CS6375 - ML - Project Assignment #2

Submitted by Kapil Gautam – KXG180032

(Bagging) Construct four models for each combination of maximum depth d = 3; 5 and bag size (k = 10; 20). Report the confusion matrix for these four settings.

Depth: 3 Bag Size: 10
Test Error = 3.76%.

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	1153	43
	Negative	34	816

Depth: 3 Bag Size: 20
Test Error = 3.76%.

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	1153	43
	Negative	34	816

Depth: 5 Bag Size: 10
Test Error = 0.15%.

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	1193	3
	Negative	0	850

Depth: 5 Bag Size: 20
Test Error = 0.15%.

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	1193	3
	Negative	0	850

(Boosting) Construct four models for each combination of maximum depth d = 1; 2 and bag size (k = 20; 40). Report the confusion matrix for these four settings.

Depth: 1 Bag Size: 20
Test Error = 9.34%.

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	1037	159
	Negative	32	818

Depth: 1 Bag Size: 40
Test Error = 9.34%.

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	1037	159
	Negative	32	818

Depth: 2 Bag Size: 20
Test Error = 4.89%.

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	1100	96
	Negative	4	846

Depth: 2 Bag Size: 40
Test Error = 4.89%.

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	1100	96
	Negative	4	846

(Scikit-learn) Use scikit-learn's bagging and AdaBoost learners and repeat the experiments as described in parts (a) and (b) above. Report the confusion matrices for these sets of settings. What can you say about the quality of your implementation's performance versus scikit's performance?

Note - In binary classification in sklearn, the count of true negatives is C[0][0] , false negatives is C[1][0], true positives is C[1][1] and false positives is C[0][1].

Bagging:

Depth : 3 Bag Size : 10 test error (%): 3.910068426197455 Confusion matrix: [[846 4] [76 1120]]	**Depth : 3 Bag Size : 20** test error(%): 4.740957966764414 Confusion matrix: [[846 4] [93 1103]]
Depth : 5 Bag Size : 10 test error(%): 0.0 Confusion matrix: [[850 0] [0 1196]]	**Depth : 5 Bag Size : 20** test error(%): 0.0 Confusion matrix: [[850 0] [0 1196]]

AdaBoost

Depth : 1 Bag Size : 20 test error(%): 0.2932551319648091 Confusion matrix: [[846 4] [2 1194]]	**Depth : 1 Bag Size : 40** test error(%): 0.0 Confusion matrix: [[850 0] [0 1196]]
Depth : 2 Bag Size : 20 test error(%): 0.0 Confusion matrix: [[850 0] [0 1196]]	**Depth : 2 Bag Size : 40** test error(%): 0.0 Confusion matrix: [[850 0] [0 1196]]

Scikit implementation for Bagging is almost the same as my implementation.

For Boosting, Scikit probably uses better decision tree criteria, which gives it a better edge over my implementation.