    #grp                                        number of groups in the cross validation
    #k                                          number of the nearest neighbors
    #dist                                       A MATRIX! With first column as chosen
distance, senond column as minkowski power

    nk <-length(k)
    ndist <- nrow(dist)
    parmat <- matrix(ncol = nk, nrow = ndist)
    for(i in 1:ndist){
        for(j in 1:nk){
            parmat[i,j] <- CVclassify(trainVariables, grp = grp, k = k[j], dist = dist[i,][1], p =
as.numeric(dist[i,][2]))
        }
    }
    rownames(parmat) <- dist[, 1]
    colnames(parmat) <- as.character(k)
```

```
  parmat
}

#A function to replace missing values.
nasub<- function (x){
  if(is.logical(x[1])){

    #Missing values will be replaced by which occurs the most.
    x[which(is.na(x))] <- as.logical(names(which.max(table(x))))
  }
  else{

    #Missing values will be replaced by mean of the whole variable.
    x[which(is.na(x))] <- mean(x, na.rm = TRUE)
  }
  x
}

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

#Decide which distance k combination shall we choose.
 k <- 1:5
 dist <- matrix(c("manhattan", "euclidean", "maximum", "minkowski", "minkowski", 2, 2, 2, 3, 4),
ncol = 2)
 testkdist(trainVariables, 5, k, dist)

#By comparison, I decide to use k=1 dist="manhattan" to fit the model.
#Training data correctrate and confusion matrix
 traincorrectrate<-CVclassify(trainVariables,5,1,dist="manhattan")

#Test data with actual results correctrate and confusion matrix
 predcorrectrate<-kNNclassify(trainVariables,testVariables,1,dist="manhattan")

#Explore missclassified observations in training data
#Misclassification index is not listed here, it is in the code!

#Plot density plot of each numeric variable for right classified and misclassified obs.
facind <- ifelse(1:nrow(trainVariables) %in% KNNmisind, FALSE, TRUE)
library("lattice")
islogic <- sapply(trainVariables[1,], is.logical)
trainVarnum <- trainVariables[,!islogic]
trainVarlog <- trainVariables[,islogic]

for(i in names(trainVarnum)){
```

```
    trainVarnum$facind <- facind
    ind <- which(names(trainVarnum) == i)
    jpeg(file = paste(i, ".jpg", sep = ""))
    plot <- densityplot( ~trainVarnum[,ind], trainVarnum, group = facind, auto.key = TRUE, xlab = i)
    print(plot)
    dev.off()

}

#get table of each logical variable for right classified and misclassified obs.
tblist <- by(as.data.frame(sapply(trainVarlog, nasub)), facind, function(x) sapply(x, mean))

#Missing value influence
missingind <- which(is.na(apply(as.matrix(trainVariables), 1, mean)))
ratio <- mean(KNNmisind %in% missingind)


#This is the code for classification tree algorithm

library("lattice")

#get numeric variables
islogic <- sapply(trainVariables[1,], is.logical)
trainVarnum <- trainVariables[, !islogic]

#emploration of variable transformation by plotting
for(i in names(trainVarnum)){
    trainVarnum$isSpam <- trainVariables$isSpam
    ind <- which(names(trainVarnum) == i)
    jpeg(file = paste(i, ".jpg", sep = ""))

    plot1 <- densityplot( ~trainVarnum[,ind], trainVarnum, group = isSpam, auto.key = TRUE, xlab =
i)
    plot2 <- densityplot( ~log(trainVarnum[,ind]), trainVarnum, group = isSpam, auto.key = TRUE,
xlab = paste("log", i, sep = ""))
    plot3 <- densityplot( ~(trainVarnum[,ind])^2, trainVarnum, group = isSpam, auto.key = TRUE,
xlab = paste("sqr", i, sep = ""))
    plot4 <- densityplot( ~sin(trainVarnum[,ind]), trainVarnum, group = isSpam, auto.key = TRUE,
xlab = paste("sin", i, sep = ""))
    print(plot1, position = c(0, 0, 1/2, 1/2), more = TRUE)
    print(plot2, position = c(0, 1/2, 1/2, 1), more = TRUE)
    print(plot3, position = c(1/2, 1/2, 1, 1), more = TRUE)
    print(plot4, position = c(1/2, 0, 1, 1/2))
    dev.off()
```

```
}

##By comparision, apply transformed variables to fit the classification tree

library("rpart")
library("rpart.plot")
trainVariables$bodyCharacterCount <- log(trainVariables$bodyCharacterCount)
trainVariables$numAttachments <- log(trainVariables$numAttachments)
trainVariables$numLinesInBody <- log(trainVariables$numLinesInBody)
trainVariables$percentCapitals <- log(trainVariables$percentCapitals)
trainVariables$subjectExclamationCount <- sin(trainVariables$subjectExclamationCount)
trainVariables$percentForwards <- log(trainVariables$percentForwards)
trainVariables$subjectQuestCount <- log(trainVariables$subjectQuestCount)
trainVariables$percentHTMLTags <- log(trainVariables$percentHTMLTags)

#Fitted the model
classtree <- rpart(isSpam~., data = trainVariables, method = "class")
rpart.plot(classtree)

#Training data correctrate and confusion matrix
predicted <- predict(classtree, newdata = subset(trainVariables, select = -isSpam), type = "class")
traincorrectrate <- mean(predicted == trainVariables$isSpam)
trainconfusionmat <- table(data.frame(predicted = predicted, truevalue = trainVariables$isSpam))

#Test data with actual results correctrate and confusion matrix
testVariables$bodyCharacterCount <- log(testVariables$bodyCharacterCount)
testVariables$numAttachments <- log(testVariables$numAttachments)
testVariables$numLinesInBody <- log(testVariables$numLinesInBody)
testVariables$percentCapitals <- log(testVariables$percentCapitals)
testVariables$subjectExclamationCount <- sin(testVariables$subjectExclamationCount)
testVariables$percentForwards <- log(testVariables$percentForwards)
testVariables$subjectQuestCount <- log(testVariables$subjectQuestCount)
testVariables$percentHTMLTags <- log(testVariables$percentHTMLTags)
predcorrectrate <- mean(predict(classtree, newdata = subset(testVariables, select = -isSpam), type
= "class") == testVariables$isSpam)
testconfusionmat <- table(data.frame(predicted = predict(classtree, newdata =
subset(testVariables, select = -isSpam), type = "class"), truevalue = testVariables$isSpam))

#Explore missclassified observations in training data
#misclassification index
treemisind <- which(predicted != trainVariables$isSpam)

#Plot density plot of each numeric variable for right classified and misclassified obs.
```

```
#Be aware to use original not transformed trainVariables.
facind <- ifelse(1:nrow(trainVariables) %in% treemisind, FALSE, TRUE)
library("lattice")
islogic <- sapply(trainVariables[1,], is.logical)
trainVarnum <- trainVariables[, !islogic]
trainVarlog <- trainVariables[, islogic]

for(i in names(trainVarnum)){
   trainVarnum$facind <- facind
   ind <- which(names(trainVarnum) == i)
   jpeg(file = paste(i, ".jpg", sep = ""))
   plot <- densityplot( ~trainVarnum[,ind], trainVarnum, group = facind, auto.key = TRUE, xlab = i)
   print(plot)
   dev.off()

}

#get table of each logical variable for right classified and misclassified obs.
tblist <- by(as.data.frame(sapply(trainVarlog, nasub)), facind, function(x) sapply(x, mean))

#Missing value influence
missingind <- which(is.na(apply(as.matrix(trainVariables), 1, mean)))
ratio <- mean(treemisind %in% missingind)


#Explore the residuals
#knnmis and treemis index are not listed here, they are in the code!

test<-testVariables[c(knnmis,treemis),]
factest<-c(rep("knn",length(knnmis)),rep("tree",length(treemis)))

#Plot density plot of each numeric variable for two types of misclassification.
library("lattice")
islogic <- sapply(test[1,], is.logical)
testVarnum <- test[, !islogic]
testVarlog <- test[, islogic]

for(i in names(testVarnum)){
   testVarnum$factest <- factest
   ind <- which(names(testVarnum) == i)
   jpeg(file = paste(i, ".jpg", sep = ""))
   plot <- densityplot( ~testVarnum[,ind], testVarnum, group = factest, auto.key = TRUE, xlab = i)
   print(plot)
   dev.off()
```

```
}

#get table of each logical variable for two types of misclassification.
tblist <- by(as.data.frame(sapply(testVarlog, nasub)), factest, function(x) sapply(x, mean))
```