ASSIGNMENT NO. 3

AIM: Assignment of Decision Tree.

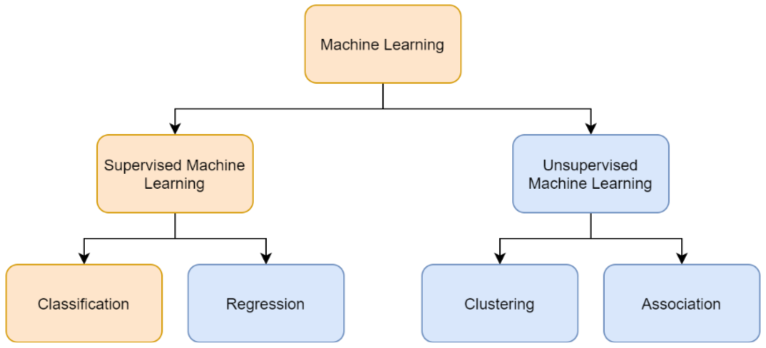
PREREQUISITE: Python programming

THEORY:

The purpose of an **information system** is to extract useful information from raw data. **Data science** is a field of study that aims to understand and analyze data by means of **statistics, big data, machine learning** and to provide support for decision makers and autonomous systems. While this sounds complicated, the tools are based on mathematical models and specialized software components that are already available (e.g. Python packages). In the following labs we will learn about.. learning. Machine Learning, to be more specific, and the two main classes: **Supervised Learning** and **Unsupervised Learning**. The general idea is to write software programs that can learn from the available data, identify patterns and make decisions with minimal human interventions, based on Machine Learning algorithms.

Machine Learning: Supervised Learning

Supervised learning is the Machine Learning task of learning a function (f) that maps an input (X) to an output (y) based on example input-output pairs. The goal is to find (approximate) the mapping function so that new data can be predicted. The function can be continuous in the case of regression, or discrete in the case of classification, requiring different algorithms. Now, we will discuss about classification methods, where the input/output variables are attributes and not limited to numbers.



Regression vs classification

The main difference between them is that the output variable in regression is numerical (or continuous, such as “dollars” or “weight”) while that for classification is categorical (or discrete, such as “red”, “blue”, “small”, “large”). For example, when provided with a dataset about houses (e.g. Boston), and you are asked to predict their prices, that is a regression task because price will be a continuous output (see Lab 7). Examples of the common regression algorithms include linear regression, Support Vector Regression (SVR), and regression trees.

Classification. Decision Trees

For example, when provided with a dataset about houses, a classification algorithm can try to predict whether the prices for the houses "sell more or less than the recommended retail price". Examples of the common classification algorithms include logistic regression, Naïve Bayes, **decision trees**, and K Nearest Neighbors.

Decision Trees (overview)

A decision tree is a classification and prediction tool having a tree like 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:

- Input: historical data with known outcomes
- Output: rules and flowcharts (generated by the algorithm)
- How it works: the algorithm looks at all the different attributes and finds out the decisions that have to be made at each step in order to reach the target value.

You can actually construct a flowchart that can be used to understand the decisions from the historical data and predict decisions for the next sample of data.

Here is an example literally comparing apples and oranges based on the size and the texture of the fruit, based on Decision Trees. The algorithm has to learn from the available, labelled examples and then predict other fruits and classify them as either apples or oranges:

Types of Decision Trees

Hunt's algorithm, which was developed in the 1960s to model human learning in Psychology, forms the foundation of many popular decision tree algorithms, such as the following:

- **ID3:** Ross Quinlan is credited within the development of ID3, which is shorthand for "Iterative Dichotomiser 3." This algorithm leverages entropy and information gain as metrics to evaluate candidate splits. Some of Quinlan's research on this algorithm from 1986 can be found here (PDF, 1.4 MB) (link resides outside of ibm.com).

- **C4.5:** This algorithm is considered a later iteration of ID3, which was also developed by Quinlan. It can use information gain or gain ratios to evaluate split points within the decision trees.

- **CART:** The term, CART, is an abbreviation for “classification and regression trees” and was introduced by Leo Breiman. This algorithm typically utilizes Gini impurity to identify the ideal attribute to split on. Gini impurity measures how often a randomly chosen attribute is misclassified. When evaluating using Gini impurity, a lower value is more ideal.

How to choose the best attribute at each node

While there are multiple ways to select the best attribute at each node, two methods, information gain and Gini impurity, act as popular splitting criterion for decision tree models. They help to evaluate the quality of each test condition and how well it will be able to classify samples into a class.

Entropy and Information Gain

It’s difficult to explain information gain without first discussing entropy. Entropy is a concept that stems from information theory, which measures the impurity of the sample values. It is defined with by the following formula, where:

$$\text{Entropy}(S) = -\sum_{c \epsilon C} p(c)\log_2 p(c)$$

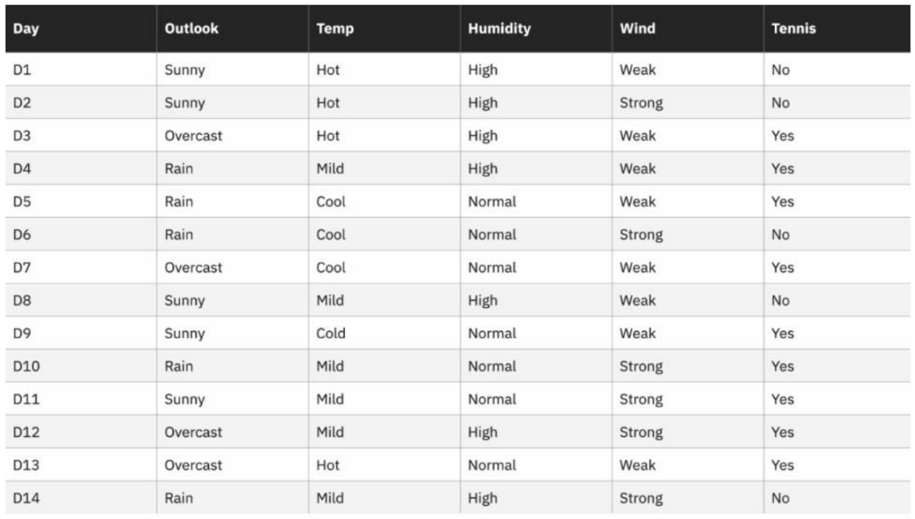
- S represents the data set that entropy is calculated
- c represents the classes in set, S
- p(c) represents the proportion of data points that belong to class c to the number of total data points in set, S

Entropy values can fall between 0 and 1. If all samples in data set, S, belong to one class, then entropy will equal zero. If half of the samples are classified as one class and the other half are in another class, entropy will be at its highest at 1. In order to select the best feature to split on and find the optimal decision tree, the attribute with the smallest amount of entropy should be used. Information gain represents the difference in entropy before and after a split on a given attribute. The attribute with the highest information gain will produce the best split as it’s doing the best job at classifying the training data according to its target classification. Information gain is usually represented with the following formula, where:

$$\text{Information Gain}(S,a) = \text{Entropy}(S) - \sum_{v \epsilon \text{vcalues}(a)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

- *a* represents a specific attribute or class label
- *Entropy(S)* is the entropy of dataset, S
- |Sv|/ |S| represents the proportion of the values in S_v to the number of values in dataset, S
- *Entropy(S_v)* is the entropy of dataset, S_v

Let’s walk through an example to solidify these concepts. Imagine that we have the following arbitrary dataset:



Day	Outlook	Temp	Humidity	Wind	Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

For this dataset, the entropy is 0.94. This can be calculated by finding the proportion of days where "Play Tennis" is "Yes", which is 9/14, and the proportion of days where "Play Tennis" is "No", which is 5/14. Then, these values can be plugged into the entropy formula above.

Entropy (Tennis) = -(9/14) log2(9/14) – (5/14) log2 (5/14) = 0.94

We can then compute the information gain for each of the attributes individually. For example, the information gain for the attribute, "Humidity" would be the following:

Gain (Tennis, Humidity) = (0.94)-(7/14)*(0.985) – (7/14)*(0.592) = 0.151

As a recap,

- 7/14 represents the proportion of values where humidity equals "high" to the total number of humidity values. In this case, the number of values where humidity equals "high" is the same as the number of values where humidity equals "normal".

- 0.985 is the entropy when Humidity = "high"

- 0.59 is the entropy when Humidity = "normal"

Then, repeat the calculation for information gain for each attribute in the table above, and select the attribute with the highest information gain to be the first split point in the decision tree. In this case, outlook produces the highest information gain. From there, the process is repeated for each subtree.

Gini Impurity

Gini impurity is the probability of incorrectly classifying random data point in the dataset if it were labeled based on the class distribution of the dataset. Similar to entropy, if set, S, is pure—i.e. belonging to one class) then, its impurity is zero. This is denoted by the following formula:

$$\text{Gini Impurity} = 1 - \sum_i (p_i)^2$$

Advantages and disadvantages of Decision Trees

While decision trees can be used in a variety of use cases, other algorithms typically outperform decision tree algorithms. That said, decision trees are particularly useful for data mining and knowledge discovery tasks. Let's explore the key benefits and challenges of utilizing decision trees more below:

Advantages

- **Easy to interpret:** The Boolean logic and visual representations of decision trees make them easier to understand and consume. The hierarchical nature of a decision tree also makes it easy to see which attributes are most important, which isn't always clear with other algorithms, like neural networks.

- **Little to no data preparation required:** Decision trees have a number of characteristics, which make it more flexible than other classifiers. It can handle various data types—i.e. discrete or continuous values, and continuous values can be converted into categorical values through the use of thresholds. Additionally, it can also handle values with missing values, which can be problematic for other classifiers, like Naïve Bayes.

- **More flexible:** Decision trees can be leveraged for both classification and regression tasks, making it more flexible than some other algorithms. It's also insensitive to underlying relationships between attributes; this means that if two variables are highly correlated, the algorithm will only choose one of the features to split on.

Disadvantages

- **Prone to overfitting:** Complex decision trees tend to overfit and do not generalize well to new data. This scenario can be avoided through the processes of pre-pruning or post-pruning. Pre-pruning halts tree growth when there is insufficient data while post-pruning removes subtrees with inadequate data after tree construction.

- **High variance estimators:** Small variations within data can produce a very different decision tree. Bagging, or the averaging of estimates, can be a method of reducing variance of decision trees. However, this approach is limited as it can lead to highly correlated predictors.

- **More costly:** Given that decision trees take a greedy search approach during construction, they can be more expensive to train compared to other algorithms.

- **Not fully supported in scikit-learn:** Scikit-learn is a popular machine learning library based in Python. While this library does have a Decision Tree module (DecisionTreeClassifier, link resides outside of ibm.com), the current implementation does not support categorical variables.

GITHUB LINK:- https://github.com/sahilb-official/machinelearninglab

REFERENCES:

1. Coursera Course on "What is Data Science?" offered by IBM.
Available at https://www.coursera.org/learn/what-is-datascience?specialization=ibm-data-science

2. https://www.ibm.com/in-en/topics/decision-trees

CONCLUSION:

Decision trees are powerful and interpretable machine learning models used for classification and regression tasks. They function by recursively splitting data based on attribute values to form a tree-like structure, making predictions based on learned rules. Decision trees are advantageous due to their ease of interpretation, minimal data preparation requirements, and flexibility in handling various data types. However, they are prone to overfitting, sensitive to data variations, and can be computationally expensive. Techniques like pruning and ensemble methods (e.g., Random Forest) can help mitigate these challenges. Despite some limitations, decision trees remain a widely used and effective tool in machine learning applications.