Final Project: Spam Filter Analysis

CSZ

December 7, 2018

Group Member Info

Group Name: CSZ

Members:

- Shuoqi Zhang (shuoqiz2)
- Yunan Shi (shi38)
- Chuanqi Chen (chuanqi2)

Introduction and literature review

Data source information:

- Source: created in the late 1990s at Hewlett-Packard Labs, contains 4601 emails.
- Link: spam

Introduction of the data:

This collection of spam e-mails came from postmaster and individuals who had filed spam while non-spam e-mails came from filed work and personal e-mails. These are useful when constructing a personalized spam filter. There are 58 variables in total, and this spam dataset contains 4601 emails, of which 1813 are considered spam while the remaining are not spam.

Attribute information:

- 1-48. frequency of the variable name.
  - If the variable name starts with num (e.g., num650), it indicate the frequency of the corresponding number (e.g., 650); if the variable name doesn't start with num, it indicates frequency of regular word(e.g., business). Frequency is the percentage of words in the email that match the word/number
- 49-54 frequency of the characters
  - ';', '(', '[', '!', '\$', and '\#'. E.g., charDollar. Frequency is the percentage of characters in the email that match the word/number.
- 55. CapitalAve: average length of uninterrupted sequences of capital letters.
- 56. CapitalLong: longest length of uninterrupted sequence of capital letters.
- 57. CapitalLength: total length of uninterrupted sequence of capital letters
- 58. type: "non-spam" or "spam"

Literature Review:

We found several related analyses with the dataset which are all accessible on Google. Most of the topics of those analysis reports are the classification of spam emails with various machine learning methods. For instance, one of the analysis we found mainly focuses on applying different random forest models like GBM and OOB to classification. The general conclusion from analysis reports is that Random Forest is perfect model with least misclassification rate. Data accuracy is around 94.5%. In addition, character Exclamation and Dollar are two main effects determining the category of emails.

Scientific Goal:

- Constructing Spam filter mainly by SVM, Logistic regression and RandomForest
- Test the performance of KNN, LDA and CART Model
- Identifying the variables that contribute to the classification
- Comparing the quality of Spam filter for each Model, to choose the best classifier
- Tuning the classifier decision rule to make it more sensible for a Spam filter since the harm of Type I error and Type II error are different

Import Packages

```
library("DataExplorer")
library("e1071")
library("kernlab")
library("caret")
library("class")
library("MASS")
library("tree")
library("randomForest")
library("ggplot2")
library("corrplot")
library("bubbles")
```

Summary statistics and Data Visualization

Import the data

```
data("spam")
dim(spam)
```

```
## [1] 4601   58
```

```
summary(spam$type)
```

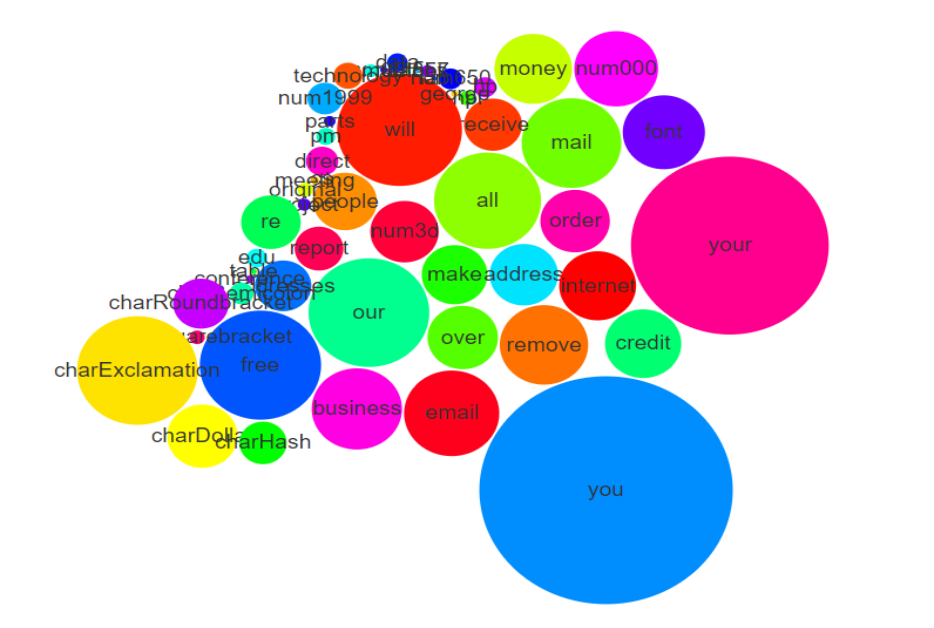
```
## nonspam    spam
```

```
##    2788    1813
```

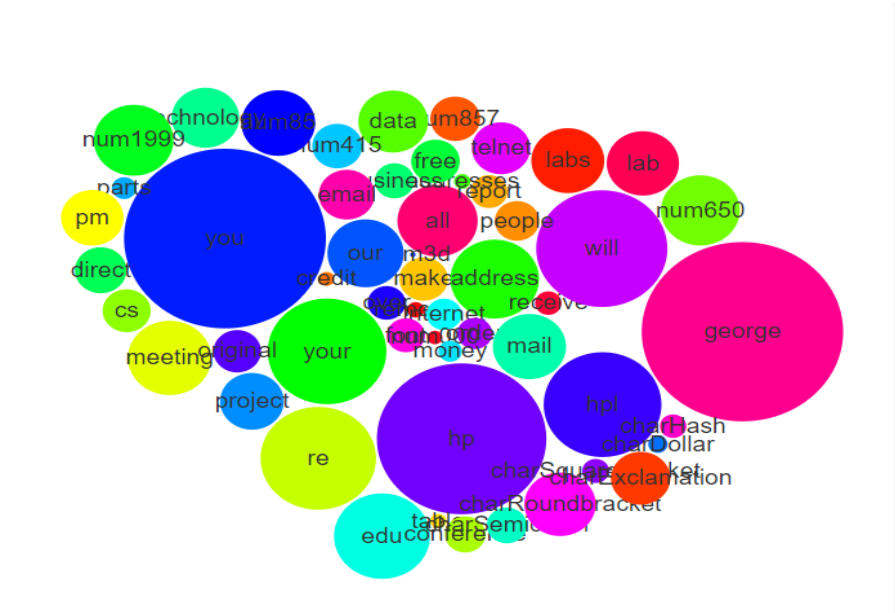
There is no missing value in the data. Response variable is a category variable with spam and non-spam. The data includes 2788 nonspam email and 1813 spam emails.

Show the average proportion of each Character or Number in the emails of spam and nonspam separately

```
nonspam = sapply(spam[spam$type == "nonspam",], as.numeric )
onlyspam = sapply(spam[spam$type == "spam",], as.numeric )
mean_spam = colMeans(onlyspam)
mean_nonspam = colMeans(nonspam)
bubbles(value = mean_spam[1:54], label = colnames(spam)[1:54],
        color = rainbow(54, alpha=NULL)[sample(54)])
```



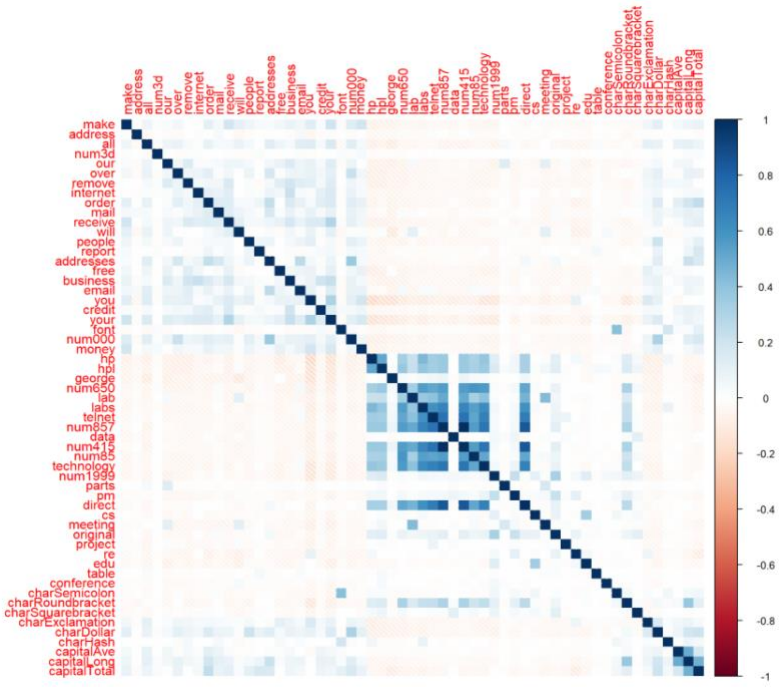

```
bubbles(value = mean_nonspam[1:54], label = colnames(spam)[1:54],
        color = rainbow(54, alpha=NULL)[sample(54)])
```



From the above two graphs, we omit the capital letters, and conclude several apparent statistical conclusions with words, numbers and characters. In a spam email, the proportion of occurrence of character like ‘you’, ‘your’, ‘will’, ‘free’ ,‘our’ and charExclamation (!) are very high which means emails with these words and character (exclamation) appearance or repeatedly appearance are likely to be spam email. For a non-spam email, frequency of ‘you’, ‘george’, ‘hp’, “hpl’,”re" and “will” is very high. In the same way, the occurrence of these words in emails are also likely categorized the emails as non-spam one. Numbers are not that obvious compared with words.

Correlation Check

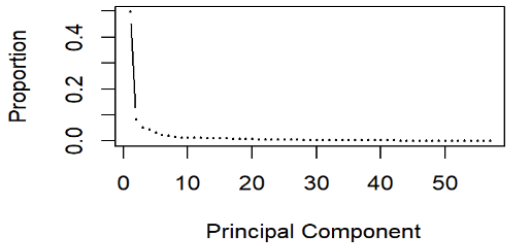
```
corrplot( cor(spam[,-58]), method = "shade")
```



From the graph we can observe clear positive pattern (blue squares) comparing with negative pattern (red squares). The variables which show in the spam email or non-spam email in the same time will have a positive correlation; variables which show in the spam email and non-spam email in different time will have a negative correlation. Most of the variables have no or lower correlation, Therefore, collinear won't be a huge problem if we use model of LDA and Logistic Regression to complete our goal.

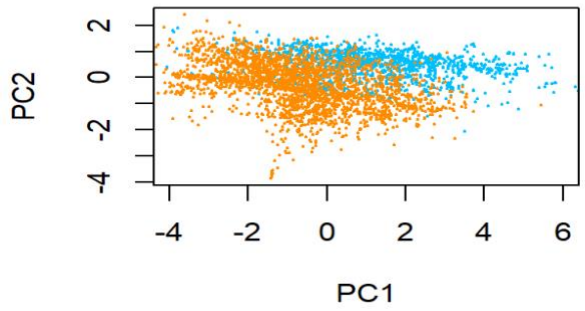
PCA to visualize the whole dataset in low dimension

```
# PCA
pr.out = prcomp(log(spam[,-58]+1))
pr.var = pr.out$sdev^2
pve = pr.var/sum(pr.var)
plot(pve , xlab=" Principal Component",
     ylab="Proportion", type="b", pch = 19, cex = 0.2)
```



From the line chart of *Proportion VS. Principal Component*, the PC1 accounts for about 50%, and pc2 accounts for 8%. Therefore, the 2D PCA visualization catches about 60% information of original dataset which is not high enough but still can help us to know the outline and tendency.

```
plot(pr.out$x[,1], pr.out$x[,2], col = c("darkorange", "deepskyblue")[spam$ty
pe], xlab = "PC1", ylab = "PC2", pch = 19, cex = 0.2, xlim = c(-4, 6))
```



The plot shows that there is a clear boundary that could classify the type which encourage us to perform the classification techniques on this dataset. In addition, for the first PC, the most dominant variable is **all** with 0.04092317, while for the second PC, the most dominant variable is **our** with 0.09414561.

Proposed Analysis

Our dataset has 57 variables we used as prediction variables corresponding to 57 dimensions which is somehow high dimension. Each cell value represents the proportion of that words in the total email, therefore the value is a decimal. Therefore, even though each observation is existing in a very high dimension, they will still be crowded or even overlapped in a small space.

For LDA, since LDA classify observations into class with the closest centroid in terms of Mahalanobis distance it may perform badly when data points are too crowded. The inverse of $\Sigma\Sigma$ also may not exist When p is very large. In addition, using the generalized inverse matrix can easily overfit the data. For KNN classification, which simply takes the mode of K nearest neighbor observations as the prediction for a specific observation. Narrow distance between two points would

be a disaster for the KNN model. Therefore, it would negatively affect the performance of KNN, LDA and other models.

In the case of this spam data, in order to control the test misclassification rate, SVM and Random Forest are our first choices since they are good at handle high-dimension problem. For SVM, it maximizes the soft margin which separates two class with different kernel methods. The SVM classifier only depends on the support vectors. Random Forest is an advanced Classification tree model with bootstrap and randomness. Random Forest exhaustively searches all the variables (dimensions) to maximize the Node Impurity Reduction. Therefore, even though our p is very large, it would only increase some computation but won't affect the performance.

Logistic Model is a very classical and useful Supervised Learning Technique. Therefore, we are going to first fit this linear model. The model selection techniques can also help us improve the model, by bias-variance trade-off, reduce the size, making it easier to interpret. We can also identify the significant variables which are also one of our Scientific Goal.

Split Data

Split the original dataset into train data and test data to train the model and then get test misclassification rate to compare each model

```
set.seed(1)

index = sample(nrow(spam), 3000)

spam_trn = spam[index, ]

spam_tst = spam[-index, ]
```

Logistic Regression

Based on the statistics summary and data visualization, we try to find out the most significant variables which may determine the category of email (spam or nonspam). Since logistics regression is a standard approach for binary classification and useful supervised learning technique, we use it as our first testing model.

Take Logistic Regression as Bayes Classifier Simply classify an observation to the class (spam/nonspam) with the larger probability. In those two cases, in order to minimize the misclassification rate, we take the cutoff = 0.5.

Perform the logistic model on all the predictors on the training dataset. Then predict the test data to get the confusion table and misclassification rate

```
logistic_model_full = glm(type ~ . , data = spam_trn, family = "binomial")

p_full = predict(logistic_model_full, newdata = spam_tst, type = "response")

pred_full = ifelse(p_full > 0.5, "spam", "nonspam")

(confusion_table_full = table(pred_full, spam_tst$type))
```

```
##
## pred_full nonspam spam
##   nonspam     929   74
```

```
##   spam         47  551
```

```
(miss_rate_full = mean(pred_full != spam_tst$type))
```

```
## [1] 0.07557776
```

Since we contain all the predictors in the model, it may cause overfitting in our logistics model. Therefore, it is necessary to do a model selection to choose a better model. Apply *AIC* standard and *backward* direction to perform the variable selection on the original full model.

```
# Perform the Variable Selection with backward AIC
logistic_model = step(logistic_model_full, trace = 0)
sort(summary(logistic_model)$coefficients[, "Pr(>|z|)"])[1:5]
```

```
##  (Intercept)         free   charDollar           hp  capitalLong
## 2.936340e-23 5.282440e-10 4.288508e-08 8.849538e-08 1.800423e-07
```

```
p = predict(logistic_model, newdata = spam_tst, type = "response")
# free, charDollar, hp, capitalLong is the four most significant variable
pred = ifelse(p > 0.5, "spam", "nonspam")
(conftable_logistic = table(pred, spam_tst$type))
```

```
##
## pred      nonspam spam
##   nonspam     934   74
##   spam         42  551
```

```
(missrate_logistic = mean(pred != spam_tst$type))
```

```
## [1] 0.07245472
```

After variable selection of full model, the misclassification rate lowered from 0.0755778 to 0.0724547 which shows out a better accuracy. The number of variables decreases a lot from 57 to 37 which make it easier for us to interpret.

In addition, from the test results, the most significant four variables are free, charDollar, hp, capitalLong.CharDollar is character “$”; capitalLong is the longest length of uninterrupted capital letters; free and hp are two words. Therefore, according to the logistic regression, we can conclude that word free and hp, “$” and the longest length of uninterrupted capital letters might be the important factors to determine an email is spam or not.

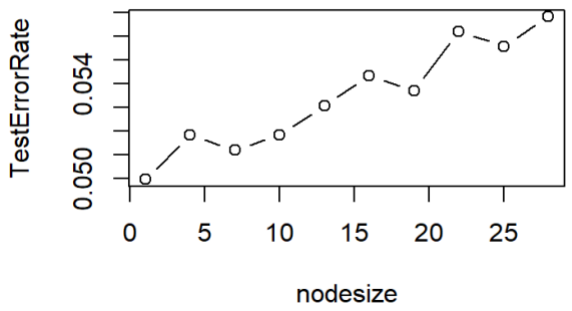
Random Forest

Since logistics model is pretty much the standard one, we still need more advanced model with high-dimension to test the data. Therefore, we use Random Forest as the second one to conclude the data and see the difference of result compared with the previous one.

The tree-based method is suitable for high-dimension data because it exhaustive search for every variables and cutoff to find the best pair $X_j, c_{X_j, c}$. The tree model partition the feature space into

“similar region” by split rule then choose the mode in every region. In addition, making a more robust and stable model, we can improve our tree model to add the technique of *bootstrap* and *randomness*. As rule of thumb, choose m=seqt(p)m=seqt(p) in the classification case. For another parameter nodesize, it controls the bias_variance tradeoff, therefore we use the cross-validation to select a reasonable value.

```
set.seed(1)

mtry = round( sqrt( length(names(spam)) -1 ) )

# Choose Best NodeSize

nodesize = seq(1, 30, by = 3)

TestErrorRate = numeric(length(nodesize))

for( i in 1:length(nodesize) ){

  rf.fit = randomForest(type ~ ., data = spam_trn, ntree = 300, mtry = mtry,
nodesize = nodesize[i])

  pred_type = predict(rf.fit, spam_tst ,type = "class")

  TestErrorRate[i] = mean(pred_type != spam_tst$type)

}

plot(nodesize, TestErrorRate, type = "b")
```




```
(best_nodesize = nodesize[TestErrorRate==min(TestErrorRate)])
```

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

The nodesize stands for the maximum number of observations in the terminal nodes. In this case it is 1. Then we fit the random Forest model with the best parameter made by cross validation

```
# Fit the RandomForest Model with the best nodesize

rf.spam = randomForest(type ~ ., data = spam_trn, ntree = 300, mtry = mtry, n
odesize = best_nodesize,importance = T)

pred_rf = predict(rf.spam, spam_tst,type = "class")

(conftable_RT = table(pred_rf, spam_tst$type))
```

```
##

## pred_rf   nonspam spam

##   nonspam     940   47

##   spam         36  578
```

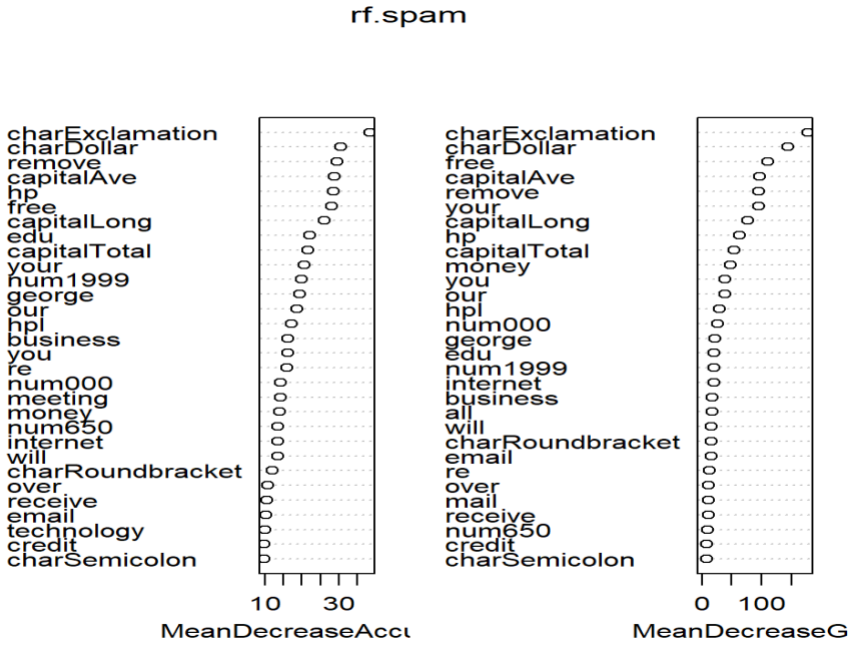
```
(missrate_RT = mean(pred_rf != spam_tst$type))
```

```
## [1] 0.0518426
```

```
# Check the importance of variables.

sort(importance(rf.spam)[, "MeanDecreaseAccuracy"], decreasing = T)[1:5]
```

```
## charExclamation      charDollar          remove      capitalAve

##        38.49454        30.47718        29.48196        28.72698

##              hp

##        28.52775
```

```
varImpPlot(rf.spam)
```



```
# charExclamation, charDollar, remove, capitalAve, hp is the most five import
ant variable
```

Higher **Variance Importance** means larger loss of accuracy due to the loss of information on variable X_j, hence more important. The graph shows two standards *Mean Decrease Accuracy* and *Mean Decrease Gini* that measures the importance of variables. The first one takes the misclassification rate while the second takes node impurity Gini as a measurement. Our goal is to lower the misclassification rate. Therefore, we choose the first one.

From the view of *Mean Decrease Accuracy,* the most five important variables are **charExclamation**, **charDollar**, **remove**, **capitalAve** and **hp**. CharExclamation is the character "!"; charDollar is the character "$"; capitalAve is the average length of uninterrupted capital letters; remove and hp are two words. Therefore, according to the RandomForest, we can conclude that word remove and hp, "$" and the average length of uninterrupted capital letters might be the important factors to determine an email is spam or not. Compared with the result we conclude in logistic regression, the important variables here have an overlap with the most significant variable of Logistic regression, which is hp and charDollar.

SVM

Random Forest is indeed a good model for applying the dataset. However, SVM only depends on support vectors and it also works very well for high dimension data since it is automatically regularized. Picking the widest separation margin is a way to automatically regularize. Therefore, we also picked SVM as our main model to see the accuracy and results.

```
# Radial Kernel
tune.out = tune(
  svm,
  type ~ .,
  data = spam_trn,
  kernel = "radial",
  ranges = list(gamma = c(0.001,0.005, 0.01,0.05),
                cost = seq(150,250, by = 20))

)
tune.out$best.parameters
##   gamma cost
## 9 0.001  190
svm.fit = tune.out$best.model
```

```
pred = predict(svm.fit, spam_tst)

conftable_SVM = table(pred, spam_tst$type)

missrate_svm = mean( pred != spam_tst$type)
```

After cross-validation tune, we choose radial kernel with gamma = 0.001 and cost = 190. The error is very low, SVM performs very well.

Assumptions about LDA and KNN is not good since their performance is not good enough for high-dimension data. In addition, they don't have the technique corresponding to the Variable selection like Logistic Regression. However, one of our goals is to "build many classification models and choose the best one". We still fit the modes to test our assumptions and compare with other models.

LDA & QDA

Attempt to utilize the linear (quadratic) discriminant analysis to classify data. The mechanism behind this is to first find $\mu^$, $\Sigma^$ and $\pi^$ then put them into Bayes equation to get Posterior Distribution. From above analysis, the high correlation and skew of the proportion of responses are not a problem. However, the data is somewhat "high dimension", therefore, the results of LDA and QDA may not be satisfying enough.

```
# LDA

lda.model = lda(type ~ ., data = spam_trn)

class_pred = predict(lda.model, spam_tst)$class

(conftable_LDA = table(class_pred, spam_tst$type))
```

```
##
## class_pred nonspam spam
##    nonspam     930  131
##    spam         46  494
```

```
(missrate_LDA = mean(class_pred != spam_tst$type))
```

```
## [1] 0.1105559
```

The test misclassification rate for the LDA model is 0.1105559 which is much higher than the corresponding selected logistic model.

There are too many (57) variables in each group spam or non-spam. In the spam group, there is a problem of rank deficiency makes it impossible to evaluate the covariance matrix. Therefore, QDA is infeasible in this case.

The results of the LDA and QDA indicates that this technique does not perform very well on the high dimensional data in this case 57-D because the inverse of $\Sigma\Sigma$ may not exist for high dimension

KNN

For KNN algorithm, the critical step is to select the best tune parameter k. The most general method is cross validation. A 3th-fold cross-validation is performed on the KNN model to test for a sequence of k. The **caret** package is used to train a sequence of models and report the accuracy.

```
# Cross Validation to Select k
set.seed(2)
train.x = spam_trn[,-58]
train.y = spam_trn$type
grid = expand.grid(k = seq(1,20,by = 2))
(knn.fit = train(train.x, train.y, method = "knn", trControl = trainControl(m
ethod = "cv", 3), tuneGrid = grid))
```

```
## k-Nearest Neighbors
## 3000 samples
##   57 predictor
##    2 classes: 'nonspam', 'spam'
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 2000, 2000, 2000
## Resampling results across tuning parameters:
##   k   Accuracy   Kappa
##    1  0.7883333  0.5573181
##    3  0.7896667  0.5576025
##    5  0.7840000  0.5450549
##    7  0.7703333  0.5156025
##    9  0.7703333  0.5166909
##   11  0.7673333  0.5082226
##   13  0.7630000  0.4986486
##   15  0.7546667  0.4806762
##   17  0.7540000  0.4790920
##   19  0.7526667  0.4764139
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```
# The best k = 1
knn_pred = predict(knn.fit, newdata = spam_tst)
(conftable_knn = table(knn_pred, spam_tst$type))
## knn_pred  nonspam spam
##   nonspam     818  160
##   spam        158  465
(missrate_knn = mean(knn_pred != spam_tst$type))
## [1] 0.1986259
```

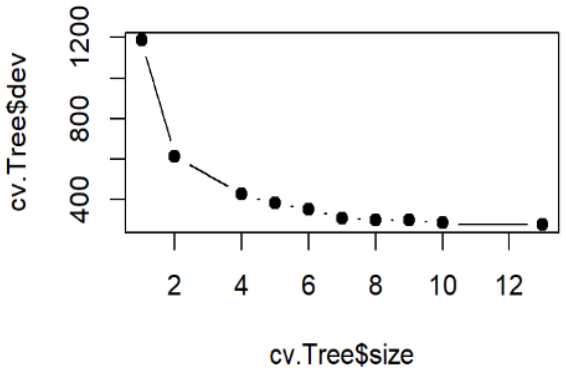
Even though KNN can classify most observations correctly, it still has much high classification rate compared with the techniques used before. Even twice higher than LDA. This is because even the data has many dimensions, many observations are similar or even the same, therefore they concentrate on each other make it harder for KNN to do the prediction

Classification Tree

After applying the normal supervised classification technique, consider the Tree-based method. Classification Tree is good for interpretation since we do the prediction after split data into different feature space based on the split rule. However, there is still a big disadvantage because of test perdition error caused by overfitting. Therefore, pruning is necessary to build an stable model. We use tree() method in the tree package to help use grow and prune the tree.

As KNN, cross-validation is still used to tune the parameter alpha which corresponds to different Tree size |T|. The plot below shows the relationship between the *nodesize* and *deviance of cross validation*.

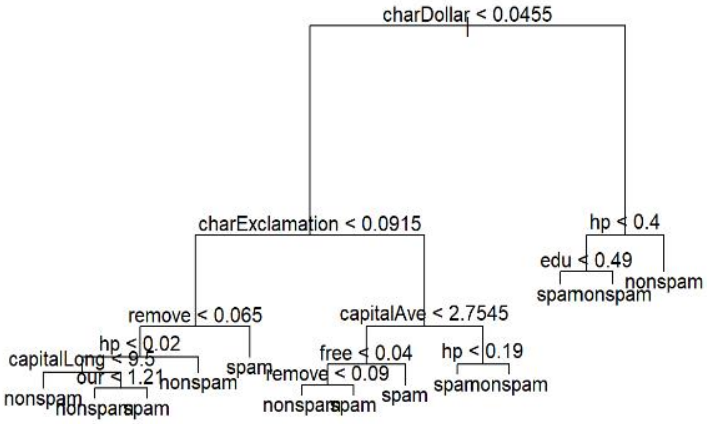
```
# Build the whole Tree
Tree_Full = tree(type ~ ., data = spam_trn)
# Cross Validation to Select Best Tune Parameter
cv.Tree = cv.tree(Tree_Full, FUN = prune.misclass)
plot(cv.Tree$size, cv.Tree$dev, type = "b", pch = 19)
```




```
best.node = cv.Tree$size[cv.Tree$dev == min(cv.Tree$dev)][1]
```

A final tree model is built based on the best tune parameter.
The below tree graph shows the split rule including selected variable X_j and cutoff c, branches and terminal nodes which are 13

```
# Prune the Tree by the best node
Tree_Final = prune.misclass(Tree_Full, best = best.node)
plot(Tree_Final)
text(Tree_Final)
```




```
# Make Prediciton & Test Missclassifictaion Rate
```

```
Pred_Final = predict(Tree_Final, spam_tst, type = "class")
(conftable_CT = table(Pred_Final, spam_tst$type))
##
## Pred_Final nonspam spam
##    nonspam     904   75
##    spam         72  550
(missrate_CT = mean( Pred_Final != spam_tst$type))
## [1] 0.09181761
```

For the classification tree, the main variables are charDollar, the character of "$", charExclamation, the character of "!", and the word, hp. The misclassification rate is similar with LDA which is better than KNN, but still cannot compare with Logistic Classifier.

Sensitivity & Specificity Analysis

In our case, the classification of spam, the harm of these errors is unequal. We can't allow important information, say, a job offer, miss our inbox and get sent to the spam folder. On the other hand, the spam email that would make it to an inbox (false negatives) are easily dealt with, just delete them. Therefore, instead of simply evaluating a classifier based on its misclassification rate (or accuracy), we'll create two additional functions to calculate sensitivity and specificity.

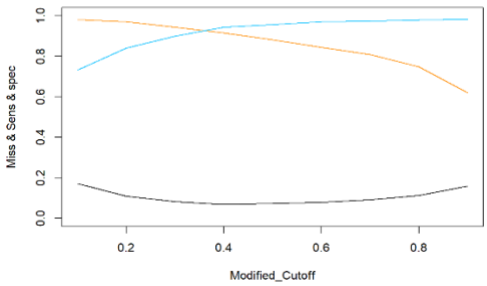
For Spam filter, we want to lower the rate which the useful(non-spam) email to be classified into spam. Therefore, we want to lower the False Positive Rate which means to improve specificity. An effective way is to modify the cut off in Logistic Classifier.

- From above all, we have many useful Models to successfully classify the spam with high accuracy.
- The Error of **False Positive** is much more harmful than **False Negative** for the Spam Filter.
- Improve **sensitivity** or **specificity** at the expense of the **overall accuracy**

```
# Sensitivity: True positive rate
get_sens = function(conf_mat) {
  conf_mat[2, 2] / sum(conf_mat[, 2])
}
# Specificity: True negative rate
get_spec = function(conf_mat) {
  conf_mat[1, 1] / sum(conf_mat[, 1])
}
# Original Logistic Classifier
get_sens(conftable_logistic)
## [1] 0.8816
get_spec(conftable_logistic)
```

```
## [1] 0.9569672
```

```
#Modify Cutoff to Improve Specificity
Modified_Cutoff = seq(0.1, 0.9, by = 0.1)
specificity = numeric(length(Modified_Cutoff))
Sensitivity = numeric(length(Modified_Cutoff))
Missrate =   numeric(length(Modified_Cutoff))

for(i in 1:length(Modified_Cutoff)){

  tst_pred_Modified = ifelse(predict(logistic_model, spam_tst, type = "respon
se") > Modified_Cutoff[i], "spam", "nonspam")
  conftable_Modified = table(tst_pred_Modified, spam_tst$type)
  Missrate[i] = mean(tst_pred_Modified != spam_tst$type)
  Sensitivity[i] = get_sens(conftable_Modified)
  specificity[i] = get_spec(conftable_Modified)

}

plot(Modified_Cutoff, Missrate, type = "l", col = "black", ylim = c(0,1), yla
b = "Miss & Sens & spec" )
points(Modified_Cutoff, Sensitivity, type = "l", col = "darkorange")
points(Modified_Cutoff, specificity, type = "l", col = "deepskyblue")
```



In the plot, black represents the test misclassification rate while orange is Sensitivity and blue is specificity. From this plot, the misclassification rate is the lowest at when cutoff = 0.5 which is corresponding to Bayes Classifier. Specificity is keeping increasing with bigger cutoff while sensitivity is decreasing. This is the idea that improves **sensitivity** at the expense of the **overall accuracy** and **specificity**. Since the significant harm of False Positive, we plan to choose cutoff = 0.9, to make it largest.

```
Results = data.frame(c(Missrate[9], missrate_logistic, missrate_LDA, missrate
_knn, missrate_CT, missrate_RT,missrate_svm),

                     c(Sensitivity[9], get_sens(conftable_logistic), get_sens
(conftable_LDA), get_sens(conftable_knn), get_sens(conftable_CT), get_sens(co
nftable_RT),get_sens(conftable_SVM) ),

                     c(specificity[9], get_spec(conftable_logistic), get_spec
(conftable_LDA), get_spec(conftable_knn), get_spec(conftable_CT), get_spec(co
nftable_RT),get_spec(conftable_SVM) ))

rownames(Results) = c("Modified Logistic","Logistic", "LDA","KNN", "Classific
ation Tree", "Random Forest","SVM")

colnames(Results) = c("Missclassification", "Sensitivity", "Specificity")

Results
```

```
##                      Missclassification Sensitivity Specificity
## Modified Logistic           0.15865084      0.6192   0.9836066
## Logistic                    0.07245472      0.8816   0.9569672
## LDA                         0.11055590      0.7904   0.9528689
## KNN                         0.19862586      0.7440   0.8381148
## Classification Tree         0.09181761      0.8800   0.9262295
## Random Forest               0.05184260      0.9248   0.9631148
## SVM                         0.06121174      0.9104   0.9569672
```

According to this Final Results, Random forest Model is the best in the view of Test Misclassification Rate, SVM with radial is the second. For a more practical use of spam filter (pay much more attention on Specificity), We choose the selected Logistic classifier with 0.9 cutoff.

Conclusion and Discussion

Summary of Scientific Findings:

- Most of the Classification Model work successfully in this Spam Filter
- Random Forest and SVM are the best for the Spam Filter in the view of misclassification rate
- charDollar (“$”), word “hp” and charExclamation (“!”) are the most effective variables for the spam filter, the “uninterrupted capita words also have some influence on the spam filter.
- Miss Rate: Random Forest is much less than Classification Tree, which indicates that bootstrap and randomness play important roles in the prediction error and model stability.
- From the test misclassification rate, it proves our assumptions before that KNN and LDA are bad for this high-dimension and crowded-points data.

- Combining with the harmful of Type I and Type II error, we tune the cutoff = 0.9 in the Logistic Model to make a more reasonable and practical spam filter.

Potential Pitfalls:

The whole dataset has only 4601 rows which is not enough to train and test our models sufficiently. To make the results more convincing, it would be better to gather more emails and collect more data information. In addition, this dataset is from Year 1999, therefore, the information extracted from each email could be outdated, to keep our spam filter as well as the analysis fashion and practical, we need to keep updating our spam database.

Concluding Remarks:

After this project, we constructed both theoretical-perfect and practical spam filter according to the dataset, what we need to do in the future is maintaining it with different emails data.

This project offers us an opportunity to apply what we learned throughout the whole semester and apply them to real-world problems.