ML

23

Name : Preksha A. Patel
Sapid : 60004210126
Branch : Computer Engineering
Div : C2, Batch - 1

Experiment no. 4

Aim : To implement CART decision tree algorithm.

Theory :

The CART algorithm is a type of classification algorithm that is required to build a decision tree based on Gini's impurity index. It is a basic machine learning algorithm & provides a wide variety of use cases.

CART is an umbrella word that refers to the following types of decision trees:

- classification Trees : when the target variable is continuous, the tree is used to find the "class" into which the target variable is most likely to fall.
- Regression Trees : these are used to forecast the value of a continuous variable.

A decision tree is a technique used for predictive analysis in the fields of statistics, data mining, and machine learning. The predictive model here is the decision tree & it is employed to progress from observations about an item that is represented by branches & finally concludes simplicity, decision trees are among the most popular machine learning methods.

The CART algorithm does that by searching for the best homogeneity for the sub nodes, with the help of the Gini Index criterion. The root node is taken as the training set & is split into two ways by considering the best attribute & threshold value. Further, the subsets are also split using the same logic.

This continues till the last pure sub-set is found in the tree or the maximum number of leaves possible in that growing tree. This is also known as Tree Pruning.

The formulae used to find Gini index are
To calculate the Gini value for the entire dataset -
$Gini(D) = 1 - \Sigma(P_i)2mi = 1$

To calculate the Gini value for a particular attribute of the dataset -
$Gini\ A(D) = |D1|/|D|\ Gini(D1) + |D2|/|D|\ Gini(D2)$

Reduction in impurity -
$\Delta Gini(A) = Gini(D) - Gini\ A(D)$

Advantages of CART algorithm:
1. The CART algorithm is nonparametric; thus, it does not depend on information from a certain sort of distribution.
2. The CART algorithm combines both testings with a test data set and cross-validation to measure the goodness of fit more precisely.
3. CART allows one to utilize the same variables many times in various regions of the tree. This skill can reveal intricate interdependencies between groups of variables.
4. Outliers in the input variable have no meaningful effect on CART.
5. One can loosen halting restrictions to allow decision trees to overgrow and then trim the tree down to its ideal size. This method reduces the likelihood of missing essential structure in the data by terminating too soon.
6. To choose the input set of variables, CART can be used in combinations with other prediction algorithm.

eg:	Age	Gender	sportive
	22	F	yes
	24	M	yes
	30	F	yes
	31	F	no
	27	F	no
	32	M	no
	25	F	yes
	30	M	no
	24	F	yes
	21	F	yes
	29	M	yes
	26	M	no
	21	M	no

→ divide data in binary like F or M and age =< 25 or age > 25.

1. F& yes = F - yes = 5, F & no = F-NO = 2
2. M & yes = M-yes = 2, M & no = M-NO = 4
3. less than 25 & yes = Less - Yes = 5, less than 25 & no = Less-no = 1
4. Greater than 25 & yes = Greater-yes = 2, Greater than 25 & no = Greater-no = 5
5. Total = 13
6. Total-of-F = 7
7. Total-of-M = 6
8. Total-of-Less = 6
9. Total-of-Greater = 7

To find Gini,

Gini-of-F = $1 - (F\text{-}Yes)^2 - (F\text{-}NO)^2 = 1 - (5/7)^2 - (2/7)^2$

$= 1 - 0.5101 - 0.08162$

$= 0.40828$

$$\text{Gini-of-M} = 1 - (\text{M-yes})^2 - (\text{M-no})^2 = (1) - (2/6)^2 - (4/6)^2$$
$$= 0.445$$

$$\text{Gini-of-Less} = 1 - (\text{Less-yes})^2 - (\text{Less-no})^2 = 1 - (5/6)^2 - (1/6)^2$$
$$= 0.02776$$

$$\text{Gini-of-Greater} = 1 - (\text{Greater-yes})^2 - (\text{Greater no})^2 = (1) - (2/7)^2 - (5/7)^2$$
$$= 0.40828$$

To find Gini Index,

$$\text{Gender} = \left(\frac{\text{Total-of-F}}{\text{Total}}\right) \times (\text{Gini-of-F}) + \left(\frac{\text{Total-of-M}}{\text{Total}}\right) \times (\text{Gini-of-M})$$

$$= \left(\frac{7}{13}\right) \times 0.40828 + \left(\frac{6}{13}\right) \times 0.445$$

$$= 0.2198 + 0.2053$$

$$= 0.4251$$

$$\text{Age} = \left(\frac{\text{Total-of-Less}}{\text{Total}}\right) \times (\text{Gini-of-Less}) + \left(\frac{\text{Total-of Greater}}{\text{Total}}\right) \times (\text{Gini-of greater})$$

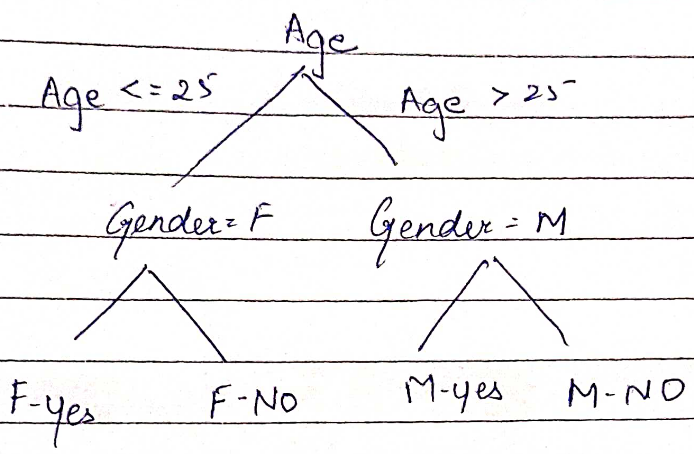
$$= \left(\frac{6}{13}\right) \times 0.02776 + \left(\frac{7}{13}\right) \times 0.40828$$

$$= 0.01281 + 0.2198$$

$$= 0.2326$$

Lowest gini Index is the answer. So our root node in decision tree will be lowest gini index node i.e. Age.

So the CART tree is given by :-



Conclusion :-

In Conclusion, CART (classification and Regression Trees) is a powerful algorithm in machine learning that builds a decision tree by recursively splitting the data based on feature values to make predictions for classification or regression tasks. It efficiently partitions the data into subsets, creating a tree structure where each node represents a decision based on features, leading to leaf nodes that provide the final prediction. CART's ability to handle complex datasets & mitigate overfitting through techniques like pruning makes it a versatile & widely used tool in predictive modeling.

MQ
27/2/24

Name:-Preksha Ashok Patel
Sapid:-60004210126
Branch:-Computer Engineering
Div:-C2,Batch:-1

MACHINE LEARNING

EXPERIMENT NO.4

CODE AND OUTPUT:-

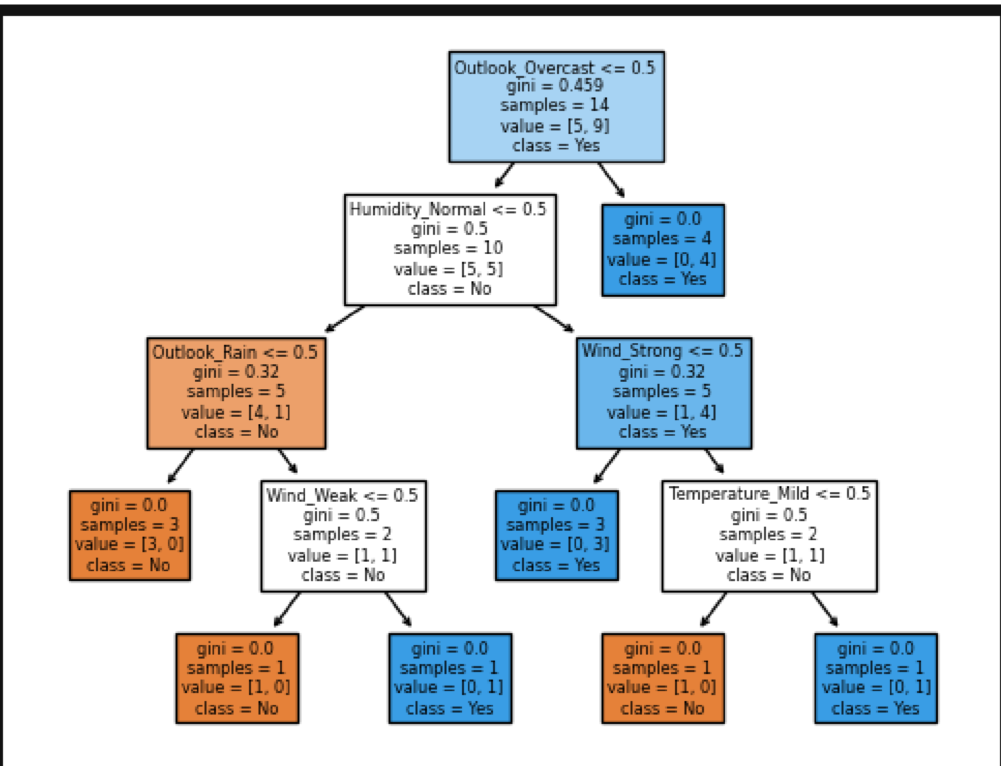
```
[12] import pandas as pd
     from sklearn import tree
     import matplotlib.pyplot as plt
```

```
dataset = pd.read_csv('/content/PlayTennis.csv')
```

```
[14] def count_unique_values(attr):
         unique_vals = pd.unique(attr)
         num_unique_vals = len(unique_vals)
         val_counts = attr.value_counts()
         return num_unique_vals, val_counts, unique_vals
```

```
[15] def calculate_gini(num_unique_vals, val_counts, rows, class_labels, unique_vals_attr, attr, class_label):
         gini_attr = 0
         type_cl_count = 0
         type_count = 0
         gini_values = []
         div_index = 0

         if num_unique_vals == 2:
             for i in range(len(unique_vals_attr)):
                 temp = dataset.loc[dataset[attr.name] == unique_vals_attr[i]]
                 type_count = len(temp)
                 p = 1
                 for j in range(len(class_labels)):
                     temp = dataset.loc[(dataset[attr.name] == unique_vals_attr[i]) & (dataset[class_label.name]
                     type_cl_count = len(temp)
                     p -= pow((type_cl_count / type_count), 2)
                 gini_attr += (type_count / rows) * p
         elif num_unique_vals > 2:
             for i in range(num_unique_vals):
                 temp1 = dataset.loc[dataset[attr.name] == unique_vals_attr[i]]
                 temp2 = dataset.loc[dataset[attr.name] != unique_vals_attr[i]]
                 type_count1 = len(temp1)
                 type_count2 = len(temp2)
                 p1 = 1
                 p2 = 1
                 for j in range(len(class_labels)):
                     temp3 = dataset.loc[(dataset[attr.name] == unique_vals_attr[i]) & (dataset[class_label.name
                     type_cl_count1 = len(temp3)
                     p1 -= pow((type_cl_count1 / type_count1), 2)
                     temp4 = dataset.loc[(dataset[attr.name] != unique_vals_attr[i]) & (dataset[class_label.name
                     type_cl_count2 = len(temp4)
                     p2 -= pow((type_cl_count2 / type_count2), 2)
                 gini_values.append((type_count1 / rows) * p1 + (type_count2 / rows) * p2)
             gini_attr = min(gini_values)
             div_index = gini_values.index(gini_attr)

         return gini_attr, div_index
```

```
[16] def construct_decision_tree(dataset):
         columns = list(dataset.columns.values.tolist())
         class_labels = dataset.iloc[:, -1]
         num_unique_class_labels, class_label_counts, unique_class_labels = count_unique_values(class_labels)
         rows = len(class_labels)
         initial_gini = 1 - pow((class_label_counts[0] / rows), 2) - pow((class_label_counts[1] / rows), 2)
         print("Initial Gini Index:", initial_gini)
         num_attrs = len(dataset.columns) - 1
         for i in range(num_attrs):
             attr = dataset.iloc[:, i]
             num_unique_vals, val_counts, unique_vals = count_unique_values(attr)
             gini_index, division_index = calculate_gini(num_unique_vals, val_counts, rows, unique_class_labels,
             print("Gini Index of attribute", attr.name, ":", gini_index)
             print("Division index:", division_index)

     # Fit decision tree classifier
     clf = tree.DecisionTreeClassifier()
     X = dataset.drop('Play Tennis', axis=1)
     y = dataset['Play Tennis']
     X_encoded=pd.get_dummies(X)
     clf = clf.fit(X_encoded, y)

     # Plot decision tree
     tree.plot_tree(clf, feature_names=X_encoded.columns, class_names=y.unique(), filled=True)
     plt.show()

     # Build display decision tree
     construct_decision_tree(dataset)
```




```
Initial Gini Index: 0.4591836734693877
Gini Index of attribute Outlook : 0.35714285714285715
Division index: 1
Gini Index of attribute Temperature : 0.44285714285714295
Division index: 0
Gini Index of attribute Humidity : 0.3673469387755103
Division index: 0
Gini Index of attribute Wind : 0.42857142857142855
Division index: 0
```