JUNAID GIRKAR

60004190057

TE COMPS A4

**EXPERIMENT - 3**

**AIM**: Implement CART Algorithm with Gini Index.

**THEORY**:

The CART algorithm is a type of classification algorithm that is required to build a decision tree on the basis of Gini’s impurity index. It is a basic machine learning algorithm and provides a wide variety of use cases. A statistician named Leo Breiman coined the phrase to describe Decision Tree algorithms that may be used for classification or regression predictive modelling issues.

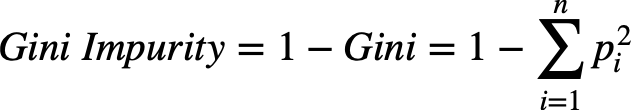
CART is an umbrella word that refers to the following types of decision trees:

* **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 and it is employed to progress from observations about an item that is represented by branches and finally concludes at the item’s target value, which is represented in the leaves. Because of their readability and simplicity, decision trees are among the most popular machine learning methods.

The structure of a decision tree consists of three main parts: Root nodes, Internal Nodes and Leaf Nodes.

**Gini Impurity**



*where Pi is the fraction of items in the class i.*

To find the best split, we need to calculate the weighted sum of Gini Impurity for both child nodes. We do this for all possible splits and then take the one with the lowest Gini Impurity as the best split.

CODE:

| import pandas as pd import numpy as np  df = pd.DataFrame() outlook = ['Sunny', 'Sunny', 'Overcast', 'Rain', 'Rain', 'Rain', 'Overcast', 'Sunny', 'Sunny', 'Rain', 'Sunny', 'Overcast', 'Overcast', 'Rain'] temp = ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'] humidity = ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'High'] wind = ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Strong'] decision = [0, 0, 1, 1, 1, 0, 1,0, 1, 1, 1, 1, 1, 0]  df['outlook'] = outlook df['temp'] = temp df['humidity'] = humidity df['wind'] = wind df['decision'] = decision  def calc\_gini\_for\_attribute(class\_name, col,df, target\_col='decision'):  total\_count = len(df[df[col].isin([class\_name])])  count\_of\_1 = len(df[(df[col].isin([class\_name])) & (df[target\_col] == 1)])  count\_of\_0 = len(df[(df[col].isin([class\_name])) & (df[target\_col] == 0)])  prob\_of\_1 = count\_of\_1 / total\_count  prob\_of\_0 = count\_of\_0 / total\_count  gini = 1 - (prob\_of\_1 \*\*2) - (prob\_of\_0\*\*2)  return gini, total\_count  calc\_gini\_for\_attribute('Sunny', 'outlook',df, target\_col='decision')  col = 'outlook' list(df[col].unique())  cols = ['outlook', 'temp', 'humidity', 'wind'] gini\_dict = {} for col in cols:  print(col)  gini\_for\_attr = 0  for value in list(df[col].unique()):  gini\_val, var\_count = calc\_gini\_for\_attribute(value, col, df)  print(f"For atr: {value}, Value = {gini\_val}")  gini\_for\_attr += var\_count/len(df) \* gini\_val   print(round(gini\_for\_attr, 3))  print('\n')  gini\_dict[col] = round(gini\_for\_attr, 3)    gini\_dict   def calc\_gini(cols: list, data):  gini\_dict = {}  for col in cols:  gini\_for\_attr = 0  for value in list(data[col].unique()):  gini\_val, var\_count = calc\_gini\_for\_attribute(value, col, data)  gini\_for\_attr += var\_count/len(data) \* gini\_val  gini\_dict[col] = round(gini\_for\_attr, 3)  return gini\_dict  calc\_gini(cols, df)  df['outlook'].values[0]  list(df.columns)  def get\_sel\_attr(df):  cols = list(df.columns)  attr\_gini = calc\_gini(cols, df)  min = 10  for col in cols:  if attr\_gini[col] < min:  min = attr\_gini[col]  sel\_attr = col  return sel\_attr   # Here we can split the df -> We need to send the selected attribute def split\_df(sel\_attr, df, father):  global id  print(sel\_attr)  list\_of\_unique\_values = list(df[sel\_attr].unique())  for value in list\_of\_unique\_values:  if check\_termination(df[df[sel\_attr] == value]):  print(f"terminating when {sel\_attr} is {value}")   id += 1  final\_tree.append({'id': id, 'data': df[df[sel\_attr] == value], 'cond': sel\_attr + " is " + value, 'children': [0, 0], 'isRoot': False, 'isLeaf': True, 'father': father})  else:  print(f"Cannot terminate when {sel\_attr} is {value}")  id +=1  new\_df = df[df[sel\_attr] == value].drop([sel\_attr], axis = 1)   # print(new\_df.head())  final\_tree.append({'id': id, 'data': df[df[sel\_attr] == value], 'cond': sel\_attr + " is " + value, 'children': [], 'isRoot': False, 'isLeaf': False, 'father': father})  new\_best\_attr = get\_sel\_attr(new\_df)  print(f"New Attr: {new\_best\_attr}")  split\_df(new\_best\_attr, new\_df, id)    return {'success': False}   # We can terminate if the probability of one class exceeds 75% def check\_termination(df):  df\_length = len(df)  zero\_count = len(df[df['decision'] == 0])  one\_count = len(df[df['decision'] == 1])  higher = zero\_count/df\_length if zero\_count/df\_length > one\_count/df\_length else one\_count/df\_length  print(higher)  if (higher > 0.9):  return True  return False   # We define the final tree variable which will store the decision tree id = 1 final\_tree = [{'id': 1, 'data': df, 'cond': 'None', 'children': [], 'isRoot': True, 'isLeaf': False, 'father': 0}] split\_df('outlook', df, id)   for obj in final\_tree:  print(f"id: {obj['id']} --- cond: {obj['cond']} --- father: {obj['father']}") |
| --- |

OUTPUT:

| outlook For atr: Sunny, Value = 0.48 For atr: Overcast, Value = 0.0 For atr: Rain, Value = 0.48 0.343   temp For atr: Hot, Value = 0.5 For atr: Mild, Value = 0.4444444444444445 For atr: Cool, Value = 0.375 0.44   humidity For atr: High, Value = 0.489795918367347 For atr: Normal, Value = 0.24489795918367355 0.367   wind For atr: Weak, Value = 0.375 For atr: Strong, Value = 0.5 0.429    {'outlook': 0.343, 'temp': 0.44, 'humidity': 0.367, 'wind': 0.429}    ['outlook', 'temp', 'humidity', 'wind', 'decision']   outlook 0.6 Cannot terminate when outlook is Sunny New Attr: humidity humidity 1.0 terminating when humidity is High 1.0 terminating when humidity is Normal 1.0 terminating when outlook is Overcast 0.6 Cannot terminate when outlook is Rain New Attr: wind wind 1.0 terminating when wind is Weak 1.0 terminating when wind is Strong {'success': False}   id: 1 --- cond: None --- father: 0 id: 2 --- cond: outlook is Sunny --- father: 1 id: 3 --- cond: humidity is High --- father: 2 id: 4 --- cond: humidity is Normal --- father: 2 id: 5 --- cond: outlook is Overcast --- father: 1 id: 6 --- cond: outlook is Rain --- father: 1 id: 7 --- cond: wind is Weak --- father: 6 id: 8 --- cond: wind is Strong --- father: 6 |
| --- |

CONCLUSION: We learnt about decision trees and implemented Classification and Regression Tree in Python. We then learnt about the different node splitting techniques and implemented Gini Indexing in our python program.