# MACHINE LEARNING ASSIGNMENT 2

TEAM MEMBERS:

* DHURV TOSHNIWAL - 2015A7PS0177H
* SAURABH KHANDELWAL - 2015A7PS0082H
* ABHIRAM GULANIKAR - 2015A7PS0105H

**Brief Introduction of ID3 Algorithm**

The ID3 algorithm begins with the original set {\displaystyle S} as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the set {\displaystyle S} and calculates the entropy (or information gain) of that attribute. It then selects the attribute which has the smallest entropy (or largest information gain) value. The set {\displaystyle S} is then split by the selected attribute to produce subsets of the data. The algorithm continues to recurs on each subset, considering only attributes never selected before.

**PSEUDOCODE**

ID3 (Examples, Target\_Attribute, Attributes)

Create a root node for the tree

If all examples are positive, Return the single-node tree Root, with label = +.

If all examples are negative, Return the single-node tree Root, with label = -.

If number of predicting attributes is empty, then Return the single node tree Root,

with label = most common value of the target attribute in the examples.

Otherwise Begin

A ← The Attribute that best classifies examples.

Decision Tree attribute for Root = A.

For each possible value, *vi*, of A,

Add a new tree branch below Root, corresponding to the test A = *vi*.

Let Examples(*vi*) be the subset of examples that have the value *vi* for A

If Examples(*vi*) is empty

Then below this new branch add a leaf node with label = most common target value in the examples

Else below this new branch add the subtree ID3 (Examples(*vi*), Target\_Attribute, Attributes – {A})

End

Return Root

**FUNCTIONS**

* *public double calc\_entropy() and public double calc\_entropy(double a,double b)*

- Using these function, we calculate the entropy of each node. Here, we have used OOP features polymorphism and method overloading.

This method is called by infogain method.

* *public double infogain(int x)*

- Using this function, we calculate the information gain achieved on fixing a particular attribute at the node. It achieves this by using the entropy method.

* *public void generate\_children(int attribute,double infogain\_attr)*

- This function takes maximum infogain and depending upon the distinct number of attributes, creates that many children of a node

* *public int classify(ArrayList<Integer> test)*

- This function is important for calculating the label of each leaf node and comparing it to the corresponding value in the training data. This function helps in the calculation of accuracy, precision and recall.

* *public static void datapre(String line,int i,int comma\_no,int start,int data\_no,int x) and public static void data\_preprocess(String file)*

*-* These two functions are responsible for the data preprocessing on our adult.data and test.data files. Here, we are targetting on inputting the entire in an arrayList and satisfying any missing data fields using the average values.

**ANALYSIS**

**ID3 ACCURACY: 76.72%**

**Reduced Error Pruning: 79.13%**

**Initializing Random Forests with 4 trees and .5 fraction of attributes and .25 fraction of training instances.**

**Random Forest Accuracy: 88.50%**

**ID3** grows each branch of tree just deeply enough to perfectly classify training samples. It overfits the data when there is noise in the data or when the number of training examples is too small to produce a representative sample of the true target function.

In **Reduced Error Pruning**, when pruning begins; tree is at maximum size and lowest accuracy over test set. As pruning proceeds, number of nodes is reduced and accuracy over test set increases. When data is limited, number of samples available for training is further reduced due to the partition of training set into actual training set and validation set.

**Random Forest** model works well even when predictive features contain irrelevant features (or noise); it can be used when the number of features is much larger than the number of samples. However, with randomizing mechanism in both bagging samples and feature selection, RFs could give poor accuracy when applied to high dimensional data. The main cause is that, in the process of growing a tree from the bagged sample data, the subspace of features randomly sampled from thousands of features to split a node of the tree is often dominated by uninformative features (or noise), and the tree grown from such bagged subspace of features will have a low accuracy in prediction which affects the final prediction of the RFs.