Random Forest Classifier

Aim: To classify the cars based on their condition using random forest classifier and compare the result with the result obtained using decision tree classifier on the same dataset

Random Forest Classifier:
Random Forest is a supervised machine learning algorithm used for classification, regression, and other tasks, leveraging multiple decision trees. During training, it constructs a collection of decision trees, each based on a random subset of the training data and features. This introduces diversity among the trees, making the model more robust and less likely to overfit. The final prediction is made by taking the majority vote (mode) of the classifications from all the trees. The algorithm uses a technique called bagging (Bootstrap Aggregating) to generate these diverse subsets.

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 the RandomForestClassifier (Gini/Entropy) on the training set.
7. Evaluate and print the training and testing scores.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

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

# Displaying the first few rows and checking the dataset shape
df.head()
df.shape

# Displaying summary statistics and dataset info
df.describe()
```

```
df.info()

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

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

# Checking for missing 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 RandomForestClassifier with Gini index
clf_gini = RandomForestClassifier(n_estimators=10,
criterion="gini", random_state=0)
clf_gini.fit(x_train, y_train)

# Evaluating the 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)))

# Fitting the RandomForestClassifier with Entropy
clf_en = RandomForestClassifier(n_estimators=10,
criterion="entropy", random_state=0)
clf_en.fit(x_train, y_train)

# Evaluating the 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)))
```

Output:

```
Training set score using gini: 1.00
Test set score using gini: 0.95


Training set score using entropy: 1.00
Test set score using entropy: 0.96
```