Morgan Dixon
CS 573 Mini-Project

For this project, I implement a Naïve Bayes classifier to classify spam email. I implement my classifier within the Weka framework and compare my implementation to few decision tree-based learning algorithms implemented in Weka. There is already a Naïve Bayes classifier within Weka, however I implement my own to get a better understanding of the algorithm. Since the spam data that I am using contains real valued attributes, values must be discretized for the classifier to work well. Weka includes a few different methods for discretizing data for their Naïve Bayes classifier, and I use each of these discretizers within my algorithm to compare and contrast their effectiveness. Please note that I did not implement the discretizers, I use those provided in Weka. To evaluate my algorithm and to compare it to other algorithms within Weka, I perform a 10-fold cross-validation on my classifier and each classifier that I test. The training/test data that I use is from http://archive.ics.uci.edu/ml/datasets/Spambase. A description of the attributes and the training data is on the website. In the data, there are 4601 total instances, 39.4% of which is labeled as spam.

The write-up of my work is as follows. First I explain my implementation of the Naïve Bayes classifier, and the different discretization algorithms that are used. I then compare the Naïve Bayes classifier on with these different algorithms against each other, and then I compare these to other existing algorithms within Weka and explain the results.

I've included the Java code for my project – It requires the Weka package. I've also included a jar file that can be executed at the command prompt. To run my code in the command prompt, type:

```
java –jar MorgansNaiveBayes.jar –t spam_data.arff –D|-K|<Empty String>
```

Where –D discretizes the attributes by learning the binary split, –K uses the kernel density estimator (I will explain how this works shortly) on the real valued attributes, and if you do not enter an option, then by default the classifier will assume a normal distribution of each attribute.

The three discretization methods work as follows. First, the default case rounds each attribute to the nearest 1/100th and places a normal distribution over all of the rounded data. This probability distribution is then used to predict the likelihood of a given attribute value when classifying an instance. In the binary split case, the attribute is converted into a single binary attribute. For example, for a real valued attribute that ranges from 0 to 1, the binary split discretizer may lump all values as below or above 0.3. The way this works is by computing the best threshold (the threshold that divides the data into spam/not spam as best as possible). Finally, Kernel density estimator is similar to the default method except that instead of assuming the data falls within a normal distribution, the distribution is approximated and smoothed given the sample data.

Below is a graph of the accuracy, precision, and recall of my Naïve Bayes classifier, and three decision tree algorithms (one is an ensemble using decision trees) used on the spam data. I chose to look at the performance of decision trees just because I was curious how a simple classifier such as Naïve Bayes would compare to some of the other common learning algorithms that are available within Weka.

Algorithm/Results	Accuracy (correct/total)	Precision - Spam	Precision- Not Spam	Recall– Spam	Recall- Not Spam
ADTree	92.132	0.909	0.929	0.889	0.942
Bagging (Using REPTree)	94.307	0.937	0.948	0.919	0.96
My Naïve Bayes Binary Split	89.15	0.901	0.897	0.835	0.94
My Naïve Bayes Kernel Density Estimator	76.347	0.628	0.983	0.983	0.621
My Naïve Bayes Normal Distribution	79.28	0.666	0.956	0.951	0.69

The first thing to notice in my data is that my Naïve Bayes algorithm using the Binary split discretizer performs better than the other two Naïve Bayes classifiers. This makes sense, as most of the attributes in the data are percentages of particular words within an email. So we can imagine it to be common that certain words are very frequent in spam and not frequent in normal email (or vice versa), implying that it is often likely that there is a natural threshold in the occurrence of words between spam and non-spam emails. Surprisingly, however, the Kernel Density Estimator method had a worse accuracy than the Normal Distribution method. This may be simply because the distribution of attribute values are better approximated by a normal given our data set.

The decision tree algorithms that I tested work as follows. First, the ADTree is built by iteratively creating decision and prediction nodes. Decision nodes discriminate between positive and negative examples, and prediction nodes specify a value to add to a tallying score as we move through the tree. When we reach a leaf, the total score gives us the classification of the data. The bagging ensemble method, which we discussed in class works by creating *m* training sets from a given set of training data, sampling examples uniformly from each set with replacement, and then the *m* models are combined. The REPTree is simply a decision tree that is pruned to minimize errors (branches are pruned until harmful).

Both decision tree-based classifiers had higher accuracy (and typically better precision and recall) than the Naïve Bayes classifiers. This was expected, since the Naïve Bayes classifier assumes independence between all of the attributes, while in decision trees, this is not necessarily true. On the other hand, given the simplicity of a Naïve Bayes classifier, the accuracy, precision, and recall are remarkably high. This demonstrates the surprising power of Naïve Bayes classifiers that assume each attribute of a class is independent of every other attribute, which is rarely true in real-life situations. The ensemble method performed the best out of all of the algorithms. An interesting future test would be to evaluate the same ensemble method where it instead uses my Naïve Bayes classifier instead of the REP tree, and see how it compares to the other algorithms. In summary, I would say that I am most impressed with the strength of Naïve Bayes classifiers given their simplicity.