

## EXPERIMENT NO. 7

**AIM:** Study and Implement Decision Trees.

### **Theory:**

Decision Trees are a type of supervised machine learning model used for both classification and regression tasks. They work by splitting the data into subsets based on the values of input features, creating a tree-like structure of decisions.

The model starts at a root node and makes splits at each node based on a feature that best divides the data according to a specific criterion. Each branch represents a choice based on an attribute, and the process continues recursively until reaching a leaf node that represents a final decision or prediction.

### **Key Concepts in Decision Trees:**

- **Root Node:** The starting point of the tree, containing the entire dataset.
- **Decision Node:** A node where data splits based on an attribute.
- **Leaf Node:** The terminal node that represents a class label in classification or a value in regression.
- **Depth of the Tree:** Refers to the number of levels from the root to the leaf nodes. A deeper tree has more splits and potentially more complexity.
- **Overfitting and Underfitting:** Deep trees can overfit, capturing noise in the training data. Shallow trees might underfit, missing important patterns.

### **Code:**

```
# In[1]:
import pandas as pd

# In[2]:
df = pd.read_csv("salaries.csv")
df.head()

# In[3]:
inputs = df.drop('salary_more_than_100k',axis='columns')

#In[4]:
target = df['salary_more_than_100k']

#In[5]:
from sklearn.preprocessing import LabelEncoder
le_company = LabelEncoder()
```

```

le_job = LabelEncoder()
le_degree = LabelEncoder()

#In[6]:
inputs['company_n'] = le_company.fit_transform(inputs['company'])
inputs['job_n'] = le_job.fit_transform(inputs['job'])
inputs['degree_n'] = le_degree.fit_transform(inputs['degree'])

#In[7]:
Inputs

#In[8]:
inputs_n = inputs.drop(['company','job','degree'],axis='columns')

#In[9]:
inputs_n

#In[10]:
target

#In[11]:
from sklearn import tree
model = tree.DecisionTreeClassifier()

#In[12]:
model.fit(inputs_n, target)

#In[13]:
model.score(inputs_n,target)

```

**Is salary of Google, Computer Engineer, Bachelors degree > 100 k ?**

```

#In[14]:
model.predict([[2,1,0]])

```

**Is salary of Google, Computer Engineer, Masters degree > 100 k ?**

```

#In[15]:
model.predict([[2,1,1]])

```

## Output Snapshots:

	company		job	degree	salary_more_than_100k		
0	google		sales executive	bachelors	0		
1	google		sales executive	masters	0		
2	google		business manager	bachelors	1		
3	google		business manager	masters	1		
4	google	computer programmer	bachelors	0			
<hr/>							
	company		job	degree	company_n	job_n	degree_n
0	google		sales executive	bachelors	2	2	0
1	google		sales executive	masters	2	2	1
2	google		business manager	bachelors	2	0	0
3	google		business manager	masters	2	0	1
4	google	computer programmer	bachelors		2	1	0
5	google	computer programmer	masters		2	1	1
6	abc pharma		sales executive	masters	0	2	1
7	abc pharma		computer programmer	bachelors	0	1	0
8	abc pharma		business manager	bachelors	0	0	0
9	abc pharma		business manager	masters	0	0	1
10	facebook		sales executive	bachelors	1	2	0
11	facebook		sales executive	masters	1	2	1
12	facebook		business manager	bachelors	1	0	0
13	facebook		business manager	masters	1	0	1
14	facebook	computer programmer	bachelors		1	1	0
15	facebook	computer programmer	masters		1	1	1

	company_n	job_n	degree_n
0	2	2	0
1	2	2	1
2	2	0	0
3	2	0	1
4	2	1	0
5	2	1	1
6	0	2	1
7	0	1	0
8	0	0	0
9	0	0	1
10	1	2	0
11	1	2	1
12	1	0	0
13	1	0	1
14	1	1	0
15	1	1	1

0	0
1	0
2	1
3	1
4	0
5	1
6	0
7	0
8	0
9	1
10	1
11	1
12	1
13	1
14	1
15	1

Name: salary\_more\_than\_100k, dtype: int64

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                        splitter='best')
```

array([0], dtype=int64)

array([1], dtype=int64)

## Learning Outcome: