DECISION TREE ,RANDOM FOREST , BOOSTING

Machine Learning Assignment

Submitted by,

Amala Deshmukh –	**2014A7PS541H**
Bhargav Kanuparthi-	**2014A7PS527H**
Karthik Menon –	**2013B3A7487H**
Keyur Jain -	**2015A7PS056H**

DECISION TREE

Pre-processing of Data:

- The initial part of cleaning up the data involved with dealing with missing values. Missing values were replaced with the most frequent element of that attribute or the mode.
- Then all the string values that the attributes took were replaced by numbers. If an attribute could take '**x**' strings it was replaced by numbers from 0 to x-1.
- Now the continuous values were split into two intervals about the value which gave the highest information gain.

Structure of Code:

Each node of the Decision Tree is stored as a structure called **node** . Each node has 5 elements :

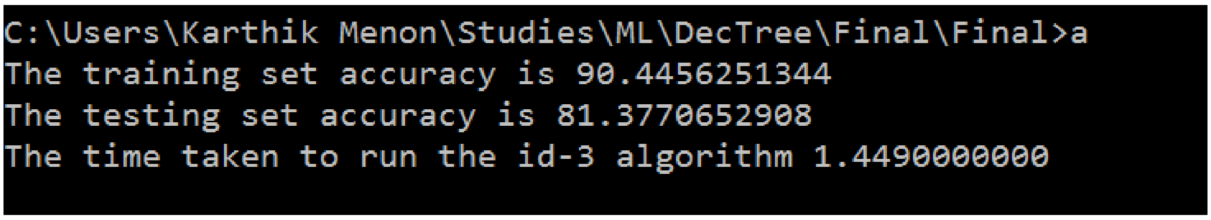
- **vector <node *> child –** is a vector containing pointers to the all the children of the current node .
- **vector <int> left –** The indices of the dataset that are valid for the current node .
- **int used[] –** The set of attributes already used by the current nodes ancestors .
- **int attr_no –** An integer variable that stores which attribute the next split in the decision tree will be based on.
- **int out –** An integer variable that stores the output of the decision tree if that node is a leaf node.

Functions:

- **init() –** used to initialize various variables
- **information_gain() –** used to calculate the information gain for a given attribute
- **getmaxgain() –** returns the attribute number which gives the maximum information gain if the node were to be split on that attribute .

- **id3() -** The id3 algorithm was implemented to train the decision tree. In the id3 algorithm we first search for an attribute that gives us the maximum information gain. If no such attribute exists or the attribute was already split by one of its ancestors the node was marked as a leaf node and its value was assigned to 1 or 0 based on their counts in the remaining dataset. If an attribute existed we created the children nodes for each value that the selected attribute took and partitioned the remaining dataset accordingly. After this the id3 algorithm was now called recursively on all the children and the entire tree was created.
- **getoutput() -** would take an input and classify it accordingly by traversing the trained tree.
- **getAccuracy() –** this returns the accuracy on the training data.
- **getAccuracyTest() –** this returns the accuracy on the testing data .

Snapshot of the Accuracy and Training time for the ID-3 algorithm



```
C:\Users\Karthik Menon\Studies\ML\DecTree\Final\Final>a
The training set accuracy is 90.4456251344
The testing set accuracy is 81.3770652908
The time taken to run the id-3 algorithm 1.4490000000
```

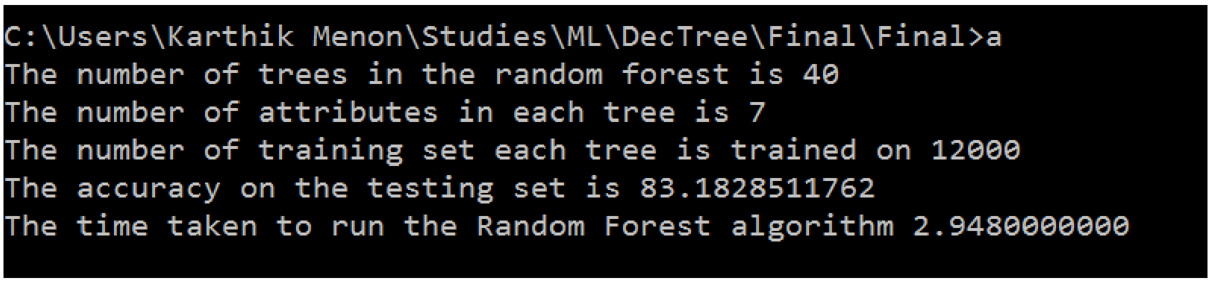
RANDOM FOREST

Additional Functions used in Random forest implementation

- **random_forest() –** This function creates **n** trees for the random forest and trains each tree on random training data , and choosing random attributes for each tree .

- **getAccuracyTest() –** This function is different from the getAccuracyTest function used in the Decision tree code , as the output of the Random forest is taken as the mode of the all the outputs given by the individual trees used in the Random forest .

Snapshot of the various Hyper-parameters , Accuracy and Training time for the Random Forest algorithm:



```
C:\Users\Karthik Menon\Studies\ML\DecTree\Final\Final>a
The number of trees in the random forest is 40
The number of attributes in each tree is 7
The number of training set each tree is trained on 12000
The accuracy on the testing set is 83.1828511762
The time taken to run the Random Forest algorithm 2.9480000000
```

BOOSTING

The boosting algorithm used is adaptive boosting.

Additional Functions used in Random forest implementation

- **getAccProbability() –** This function generates a random according to the probability distribution of the instances of the dataset (given by the weights).
- **formClassifier() -** This function picks samples with replacement from the entire dataset according to their weights and forms a classifier for the same. It them computes the coefficient for the classifier and updates the weights of all the instances in the dataset.
- **adaBoost() -** This function assigns a uniform probability distribution to all the instances in the dataset and forms L classifiers.
- **getBoostedOutput() -** This functions gets the output for an input instance for each classifier. The output of the Boosted tree then depends on the sign of the sum of the product of the output of each classifier and the corresponding coefficient.

COMPARISON

Algorithm	Accuracy	Training Time
Decision Tree	81.377	1.449s
Random Forest	83.182	2.948s
Boosting	82.204	56.124s