

Chris Lim 0435570  
CSE573 – Project 1  
12/11/2008

## Introduction

In this assignment, my aim was first to understand the performance of Naïve Bayes and Decision Tree classifiers (along with their various ensemble variations) on a spam filtering task given various levels of training data and second to gain experience using the WEKA Machine Learning package.

I originally tried to obtain spam datasets from WEBSpAM UK2007 according to the link provided in the assignment description, but it required several hoops to get the actual data, so I reverted to the data provided by a previous quarter of CSE473. I wrote a script to transform the data into the .arff format required by WEKA and was anticipating the feature selection stage, but discovered that the data had pre-selected features (with none of the original content), so I proceeded directly to experimenting with different classifiers. I examined their performance on the original training and test sets as well as the combination of these two with 10-fold cross-validation. In the end, I obtained a performance better than the one described in the dataset description (I obtained 5.2% misclassification error rate compared to ~7% in the description) using Boosted Decision Trees. In the rest of this assignment, I describe the procedure I followed to use Weka, the experiment I conducted and the results from using different classifiers on the spam filter task. In my conclusion, I present an informal analysis of the results and also describe my experience with using the Weka package.

## Procedure

In order to run the experiment, I used the following procedure:

1. Obtain the training and test data from the following links:  
Training: <http://www.cs.washington.edu/education/courses/cse473/04sp/hw/hw3/spambase.train>  
Test: <http://www.cs.washington.edu/education/courses/cse473/04sp/hw/hw3/spambase.test>
2. Run the included script make-arff.pl with the following in a command line:  
perl make-arff.pl < spambase.train > spambase.train.arff  
perl make-arff.pl < spambase.test > spambase.test.arff  
Note: I also concatenated spambase.train and spambase.test to get one big file and trained and tested on that file spambase.all using cross-validation.
3. Download and Install the WEKA ML package from this location:  
<http://prdownloads.sourceforge.net/weka/weka-3-4-13.exe>
4. Copy the spambase.train.arff, spambase.test.arff, spambase.all.arff files to the machine with WEKA installed and run WEKA. Select the Experimenter and load the included spambase\_experiment (final).exp file.
5. Update the path to the data files and the path to the output file. Click the Run tab and press Start
6. Compile and examine the results in the Analyze tab
7. Repeat experiment using a randomized split of the total dataset into 66% training and 90% training
8. Follow-up any interesting classifiers with deeper analysis in the WEKA explorer

## Experiment

The dataset I used was provided to the Spring 2004 class of CSE473 at UW and consisted of the UCI Spambase corpus (<http://archive.ics.uci.edu/ml/datasets/Spambase>), which was pre-split into training and test data. I combined the training and test data into a single file and wrote a script to transform it into an appropriate format for WEKA. The feature values were pre-selected (I did not select them and did not have access to the raw data).

For my experiment, I split the data 3 ways: 66% training, 90% training (similar to the original split), and with 10-fold cross validation.

Here are the details of the attached spambase\_experiment\_practice (final).exp file (Note the file only does 10-fold cross-validation, but to try the other experiments, you can simply load it and change the setting in a convenient textbox):











I performed a 10-fold cross-validation experiment on the total combined data from the training and test sets with the following configurations of learners:

- Decision Tree with and without reduced error pruning
- Naïve Bayes Classifier
- Ensemble using 10 iterations of AdaBoost to get 10 Naïve Bayes classifiers
- Ensemble using 10 iterations of AdaBoost to get 10 Decision Trees
- Voting using 1 Naïve Bayes classifier, 1 Decision Tree, and 1 Majority Classifier
- An ensemble of 10 Decision Trees created via Bagging
- An ensemble of 10 Naïve Bayes classifiers created via Bagging
- An ensemble built by stacking with 1 Naïve Bayes and 1 Decision Tree base learners and a Naïve Bayes Meta-learner
- An ensemble built by stacking with 1 Naïve Bayes and 1 Decision Tree base learners and a Decision Tree Meta-learner

After getting the initial results, I tried several other experiments to see what would happen, using the WEKA explorer on the best performing classifiers and this allowed me to get more detailed information about each run and visualizations of performance.

## Results

This legend denotes which learner corresponds to which classifier ID in the figures that follow.

Legend			
 1	Decision Tree	 6	Voting between Naïve Bayes, Decision Trees, and Majority Classifier
 2	Decision Tree with Reduced Error Pruning	 7	Bagging for 10 Decision Trees
 3	Naïve Bayes	 8	Bagging for 10 Naïve Bayes
 4	Boosted Naïve Bayes	 9	Stacking Naïve Bayes and Decision Tree with Naïve Bayes meta-learner
 5	Boosted Decision Trees	 10	Stacking Naïve Bayes and Decision Tree with Decision Tree meta-learner

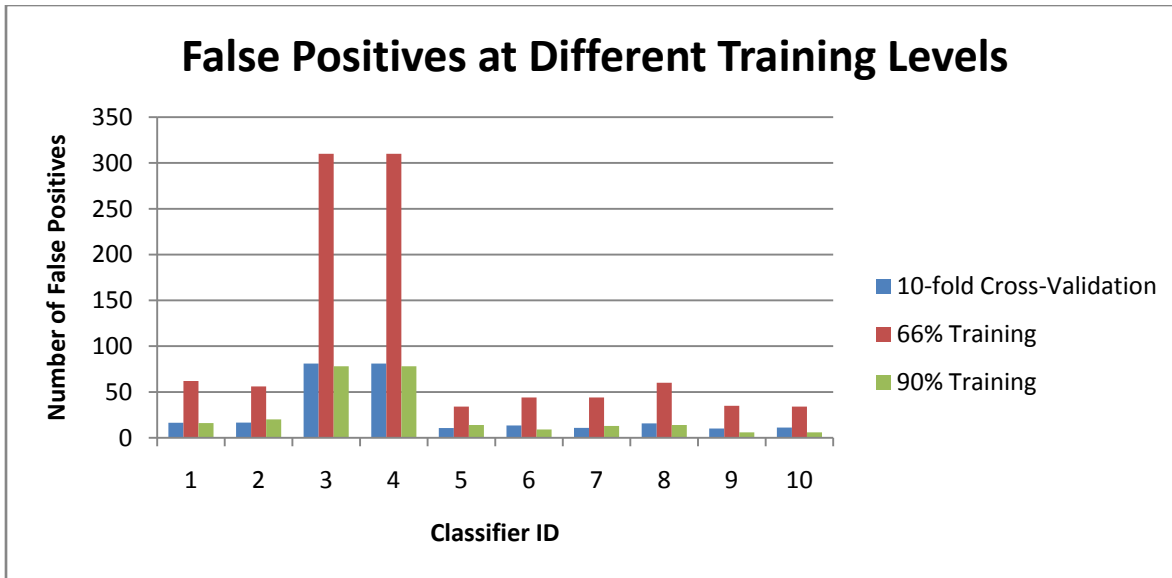


Figure 1: Number of False Positives per classifier for each type of training (lower is better)

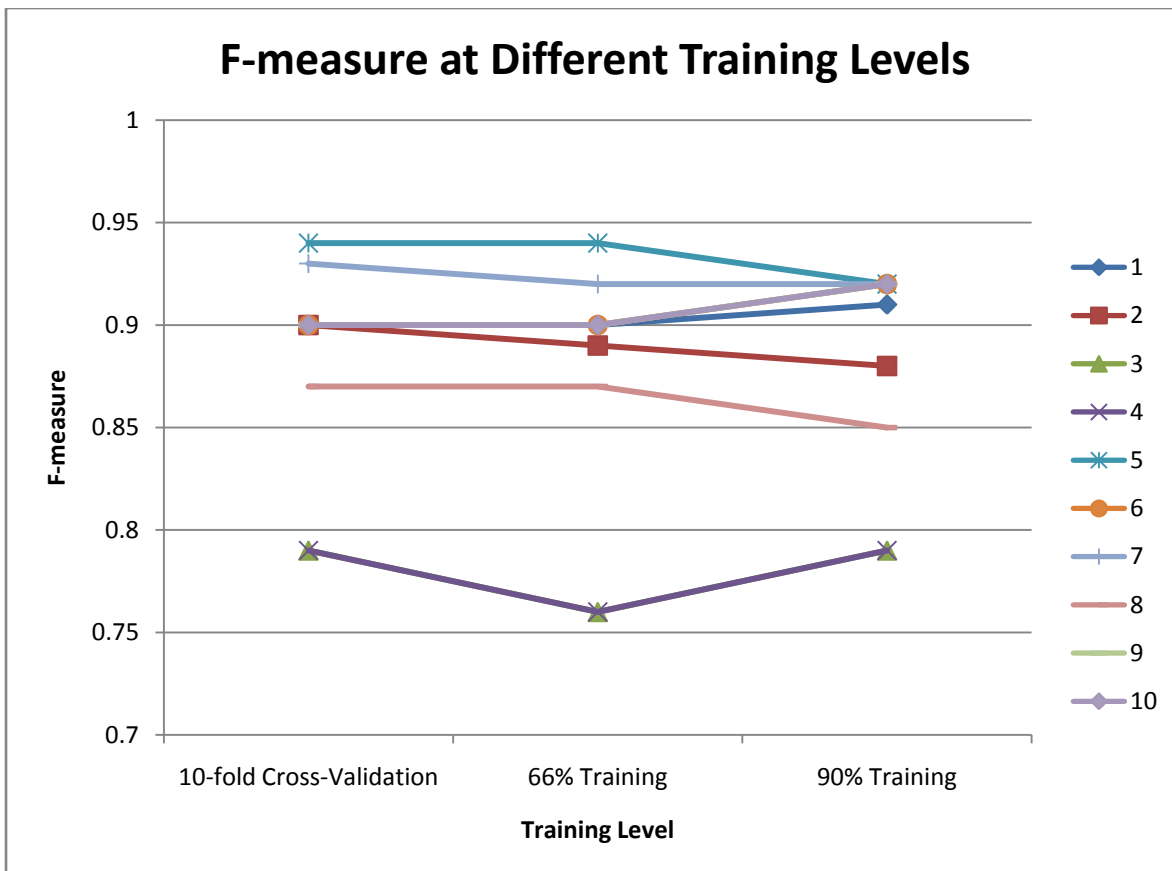


Figure 2: F-measure of each classifier for each type of training (higher is better)

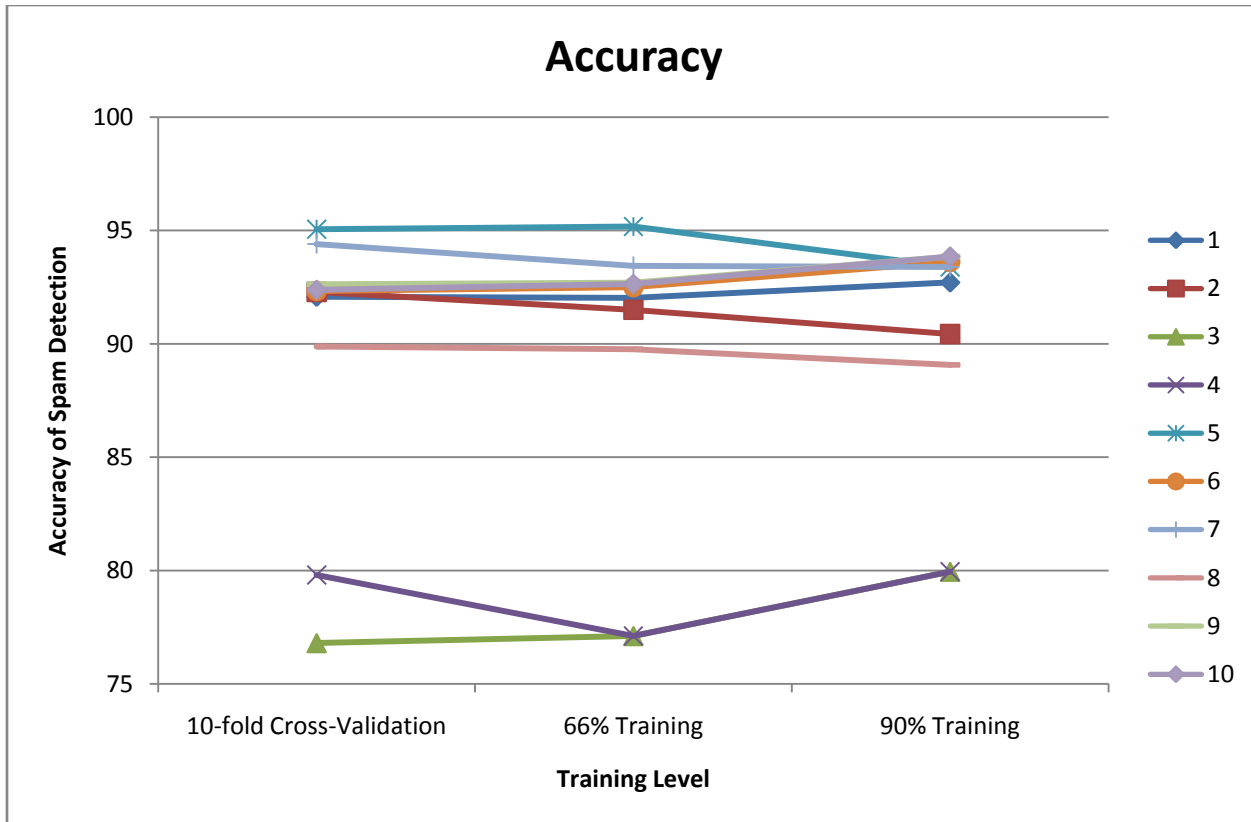


Figure 3: Accuracy of each classifier at identifying spam (higher is better)

I have attached an excel spreadsheet containing the results for this experiment, providing numbers such as F-measures, Precision/Recall figures, number of false positives, and accuracy at identifying spam. I chose to include graphs of the F-measure and accuracy here because they are general metrics to help us determine which learners perform best and the False Positives graph is particularly important because marking a legitimate e-mail as spam is highly undesirable (see spambase-docs.txt).

### Discussion & Conclusion

Overall, the Boosted Decision Trees (ID: 5) performed the best with an accuracy of 95.06% on the 10-fold cross-validation test and an F-measure of .94 on the same task. It consistently performed very well for each type of training, be it 10-fold cross validation, when the training was 66% of the total data or 90% of the total. This was very surprising to me because I had heard that the standard spam filtering technique is to use Naïve Bayes classifiers. Furthermore, using a Boosted Naïve Bayes classifier (ID: 4) had very bad results, which were basically equivalent to the results for a single Naïve Bayes classifier (ID: 3) and which further surprised me because I had expected it to be comparable to the Boosted Decision Tree classifier. Perhaps decision trees performed generally well simply because they fit data very well and it would be interesting to experiment on another held-out set of spam to see if the trees might be overfitting (Note: using reduced error pruning resulted in slightly lower performance compared to the regular Decision Tree, except that it had fewer false positives at the 66% training level). If we had access to the raw data, it would also be instructive to be able to go through the feature selection phase and compare performance with various feature sets.

When we examine false positives, we find that the boosted decision trees again perform near the top (2<sup>nd</sup> place with 10.7 false positives on the 10-fold cross-validation evaluation), although they are slightly beaten by the stacked learner, consisting of 1 Naïve Bayes and 1 Decision tree and a Naïve Bayes meta-learner (ID: 9), which had 10.2 false positives for the same evaluation. Unfortunately, the stacked learner did not perform as well in general (it is in the top 4). False positives are of particular interest because marking good mail as spam is very bad and we may prefer a learner with lower false positives even if they allow more spam through. (Note: the dataset description says 0 false positives meant 20-25% of spam messages passed through, which is also bad).

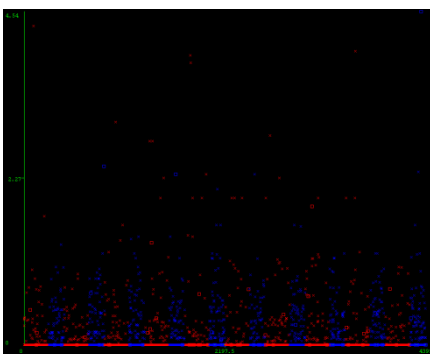
Given the overall excellence of the Boosted Decision Tree, I wondered why the spam filtering industry uses Naïve Bayes classifiers (which according to my experiments did not generally perform well). After some searching online I found a book titled *Programming Collective Intelligence* by Toby Segaran (p 284), in which the author explains that Decision Tree Algorithms are not practical for spam filters because they do not support incremental training (though active research is going into algorithms for decision trees that support incremental training). Therefore, as spammers adapt and new e-mail messages come in, the decision tree cannot adapt to the new data, but is stuck with its original training data. Naïve Bayes classifiers on the other hand support incremental training and are able to keep up with the latest attacks without having to be rebuilt with each new e-mail message. Therefore, the conclusion seems to be that although decision trees generally perform very well, they are not practical for spam filtering applications because they can easily become obsolete and cannot be continually rebuilt to take into account the continuous stream of new data.

Perhaps a viable short-term alternative could be to use a stacked learner composed of a boosted decision tree and a naïve bayes classifier with a naïve bayes meta-learner. The meta-learner and Naïve Bayes classifier can be incrementally trained while the boosted decision tree would only be rebuilt once every so often. This may enable us to capture the high performance of decision trees while maintaining the adaptability and robustness of Naïve Bayes classifiers (the meta-learner can also adapt to each new decision tree we build).<sup>i</sup>

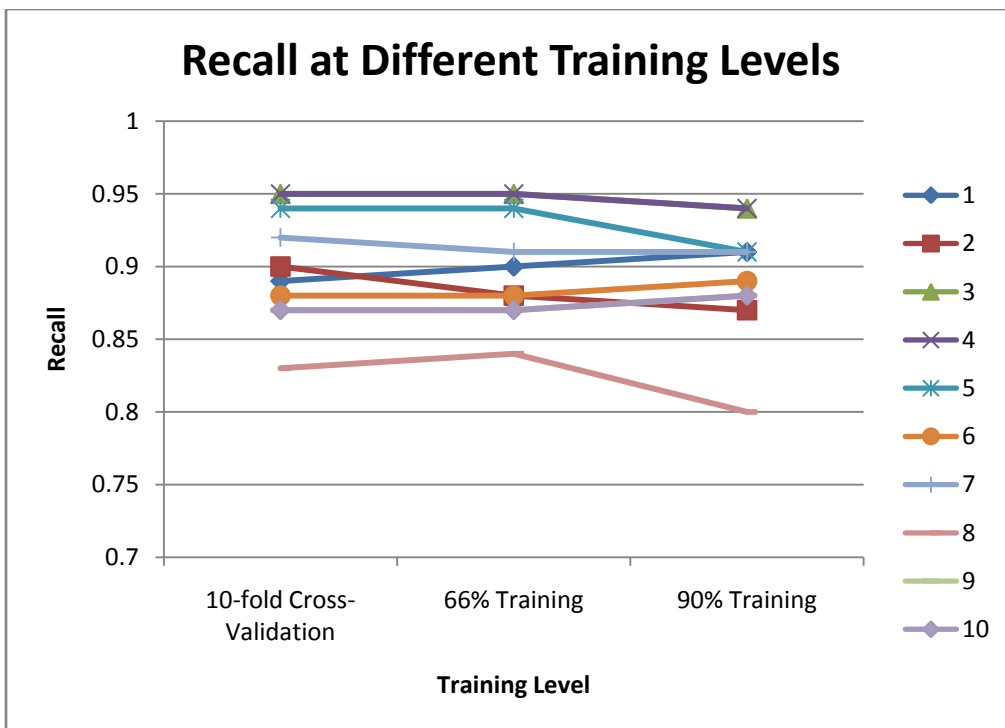
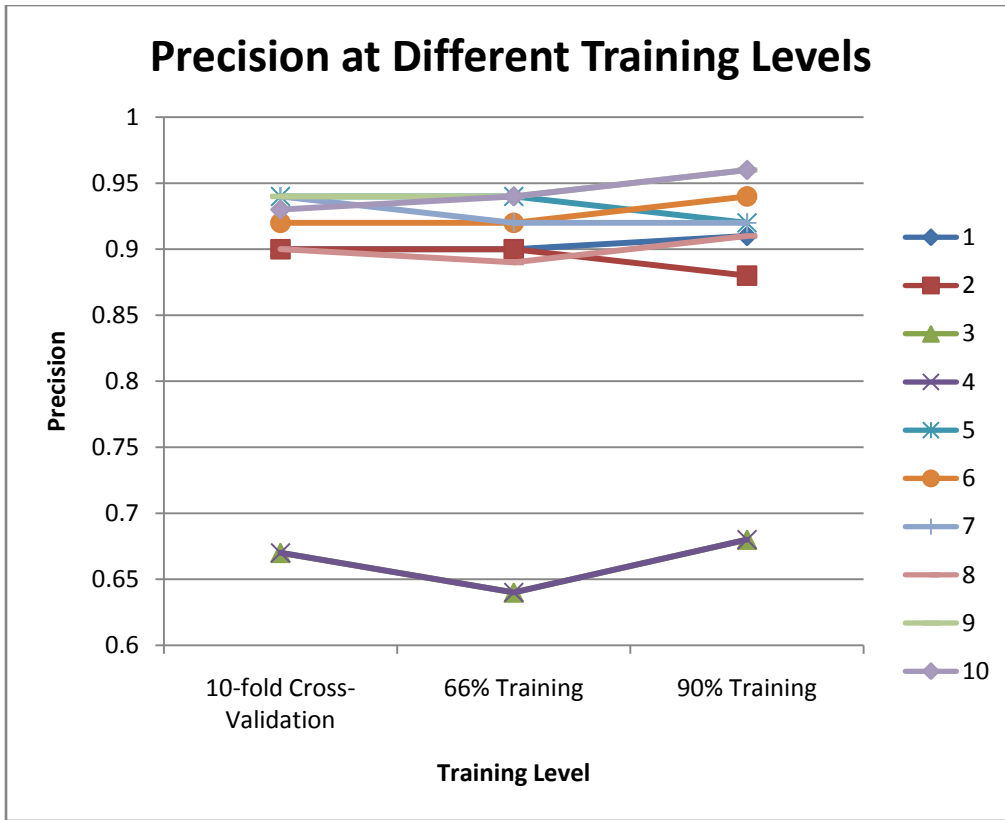
Overall, I had a good experience using WEKA. Although it was initially intimidating, the included documentation was sufficient to get me started and I began to realize the power of having such a convenient tool at my disposal; WEKA enabled me to try many interesting experiments with a short turnaround time, which I would never have conceived of otherwise. Although it could have better documentation, visualizations and output formats, my experience using WEKA was very instructive and beneficial. SDG.

## Appendix

Here are some additional figures.



This is a sample visualization from the WEKA explorer. It can show misclassification errors based on each instance compared to a particular feature. Although the graph could be more attractive, it is already useful as is.



<sup>i</sup> This is an endnote because of the 6 page limit. I ran an experiment using a stacked learner with a Naïve Bayes meta learner, boosted decision tree and naïve bayes base learners. It performed slightly better than plain boosted decision trees.