Assignment 1: Decision Trees

Ricco Amezcua
CS422 Data Mining
Department of Computer Science
Illinois Institute of Technology

March 10, 2013

Abstract

This is a report for the second assignment of CS422 Data Mining. In this report, a set of data containing emails and its various qualites are observed. A decision tree is created and used to classify the emails as spam or not spam. Then, an attempt is made to make a decision tree algorithm.

1 Problem Statement

In this report, data collected from emails is looked at. Various details about the emails have already been collected and are available in the data set. Also, whether the email is spam or not spam has been included. Using this data set, a decision tree algorithm is used to create a decision tree using 80% of the data set. The decision tree is then tested on the other 20% of the data set. From this, a confusion matrix is created and the accuracy, recall, and precision is found. The importance of this report is to show the effectiveness of the decision tree algorithm as a classifier.

This process is first done using the rpart package in r. The creation of a decision tree algorithm was attempted but not completed in time.

2 Proposed Solution

]The data is split in to two parts: %80 for training and %20 for testing. The samples are chosen randomly without replacement.

The first question used the *rpart* package to create a decision tree. Its graph was looked at to determine the best value for its pruning.

While the decision tree algorithm was not implemented, if it was, the following would be the algorithm. First the main algorithm would be *GrowTree* and it would take a data frame as its argument. It would then check the stopping condition, which is if all of the emails in the data frame have the same class or all the attributes have the same value. If this is true, a leaf is created, classified by the majority class, and then returned. Else, a new node in the tree is created and a best split algorithm is used to find the attribute that will give the best split and the split point that will give the best split. The rating of the attributes is done using either Gini, Entropy, or the Classification Error. Once the attribute is found, it is used to split the data frame into two sets, those below the split point and those equal or above the split point. The function *GrowTree* is then called on both children. Once they return, they'll be associated as the children of the new node. Then, the new node is returned.

The *Prune* function would look at the growth between a child and a parent. It would take an argument which would be a threshold. If the a branch has a growth smaller than the threshold, it would be pruned from the decision tree.

3 Implementation Details

The R scripts can be found inside the *code* folder. The scripts are separated in to two scripts: question 1, question 2 . They are ran by running all of the commands in order. The code must be in the same file as the data files. The data matrices can be found in the *data* folder.

The second question had the largest problem. The implementation of the *BestSplit* algorithm was completed and works on a data set once, however it is a large bottleneck. When ran on a large data set, it takes a very long time to complete once, let alone enough times to create a tree. No solution was found to fix the algorithm, and therefore the *GrowTree* algorithm could not be completed.

4 Results and Discussion

Figure 4.1 shows the decision tree that was created using the rpart function. The tree is quite dense, however, the the algorithm was able to get the data down to at most a %20-%80 split in the data, and in some places %10-%90. Looking at Table 1, which contains the metrics from the confusion matrices, the unpruned tree has good accuracy, precision, and recall for the training set, which is to be expected. The metrics for the data set dip slightly for testing set, but only by at most %2.

Using the plot shown in Figure 4.2, the error rate was determined to be 0.017. Using this error rate, the tree was pruned by this much. This gives the pruned tree at Figure 4.3. Looking again at Table 1, the accuracy, precision, and recall has dropped %2 for the training set. The scores for the testing set are much lower for the pruned tree than the unpruned tree. This can be for various reasons. One is that the tree was pruned too much. Another is that the testing set was a good representation of the training set. If this is the case, it would be interesting to have more data to test to see if the accuracy, precision, and recall stay high for other samples of data.

Table 1: Metrics of Confusion Matrices

Matrix	Accuracy	Precision	Recall
Training	0.90	0.90	0.94
Testing	0.88	0.89	0.92
Pruned Training	0.88	0.88	0.92
Pruned Testing	0.83	0.85	0.91

5 References

1. http://cran.r-project.org/manuals.html

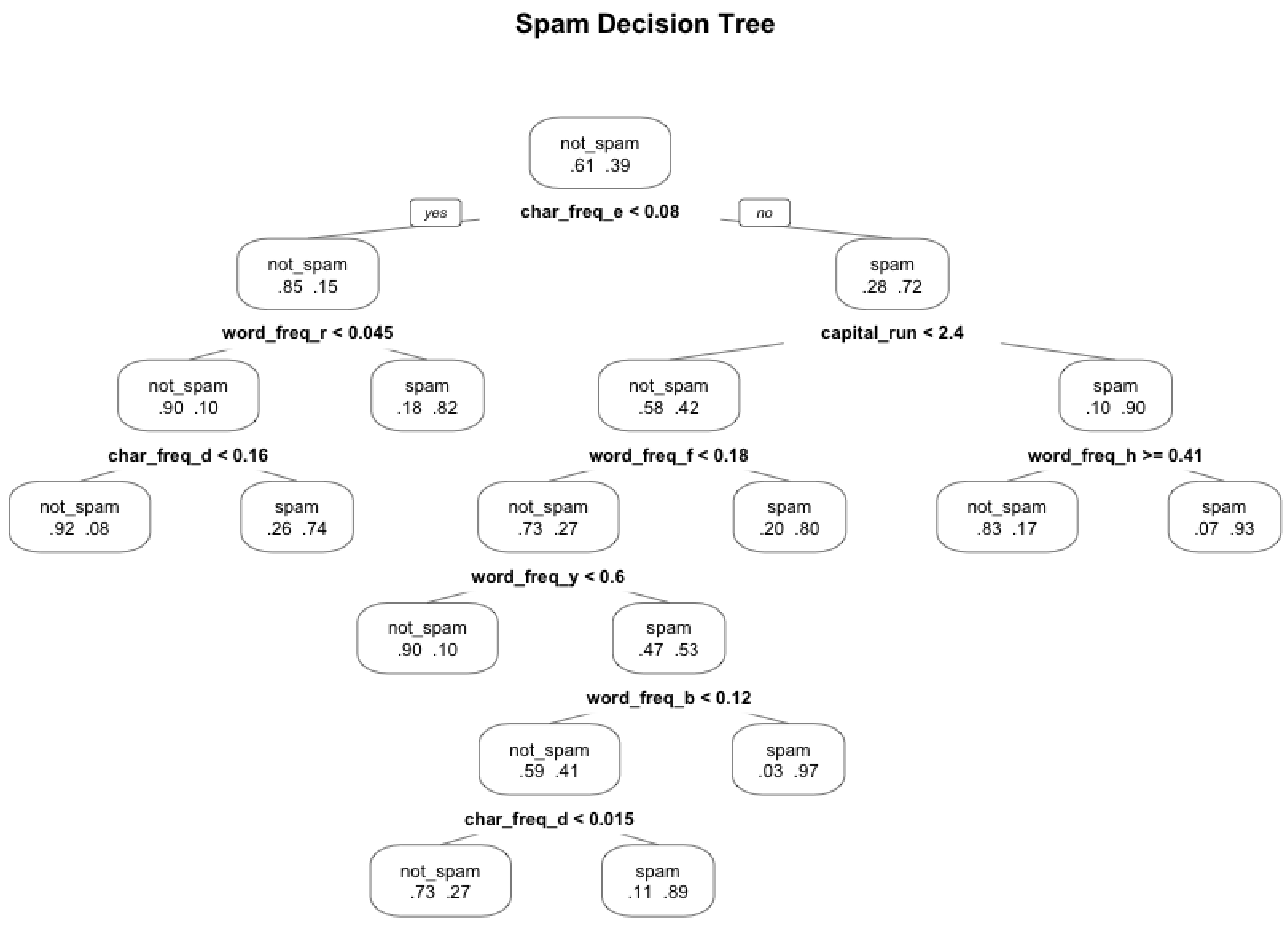
Spam Decision Tree
not_spam .61 .39
yes
char_freq_e < 0.08
no
not_spam .85 .15
spam .28 .72
word_freq_r < 0.045
capital_run < 2.4
not_spam .90 .10
spam .18 .82
not_spam .58 .42
spam .10 .90
char_freq_d < 0.16
word_freq_f < 0.18
word_freq_h >= 0.41
not_spam .92 .08
spam .26 .74
not_spam .73 .27
spam .20 .80
not_spam .83 .17
spam .07 .93
word_freq_y < 0.6
not_spam .90 .10
spam .47 .53
word_freq_b < 0.12
not_spam .59 .41
spam .03 .97
char_freq_d < 0.015
not_spam .73 .27
spam .11 .89

Figure 4.1:

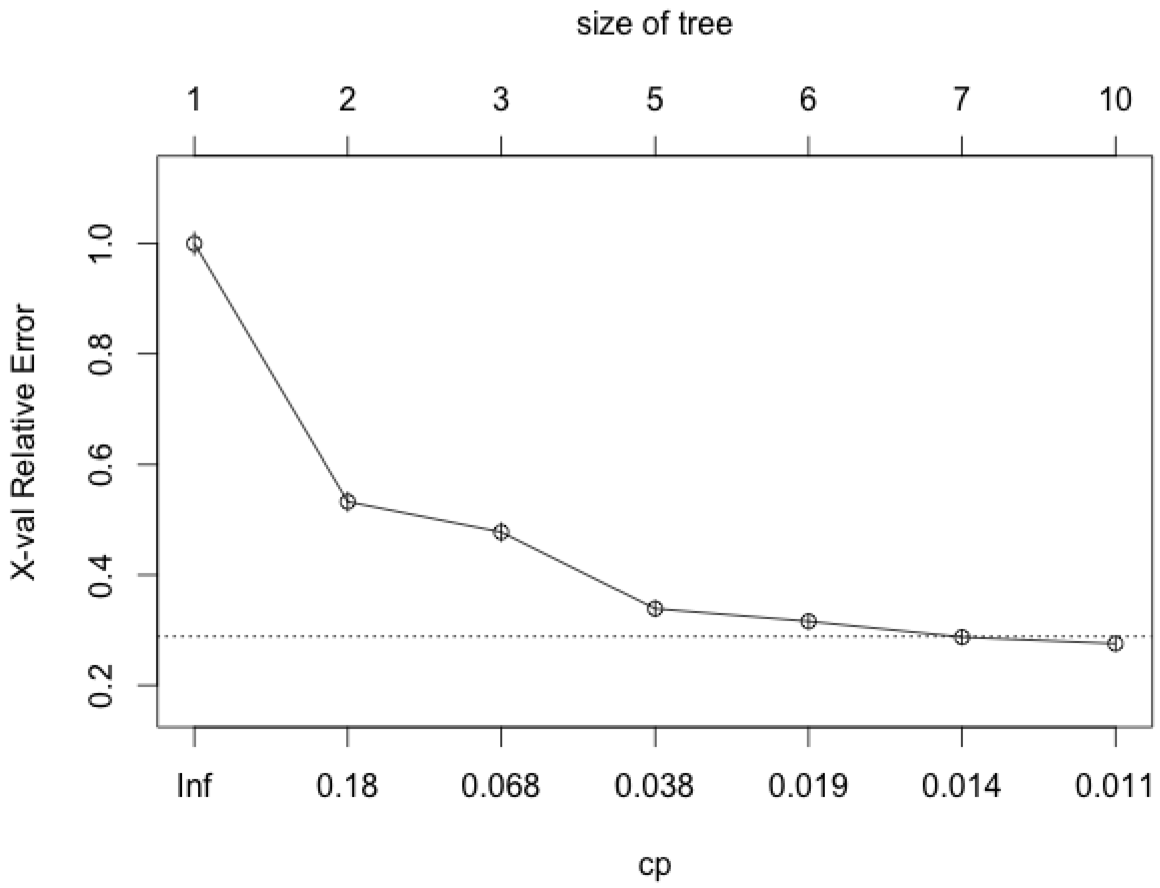
size of tree
1
2
3
5
6
7
10
X-val Relative Error
1.0
0.8
0.6
0.4
0.2
Inf
0.18
0.068
0.038
0.019
0.014
0.011
cp

Figure 4.2:

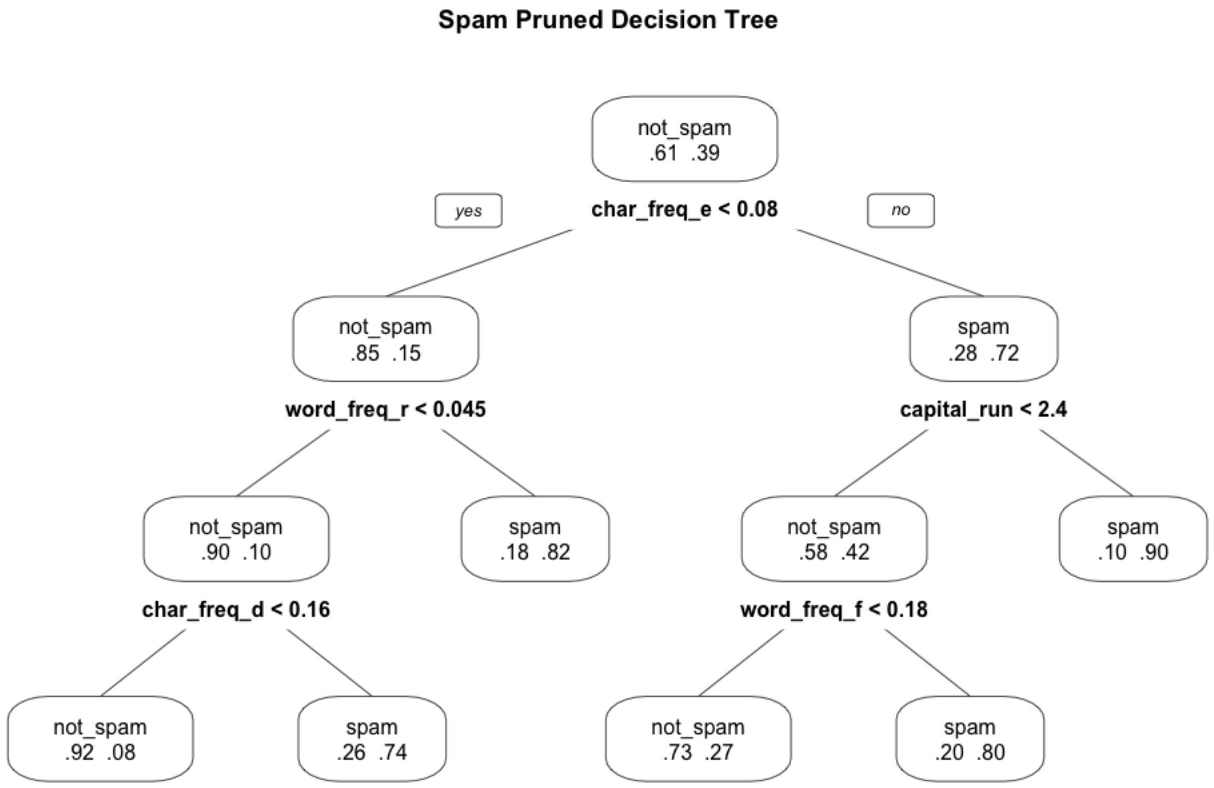
Spam Pruned Decision Tree
not_spam
.61 .39
yes
char_freq_e < 0.08
no
not_spam
.85 .15
spam
.28 .72
word_freq_r < 0.045
capital_run < 2.4
not_spam
.90 .10
spam
.18 .82
not_spam
.58 .42
spam
.10 .90
char_freq_d < 0.16
word_freq_f < 0.18
not_spam
.92 .08
spam
.26 .74
not_spam
.73 .27
spam
.20 .80

Figure 4.3: