Homework 4, CS 412 Data Mining
(Assignment 4, report)
Name: Jason Anema, Ph.D.
NetID: *janema2*

Used Python version 2.7.6 and only used built-in libraries (datetime, random, sys).

Brief introduction to classification methods and classification framework:

Decision Tree classifier:
Implemented a decision tree algorithm from scratch (only using built-in libraries) in python. Vertices were split if they had more than a minimum number of data points in their partition, and all data points did not have the same label. The splitting of a vertex was based on the attribute with minimal $Gini_A$ for attributes (so as to maximize the reduction in impurity at each vertex.) If a vertex's dataset had less that a pre-set minimum number of data points, that vertex is labeled as a leaf with label being decided on by majority rules (with random selection to break a tie). If a vertex's dataset all had the same labels, that vertex is labelled as a leaf with that label. A maximal depth of the tree was set to avoid overfitting (and control runtimes). Both the decision tree, and the function to make a prediction were defined recursively. The decision tree was trained on training data, and parameters (max depth of tree, and minimum number of data points in a vertex's dataset) were tuned to maximize accuracy of the decision tree's classification of the test dataset.

Random Forest:
For my Random Forest classifier, I implemented Forest-RI. This means my random forest had each tree trained by data bagging a random sample (with replacement) from the training dataset, and with the same size of the training dataset (counting multiple entries). I also used random attribute selection at each vertex of the tree for deciding which attribute to split on (when a split was called for, which the same conditions as for a decision tree). After building the random forest of decision trees in this way, classification was made on majority rules from the individual trees in the forest' predictions.

Model Evaluation Measures:

Decision Tree:

(Dec. Tree) balance.scale.train
overall accuracy = 0.83

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.933	0.997	0.5	0.037	0.069	0.143	0.045
2	0.868	0.864	0.848	0.871	0.859	0.853	0.866
3	0.86	0.822	0.816	0.904	0.858	0.833	0.885

(Dec. Tree) balance.scale.test
overall accuracy = 0.662

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.893	0.990	0.0	0.0	div by zero	div by zero	div by zero
2	0.707	0.667	0.653	0.755	0.7	0.671	0.732
3	0.724	0.734	0.686	0.713	0.699	0.691	0.707

(Dec. Tree) nursery.train
overall accuracy = 1.0

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0

(Dec. Tree) nursery.test
overall accuracy = 0.976

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.977	0.982	0.963	0.966	0.964	0.964	0.965
2	0.986	0.992	0.724	0.746	0.735	0.728	0.742
3	0.990	0.995	0.988	0.980	0.984	0.986	0.982
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	0.999	0.999	0.0	div by zero	div by zero	div by zero	div by zero

(Dec. Tree) led.train
overall accuracy = 0.859

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.859	0.901	0.772	0.765	0.769	0.771	0.766
2	0.859	0.765	0.897	0.901	0.899	0.898	0.900

(Dec. Tree) led.test
overall accuracy = 0.863

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.863	0.904	0.783	0.772	0.778	0.781	0.774
2	0.863	0.772	0.898	0.904	0.901	0.900	0.903

(Dec. Tree) synthetic.social.train
overall accuracy = 0.590

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.784	0.854	0.556	0.568	0.562	0.559	0.566
2	0.787	0.863	0.579	0.560	0.569	0.575	0.564
3	0.784	0.839	0.571	0.622	0.596	0.581	0.611
4	0.826	0.898	0.664	0.609	0.635	0.652	0.620

(Dec. Tree) synthetic.social.test
overall accuracy = 0.488

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.73	0.824	0.496	0.474	0.485	0.491	0.478
2	0.726	0.838	0.433	0.380	0.404	0.421	0.389
3	0.766	0.818	0.496	0.595	0.541	0.513	0.572
4	0.754	0.838	0.518	0.510	0.514	0.516	0.511

Random Forest:

(Random Forest) balance.scale.train
overall accuracy = 0.923

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.933	1.0	div by zero	0.0	div by zero	div by zero	div by zero
2	0.953	0.930	0.924	0.978	0.950	0.934	0.967
3	0.96	0.925	0.921	1.0	0.959	0.936	0.983

((Random Forest) balance.scale.test
overall accuracy = 0.813

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.902	1.0	div by zero	0.0	div by zero	div by zero	div by zero
2	0.849	0.821	0.804	0.882	0.841	0.818	0.865
3	0.876	0.839	0.823	0.921	0.869	0.841	0.899

(Random Forest) nursery.train
overall accuracy = 1.0

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0

(Random Forest) nursery.test
overall accuracy = 0.981

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.981	0.984	0.968	0.974	0.971	0.969	0.973
2	0.99	0.995	0.816	0.785	0.8	0.81	0.791
3	0.991	0.994	0.988	0.984	0.986	0.987	0.985
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	0.0	div by zero	div by zero	div by zero	div by zero

(Random Forest) led.train
overall accuracy = 0.859

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.859	0.902	0.774	0.762	0.768	0.772	0.764
2	0.859	0.762	0.896	0.902	0.899	0.897	0.901

(Random Forest) led.test
overall accuracy = 0.863

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.863	0.905	0.785	0.769	0.777	0.782	0.772
2	0.863	0.769	0.897	0.905	0.901	0.899	0.903

(Random Forest) synthetic.social.train
overall accuracy = 0.896

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.941	0.969	0.899	0.855	0.876	0.89	0.863
2	0.952	0.986	0.954	0.849	0.898	0.931	0.868
3	0.958	0.966	0.903	0.936	0.919	0.909	0.929
4	0.942	0.941	0.841	0.944	0.89	0.86	0.921

(Random Forest) synthetic.social.test
overall accuracy = .700

class label	Accuracy	Specificity	Precision	Recall	F_1 Score	$F_{.5}$ Score	F_2 Score
1	0.836	0.922	0.739	0.601	0.663	0.707	0.624
2	0.852	0.938	0.754	0.588	0.661	0.714	0.615
3	0.861	0.879	0.667	0.802	0.728	0.69	0.771
4	0.851	0.862	0.67	0.82	0.737	0.695	0.785

Parameters chosen during implementation and why:

Decision Tree:

For each dataset, the parameters:

- max_depth = maximum depth of tree
- min_node_size = minimum size of data partition at a vertex for a split to occur

were set in order to maximize the decision tree's total accuracy (after being trained on the training dataset) on the test data set.

For each dataset, these parameters were chosen to be:

balance.scale:
max_depth = 4 , and min_node_size = 2
nursery:
max_depth = 8 , and min_node_size =0
led:
max_depth = 7 , and min_node_size = 2
synthetic.social:
max_depth = 6 , and min_node_size = 10

Random Forest:

For each dataset, the parameters:

- max_depth = maximum depth of tree
- min_node_size = minimum size of data partition at a vertex for a split to occur
- number_random_attributes = number of attributes to chose from at each vertex of each tree in forest (if there are fewer than this number, use all remaining attributes)
- number_trees = number of trees in the random forest

were set in order to maximize the random forest's total accuracy (after being trained on the training dataset) on the test data set.

balance.scale:
max_depth = 4 , min_node_size = 2, number_random_attributes = 1, number_trees = 200
nursery:
max_depth = 8, min_node_size =0, number_random_attributes = 4, number_trees = 50
led:
max_depth = 7 , min_node_size = 2, number_random_attributes =1 , number_trees = 100
synthetic.social:
max_depth = 6 , min_node_size = 10, number_random_attributes =8 , number_trees = 100

Conclusion on whether ensemble method improves performance of the basic classification method you chose, why or why not:

Summary of overall accuracies for test datasets:

test dataset	Decision Tree (overall acc.)	Random Forest (overall acc.)
balance.scale.test	0.662	0.813
nursery.test	0.976	0.981
led.test	0.863	0.863
synthetic.social.test	0.488	0.700

The ensemble method (Forest-RI) of Random Forest greatly increased the overall accuracy for the test datasets balance.scale and synthetic.social, made minor improvements to overall accuracy for nursery, and performed equally well on led, as compared to the basic classification method of Decision Tree. See the table above for a summary of these performances.

Random Forest (Forest-RI) uses bagging in tandem with random attribute selection. This provides a diversity of decision trees, each of which predicts the class label for a tuple, the Random Forest outputs the majority rules prediction from these trees. That is to say, a tuple will only be misclassified if a majority of trees misclassifies that tuple as another class label. This is why Random Forest tends to be more accurate that a single decision tree, and at least doesn't tend to perform worse than a single decision tree.

On nursery.test, Random Forest only made very marginal improvements to the overall accuracy versus the decision tree. This is likely because the overall accuracy of a decision tree was already to high.

On led.test, Random Forest and Decision trees produced very similar overall accuracies after optimizing each's parameters. One possible explanation for the lack of increase in overall accuracy for Random Forest is the individual trees in Random Forest may still have (even with bagging and random attribute selection) some measure of dependence between them.

Have a nice break!!

Best regards,

Jason