



Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19MN4C2)

AY: 2022-23

Experiment 4

(Decision Tree)

Name: Ayush Jain

SAP ID: 60004200132

Aim: Implement Decision Tree on the given Datasets to build a classifier and Regressor. Apply appropriate pruning method to overcome overfitting.

Theory:

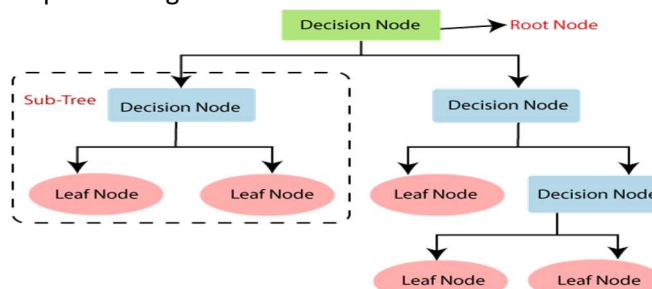
Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome**. In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node**.

Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees. Below diagram explains the general structure of a decision tree:



Decision Tree Terminologies

Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.



Department of Computer Science and Engineering (Data Science)

Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

Branch/Sub Tree: A tree formed by splitting the tree.

Pruning: Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

Steps in building a Tree

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

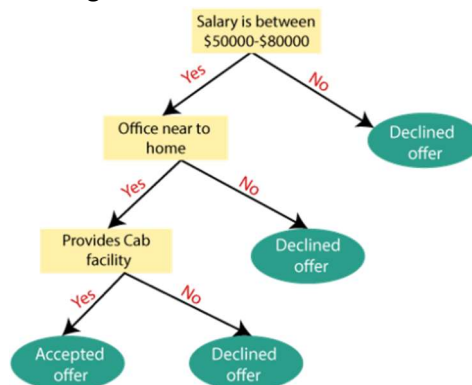
Step-2: Find the best attribute in the dataset using **Attribute Selection Measure (ASM)**.

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM**. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

1. Information Gain:

Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class.



Department of Computer Science and Engineering (Data Science)

According to the value of information gain, we split the node and build the decision tree.

A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)]

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data.

Entropy can be calculated as:

$$\text{Entropy}(s) = -P(\text{yes})\log_2 P(\text{yes}) - P(\text{no})\log_2 P(\text{no})$$

Where,

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

2. Gini Index:

Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.

An attribute with the low Gini index should be preferred as compared to the high Gini index.

It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum P_j^2$$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree. A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- Cost Complexity Pruning
- Reduced Error Pruning.

Lab Assignments to complete in this session:

Use the given dataset and perform the following tasks:

Dataset 1: Iris.csv

1. Implement Decision tree classifier from using Dataset 1 by defining Node class and Tree class.
2. Use python libraries to build a decision tree classifier on Dataset 1. Analyze the results using confusion matrix and accuracy.
3. Write a code to show overfitting in the decision tree classifier built using Dataset 1. Use sklearn and matplotlib.

**Department of Computer Science and Engineering (Data Science)**

Code:

```

import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

```

```

data=load_iris()
df=pd.DataFrame(data=data.data,columns=data.feature_names)
df['target']=pd.Series(data.target)
df

```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

```

x= df.iloc[:,1:-1].values
y= df.iloc[:, -1].values
print(x)
print(y)

```

```

[[3.5 1.4 0.2]
 [3.  1.4 0.2]
 [3.2 1.3 0.2]
 [3.1 1.5 0.2]
 [3.6 1.4 0.2]
 [3.9 1.7 0.4]
 [3.4 1.4 0.3]
 [3.4 1.5 0.2]
 [2.9 1.4 0.2]
 [3.1 1.5 0.1]
 [3.7 1.5 0.2]
 [3.4 1.6 0.2]
 [3.  1.4 0.1]
 [3.  1.1 0.1]
 [4.  1.2 0.2]
 [4.4 1.5 0.4]
 [3.9 1.3 0.4]
 [3.5 1.4 0.3]
 [3.8 1.7 0.3]
 [3.8 1.5 0.3]
 [3.4 1.7 0.2]
 [3.7 1.5 0.4]
 [3.6 1.  0.2]
 [3.3 1.7 0.5]
 [3.4 1.9 0.2]
 ...

```

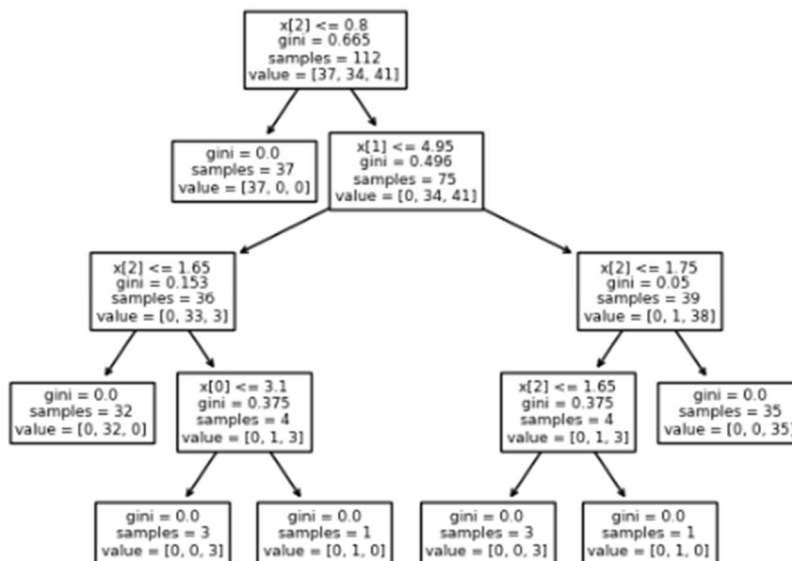
**Department of Computer Science and Engineering (Data Science)**

```
▶ x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
print(x_train)
```

```
● [[3. 4.2 1.5]
 [2.6 4. 1.2]
 [3. 5.5 2.1]
 [3.2 1.3 0.2]
 [3.1 5.1 2.3]
 [3.5 1.6 0.6]
 [3.7 1.5 0.2]
 [2. 3.5 1. ]
 [3. 5.5 1.8]
 [3.3 5.7 2.5]
 [2.2 5. 1.5]
 [2.5 5.8 1.8]
 [2.5 3.9 1.1]
 [3. 6.1 2.3]
 [3.3 4.7 1.6]
 [2.4 3.8 1.1]
 [2.7 4.9 1.8]
 [2.8 5.1 1.5]
 [2.5 4.5 1.7]
 [2.5 5. 1.9]
 [3.2 4.7 1.4]
 [3. 5.2 2. ]
 [3.4 4.5 1.6]
 [3.1 1.6 0.2]
 [2.7 5.1 1.9]
 [2.7 4.2 1.3]
 [2.9 3.6 1.3]]
```

```
▶ from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf=clf.fit(x_train,y_train)
tree.plot_tree(clf)
```

```
● [Text(0.4, 0.9, 'x[2] <= 0.8\ngini = 0.665\nsamples = 112\nvalue = [37, 34, 41]'),
 Text(0.3, 0.7, 'gini = 0.0\nsamples = 37\nvalue = [37, 0, 0]'),
 Text(0.5, 0.7, 'x[1] <= 4.95\ngini = 0.496\nsamples = 75\nvalue = [0, 34, 41]'),
 Text(0.2, 0.5, 'x[2] <= 1.65\ngini = 0.153\nsamples = 36\nvalue = [0, 33, 3]'),
 Text(0.1, 0.3, 'gini = 0.0\nsamples = 32\nvalue = [0, 32, 0]'),
 Text(0.3, 0.3, 'x[0] <= 3.1\ngini = 0.375\nsamples = 4\nvalue = [0, 1, 3]'),
 Text(0.2, 0.1, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(0.4, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.8, 0.5, 'x[2] <= 1.75\ngini = 0.05\nsamples = 39\nvalue = [0, 1, 38]'),
 Text(0.7, 0.3, 'x[2] <= 1.65\ngini = 0.375\nsamples = 4\nvalue = [0, 1, 3]'),
 Text(0.6, 0.1, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(0.8, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.9, 0.3, 'gini = 0.0\nsamples = 35\nvalue = [0, 0, 35]')]
```

**Department of Computer Science and Engineering (Data Science)**

```
[ ] pred=clf.predict(x_test)
print('Predicted values for test set are: \n',pred)
print('Target values of test set are: \n',y_test)
```

Predicted values for test set are:

```
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 2]
```

Target values of test set are:

```
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 1]
```

```
[ ] from sklearn.metrics import confusion_matrix,accuracy_score
cf=confusion_matrix(clf.predict(x_test),y_test)
acc= accuracy_score(clf.predict(x_test),y_test)
print('accuracy is: ',acc)
print('confusion matrix is: \n',cf)
```

accuracy is: 0.9736842105263158

confusion matrix is:

```
[[13  0  0]
 [ 0 15  0]
 [ 0  1  9]]
```

Conclusion: We successfully implemented Decision Tree on the given dataset to build a classifier and Regressor.