ProdigyInfoTech

Let's walk through building a **Decision Tree Classifier** using the **Bank Marketing dataset** from the UCI Machine Learning Repository to predict whether a customer will **subscribe to a term deposit (purchase the product)** based on demographic and behavioral features.

- **Step 1: Load and Understand the Dataset**

Dataset source:
UCI Bank Marketing Dataset

The data is related to **direct marketing campaigns (phone calls)** of a Portuguese banking institution. The classification goal is to predict if the client will **subscribe (yes/no) to a term deposit**.

Target variable: y
Classes: yes (purchase) / no (did not purchase)

- **Step 2: Setup**

We'll use Python with common libraries.

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier, plot_tree

from sklearn.metrics import classification_report, confusion_matrix, accuracy_scor
```

- **Step 3: Load and Explore Data**

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank.csv"
df = pd.read_csv(url, sep=';')

print(df.head())
print(df.info())
print(df['y'].value_counts())
```

- **Step 4: Data Preprocessing**

Convert categorical variables to numeric using Label Encoding or One-Hot Encoding:

```
# Label encode the target
df['y'] = df['y'].map({'yes': 1, 'no': 0})

# One-hot encode categorical features
df_encoded = pd.get_dummies(df.drop('y', axis=1), drop_first=True)

# Final feature set
X = df_encoded
y = df['y']
```

Split into training and testing sets:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- **Step 5: Train Decision Tree Model**

```
clf = DecisionTreeClassifier(max_depth=5, random_state=42)
clf.fit(X_train, y_train)
```

- **Step 6: Evaluate the Model**

```
y_pred = clf.predict(X_test)


print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

🔹 **Step 7: Visualize the Decision Tree**

```
plt.figure(figsize=(20,10))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=['No', 'Yes'], rounded=True)
plt.show()
```

🔹 **Optional: Feature Importance**

```
importances = pd.Series(clf.feature_importances_, index=X.columns)
importances = importances[importances > 0].sort_values(ascending=False)


plt.figure(figsize=(10, 6))
sns.barplot(x=importances.values, y=importances.index)
plt.title("Feature Importances in Decision Tree")
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.tight_layout()
plt.show()
```
