```
# Import necessary 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.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.tree import plot_tree

pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.7-py3-none-any.whl (8.0 kB)
Requirement already satisfied: pandas>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2.0.3)
Requirement already satisfied: certifi>=2020.12.5 in
/usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2024.6.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0-
>ucimlrepo) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0-
>ucimlrepo) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0-
>ucimlrepo) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0-
>ucimlrepo) (1.25.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas>=1.0.0->ucimlrepo) (1.16.0)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.7
```

```
from ucimlrepo import fetch_ucirepo
# fetch dataset
bank_marketing = fetch_ucirepo(id=222)
# data (as pandas dataframes)
X = bank_marketing.data.features
y = bank_marketing.data.targets
# metadata
print(bank_marketing.metadata)
# variable information
print(bank_marketing.variables)
```

```
{'uci_id': 222, 'name': 'Bank Marketing', 'repository_url':
'https://archive.ics.uci.edu/dataset/222/bank+marketing', 'data_url':
```

'https://archive.ics.uci.edu/static/public/222/data.csv', 'abstract': 'The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).', 'area': 'Business', 'tasks': ['Classification'], 'characteristics': ['Multivariate'], 'num_instances': 45211, 'num_features': 16, 'feature_types': ['Categorical', 'Integer'], 'demographics': ['Age', 'Occupation', 'Marital Status', 'Education Level'], 'target_col': ['y'], 'index_col': None, 'has_missing_values': 'yes', 'missing_values_symbol': 'NaN', 'year_of_dataset_creation': 2014, 'last_updated': 'Fri Aug 18 2023', 'dataset_doi': '10.24432/C5K306', 'creators': ['S. Moro', 'P. Rita', 'P. Cortez'], 'intro_paper': {'title': 'A data-driven approach to predict the success of bank telemarketing', 'authors': 'Sérgio Moro, P. Cortez, P. Rita', 'published_in': 'Decision Support Systems', 'year': 2014, 'url': 'https://www.semanticscholar.org/paper/cab86052882d126d43f72108c6cb41b295cc8a9e', 'doi': '10.1016/j.dss.2014.03.001'}, 'additional_info': {'summary': "The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. \n\nThere are four datasets: \n1) bank-additional-full.csv with all examples (41188) and 20 inputs, ordered by date (from May 2008 to November 2010), very close to the data analyzed in [Moro et al., 2014]\n2) bank-additional.csv with 10% of the examples (4119), randomly selected from 1), and 20 inputs.\n3) bank-full.csv with all examples and 17 inputs, ordered by date (older version of this dataset with less inputs). \n4) bank.csv with 10% of the examples and 17 inputs, randomly selected from 3 (older version of this dataset with less inputs). \nThe smallest datasets are provided to test more computationally demanding machine learning algorithms (e.g., SVM). \n\nThe classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).", 'purpose': None, 'funded_by': None, 'instances_represent': None, 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': 'Input variables:\n   # bank client data:\n   1 - age (numeric)\n   2 - job : type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur","student",\n   "blue-collar","self-employed","retired","technician","services") \n   3 - marital : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed)\n   4 - education (categorical: "unknown","secondary","primary","tertiary")\n   5 - default: has credit in default? (binary: "yes","no")\n   6 - balance: average yearly balance, in euros (numeric) \n   7 - housing: has housing loan? (binary: "yes","no")\n   8 - loan: has personal loan? (binary: "yes","no")\n   # related with the last contact of the current campaign:\n   9 - contact: contact communication type

(categorical: "unknown","telephone","cellular") \n 10 - day: last contact day of the month (numeric)\n 11 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")\n 12 - duration: last contact duration, in seconds (numeric)\n   # other attributes:\n 13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)\n 14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)\n 15 - previous: number of contacts performed before this campaign and for this client (numeric)\n 16 - poutcome: outcome of the previous marketing campaign (categorical: "unknown","other","failure","success")\n\n Output variable (desired target):\n 17 - y - has the client subscribed a term deposit? (binary: "yes","no")\n', 'citation': None}}

	name	role	type	demographic	\
0	age	Feature	Integer	Age	
1	job	Feature	Categorical	Occupation	
2	marital	Feature	Categorical	Marital Status	
3	education	Feature	Categorical	Education Level	
4	default	Feature	Binary	None	
5	balance	Feature	Integer	None	
6	housing	Feature	Binary	None	
7	loan	Feature	Binary	None	
8	contact	Feature	Categorical	None	
9	day_of_week	Feature	Date	None	
10	month	Feature	Date	None	
11	duration	Feature	Integer	None	
12	campaign	Feature	Integer	None	
13	pdays	Feature	Integer	None	
14	previous	Feature	Integer	None	
15	poutcome	Feature	Categorical	None	
16	y	Target	Binary	None	

	description	units	missing_values
0	None	None	no
1	type of job (categorical: 'admin.','blue-colla...	None	no
2	marital status (categorical: 'divorced','marri...	None	no
3	(categorical: 'basic.4y','basic.6y','basic.9y'...	None	no
4	has credit in default?	None	no
5	average yearly balance	euros	no
6	has housing loan?	None	no

```
7                                    has personal loan?   None
no
8   contact communication type (categorical: 'cell...   None
yes
9                          last contact day of the week   None
no
10  last contact month of year (categorical: 'jan'...   None
no
11   last contact duration, in seconds (numeric). ...   None
no
12  number of contacts performed during this campa...   None
no
13  number of days that passed by after the client...   None
yes
14  number of contacts performed before this campa...   None
no
15  outcome of the previous marketing campaign (ca...   None
yes
16          has the client subscribed a term deposit?   None
no
```

```
# Preprocess the data
# Convert categorical columns to numerical using one-hot encoding
X = pd.get_dummies(X, drop_first=True)

# Split the dataset 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)

# Initialize the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=42)

# Train the model
clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier(random_state=42)
```

```
# Make predictions
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

```
Accuracy: 0.8694337953406075
Confusion Matrix:
[[11057   909]
 [  862   736]]
Classification Report:
              precision    recall  f1-score   support

          no       0.93      0.92      0.93     11966
         yes       0.45      0.46      0.45      1598

    accuracy                           0.87     13564
   macro avg       0.69      0.69      0.69     13564
weighted avg       0.87      0.87      0.87     13564


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