Machine Learning (BITS F464)
Assignment 1
(Decision Tree, Random Forest, Boosting Techniques)

Documentation

Team Members:

1. Arram Saichand 2014A7PS046H
2. B.Siva Naga Sasank 2014A7PS050H
3. Kampara Sriteja: 2014A7PS149H
4. Eda Amos William Prasada Rao 2014A8PS467H

Assignment Files:

1.ID3.java

2.Mapper.java

3.Gini.java

4.DataSet.java

5.RandomForest.java

6.AdaBoost.java

7.Pair.java

1.ID3.java

-It is comprised of 4 classes.

1)Instance

2)Node

3)ID3

4)Split

Purpose: Creates a decision tree based on given training data and tests on test data and calculates accuracy.

Usage:

javac ID3.java

java ID3

Classes:

1.1. Instance Class:

Purpose: Encapsulates the information about all the instances.

Methods:

i) **public** Instance(String line, **int** id) : Class constructor takes in an instance from file and processes it

1.2 Node Class:

Purpose: Encapsulates information about each Node in tree and its parent and children.

Methods:

1. Node(Node parent, List<Instance> instances) : Initialises its variables with given arguments.
2. **public int** classify(Instance t) : Classifies the instances after building the tree. Takes object of Instance as input and classifies it.

1.3 Split Class:

Purpose: This class is used to find the split values for Continued value attributes using Gini Index.

Methods:

i) Split() : In this constructor we are populating outputs in an array and passing it to other function.

1.4 ID3 Class:

Purpose: The main class which generates the tree. It has main method in it from where execution begins.

Methods:

i) **public double** calc(**int** a, **int** b) : It is present in abstract class ImpurityFunction where it will calculate entropy taking values a and b.

ii) **public** Node generate(List<Instance> instances, ImpurityFunction f) : This function starts the process of generating tree by taking list of instances and impurity function as inputs and returns Node of the tree.

iii) **void** expand(Node node, ImpurityFunction impurityFunction, **int** depth) : This function is used to expand the tree by taking node and impurity function and depth as inputs .This function is recursively called to expand the whole tree.

iv) **public void** learn(List<Instance> instances) : This calls **generate** method taking list of instances as input.

v) **public** List<Integer> classify(List<Instance> testInstances) : This method is used to classify the instances by calling **classify** method of Node class.

vi) **public static void** load(String trainfile, String testfile,
List<Instance> trainInstances, List<Instance> testInstances):

Method used to load training instances and testing instances from train and test files. Takes file names and List of instances as inputs.

vii) **public static double** computeAccuracy(List<Integer> predictions,
List<Instance> testInstances) :
Method used to compute accuracy of the tree by taking predictions and test instances as input.

2.Mapper.java

-Has only 1 class Mapper.

2.1 Mapper.class:

Purpose: A utility class which maps the features of an attribute to corresponding integer values.

Methods:

i) Mapper() : This maps the features of every attribute to some integer value using HashMap.

3.Gini.java

-Has Gini class.

3.1 Gini Class

Purpose: Class Gini to calculate split values for continous attributes.

Methods :

i) **public double** gini(**int** a[],**int** b[]) : A method to calculate gini value for given values and return split value with lowest gini-index.
ii) **public void** populate(Pair[][] table,**double** value,**int** i) : A method to populate the table array for each split value. Takes 2D array of Pair object as input ,double value and an integer.
iii) **public double** gini_index(Pair p1,Pair p2) : A method to calculate Gini Index. It takes two objects of pair as inputs and calculates Gini index.
iv) **public int** bestsplit(**double** a[]) : A method to return best split value which has minimum giniindex.Takes array of gini indices and returns an index with split value.

4.Pair.java

-Has pair class

Purpose: Utility class to inplement a pair data structure.

Methods:

i) Pair () : Given an integer and double as parameters it holds them in a pair for easy access.

5.Dataset.java

-Has Dataset class

Purpose :Used for pre-processing the dataset and storing them in list structures for accessing.

Methods:

i) Dataset () : Used for processing the dataset and storing the instances after removing spaces and specific attribute values of instances by tokenizing the instance in the required data structures.
ii) **public static** ArrayList<String> getInstance() : Used for storing the instances in a data structure.

6.RandomForest.java

-Has RandomForest class which extends the ID3 class.

Purpose :Used to implement the RandomForest algorithm.

Methods:

i) RandomForest () : Used for generating 4 random numbers for choosing the attributes.
ii) **void** expand**(Node** node**, ImpurityFunction** impurityFunction**, int** depth**)**: Used for selecting the best attribute which has the maximum information gain. It is also used to grow the tree recursively.
iii) **public static void** load(**List**<Instance> trainInstances): Used to load the instances from the dataset randomly. Also some instances may be occurring more than once.
iv) **public static void** loadTest(**String** testfile,List<Instance> testInstances): Used to load the test instances.
v) **public void** treeDecision(**List**<Instance> testInstances):
vi) public static void main(String []args): Used to call the above methods and also for displaying the accuracy.

7.AdaBoost.java

-Has AdaBoost class which extends the RandomForest class.

Purpose: Used to boost the trees in the RandomForest for better performance.

Methods:

i) AdaBoost () : Used to initialize the weights of the training instances to some constant equal value.
ii) **public static void** calcWeightedError(**int** d) : Used to calculate the weighted errors of the instances.
iii) **public static int** calcWHat() : Used to calculate $w^{\wedge}$ for each iteration of the adaboost algorithm.
iv) **public static void** updateAlpha(**int** minindex) : Used to update the weights of all the instances after one iteration of the adaboost.

v) **public static** List<Integer> adaBoost(): Used to classify the instances after completion of the adaboost algorithm.

vii) **public static void** main(**String** []args) : Used to call the above methods and also for displaying the accuracy.