Sardar Patel Institute of Technology, Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII (2023-2024)
ETEL71A - Machine Learning and AI

Experiment: Decision Tree (ID3) algorithm

Name: Narayan Gawas **UID:** 2021201070

Objective: Write Python program to demonstrate the working of the decision tree based ID3 algorithm by using appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Outcomes:

1. Find entropy of data and follow steps of the algorithm to construct a tree.
2. Representation of hypothesis using decision tree.
3. Apply Decision Tree algorithm to classify the given data.
4. Interpret the output of Decision Tree.

System Requirements: Linux OS with Python and libraries or R or windows with MATLAB

Theory:

The decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

Entropy

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

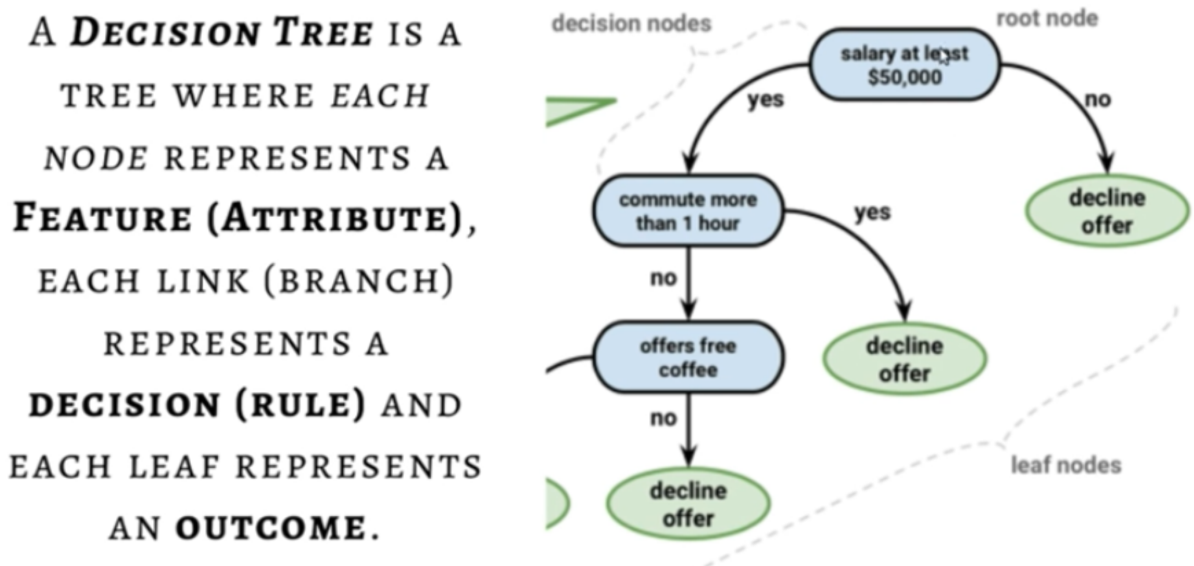
E(S) is the Entropy of the entire set, while the second term E(S, A) relates to an Entropy of an attribute A.

$$E(S) = \sum_{x \in X} -P(x) \log_2 P(x) \qquad E(S, A) = \sum_{x \in X} [P(x) * E(S)]$$

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

$$IG(S, A) = E(S) - E(S, A)$$



Code with Output:

```
import numpy as np
import pandas as pd
eps = np.finfo(float).eps
from numpy import log2 as log
outlook = 'sunny,sunny,overcast,rainy,rainy,rainy,overcast,sunny,sunny,rainy,sunny,overcast,overcast,rainy'.split(',')
temp = 'hot,hot,hot,mild,cool,cool,cool,mild,cool,mild,mild,mild,hot,mild'.split(',')
humidity = 'high,high,high,high,normal,normal,normal,high,normal,normal,normal,high,normal,high'.split(',')
```

```
windy = 'weak,strong,weak,weak,weak,strong,strong,weak,weak,weak,strong,strong,weak,strong'.split(',')
play = 'no,no,yes,yes,yes,no,yes,no,yes,yes,yes,yes,yes,no'.split(',')
dataset ={'outlook':outlook,'temp':temp,'humidity':humidity,'windy':windy,'play':play}
df = pd.DataFrame(dataset,columns=['outlook','temp','humidity','windy','play'])
Df
```

	outlook	temp	humidity	windy	play
0	sunny	hot	high	weak	no
1	sunny	hot	high	strong	no
2	overcast	hot	high	weak	yes
3	rainy	mild	high	weak	yes
4	rainy	cool	normal	weak	yes
5	rainy	cool	normal	strong	no
6	overcast	cool	normal	strong	yes
7	sunny	mild	high	weak	no
8	sunny	cool	normal	weak	yes
9	rainy	mild	normal	weak	yes
10	sunny	mild	normal	strong	yes
11	overcast	mild	high	strong	yes
12	overcast	hot	normal	weak	yes
13	rainy	mild	high	strong	no

```
entropy_node = 0  #Initialize Entropy
values = df.play.unique()  #Unique objects - 'Yes', 'No'
for value in values:
    fraction = df.play.value_counts()[value]/len(df.play)
```

```
    entropy_node += -fraction*np.log2(fraction)

print(f'Values: {values}')
print(f'entropy_node: {entropy_node}')

def ent(df,attribute):
    target_variables = df.play.unique()  #This gives all 'Yes' and 'No'
    variables = df[attribute].unique()    #This gives different features in that
attribute (like 'Sweet')
```

```
Values: ['no' 'yes']
entropy_node: 0.9402859586706309
```

```
    entropy_attribute = 0
    for variable in variables:
        entropy_each_feature = 0
        for target_variable in target_variables:
            num = len(df[attribute][df[attribute]==variable][df.play
==target_variable]) #numerator
            den = len(df[attribute][df[attribute]==variable])  #denominator
            fraction = num/(den+eps)  #pi
            entropy_each_feature += -fraction*log(fraction+eps) #This calculates
entropy for one feature like 'Sweet'
        fraction2 = den/len(df)
        entropy_attribute += -fraction2*entropy_each_feature   #Sums up all the
entropy ETaste

    return(abs(entropy_attribute))
a_entropy = {k:ent(df,k) for k in df.keys()[:-1]}
a_entropy
```

```
{'outlook': 0.6935361388961914,
 'temp': 0.9110633930116756,
 'humidity': 0.7884504573082889,
 'windy': 0.892158928262361}
```

```
def ig(e_dataset,e_attr):
    return(e_dataset-e_attr)
IG = {k:ig(entropy_node,a_entropy[k]) for k in a_entropy}
IG
```

```
{'outlook': 0.2467498197744391,
 'temp': 0.029222565658955313,
 'humidity': 0.15183550136234203,
 'windy': 0.04812703040826993}
```

```
def find_entropy(df):
    Class = df.keys()[-1]   #To make the code generic, changing target variable
class name
    entropy = 0
    values = df[Class].unique()
    for value in values:
        fraction = df[Class].value_counts()[value]/len(df[Class])
        entropy += -fraction*np.log2(fraction)
    return entropy



def find_entropy_attribute(df,attribute):
  Class = df.keys()[-1]   #To make the code generic, changing target variable class
name
  target_variables = df[Class].unique()  #This gives all 'Yes' and 'No'
  variables = df[attribute].unique()    #This gives different features in that
attribute (like 'Hot','Cold' in Temperature)
  entropy2 = 0
  for variable in variables:
      entropy = 0
```

```
        for target_variable in target_variables:
            num = len(df[attribute][df[attribute]==variable][df[Class]
==target_variable])
            den = len(df[attribute][df[attribute]==variable])
            fraction = num/(den+eps)
            entropy += -fraction*log(fraction+eps)
        fraction2 = den/len(df)
        entropy2 += -fraction2*entropy
    return abs(entropy2)


def find_winner(df):
    Entropy_att = []
    IG = []
    for key in df.keys()[:-1]:
#         Entropy_att.append(find_entropy_attribute(df,key))
        IG.append(find_entropy(df)-find_entropy_attribute(df,key))
    return df.keys()[:-1][np.argmax(IG)]


def get_subtable(df, node,value):
    return df[df[node] == value].reset_index(drop=True)


def buildTree(df,tree=None):
    Class = df.keys()[-1]   #To make the code generic, changing target variable
class name
    #Here we build our decision tree
    #Get attribute with maximum information gain
    node = find_winner(df)
```

```
    #Get distinct value of that attribute e.g Salary is node and Low,Med and High
are values
    attValue = np.unique(df[node])
    #Create an empty dictionary to create tree
    if tree is None:
        tree={}
        tree[node] = {}
   #We make loop to construct a tree by calling this function recursively.
    #In this we check if the subset is pure and stops if it is pure.
    for value in attValue:
        subtable = get_subtable(df,node,value)
        clValue,counts = np.unique(subtable[Class],return_counts=True)
        if len(counts)==1:#Checking purity of subset
            tree[node][value] = clValue[0]
        else:
            tree[node][value] = buildTree(subtable) #Calling the function
recursively
    return tree
t  = buildTree(df)
import pprint
pprint.pprint(t)
```

```
{'outlook': {'overcast': 'yes',
             'rainy': {'windy': {'strong': 'no', 'weak': 'yes'}},
             'sunny': {'humidity': {'high': 'no', 'normal': 'yes'}}}}
```

Conclusion: this experiment introduced the ID3 (Iterative Dichotomiser 3) algorithm for decision tree learning. We implemented a simplified version in Python to build a decision tree from a toy dataset and demonstrated how it can be used to classify new samples. This experiment provided a foundational understanding of decision tree construction and classification, serving as a valuable introduction to these fundamental machine learning concepts.