**ASSIGNMENT 1**

**DECISION TREE USING C4.5 Algorithm**

**Interfaces:**

**\_\_init\_\_(self, pathToData,pathToNames):** it is a constructor for objects

**getData(self):** This function fetch data from ‘iris.names’ which stores label names and features and type of feature variables.

Parameters: Self: it is an object of the class.

**preprocessData(self):** This function performs discretization to the continuous attributes.

Parameter: Self: it is an object of the class.

**printTree(self):** This function prints the tree using printNodeDetails(self, node, indent="") function.

Parameter: Self: it is an object of the class.

**printNodeDetails(self, node, indent=""):**  This function prints the tree, creates path of the tree taking threshold into consideration .

Parameter: Self: it is an object of the class.

Node: Node of a tree which starts with root and recursively takes decision for every node.

**createTree(self):** This function creates the tree calling CreateTreeRecursively(self, curData, curAttributes)

Parameter: Self: it is an object of the class.

**CreateTreeRecursively(self, curData, curAttributes):**  This function is recursively called creating a tree for all the nodes that belong to same class.

Parameter:

Self: it is an object of the class.

curData: current data

curAttributes: attributes for the current data input.

**getMajorityClass(self, curData):** this function checksmajority of classes in the current data set.

Parameter: Self: it is an object of the class.

curData: current data

**isAllIdenticalClass(self, data):** This function checks if predicted class and actual class is same.

Parameter: Self: it is an object of the class.

Data: test data.

**isAttrDiscrete(self, attribute):** This function check whether the attribute is discrete or not.

Parameter: Self: it is an object of the class.

Attribute: Attribute to be checked.

**splitAttr(self, curData, curAttributes):** For each attribute, this function finds the normalized information gain ratio from splitting on that attribute.

Parameter: Self: it is an object of the class.

curData: current data.

**info\_gain(self, unionSet, subsets):** This function calculates information gain for every node.

**entropy(self, dataSet):**  This function calculates entropy for every attribute.

Parameter: Self: it is an object of the class.

dataset: current dataset

**log(self, x):** This function calculated the log to base 2

Parameter: Self: it is an object of the class.

X: value

**get\_index(self, label):** This function returns the index of the attribute w.r.t the label passed.

Parameter: Self: it is an object of the class.

Label: class label

**predict(self, values):** This function calculates prediction for every node

Parameter: Self: it is an object of the class.

Values: current row of the data

**predictRecursive(self, node, values):** This function calculates prediction for every node.

Parameter: Self: it is an object of the class.

Node: current node

Values: current row of the data

**cal\_accuracy(self, testData)**: This function calculates the accuracy on the test data.

Parameters:

Self: it is an object of the class.

testdata: test data matrix.

**createAndStoreData(train, filename)**: This function to store the test and train data into a text file so that we can give it as input while printing the tree.

Parameters:

train: train data matrix

filename: Path and filename that will be created by the function in the system.

**c45\_using\_80\_20\_dataSplitting(dataset):**  splits the data into training and testing where 80% is training data and 20% is test data. C4.5 algorithm is applied on this train and test data.

**c45\_using\_kfold(dataset, k):** splits the data into k folds and performs c4.5 algorithm on the folds k times.

**cross\_validation\_split(dataset, folds=3):** This function divides the data into k folds.

**Steps of experiment:**

1. Split data into train and test using k-fold and data splitting function 80-20.
2. Create C45 object with train data and labels.
3. Fetch Data – calling getdata()
4. Pre Processing the data like discretization of data.
5. Generate tree recursively using printTree()
6. Calculating the accuracy by predicting the classes and comparing it with the actual class.

**Output:**

--------------------------------------------------------

Decision tree prediction using k fold cross validation:

Tree 0 : -------------------------------------------------

sepal width <= 0.8 : Iris-setosa

sepal width > 0.8 :

petal length <= 4.95 :

petal width <= 5.050000000000001 : Iris-virginica

petal width > 5.050000000000001 : Iris-versicolor

petal length > 4.95 :

sepal length <= 2.75 :

petal width <= 5.85 : Iris-virginica

petal width > 5.85 : Iris-versicolor

sepal length > 2.75 : Iris-virginica

93.33333333333333

Tree 1 : -------------------------------------------------

sepal width <= 0.8 : Iris-setosa

sepal width > 0.8 :

petal length <= 4.95 :

petal width <= 5.15 : Iris-virginica

petal width > 5.15 : Iris-versicolor

petal length > 4.95 :

sepal length <= 2.75 :

petal width <= 5.85 : Iris-virginica

petal width > 5.85 : Iris-versicolor

sepal length > 2.75 : Iris-virginica

93.33333333333333

Tree 2 : -------------------------------------------------

sepal width <= 0.8 : Iris-setosa

sepal width > 0.8 :

petal length <= 5.35 :

petal width <= 5.050000000000001 : Iris-virginica

petal width > 5.050000000000001 :

sepal length <= 2.75 : Iris-versicolor

sepal length > 2.75 : Iris-versicolor

petal length > 5.35 : Iris-virginica

90.0

Tree 3 : -------------------------------------------------

sepal width <= 0.8 : Iris-setosa

sepal width > 0.8 :

petal length <= 4.85 : Iris-versicolor

petal length > 4.85 :

sepal length <= 2.75 :

petal width <= 5.85 : Iris-virginica

petal width > 5.85 : Iris-versicolor

sepal length > 2.75 : Iris-virginica

96.66666666666667

Tree 4 : -------------------------------------------------

sepal width <= 0.7 : Iris-setosa

sepal width > 0.7 :

petal length <= 4.95 :

petal width <= 5.050000000000001 : Iris-virginica

petal width > 5.050000000000001 : Iris-versicolor

petal length > 4.95 :

sepal length <= 2.75 :

petal width <= 5.85 : Iris-virginica

petal width > 5.85 : Iris-versicolor

sepal length > 2.75 : Iris-virginica

93.33333333333333

Average accuracy : 93.33333333333333

--------------------------------------------------------

Decision tree prediction using 80-20 data split:

Tree : -------------------------------------------------

sepal width <= 0.8500000000000001 : Iris-setosa

sepal width > 0.8500000000000001 :

petal length <= 5.35 : Iris-versicolor

petal length > 5.35 : Iris-virginica

Accuracy: 89.65517241379311

--------------------------------------------------------