CSE 6363
MACHINE LEARNING
PROGRAMMING ASSIGNMENT – 1

NAME: SAGAR SHARMA
UTA ID: 1001626958
NET ID: SXS6958

Language Used: **Python 3**

File included:

- c45.py
- main.py
- iris.data.txt
- iris.names.txt
- housevotes84.data.txt
- housevotes84.names.txt
- ReadMe.pdf

Note: Please keep all the files in the same folder

Commands to run the program from the windows command line:

python main.py "splitting/validation method option"

Command examples

For implementing the **80-20 split** method for train-test data sets, type the following command

python main.py 80-20

For implementing the **3 fold cross validation method as asked by the professor**, type the following command

python main.py 3fold

Note: if you do not provide the last option properly explained in the above two formats, an error will be raised

There is a prompt for entering the **file names**:

Note: Please enter the file names exactly as mentioned below

for the iris data set
data file name: **iris.data.txt**
header/names file name: **iris.names.txt**

for the housevotes84 data set
data file name: **housevotes84.data.txt**
header/names file name: **housevotes84.names.txt**

Program flow explained:

def __init__(self, datafile, namesfile):
#used to define the data objects we are going to use through the whole program
#contains the path to the data comma seperated file

def get_format_data(self):
fetch the data set and info about the data set from the files
#and retrieve and store the data set in a particular format according to feature data types and class labels in the domain
#open the file which contains the class labels info and feature-value info for reading purpose

def preformatdata(self):
#convert the elements in the rows of the data set, if the features in the data set are continuous

def postformatdata(self):
convert the elements in the rows of the data set after training the d tree,
#if the features in the data set are continuous

def Print_Tree(self):
#print the nodes in the trees, each node represents a rule

def Print_Node(self, node, lev = 0 , indent=""):
#prints all the nodes returned by the generate tree

def Generate_Tree(self):
generate the d tree using the training data set and the atribute names list
gets nodes returned from recursiveGenerateTree method

def Recursive_Tree(self, current_Data, current_Attributes):
the d tree gets build from bottom to top recursively
    # the DT builds from bottom to top

def Majority_Class(self, current_Data):
find the majority class in the train examples

def All_Same_Class(self, w_data):
#check if all the samples have the same class label
    # check if all examples correspond to one class label

def Is_Attribute_Discrete(self, attribute):
check for if the attribute in the data set is discrete or not

def splitAttribute(self, current_Data, current_Attributes):
#given train data set and attribute names list, return the best attribute
    #which has the highest information gain. return the atribute name, threshold if it is a continuous
    #and the splitted train data based on that attribute value names

def gain(self,MainSet, data_subsets):
    #compute the info gain for an attribute

def entropy(self, W_Data_Set):
    #compute the entropy for a attribute-value pair given a corresponding data set representing the particular attribute-value pair

def log(self, x):
    # log to the base 2 method

def test_data(self):
method to apply rules formed by the decision tree on the validation set and calulcating the accuracy for each turn
    # retrieves the rules formed by the decision trees using regular expressions

class Node:
#node class to create node objects for d trees