TASK-3 (Build a Decision Tree Classifier)

PROGRAM:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.tree import plot_tree
import  requests
import  zipfile
import io

# Load the dataset from the UCI Machine Learning Repository
# Corrected URL
url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-a
dditional.zip"

# Use pandas to read the zip file directly, assuming
'bank-additional-full.csv' is inside
import zipfile
with zipfile.ZipFile(io.BytesIO(requests.get(url).content), 'r') as
zip_ref:
    zip_ref.extractall('.')
df = pd.read_csv('bank-additional/bank-additional-full.csv', sep=';')

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

# Display basic information about the dataset
print(df.info())

# Preprocessing the data
# Handle missing values (if any)
print(df.isnull().sum())

# Encode categorical variables
```

```
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    df[column] = label_encoders[column].fit_transform(df[column])

# Split the data into training and test sets
X = df.drop('y', axis=1)
y = df['y']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Train a decision tree classifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict on the test set
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

print('Classification Report:')
print(classification_report(y_test, y_pred))

print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))

# Visualize the decision tree
plt.figure(figsize=(20, 10))
plot_tree(clf, feature_names=X.columns,
class_names=label_encoders['y'].classes_, filled=True)
plt.show()
```