CLASS: B.E. E &TC **SUBJECT: ML**
EXPT. NO.: 10 **DATE: 22/02/21**
ROLL NO.: 42428

TITLE: **To create a machine learning algorithm of Decision Tree for Classification**

CO 1: Understand the basic concepts in machine learning like parametric/non- parametric modeling, classification, clustering, linear/ nonlinear regression and supervised/unsupervised learning to broadly classify various types of machine learning algorithms. Given an applicable feature vector, select an appropriate machine learning approach to design an analytical model to make expected predictions.

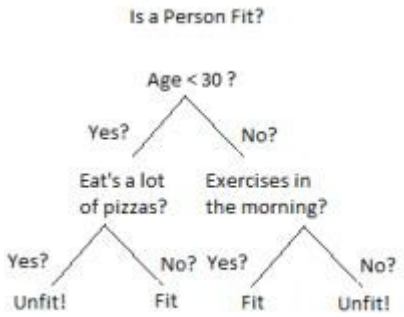
CO 4: Carry out experiments as an individual and in a team, comprehend and write a laboratory record and draw conclusions at a technical level.

AIM:
To implement:
To create a Machine Learning Algorithm of Decision Tree for classification

SOFTWARES REQUIRED: MATLAB 7.0 or Python

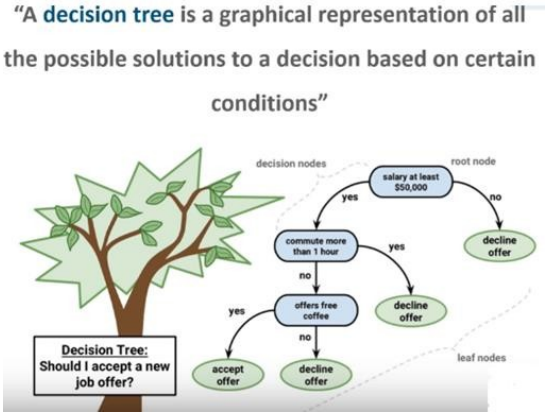
THEORY:

Decision Trees are a type of Supervised Machine Learning where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.



An example of a decision tree can be explained using above binary tree. Let's say you want to predict whether a person is fit given their information like age, eating habit, and physical activity, etc. The decision nodes here are questions like 'What's the age?', 'Does he exercise?', 'Does he eat a lot of pizzas'? And the leaves, which are outcomes like either 'fit', or 'unfit'. In this case this was a binary classification problem (a yes no type problem).

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



Construction of Decision Tree:

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each

derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. In general decision tree classifier has good accuracy.

Decision Tree Representation:

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure.
This process is then repeated for the sub-tree rooted at the new node.
The decision tree in above figure classifies a particular morning according to whether it is suitable for playing tennis and returning the classification associated with the particular leaf.(in this case Yes or No).
For example, the instance

(Outlook = Rain, Temperature = Hot, Humidity = High, Wind = Strong)

Would be sorted down the left most branch of this decision tree and would therefore be classified as a negative instance.
In other words we can say that decision tree represent a disjunction of conjunctions of constraints on the attribute values of instances.

(Outlook = Sunny ^ Humidity = Normal) v (Outlook = Overcast) v (Outlook = Rain ^ Wind = Weak)

Important definitions:

1. Pruning: Opposite of splitting, basically removing unwanted branches from the tree
2. Parent/Child node: Root node is the parent node and all the other nodes branched from it is known as child node.

3. Root node: It represents the entire population or sample and this further gets divided into two or more homogenous sets.
4. Branch/Sub tree: formed by splitting the tree nodes
5. Splitting: It is dividing the root or sub node into different parts on the basis of some condition.
6. Leaf node: It is a node that cannot be further segregated into further nodes.
7. Entropy: Entropy is the measure of randomness in the information being processed.

Entropy(s)=-P(yes)log2P(yes) – P(no) log2P(no)

Where,

- S is the total sample space,
- P(yes) is probability of yes

If number of yes = number of no i.e. P(s) = 0.5

- Entropy(s) = 1

If it contains all yes or all no i.e. P(s) =1 or 0

- Entropy(s) = 0

8. Information gain: Information gain calculates the reduction in entropy or surprise from transforming a dataset and is often used in training decision trees.

Information gain =Entropy(s) – {(weighted avg.) x Entropy (each feature)}

Strengths and Weakness of Decision Tree approach:

The strengths of decision tree methods are:

1. Decision trees are able to generate understandable rules.
2. Decision trees perform classification without requiring much computation.
3. Decision trees are able to handle both continuous and categorical variables.
4. Decision trees provide a clear indication of which fields are most important for prediction or classification.

The weaknesses of decision tree methods:

1. Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
2. Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.
3. Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

ALGORITHM:

1. Calculate entropy for dataset.
2. For each attribute/feature
   i. Calculate entropy for all its categorical values.
   ii. Calculate information gain for the feature.
3. Find the feature with maximum information gain as a Root node.
4. Repeat it until we get the desired tree.

CONCLUSION:

In this experiment we learned about the parametric and non-parametric modeling, classification, concept of clustering, linear and nonlinear regression along with this the supervised and unsupervised learning to classify various types of machine learning algorithms. And for feature vector we select an appropriate machine learning approach to design an analytical model to make expected predictions.

REFERENCES:

i. Laurene Fausett ,"Fundamentals of Neural Networks: Architectures, Algorithms and Applications", Pearson Education, Inc, 2008.

ii. S. N. Sivanandam , S. Sumathi, S. N. Deepa, "Introduction to Neural Networks using MATLAB", McGraw Hill, 2006.

iii. S. N. Sivanandam, S. N. Deepa, "Principles of Soft Computing" , John Wiley & Sons, 2007

iv. Phil Kim, "MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence", a Press 2017.

(Course Teacher)

Code :

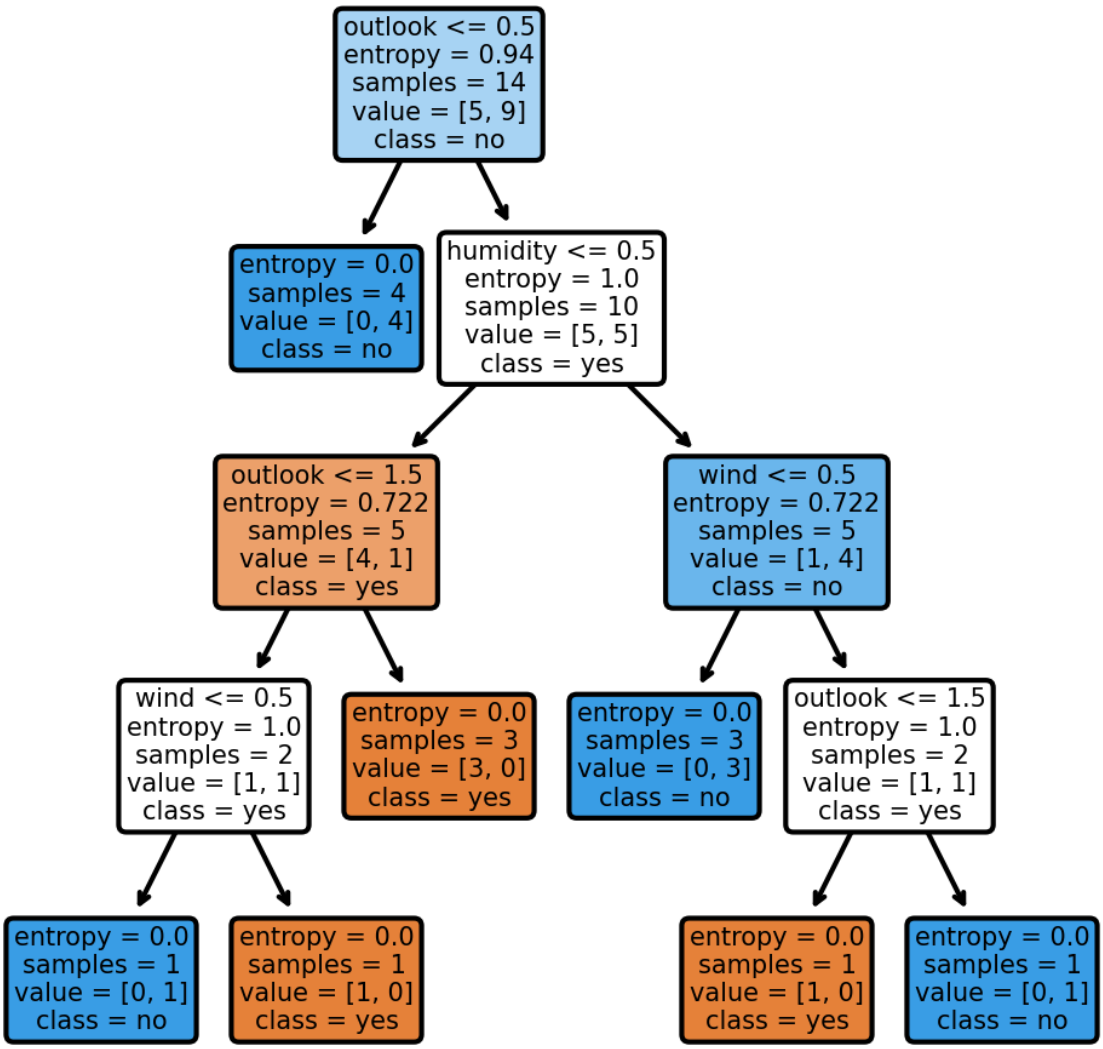
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('Dataset1.csv')
print(df)
#convert string data into float
lab = LabelEncoder()
df['Outlook'] = lab.fit_transform(df['Outlook'])
df['Temp'] = lab.fit_transform(df['Temp'])
df['Humidity'] = lab.fit_transform(df['Humidity'])
df['Windy'] = lab.fit_transform(df['Windy'])
df['Play'] = lab.fit_transform(df['Play'])

#convert to numpy
a=df.to_numpy()

cols=df.columns
x_train = a[:,0:cols.size-1]
y_train = a[:,cols.size-1]

cn = ['yes' , 'no']

#print(x_train);
#print(y_train.shape);


classifier = DecisionTreeClassifier(criterion='entropy')
classifier = classifier.fit(x_train, y_train)
tree.plot_tree(classifier,feature_names=cols[0:cols.size-
1] , class_names=cn,filled=True)
plt.show()
```

Dataset :

Outlook,Temp,Humidity,Windy,Play
Sunny,Hot,High,F,N
Sunny,Hot,High,T,N
Overcast,Hot,High,F,Y
Rainy,Mild,High,F,Y
Rainy,Cool,Normal,F,Y
Rainy,Cool,Normal,T,N
Overcast,Cool,Normal,T,Y
Sunny,Mild,High,F,N
Sunny,Cool,Normal,F,Y
Rainy,Mild,Normal,F,Y
Sunny,Mild,Normal,T,Y
Overcast,Mild,High,T,Y
Overcast,Hot,Normal,F,Y
Rainy,Mild,High,T,N

Output :

	Outlook	Temp	Humidity	Windy	Play
0	Sunny	Hot	High	F	N
1	Sunny	Hot	High	T	N
2	Overcast	Hot	High	F	Y
3	Rainy	Mild	High	F	Y
4	Rainy	Cool	Normal	F	Y
5	Rainy	Cool	Normal	T	N
6	Overcast	Cool	Normal	T	Y
7	Sunny	Mild	High	F	N
8	Sunny	Cool	Normal	F	Y
9	Rainy	Mild	Normal	F	Y
10	Sunny	Mild	Normal	T	Y
11	Overcast	Mild	High	T	Y
12	Overcast	Hot	Normal	F	Y
13	Rainy	Mild	High	T	N


outlook <= 0.5
entropy = 0.94
samples = 14
value = [5, 9]
class = no
entropy = 0.0
samples = 4
value = [0, 4]
class = no
humidity <= 0.5
entropy = 1.0
samples = 10
value = [5, 5]
class = yes
outlook <= 1.5
entropy = 0.722
samples = 5
value = [4, 1]
class = yes
wind <= 0.5
entropy = 0.722
samples = 5
value = [1, 4]
class = no
wind <= 0.5
entropy = 1.0
samples = 2
value = [1, 1]
class = yes
entropy = 0.0
samples = 3
value = [3, 0]
class = yes
entropy = 0.0
samples = 3
value = [0, 3]
class = no
outlook <= 1.5
entropy = 1.0
samples = 2
value = [1, 1]
class = yes
entropy = 0.0
samples = 1
value = [0, 1]
class = no
entropy = 0.0
samples = 1
value = [1, 0]
class = yes
entropy = 0.0
samples = 1
value = [1, 0]
class = yes
entropy = 0.0
samples = 1
value = [0, 1]
class = no