Decision Tree Classifier

Aim: To classify the cars based on their condition using decision tree classifier

Decision Tree Classifier:
A decision tree is a tree-like model used for decision-making or prediction tasks. It consists of nodes that represent tests or decisions based on attributes, branches that indicate the outcome of these tests, and leaf nodes that represent the final prediction or decision. Each internal node corresponds to a test on an attribute, while each branch represents the possible outcomes of the test. The leaf nodes typically represent the final class label or predicted value.
The process of constructing a decision tree involves the following steps:

1. Choosing the Best Attribute: The most suitable attribute for splitting the data is chosen based on a criterion such as Gini impurity, entropy, or information gain.
2. Splitting the Data: The dataset is divided into subsets based on the chosen attribute.
3. Recursion: This process is recursively applied to each subset, creating new internal nodes or leaf nodes, until a stopping condition is reached (e.g., all instances in a node belong to the same class or a maximum tree depth is reached).

Algorithm:

1. Import necessary libraries.
2. Load the car evaluation dataset and assign column names.
3. Split the dataset into features (x) and target (y).
4. Encode categorical features using LabelEncoder.
5. Split the data into training and testing sets.
6. Fit a DecisionTreeClassifier (Gini/Entropy) on the training set.
7. Evaluate and print the training and testing scores.
8. Plot the decision tree.

Code:

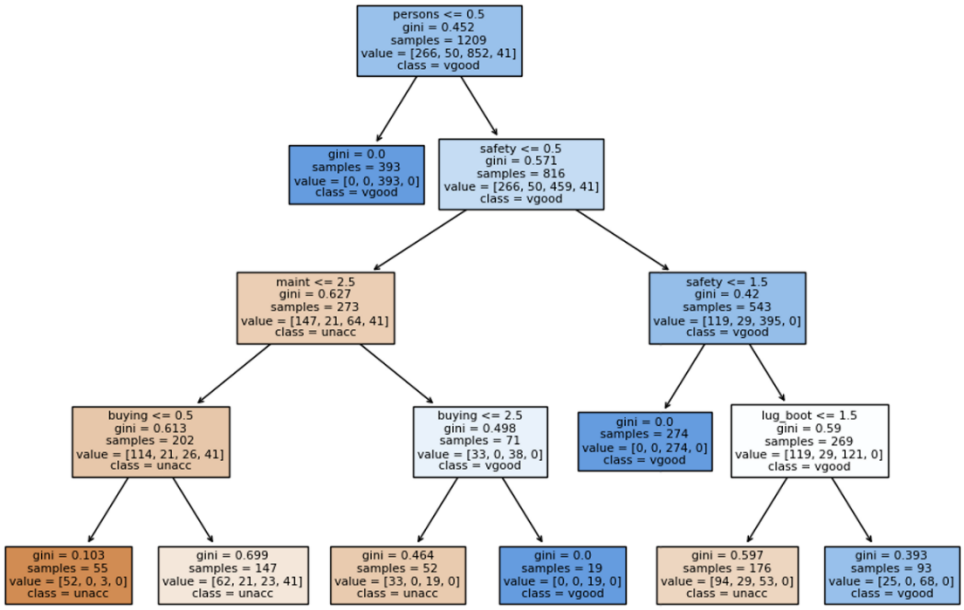
```
# Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn import tree

# Loading the dataset
df = pd.read_csv("car_evaluation.csv", header=None)

# Displaying the first few rows of the dataset
df.head()

# Displaying the shape of the dataset
df.shape

# Displaying summary statistics
df.describe()

# Displaying dataset information
df.info()

# Assigning column names
col_names = ['buying', 'maint', 'doors', 'persons',
'lug_boot', 'safety', 'class']
df.columns = col_names

# Displaying the value counts for each column
for col in df.columns:
    print(df[col].value_counts())

# Checking for null values
df.isnull().sum()

# Splitting the dataset into features (x) and target (y)
y = df['class']
x = df.drop(columns=['class'])

# Encoding categorical features
encoder = LabelEncoder()
for col in x.columns:
    x[col] = encoder.fit_transform(x[col])

# Splitting the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.3, random_state=42)

# Fitting the DecisionTreeClassifier with Gini index
clf_gini = DecisionTreeClassifier(criterion='gini',
max_depth=4, random_state=0)
clf_gini.fit(x_train, y_train)
```

```
# Evaluating training and test scores using Gini
print('Training set score using Gini:
{:.2f}'.format(clf_gini.score(x_train, y_train)))
print('Test set score using Gini:
{:.2f}'.format(clf_gini.score(x_test, y_test)))

# Plotting the decision tree (using Gini)
plt.figure(figsize=(12, 8))
tree.plot_tree(clf_gini,
               feature_names=x.columns,
               class_names=y.unique(),
               filled=True)

# Fitting the DecisionTreeClassifier with Entropy
clf_en = DecisionTreeClassifier(criterion='entropy',
max_depth=4, random_state=0)
clf_en.fit(x_train, y_train)

# Evaluating training and test scores using Entropy
print('Training set score using Entropy:
{:.2f}'.format(clf_en.score(x_train, y_train)))
print('Test set score using Entropy:
{:.2f}'.format(clf_en.score(x_test, y_test)))

# Plotting the decision tree (using Entropy)
plt.figure(figsize=(12, 8))
tree.plot_tree(clf_en,
               feature_names=x.columns,
               class_names=y.unique(),
               filled=True)
```

Output:

```
Training set score using gini: 0.82
Test set score using gini: 0.83
```




```
Training set score sing entropy: 0.82
Test set score using entropy: 0.83
```