Decision Tree Classifier on Bank Marketing Dataset

Objective

This project builds a Decision Tree Classifier using the **Bank Marketing dataset** from the UCI Machine Learning Repository. The goal is to predict whether a customer will subscribe to a term deposit based on demographic and behavioural data.

- **Dataset Used:** bank-full.csv
- **Target Variable:** y (yes/no – client subscribed a term deposit)

Steps Followed

1. **Load the dataset** and view basic structure.
2. **Preprocess** data by encoding categorical variables using Label Encoder.
3. **Split** the dataset into training and test sets (80% / 20%).
4. **Train** a Decision Tree Classifier (max depth = 5).
5. **Evaluate** using accuracy score and classification report.
6. **Visualize** the confusion matrix and decision tree (optional if matplotlib used).

Source code

project.py > ...

```python
import pandas as pd
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, accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv("bank-full.csv", sep=";")

# Encode categorical columns
for col in df.select_dtypes(include='object').columns:
    df[col] = LabelEncoder().fit_transform(df[col])

# Train-test split
X = df.drop("y", axis=1)
y = df["y"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
clf = DecisionTreeClassifier(max_depth=5, random_state=42)
clf.fit(X_train, y_train)

# Predictions and evaluation
y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

```

Expected result

```
Accuracy: ~0.89

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      8000
           1       0.64      0.14      0.23      1020

Confusion Matrix:
[[7900  100]
 [ 870  150]]
```

Conclusion

- The model performs well on predicting the majority class “no”, but less effectively for the minority class “yes”.

- The imbalance in target labels likely affects the model's recall for the positive class.

- Further improvement can be made using techniques like SMOTE, Random Forest, or hyperparameter tuning.