Programming Assignment 3

Predict 412 Sec 56

Jake Chen

1 Introduction

1.1 Purpose

The purpose of this project is to compare the performance of multiple classification methods. Each of these classification methods compared using the same dataset and the same method. Here we will be creating three models: logistic regression, decision tree, and Random Forest. All three will be evaluated using both a confusion matrix as well as an ROC curve.

1.2 Dataset

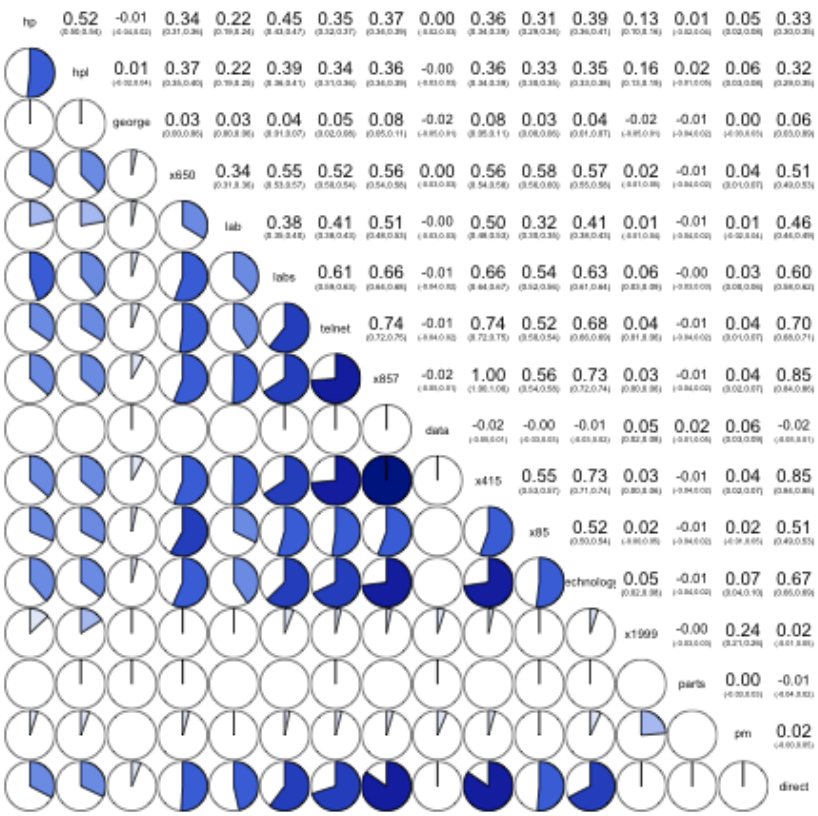
The dataset we will be using is “spambase” from the MMST package (it can also be found in the UCI repository as “Spambase Data Set”). This dataset contains information around emails both marked as spam and legitimate. The target variable is the classification of whether or not the email is spam, and the explanatory variables are numeric data pertaining to the proportions of specific words in the email, such as “free” and “order”. In total there are 4601 email records and 59 total variables (2 target, 57 explanatory).

2 Exploratory Data Analysis

In the EDA phase, we primarily look at 2 things: the relationship between the explanatory variables, and the relationship between the explanatory and target variables. This will hopefully provide us with a broad overview of the dataset as a whole and in the context of our classification problem.

2.1 Between Explanatory Variables

First we look at the relationships between explanatory variables. This is done using the "corrgram" function inside the corrgram library. Using this function we can view a matrix of the relationships between each and every variable. However, this function can be time consuming because of the large number of both variables and observations. To accommodate the function, we take a small random subset of the full data. Just by looking at 10% of the data we can already see some potentially concerning multicollinearity present in the dataset. Specifically we see relationships between the variables "x857" through "technology". This subset of the correlogram is shown below:



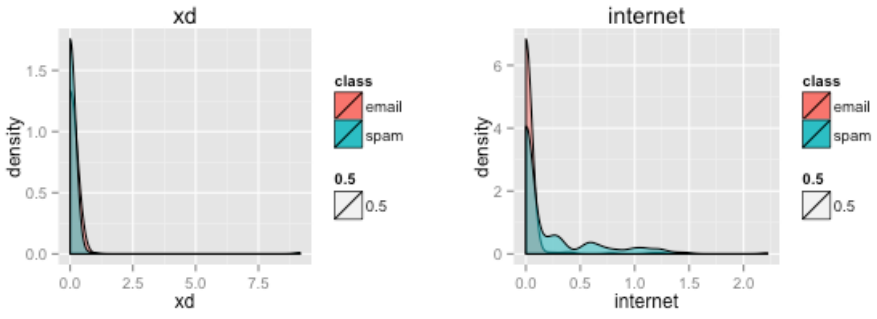
As we dig into this subset we find that there is a 1:1 correlation between the variables "x857" and "x415", even in the full dataset. It doesn't make sense to keep this redundancy, so we will remove the "x415" variable from the dataset.

2.2 Between Explanatory and Target

Next we look at the relationship between the explanatory variables and the target variable. Because this is a classification problem, we chose to use density charts of the variables, broken down and color coded by the target class. The ggplot2 package was used to facilitate the graphics. Using this type of visualization we will want to look for the following:

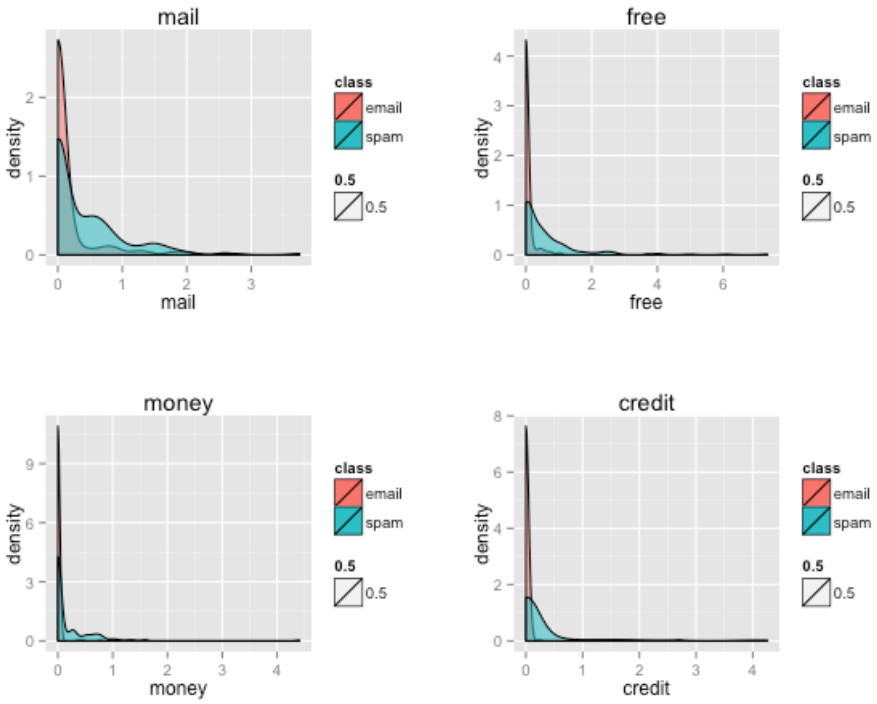
1) What is the distribution of the variable in general?
2) Is there a difference between the spam and legitimate distributions?

In terms of the distributions in general, we notice that all of the distributions spike at 0 and decrease sharply, with some variables having some more peaks at larger values. For example:



This distribution should make sense given the nature of our dataset. Because these variables look at the occurrence of words in the email, only distributions for words like "the" and "a" will not skew towards the corner.

As for the differences between spam and legitimate distributions, we can see a few variables that visually differentiate between the two classes, for example:



In all of these four variables above, the “spam” distribution is wider or contains peaks with non-zero means. All four of these words make intuitive sense in the context of spam/junk-mail. It is interesting to see these words appear visually as indicators of spam.

3 Modeling Process and Results

3.1 Modeling Iteration 1

In this first iteration of modeling we will use the default settings for our three methods. This will allow us to establish a baseline between the three, as well as for future iterations building on top of these three.

3.1.1 Split Test/Train

We first randomly split the full dataset into training and testing datasets. This simple method is suitable for us because of 2 things: the size of the dataset means that bias will not likely be introduced, and a training/testing dataset is the fastest in terms of the iterative process. Here we chose to split the dataset into an 80/20 training/testing. This number preserves a large number of observations in the training set while still providing an ample breakdown between the classes in the test set.

3.1.2 Logistic Regression

First we apply the Logistic Regression method to the training dataset. Based on the parameters we can see the words that do indeed track the difference between the classes. As expected, words such as “free”, “remove”, and “our” are statistically significant variables that indicate spam.

Still, what’s concerning with the logistic regression is the previously discovered multicollinearity. As a general linear model, logistic regression is susceptible to the fact that multicollinearity could introduce or exacerbate bias. If we wish to utilize this as our final model we will have to solve any concerns regarding multicollinearity. However, at this stage the model is nonetheless a valuable benchmark for the other methods.

Confusion Matrix

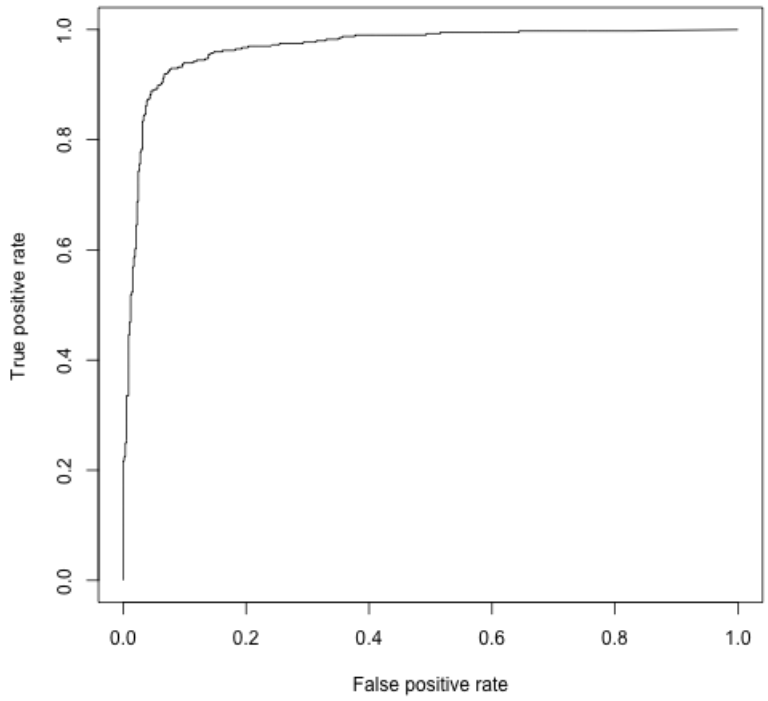
We first create a confusion matrix for the model. Here we apply a cutoff probability of 50%. Specifically we compare the predicted classes and the actual classes in the test set. The matrix is shown below:

```
##        y_hat
##           0   1 Sum
##    0    520  34 554
##    1     43 333 376
##    Sum 563 367 930
```

Our model performs relatively well on the testing data. The logistic regression was able to accurately predict 91.72% of the classes. There are 34 instances of false positives and 43 false negatives. Overall this is a passable, but not stellar, performance.

ROC Curve

While the confusion matrix above assumes a cutoff at 50% probability, an ROC can look at the entire range of cutoffs. Below we see the ROC Curve for the logistic regression:



The curve again shows that the model performs adequately but leaves some more to be desired. The AUC further reinforces this claim at 0.9715. Again quite good and likely good enough, but can potentially be improved still.

3.1.3 Decision Tree

Next we apply a decision tree method to the dataset. Because the final method is a Random Forest, a decision tree will provide us some initial insight into the structure and breakdown of the variables. This insight will be completely lost in the convoluted, but accurate, calculations of the Random Forest.

Our initial decision tree has 8 splits and 10 leaves. The most significant variables seem to be “remove”, “hp”, and “x..1”. The variables are slightly different than what we expected but do reflect the parameter estimates in the logistic regression. Still, these results are valuable in understanding the underlying process behind our tree.

Confusion Matrix

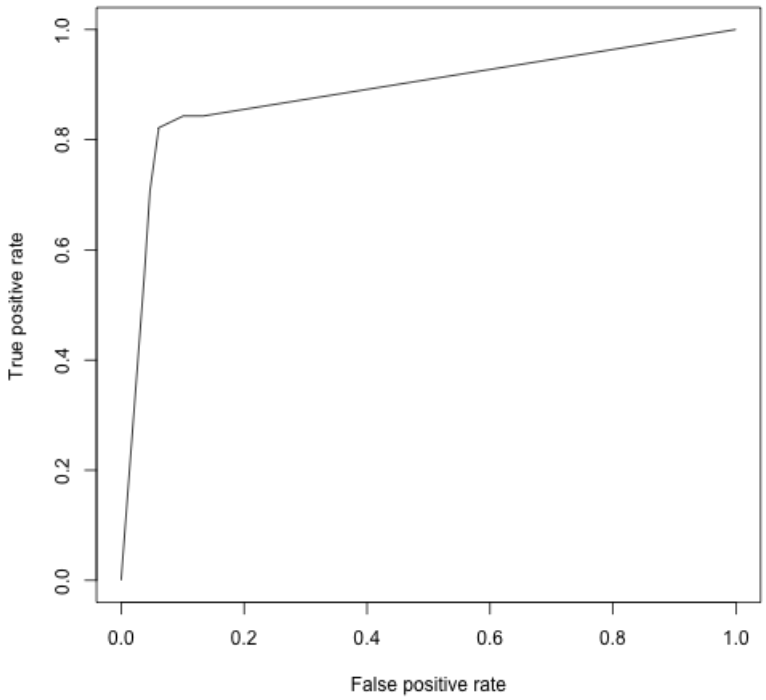
Again, let’s start with a confusion matrix, shown below:

```
##          y_hat
##           email spam Sum
##    email    520   34 554
##    spam      67  309 376
##    Sum      587  343 930
```

Unfortunately we see that the decision tree is not performing as well as the logistic regression. The results of the tree include 34 FP and 67 FN, with a subsequent accuracy of only 89.14%. Not too far from the logistic regression but a decrease nonetheless.

ROC Curve

The ROC Curve for the decision tree is show below:



The AUC for this curve is 0.8862, again lower than that of the logistic regression.

3.1.4 Random Forest (default parameters)

Finally, we'll run a Random Forest now that we have both a logistic regression and a decision tree to compare it to. Because it is very much a black-box algorithm, the output is quite sparse.

Confusion Matrix

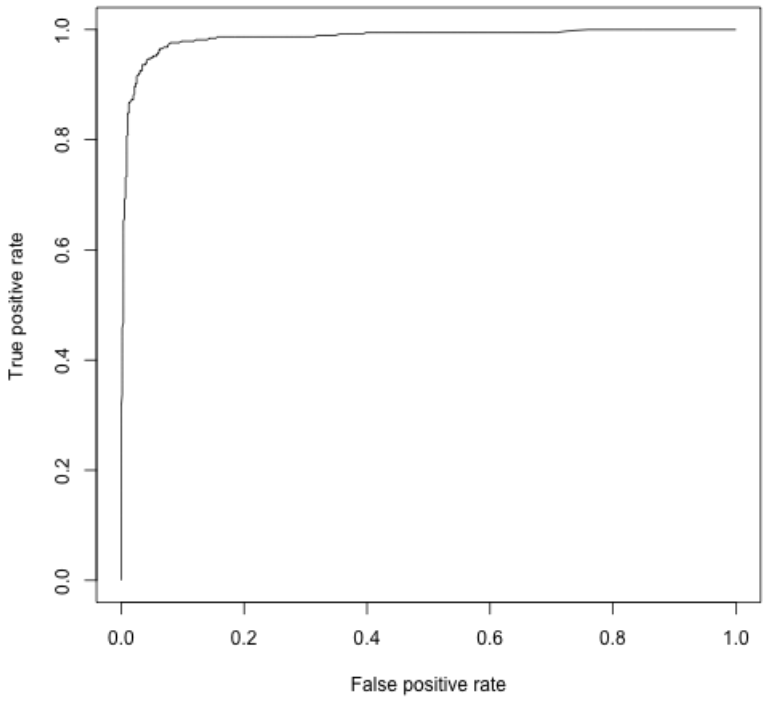
The confusion matrix when we apply out Random Forest model to the test set is shown below:

```
##          y_hat
##           email spam Sum
##    email    534   20 554
##    spam      24  352 376
##    Sum      558  372 930
```

As expected, the performance of the Random Forest is very good, exceeding that of the logistic regression. The results contain 20 false positive and 24 false negative observations, compared to the 34/43 for logistic and 34/67 for decision tree.

ROC Curve

The ROC curve for the decision tree also looks great:



The AUC for this curve is 0.9851, higher than of the logistic regression (0.9715), again showing the great performance of the Random Forest method.

3.1.5 Random Forest (modified)

In addition, we thought it might be useful to see if modifying the parameters of the Random Forest will improve the performance. The default Random Forest uses only 7 variables at a time to build a tree. Does increasing the number of variables used

increase the accuracy of the forest? We applied the same metrics and found that the modifications did not result in a drastic change in performance.

The modified Random Forest performed very similarly to the default Random Forest.

The Confusion Matrix for the modified forest is show here:

```
##          y_hat
##           email spam Sum
##    email    531   23 554
##    spam      22  354 376
##    Sum      553  377 930
```

There were 23 false positives and 22 false negatives, similar to the 20 and 24 of the default tree.

The ROC curve has an AUC or 0.9843, again very similar to the default forest. As a result, we felt it wasn't necessary to continue with the modified forest.

3.2 Modeling Iteration 2

For the second iteration, we thought it would be interesting to apply a Principle Components Analysis to the original dataset to try to 1) address the underlying multicollinearity and 2) reduce the number of variables inputted into the models. The first thing we noticed is that the top 3 principle components (PCs) captured 99.99% of the total variance. We thus decided to just use these top 3 PCs in our modeling process.

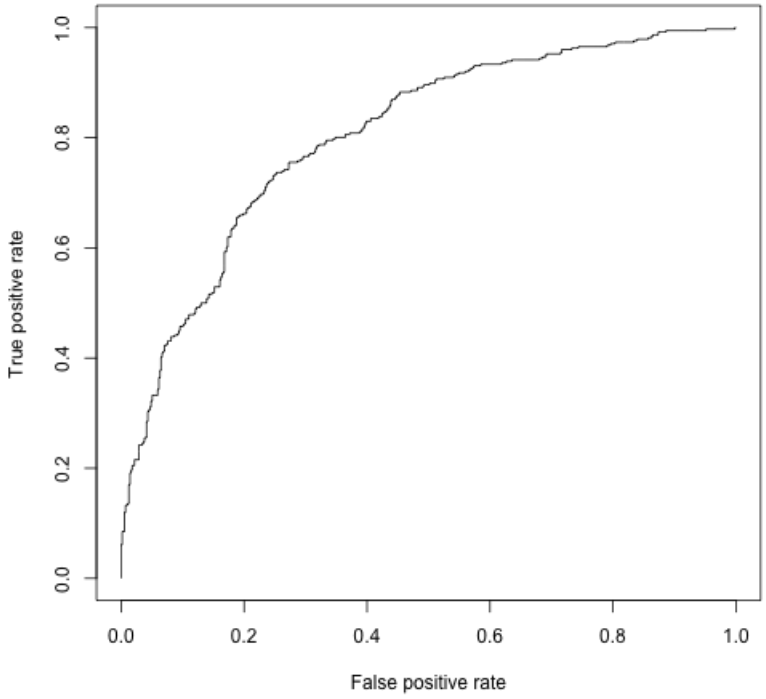
Logistic Regression

When we applied the logistic regression to this 3PC dataset, we found that the accuracy greatly plummeted. The confusion matrix, shown below, now contained a surprising 209 false negative observations.

```
##         y_hat
##            0   1 Sum
##    email 503  51 554
##    spam  209 167 376
##    Sum   712 218 930
```

It seemed that the information about the negatives were now lost after the PCA process.

The ROC Curve is also abysmal. The AUC is not 0.8035, a drastic drop from before.



Decision Tree and Random Forest

The decision tree did not fare any better. The false negative rate also severely increased, as did the false positive rate. The ROC Curve was also horrid. So far applying PCA was greatly impacting our models for the worse.

```
##         y_hat
##          email spam Sum
##    email   463   91 554
##    spam    179  197 376
##    Sum     642  288 930
```

Even the performance for the forest decreased dramatically. The AUC is now 0.8841 compared to the original 0.9851.

3.2.1 Summary

It seems that applying a PCA analysis and taking only the 3 PCs was drastically reducing the information available to the modeling methods. Upon further analysis though, the cause became very clear. The top 3 PCs are constructed only using the 'crla', 'crll', and 'crtt' attributes. This means that if we use the top 3 CPs, the information provided by the other variables are completely ignored. We also see that in terms of magnitude, these three variables are in the thousands whereas the other variables are in the ones and tens. This can explain the large amount of variance provided by these 3 variables. We then went on to use 15PCs in the analysis. While the performance was improved compared to the 3PC models, it still could not compare to the first iteration of testing.

4 Conclusion

In conclusion, we were successful in comparing the performance of three modeling techniques including logistic regression, decision tree, and Random Forest. The Random Forest performed the best out of all of these methods. In addition, we discovered that there was no need to apply Principle Components Analysis to the dataset.

5 Appendix

````
Prompt
======
This individual assignment asks students to use a statistical experiment to
compare alternative classification methods. Students are encouraged to install
packages with data-adaptive, machine learning algorithms. Ensemble methods
(committee machines) may also be employed. Various criteria, tabular, and
graphical (including receiver operating characteristic curves) are used for
evaluating classifiers. Try at least one traditional and one machine learning
method.

1. Load Preparation
===================
```{r}
library(MMST)
data(spambase)
```

2. EDA
======
```{r}
str(spambase)
head(spambase)
summary(spambase)
```

## 2.1 Between Explanatory Vars
Let's look at the relationships using a correlogram. Due to the large sample
size, it might be best to take a smaller sample first to not kill the machine.
```{r}
require(corrgram)
mask <- sample(c(T,F), nrow(spambase), replace=T, prob=c(0.1, 0.9))
temp <- spambase[mask,]
corrgram(temp, upper.panel=panel.conf, lower.panel=panel.pie)
```
There seems to be heavy multicollinearity between x857 through technology
(minus data). Especially concerning is the relationship between "x857" and
"x415". Let's run a correlogram on this smaller slice using the full dataset.
```{r}
corrgram(spambase[,25:40], upper.panel=panel.conf, lower.panel=panel.pie)
```
We see that "x857" and "x415" are indeed 1:1 correlated with one another. It
doesn't makes sense to keep this redundancy.

***Aside: Remove detected redundancy***
```{r}
spambase <- spambase[,-34]
```

Still, there remains some strong correlations within multiple variables. Let's
keep this in mind while we continue exploring the data.

## Vs Target
```{r, fig.width=4, fig.height=3}
require(ggplot2)
for(i in 1:nrow(temp)){
  p <- ggplot(temp, aes(x=temp[,i], fill=class, alpha=0.5)) +
    geom_density() +
    labs(title=names(temp)[i], x=names(temp)[i])
  print(p)
}
```
````

There seems to be some variables that have different densities between the classes. These variables may be useful when put into the classification models.

3. Modeling Iter. 1

3.1 Split Test/Train

```{r}
mask <- sample(c(T,F), nrow(spambase), replace=T, prob=c(0.8, 0.2))
train <- spambase[mask,]
test <- spambase[!mask,]
```

Are there enough test cases in the test set?

```{r}
table(test$class)
```

There are.

3.2 Logistic Regression

3.2.1 Fit the model

```{r}
fit <- glm(class~., family=binomial, train[-57])
summary(fit)
```

3.2.2 Test the model

```{r}
fit.pred <- predict(fit, test, type='response')
```

3.2.2.1 Confusion table using 0.5 cutoff rate

```{r}
y_hat <- 1*(fit.pred>0.5)
t <- table(test$classdigit, y_hat)
addmargins(t)
```

It's not too bad. False positive is 22/925 and false negative is 39/925. This is only at 0.5 cutoff though. Let's use an ROC Curve instead.

3.2.2.2 Using ROC Curve

```{r}
require(ROCR)
pred <- prediction(fit.pred, test$classdigit)
perf <- performance(pred, 'tpr', 'fpr')
plot(perf)
auc.tmp <- performance(pred,"auc")
print(fit.auc <- as.numeric(auc.tmp@y.values))
```

The AUC is very solid @ 0.9696

3.3 Decision Tree

3.3.1 Fit the model

```{r}
require(rpart)
tree <- rpart(class~., train[-57])
tree
plot(tree)
```

3.3.2 Test the model

```{r}
tree.pred <- predict(tree, test)
```

3.3.2.1 Confusion table

```{r}
y_hat <- predict(tree, test, type='class')
t <- table(test$class, y_hat)
addmargins(t)
```
The decision tree performs well, but not as well as the logistic regression. It has 45 FP and 48 FN.
3.4.2.2 Using ROC Curve
```{r}
require(ROCR)
pred <- prediction(as.matrix(tree.pred)[,2], test$classdigit)
perf <- performance(pred, 'tpr', 'fpr')
plot(perf)
auc.tmp <- performance(pred,"auc")
print(fit.auc <- as.numeric(auc.tmp@y.values))
```
The AUC is decently good @ 0.9073

3.4 Random Forest (default)
3.4.1 Fit the model
```{r}
require(randomForest)
rf <- randomForest(class~., train[-57])
rf
```
3.4.2 Test the model
3.4.2.1 Confusion table
```{r}
y_hat <- predict(rf, test)
t <- table(test$class, y_hat)
addmargins(t)
```
The random forest performs better than the others, with a FP or 10 and FN of 28.
3.3.2.2 Using ROC Curve
```{r}
require(ROCR)
rf.pred <- predict(rf, test, type='prob')
pred <- prediction(as.matrix(rf.pred)[,2], test$classdigit)
perf <- performance(pred, 'tpr', 'fpr')
plot(perf)
auc.tmp <- performance(pred,"auc")
print(fit.auc <- as.numeric(auc.tmp@y.values))
```
The ROC for the random forest is great with its AUC of 0.9904!

3.5 Random Forest (tweaked)
3.5.1 Fit the model
```{r}
require(randomForest)
rf <- randomForest(class~., train[-57], mtry=15)
rf
```
3.5.2 Test the model
3.5.2.1 Confusion table
```{r}
y_hat <- predict(rf, test)
t <- table(test$class, y_hat)
addmargins(t)
```
3.5.2.2 Using ROC Curve

```{r}
require(ROCR)
rf.pred <- predict(rf, test, type='prob')
pred <- prediction(as.matrix(rf.pred)[,2], test$classdigit)
perf <- performance(pred, 'tpr', 'fpr')
plot(perf)
auc.tmp <- performance(pred,"auc")
print(fit.auc <- as.numeric(auc.tmp@y.values))
```
The tweaked random forest works about as well as the default. There's probably no point to modifying the default parameters.

4. Modeling Iter. 2
==========
This time we'll first apply a PCA to the dataset.
4.1 PCA

4.1.1 Build PC mappings
```{r}
pc <- princomp(spambase[,-57:-58])
summary(pc)
plot(pc)
```
Just 3 PCs capture as much as 99.99% of the variance. Let's just take these three columns to speed up the modeling process.
4.1.2 Build train and test sets
```{r}
train.pc <- as.data.frame(predict(pc, train))
train.pc <- train.pc[,1:3]
train.pc <- cbind(train.pc, train$class)
names(train.pc)[4] <- 'class'
test.pc <- as.data.frame(predict(pc, test))
test.pc <- test.pc[,1:3]
test.pc <- cbind(test.pc, test$class)
names(test.pc)[4] <- 'class'
```

4.2 Logistic Regression

4.2.1 Fit the model
```{r}
fit <- glm(class~., family=binomial, train.pc)
summary(fit)
```
4.2.2 Test the model
```{r}
fit.pred <- predict(fit, test.pc, type='response')
```
4.2.2.1 Confusion table using 0.5 cutoff rate
```{r}
y_hat <- 1*(fit.pred>0.5)
t <- table(test.pc$class, y_hat)
addmargins(t)
```
The performance of the logistic regression decreased drasitically after a PCA. It's FP is now 31 and its FN is now a staggering 193... Perhaps the ROC will have more information.
4.2.2.2 Using ROC Curve
```{r}
require(ROCR)
pred <- prediction(fit.pred, test.pc$class)
perf <- performance(pred, 'tpr', 'fpr')
```

```
plot(perf)
auc.tmp <- performance(pred,"auc")
print(fit.auc <- as.numeric(auc.tmp@y.values))
```
The AUC is now 0.8417 compared to 0.9696 from before. Given that a PCA should be more accurate, I'm starting to be concerned that the logistic may actually be overfitting. Let's see the other algorithms to see what happens.

4.3 Decision Tree

4.3.1 Fit the model
```{r}
require(rpart)
tree <- rpart(class~., train.pc)
tree
plot(tree)
```
4.3.2 Test the model
```{r}
tree.pred <- predict(tree, test.pc)
```
4.3.2.1 Confusion table
```{r}
y_hat <- predict(tree, test.pc, type='class')
t <- table(test.pc$class, y_hat)
addmargins(t)
```
The decision tree has also decreased in accuracy. We see the same same increase in FN (48 -> 127) as we did in logistic regression. Perhaps PCA wasn't the best idea after all, as it seems to be losing information.

4.3.2.2 Using ROC Curve
```{r}
require(ROCR)
pred <- prediction(as.matrix(tree.pred)[,2], test.pc$class)
perf <- performance(pred, 'tpr', 'fpr')
plot(perf)
auc.tmp <- performance(pred,"auc")
print(fit.auc <- as.numeric(auc.tmp@y.values))
```
AUC has also decreased to 0.82 compared to the previous 0.9073.

4.4 Random Forest

4.4.1 Fit the model
```{r}
require(randomForest)
rf <- randomForest(class~., train.pc)
rf
```
4.4.2 Test the model
4.4.2.1 Confusion table
```{r}
y_hat <- predict(rf, test.pc)
t <- table(test.pc$class, y_hat)
addmargins(t)
```
Even the RF is decreasing in accuracy, from a FP or 10 and FN of 28 to 67 and 83.

4.4.2.2 Using ROC Curve
```{r}
require(ROCR)
rf.pred <- predict(rf, test.pc, type='prob')
pred <- prediction(as.matrix(rf.pred)[,2], test.pc$class)
```

```
perf <- performance(pred, 'tpr', 'fpr')
plot(perf)
auc.tmp <- performance(pred,"auc")
print(fit.auc <- as.numeric(auc.tmp@y.values))
```
The ROC for the random forest has decreased to 0.888 from 0.9904.

5. Modeling Iter. 3

The model building was horrible when only 3 CPs were included. Seems a lot of information was lost. When we dig into the linear equations, this information loss makes sense. The top 3 PCs are constructed only using the 'crla', 'crll', and 'crtt' attributes. This means that if we use the top 3 CPs, the information provided by the other variables are completely ignored. We also see that in terms of magnitude, these three variables are in the thousands whereas the other variabes are in the ones and tens. This can explain the large amount of variance provided by these 3 variables.

Let's include a few more PCs to try to bring in some of the information.

5.1 PCA

5.1.1 Build PC mappings
```{r}
pc <- princomp(spambase[,-57:-58])
summary(pc)
plot(pc)
```

5.1.2 Build train and test sets
```{r}
train.pc <- as.data.frame(predict(pc, train))
train.pc <- train.pc[,1:15]
train.pc <- cbind(train.pc, train$class)
names(train.pc)[16] <- 'class'
test.pc <- as.data.frame(predict(pc, test))
test.pc <- test.pc[,1:15]
test.pc <- cbind(test.pc, test$class)
names(test.pc)[16] <- 'class'
```

5.2 Logistic Regression

5.2.1 Fit the model
```{r}
fit <- glm(class~., family=binomial, train.pc)
summary(fit)
```
5.2.2 Test the model
```{r}
fit.pred <- predict(fit, test.pc, type='response')
```
5.2.2.1 Confusion table using 0.5 cutoff rate
```{r}
y_hat <- 1*(fit.pred>0.5)
t <- table(test.pc$class, y_hat)
addmargins(t)
```
Now the FP is 34 and the FN and 40. Still not as good as initially. Maybe it's overfitting or we still lost information in the PCA.
5.2.2.2 Using ROC Curve
```{r}
```

```
require(ROCR)
pred <- prediction(fit.pred, test.pc$class)
perf <- performance(pred, 'tpr', 'fpr')
plot(perf)
auc.tmp <- performance(pred,"auc")
print(fit.auc <- as.numeric(auc.tmp@y.values))
```
The AUC is now 0.9614 compared to the 0.9696 initially. This is better than the previous model with 3 PCs but still not as good as the initial raw model.

5.3 Decision Tree

5.3.1 Fit the model
```{r}
require(rpart)
tree <- rpart(class~., train.pc)
tree
plot(tree)
```
5.3.2 Test the model
```{r}
tree.pred <- predict(tree, test.pc)
```
5.3.2.1 Confusion table
```{r}
y_hat <- predict(tree, test.pc, type='class')
t <- table(test.pc$class, y_hat)
addmargins(t)
```
The FP is now 64 with the FN at 65. Still not as good as the original.
5.3.2.2 Using ROC Curve
```{r}
require(ROCR)
pred <- prediction(as.matrix(tree.pred)[,2], test.pc$class)
perf <- performance(pred, 'tpr', 'fpr')
plot(perf)
auc.tmp <- performance(pred,"auc")
print(fit.auc <- as.numeric(auc.tmp@y.values))
```
AUC is now 0.88 compared to the initial 0.9073.

5.4 Random Forest

5.4.1 Fit the model
```{r}
require(randomForest)
rf <- randomForest(class~., train.pc)
rf
```
5.4.2 Test the model
5.4.2.1 Confusion table
```{r}
y_hat <- predict(rf, test.pc)
t <- table(test.pc$class, y_hat)
addmargins(t)
```
The RF is lower in accuracy, from a FP or 10 and FN of 28 to 20 and 34.
5.4.2.2 Using ROC Curve
```{r}
require(ROCR)
rf.pred <- predict(rf, test.pc, type='prob')
pred <- prediction(as.matrix(rf.pred)[,2], test.pc$class)
perf <- performance(pred, 'tpr', 'fpr')
```

```
plot(perf)
auc.tmp <- performance(pred,"auc")
print(fit.auc <- as.numeric(auc.tmp@y.values))
```
The ROC for the random forest has increase to 0.987, but it's still not as good as the initial 0.9904.

6. Conclusion

After running the PCA and taking the top 3 and 15 PCs, we still are not getting the accuracy from the initial modeling w/out PCA. Currently the default RF built on the raw data performs extremely well already and is a model that we can continue to use.