## Assignment No. 4

**Problem Statement:**Decision Tree Classification and Prediction.

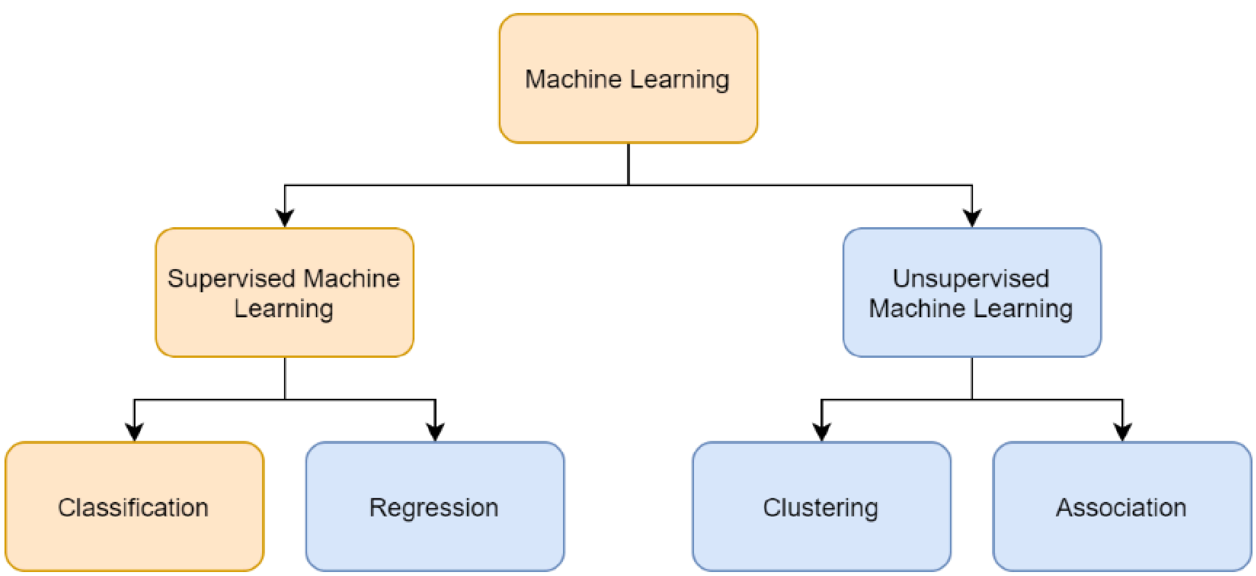
**Objective:**To implement and analyze Decision Tree classification on a dataset, understanding how it splits data based on attributes and makes predictions. The process includes exploring decision tree algorithms, evaluating entropy, information gain, and Gini impurity, and visualizing the tree structure.

**Prerequisite:**

- Python environment (Jupyter Notebook/IDE).
- Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn.
- Basic knowledge of machine learning and classification.

**Theory:**

**Understanding Decision Trees**
Decision Trees are a type of supervised learning algorithm used for classification and regression tasks. They work by recursively splitting data based on attribute values, creating a tree-like model to predict outcomes. Key aspects include:




1. **Classification vs Regression**
   a. **Classification:** Predicts discrete labels (e.g., "Spam" or "Not Spam").
   b. **Regression:** Predicts continuous values (e.g., house prices).
2. **Decision Tree Structure**

   a. **Root Node:** The starting point of the tree.
   b. **Internal Nodes:** Decision points based on attribute values.
   c. **Leaf Nodes:** Final classification outcomes.
3. **Types of Decision Tree Algorithms**
   a. **ID3 (Iterative Dichotomiser 3):** Uses entropy and information gain.
   b. **C4.5:** An improvement over ID3, using gain ratios.
   c. **CART (Classification and Regression Trees):** Uses Gini impurity for splitting.

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

4. **Entropy and Information Gain**
   a. Entropy measures impurity in a dataset. A pure set has entropy = 0.
   b. Information Gain calculates the reduction in entropy after a split.
   c. The attribute with the highest information gain is chosen for splitting.
5. **Gini Impurity**
   a. Measures how often a randomly chosen element would be incorrectly classified.
   b. A lower Gini impurity indicates a better split.
6. **Advantages and Disadvantages**
   a. Advantages: Easy to interpret, requires little data preprocessing, flexible for different data types.
   b. Disadvantages: Prone to overfitting, high variance, computationally expensive.

**Code and Output:**

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
import warnings

warnings.filterwarnings('ignore')
```

In [2]:

```python
import warnings

warnings.filterwarnings('ignore')
```

In [3]:

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

In [4]:

```python
df.shape
```

Out[4]:

```
(1727, 7)
```

In [5]:

```python
df.head()
```

Out[5]:

| | vhigh | vhigh.1 | 2 | 2.1 | small | low | unacc |
|---|---|---|---|---|---|---|---|
| **0** | vhigh | vhigh | 2 | 2 | small | med | unacc |
| **1** | vhigh | vhigh | 2 | 2 | small | high | unacc |
| **2** | vhigh | vhigh | 2 | 2 | med | low | unacc |
| **3** | vhigh | vhigh | 2 | 2 | med | med | unacc |
| **4** | vhigh | vhigh | 2 | 2 | med | high | unacc |

In [6]:

```python
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']


df.columns = col_names

col_names
```

Out[6]:

```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
```

In [7]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   buying    1727 non-null   object
 1   maint     1727 non-null   object
 2   doors     1727 non-null   object
 3   persons   1727 non-null   object
 4   lug_boot  1727 non-null   object
 5   safety    1727 non-null   object
 6   class     1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

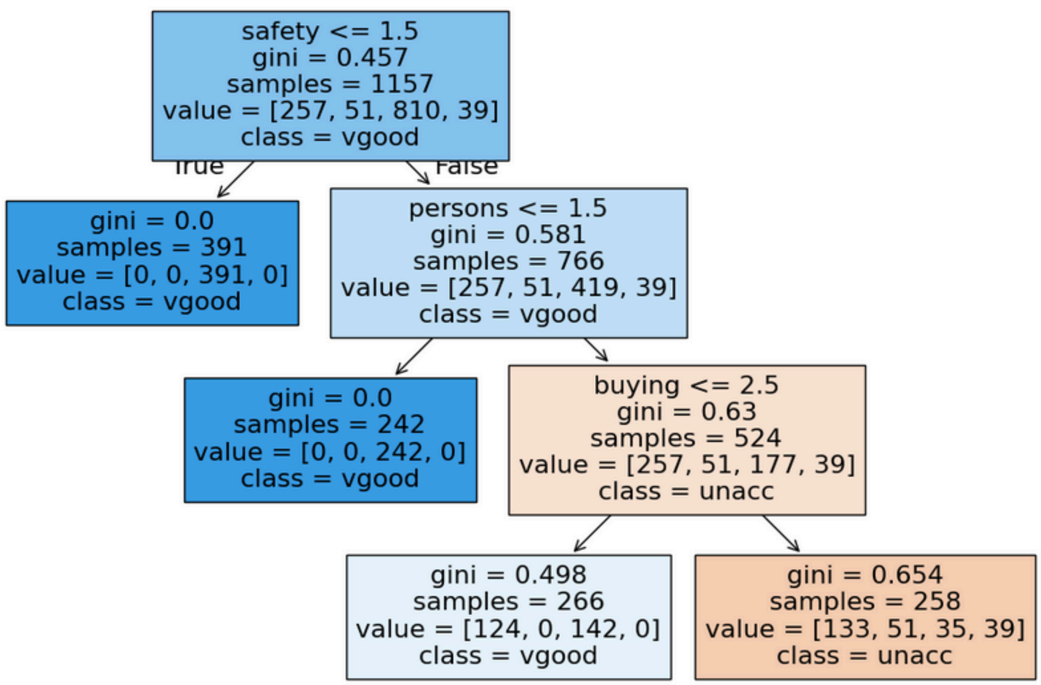
In [8]:

```
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']


for col in col_names:

    print(df[col].value_counts())
```

```
buying
high     432
med      432
low      432
vhigh    431
Name: count, dtype: int64
maint
high     432
med      432
low      432
vhigh    431
Name: count, dtype: int64
doors
3        432
4        432
5more    432
2        431
Name: count, dtype: int64
persons
4       576
more    576
2       575
```



**Conclusion:**

This task explored Decision Trees for classification. The algorithm successfully learned from data, making predictions based on recursive splits. Evaluations using entropy, information gain, and Gini impurity helped optimize tree performance. Decision Trees offer intuitive visualizations but require careful tuning to prevent overfitting.