Kevin Chian
23776620
CS189
Decision Trees Write-Up

For my decision trees I implemented a series of techniques to help me get a higher prediction rate. I found that it was good to create leaf nodes to stop tree growth in two situations: either then entropy of the current subset was below 0.3 or when my segmentation code returned a None for best feature to split on, indicating that the data had become unclear enough to the point that splitting didn't help much. At each leaf node I labeled the classification value as the majority label in the remaining subset at the node.

At each decision node, I selected at random sqrt(n) features to test on, where n was the total number of features for each observation (in my case 4711). For each randomly selected feature, I sorted the features and labels then ran through all the unique feature values that occur. I looked at the number of data points and corresponding spam labels that had the feature less than or equal to the unique feature value. I ran a information gain metric like the one we saw in lecture, with total entropy - (percent of data on left * left entropy) - (percent of data on right * right entropy). Entropy is the exact equation given in the lecture slides. Maximizing over all values of information gain, I took the best feature and split value and that became my decision node. Random feature sampling helps with creating more diversity in each tree and will assist in classification accuracy.

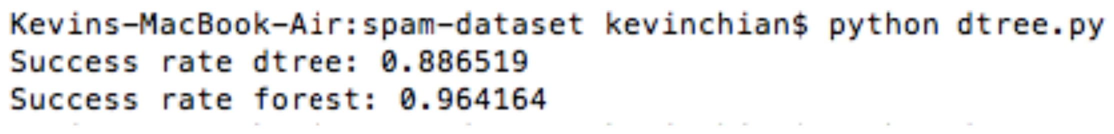
To avoid recomputing the spam count and number of data points to the left and right of any given split of any given feature, I kept a running count of how much spam and data had gone by for each new unique split value. This I stored as a temp dictionary for each feature, and then I could use the values to quickly compute entropy and information gain metrics, without having to sum up all the spam labels each time. This allowed for slightly faster training.

When implementing random forests, I also used a bagging technique I found in a paper. I took n random samples of data points with replacement from the original dataset and made those my "new datasets" for each of the decision trees. As each decision tree has a slightly different dataset it would have a slightly different tree and so would help with classification by giving it more diversity.

I then tested the best results given one decision tree, 10 trees, 25 trees, 40 trees, and 50 trees. I found that 25 trees and up gave pretty good results, and ran my waggle submission with both 25 trees and 50 trees. I used a validation set of 1172 data points, with the remaining 4000 going to training. These data points were randomly selected from the original 5172 data points.

For features, I went into featurize.py and implemented a bag of words model. I went through every ham and spam file and summed up all the word occurrences in a global dictionary. Then I pruned all word occurrences that happened less than 15 times. The remaining ones would serve as my features, and I then used the provided code to produce feature vectors, only with 4711 features instead of the original 32.

Using my validation set of 1172 randomly chosen points and labels, I got a final hit rate of:



```
Kevins-MacBook-Air:spam-dataset kevinchian$ python dtree.py
Success rate dtree: 0.886519
Success rate forest: 0.964164
```

I chose to use 25 decision trees in my random forest for the results above.
My best Kaggle score is 0.93921, which was gotten using 50 trees.

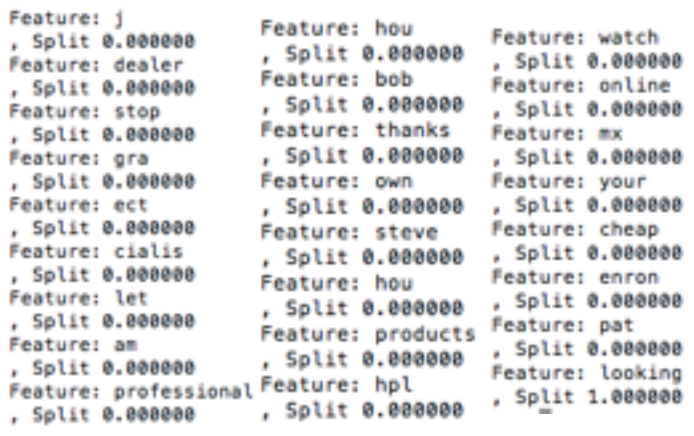
Path of one data point through decision tree:

Feature	Split
questions	<=0
works	<=0
uncertainties	<=0
meter	<=0
nom	>0

Prediction: Not spam

Splits at all roots of a 25 tree random forest:

Feature	Split (<=)	Feature	Split (<=)	Feature	Split (<=)
j	0	cialis	0	bob	0
dealer	0	let	0	thanks	0
stop	0	am	0	own	0
gra	0	professional	0	steve	0
ect	0	hou (x2)	0	products	0
hpl	0	watch	0	online	0
mx	0	your	0	cheap	0
enron	0	pat	0	looking	1



```
Feature: j
, Split 0.000000
Feature: dealer
, Split 0.000000
Feature: stop
, Split 0.000000
Feature: gra
, Split 0.000000
Feature: ect
, Split 0.000000
Feature: cialis
, Split 0.000000
Feature: let
, Split 0.000000
Feature: am
, Split 0.000000
Feature: professional
, Split 0.000000
Feature: hou
, Split 0.000000
Feature: bob
, Split 0.000000
Feature: thanks
, Split 0.000000
Feature: own
, Split 0.000000
Feature: steve
, Split 0.000000
Feature: hou
, Split 0.000000
Feature: products
, Split 0.000000
Feature: hpl
, Split 0.000000
Feature: watch
, Split 0.000000
Feature: online
, Split 0.000000
Feature: mx
, Split 0.000000
Feature: your
, Split 0.000000
Feature: cheap
, Split 0.000000
Feature: enron
, Split 0.000000
Feature: pat
, Split 0.000000
Feature: looking
, Split 1.000000
```

External References:

Wagner’s paper on Random Forests for CS170 - http://www-inst.cs.berkeley.edu/~cs170/fa14/misc/learn-rf.pdf
Wikipedia Bag of Words - http://en.wikipedia.org/wiki/Bag-of-words_model