Josh Veltri (jkv13)
Justin Wang (jsw104)

Programming Assignment 4

(a) *For all problems and any learning algorithm, compare the accuracy of the ensemble versions (10 iterations) to the base learners. Produce a table with the accuracies of the base learner, the bagged learner, and the boosted learner. Perform paired t-tests to determine if any of the ensemble methods are significantly better than the base learner with 95% confidence.*

Using nbayes:

	Voting	Volcanoes	Spam
Base Learner	Fold Accuracies 0.929412 0.941860 0.988372 0.918605 0.908046	Fold Accuracies 0.628378 0.664414 0.619369 0.594595 0.651685	Fold Accuracies 0.694459 0.681030 0.673691 0.679759 0.682189
Bagging	Fold Accuracies 0.917647 0.941860 0.988372 0.918604 0.931034	Fold Accuracies 0.668919 0.675676 0.650901 0.605856 0.683146	Fold Accuracies 0.625401 0.625427 0.625385 0.625427 0.625386
Boosting	Fold Accuracies 0.929412 0.953488 0.976744 0.918604 0.919540	Fold Accuracies 0.786036 0.722973 0.680180 0.718468 0.730337	Fold Accuracies 0.670704 0.673336 0.668540 0.669521 0.674093

Paired t Tests:

Voting

Base Learner vs. Bagging: The 95% confidence interval for (BaseLearner_accuracy-Bagging_accuracy) is -0.01979 to 0.01348. This interval contains zero, so there is no difference at the 95% confidence level between the base and bagged learner accuracy (paired by fold).

Base Learner vs. Boosting: The 95% confidence interval for (BaseLearner_accuracy-Boosting_accuracy) is -0.01433 to 0.00974. This interval contains zero so there is no difference at the 95% confidence level between the base and boosted learner accuracy (paired by fold).

Volcanoes

Base Learner vs. Bagging: The 95% confidence interval for (BaseLearner_accuracy-Bagging_accuracy) is -0.04167 to -0.008748. This interval is entirely below zero, so the bagged learner is superior at the 95% confidence level to the base learner.

Base Learner vs. Boosting: The 95% confidence interval for (BaseLearner_accuracy-Boosting_accuracy) is -0.1497 to -0.04207. This interval is entirely below zero, so the boosted learner is superior at the 95% confidence level to the base learner.

Spam

Base Learner vs. Bagging: The 95% confidence interval for (BaseLearner_accuracy-Bagging_accuracy) is 0.04741 to 0.0662. This interval is entirely above zero, so the base learner is superior at the 95% confidence level to the bagged learner.

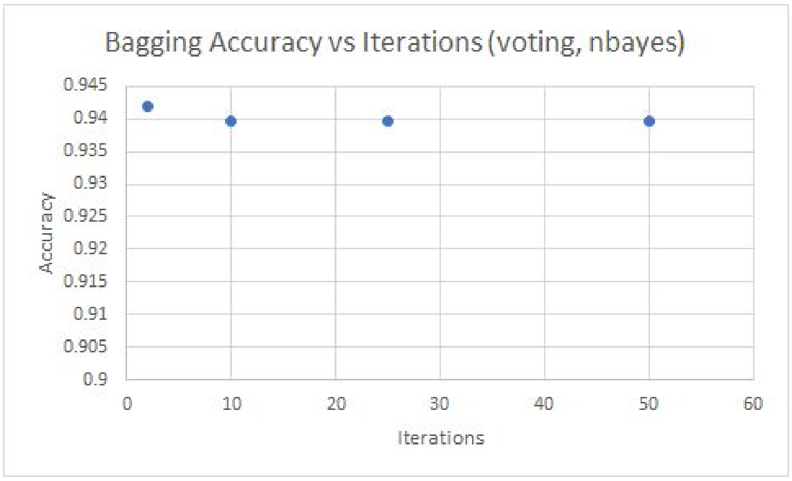
Base Learner vs. Boosting: The 95% confidence interval for (BaseLearner_accuracy-Boosting_accuracy) is 0.00184 to 0.02013. This interval is entirely above zero, so the base learner is superior at the 95% confidence level to the boosted learner.

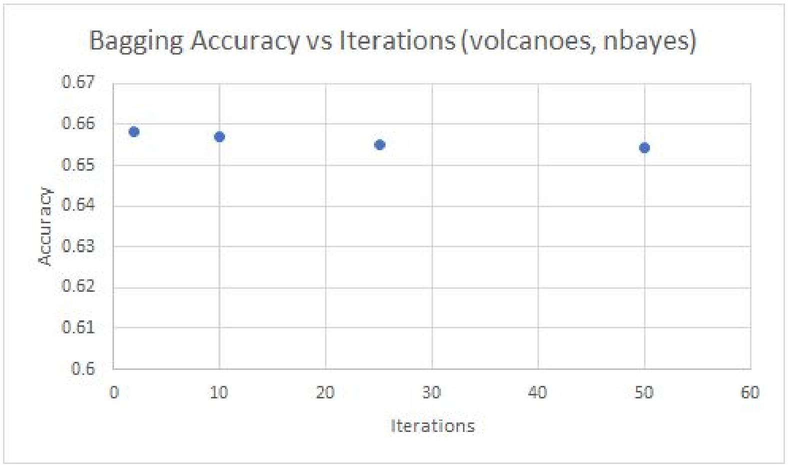
(b) *For any two problems and any learning algorithm, evaluate how the accuracy of bagging changes with the number of iterations. Pick at least three iteration values between 2 and 50, and plot the accuracy on a graph. Do you see any difference by problem?*

Accuracies using nbayes

	2 iterations	10 iterations	25 iterations	50 iterations
voting	0.94188	0.93956	0.93956	0.93956
volcanoes	0.65829	0.65690	0.65509	0.65420

For bagging, the accuracies are nearly constant for all different numbers of iterations on both voting and volcanoes.



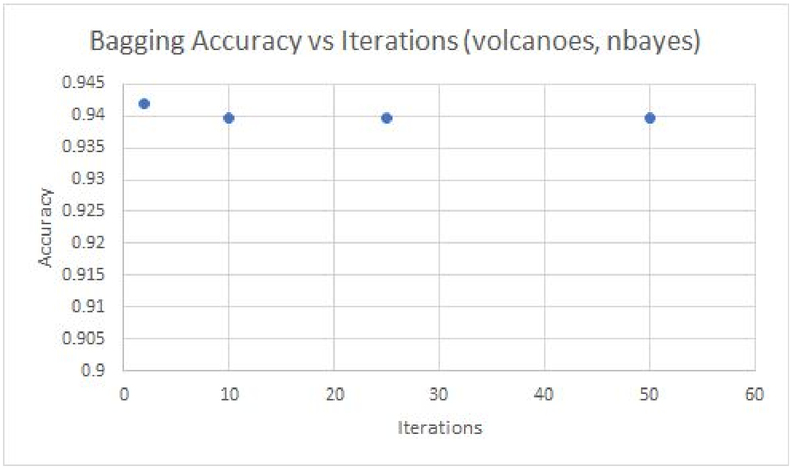


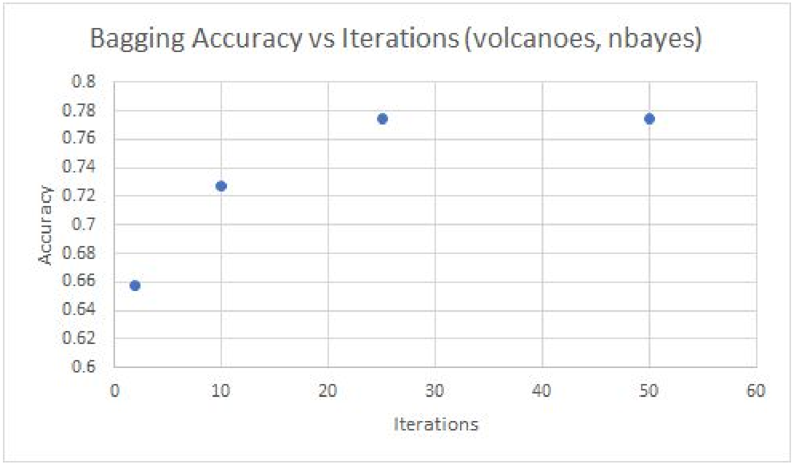
(c) *Repeat (b) for boosting.*

Accuracies using nbayes

	2 iterations	10 iterations	25 iterations	50 iterations
voting	0.94188	0.93956	0.93956	0.93956
volcanoes	0.65781	0.72760	0.77487	0.77487

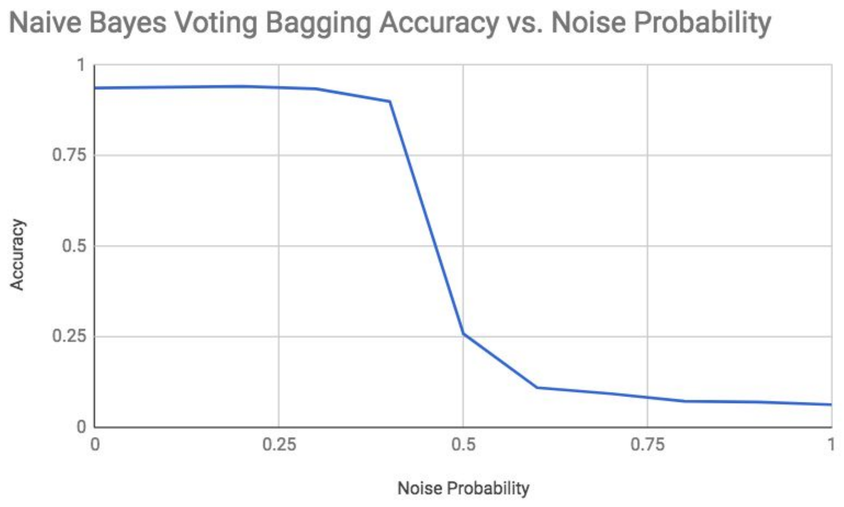
For boosting, the pattern varies by problem. On voting, the accuracy is approximately constant regardless of the number of boosting iterations. On volcanoes, the accuracy increases from 0.65781 with 2 iterations to 0.77487 with 25 iterations and plateaus there.

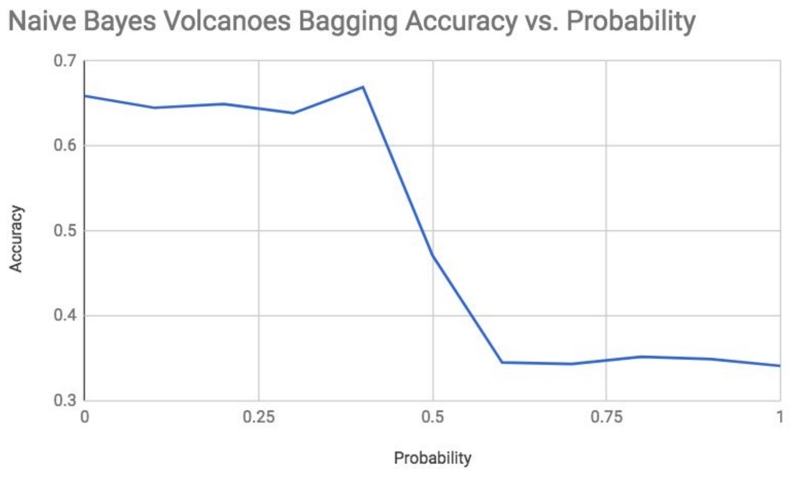




(d) *Evaluate the sensitivity of bagging to noise as follows. When training, after constructing the training sample, flip an example's label with probability p. Then use this noisy sample in your bagging algorithm and evaluate the resulting classifier on the usual (noise free) test set. For any two problems and any learning algorithm, plot a graph with p on the x-axis and the test-set accuracy of bagging (30 iterations) on the y-axis. You can use results from the previous questions for a p=0 point. Discuss how resilient bagging is to noise based on your observations.*

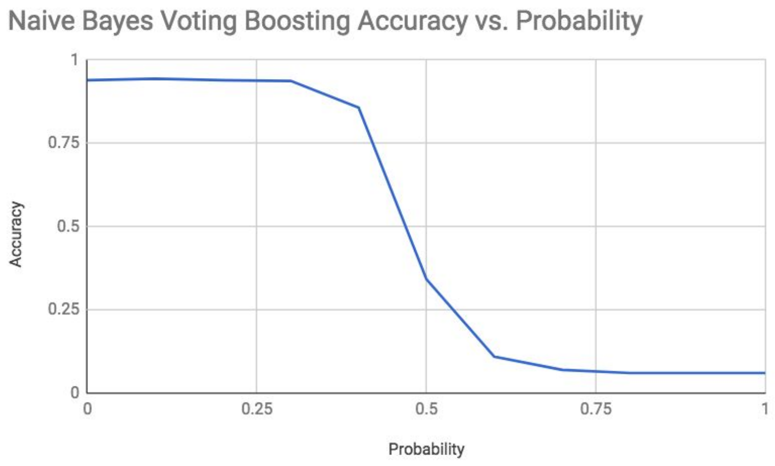
We found from our experiments that Bagging was extremely resilient to noise in the training set. So much so that there is little to no effect on the accuracy until sharply between 0.4 and 0.6 probability in which most of the examples are classified incorrectly and the ensemble algorithm actually begins predicting at a high accuracy the opposite label.

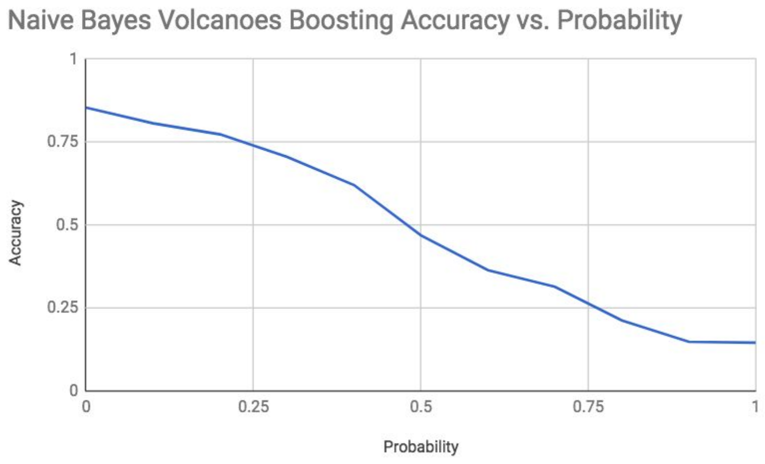




(e) *Repeat (d) for boosting.*

Boosting is fairly resilient to noise in the training set, although not nearly as resilient as bagging. We still see the same pattern of relatively small decrease in accuracy with increased noise, and then a sharp drop around the midpoint, but the slope is much more steady and there's much less of a steep drop at the midpoint. This is much more prominent on the volcanoes data set than on the voting data set.





(f) *Write down any further insights or observations you made while implementing and running the algorithms, such as time and memory requirements, the complexity of the code, etc.*

Bagging increases the computation time requirement linearly with the number of iterations. This is because a separate classifier is trained for each iteration. It therefore also increases the amount of memory required to store classifiers linearly with the number of iterations. Whether this is actually a significant amount of memory varies between learning algorithms. Some, like logistic regression require very little memory to store a classifier. Others, like decision trees, can require quite a lot of memory.

Boosting, similarly, requires training one classifier per iteration and therefore the time and memory requirements increase linearly with the number of iterations. Boosting is, overall, more expensive (both in terms of time and memory) than bagging because example weights must be stored and the example and classifier weight computations involve a large number computationally expensive exponentiation and logarithms.

We needed to refactor the base learners quite a bit to use a common interface so that we could use the same bagging and boosting code with each. This ultimately worked out quite well. We ended up with two wrapper classes, BaggedClassifier and BoostedClassifier, that both expose the same train() and evaluateExample() methods we use with the base classifiers and contain and make use of an instance of the specified base classifier. This allowed us to also use nearly the same ‘main’ as with previous assignments.

The difficulty of modifying base classifiers to handle weighted data varied somewhat from learner to learner, but was generally pretty easy overall. The dtree and nbayes learners presented the most difficulty because we had been using a simple array to store the features and classification and needed to transition to an Example class that contained the feature array as well as the example weight. This required a large number of very simple changes throughout the classifiers. We did not encounter this difficulty with ann and logreg because when we normalize our examples for use with those classifiers, were already placing the attributes within another object. We were able to simply add the example weight to this object.