

## **Experiment No. 5**

<b>Expt No. 5</b>	
<b>Date :</b>	<b>Implementation of Decision Tree Algorithm</b>

### **Aim:**

To implement and understand the working of the Decision Tree algorithm for classification tasks in Machine Learning.

### **System Requirements:**

- Operating System: Windows 8 or above / Linux / macOS
- Memory (RAM): Minimum 4 GB
- Processor: Minimum 2.33 GHz (Dual Core or higher)

### **Software/Tools Required:**

Jupyter Notebook / Anaconda Navigator / Google Colaboratory / Spyder Python 3.x with the following libraries:

- NumPy
- Pandas
- Matplotlib
- Seaborn
- Scikit-learn

### **Purpose of the Experiment:**

- To understand the application of the Decision Tree algorithm for classification tasks, using the Salaries and Tennis dataset.

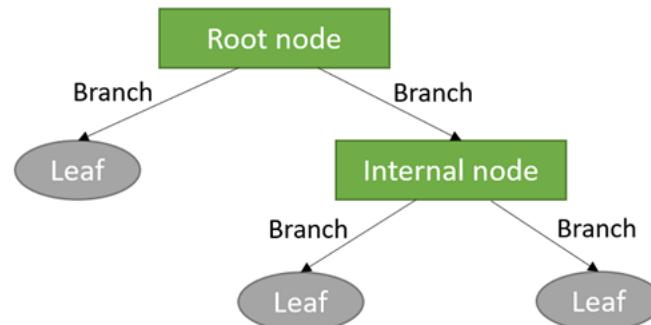
- To explore the process of encoding categorical variables and preparing the dataset for machine learning model training.
- To evaluate the performance of the Decision Tree model using metrics like F1-score.

## **Expected Outcomes:**

- Understand the concept and principles of the Decision Tree algorithm.
- Implement and understand the structure and components of a decision tree, including root nodes, internal nodes, branches, and leaf nodes.
- Evaluate the performance of the decision tree using appropriate metrics.
- Visualize the decision-making process of the model.

## **Theory:**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.



Structure of a Decision Tree:

**Root Node:**

The root node represents the entire dataset and is the starting point of the decision process. The dataset is split into subsets based on the most significant feature at the root.

**Internal Nodes:**

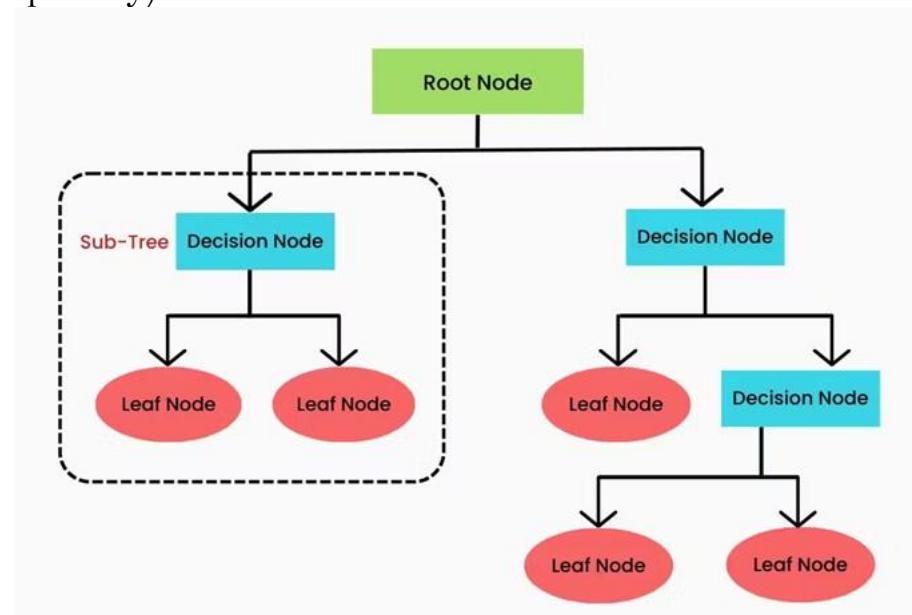
These represent the attributes or features of the dataset. At each internal node, a decision rule is applied based on the feature's value to divide the data into subsets.

**Branches:**

Branches connect the nodes and represent the outcomes of a decision rule applied at the internal nodes. They guide the data flow towards further splitting or reaching a leaf node.

**Leaf Nodes:**

Leaf nodes are the terminal points of the tree where no further splitting is done. In classification tasks, each leaf node corresponds to a class label (e.g., 'Yes' or 'No'). In regression tasks, it corresponds to a predicted continuous value (e.g., a numeric value like a price or quantity).



**Working of a Decision Tree:**

The process of building a decision tree follows these key steps:

1. Selecting the Best Feature:

At each node, the algorithm chooses the feature that best splits the data. The decision is based on criteria such as Information Gain, Gini Impurity, or Variance Reduction (for regression).

2. Information Gain: A measure of how much uncertainty is reduced when the data is split based on a particular feature. A feature with higher information gain is preferred because it provides a clearer division of the data.
3. Gini Impurity: A metric used for classification tasks that measures the impurity of a dataset. The algorithm aims to minimize this impurity at each split.
4. Variance Reduction: For regression tasks, the algorithm minimizes the variance of the predicted values within each split.
5. Recursive Data Splitting:

The tree continues to recursively split the dataset at each internal node, using the selected feature until a stopping condition is met. This could be:

6. Reaching a maximum depth.

Having fewer than a specified number of samples in a node.  
Achieving a minimum reduction in impurity.

7. Leaf Node Assignment:

Once the data can no longer be split (i.e., when it meets the stopping criteria), the algorithm assigns a class label (for classification) or a predicted value (for regression) to the leaf nodes.

### **Real-World Applications of Logistic Regression**

Here are four real-world applications of logistic regression:

1. Medical Diagnosis: Decision trees are used to classify medical conditions based on patient symptoms, test results, and medical history (e.g., diagnosing diseases like cancer or diabetes).
2. Credit Scoring: Financial institutions use decision trees to assess the risk of loan default by analyzing factors like income, credit history, and employment status.
3. Customer Segmentation: Marketing teams use decision trees to segment customers based on buying behavior and preferences, enabling targeted advertising and personalized recommendations.

4. Fraud Detection: Decision trees are used to detect fraudulent activities by analyzing transaction patterns, location, and user behavior, helping prevent financial fraud in real-time.

## 1. Problem Statement

The objective of this experiment is to implement a Decision Tree classifier to predict whether an individual earns a salary greater than 100K based on features such as company, job, and degree. The dataset contains categorical features that need to be encoded into numerical values for use in the model. The steps involved are data preprocessing, training the Decision Tree model, and evaluating its performance.

You can download the dataset from the following link:

<https://github.com/25-Madhuri/Dataset/blob/main/salaries.csv>

### Sample Code

#### Import Libraries

```
# Importing necessary libraries for data manipulation, visualization, and modeling
```

```
import pandas as pd
```

```
# Load the Salaries dataset from a CSV file
```

```
df = pd.read_csv('/content/salaries.csv')
```

```
# Displaying the first 5 rows of the DataFrame to inspect the structure of the dataset
```

```
df.head()
```

```
# Dropping the 'salary_more_then_100k' column from the dataset to create the feature set 'inputs'
```

```
inputs = df.drop('salary_more_then_100k', axis='columns')
```

inputs

*# Extracting the 'salary\_more\_then\_100k' column as the target variable for classification*

```
target = df['salary_more_then_100k']  
target
```

*# Importing the necessary libraries for data manipulation and encoding categorical features into numerical values.*

```
from sklearn.preprocessing import LabelEncoder
```

```
le_company = LabelEncoder()  
le_job = LabelEncoder()  
le_degree = LabelEncoder()
```

*# Applying LabelEncoder to convert the categorical columns ('company', 'job', 'degree') into numeric values.*

```
inputs['company_n'] = le_company.fit_transform(inputs['company'])  
inputs['job_n'] = le_company.fit_transform(inputs['job'])  
inputs['degree_n'] = le_company.fit_transform(inputs['degree'])
```

inputs

*# Dropping the original categorical columns after encoding them to keep only the numeric features for model training*

```
inputs_n = inputs.drop(['company','job','degree'],axis='columns')  
inputs_n
```

*# Importing the DecisionTreeClassifier from sklearn to create and train the Decision Tree model*

```
from sklearn import tree
```

```

model = tree.DecisionTreeClassifier()
model.fit(inputs_n,target)

model.score(inputs_n,target)

# Making a prediction using the trained model with sample input values  
(company_n=2, job_n=1, degree_n=0)

model.predict([[2,1,0]])

# Making a prediction using the trained model with sample input values  
(company_n=2, job_n=1, degree_n=1)

model.predict([[2,1,1]])

```

### **Observation**

- I. Record the following in observation table:

Sr.No	Metric	Description	Observations
1.	Shape of the Original Dataset	Record the number of rows and columns in the dataset (df) before any processing.	
2.	Encoded Values for 'company', 'job', 'degree'	Record the numeric values assigned to 'company', 'job', and 'degree' after applying LabelEncoder.	
3.	Shape of Encoded Dataset (inputs_n)	Record the number of rows and columns after encoding the categorical features and dropping the original columns.	
4.	Model Accuracy on Full Dataset	Record the accuracy score of the Decision Tree model on the entire dataset using model.score(inputs_n, target).	
5.	Prediction for a Sample Input	Record the prediction for a sample input (e.g., model.predict([[2, 1, 0]])).	

II. Perform the following tasks and record results by applying Decision Tree Algorithm. Use the Tennis dataset from the link below for each task:

<https://github.com/25-Madhuri/Dataset/blob/main/tennis.csv>

### Conclusion –

The Decision Tree model, implemented using the entropy criterion, successfully classifies data based on the encoded features of 'company', 'job', and 'degree'. The model's structure is visualized to show how different feature combinations lead to decisions regarding salary classifications.

### Viva Questions



Sr.No	Question	CO
1.	What is a decision tree and how does it work?	1ICPC406_3
2.	What are the main components of a decision tree?	1ICPC406_3
3.	How is a decision tree different from other machine learning algorithms?	1ICPC406_3
4.	What types of problems are decision trees typically used for?	1ICPC406_3
5.	What are the key advantages and disadvantages of using a decision tree?	1ICPC406_3
6.	How is a decision tree constructed? What are the steps involved in it?	1ICPC406_3
7.	Explain the concept of information gain and how it is used in decision tree?	1ICPC406_3
8.	What are Gini impurity and Entropy and how are they used in decision tree algorithm?	1ICPC406_3
9.	What is a decision tree and how does it work?	1ICPC406_3

## References –

### a. Textbook –

- i. Machine Learning with Python – An approach to Applied ML – Abhishek Vijayvargiya, BPB Publications, 1<sup>st</sup> Edition 2018
- ii. Machine Learning, Tom Mitchell, McGraw Hill Education, 1<sup>st</sup> Edition 1997

### b. Online references –

- i. <https://scikit-learn.org/stable/modules/tree.html>
- ii. <https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/>
- iii. <https://www.coursera.org/in/articles/decision-tree-machine-learning>