In [1]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import classification_report, confusion_matrix, accuracy_sc
import matplotlib.pyplot as plt

# Load dataset (assuming bank-full.csv is downloaded locally)
df = pd.read_csv("C:/Users/user/Desktop/bank-full.csv", sep=';')
df.head()
```

Out[1]:

	age	job	marital	education	default	balance	housing	loan	contact	da
0	58	management	married	tertiary	no	2143	yes	no	unknown	
1	44	technician	single	secondary	no	29	yes	no	unknown	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	
4	33	unknown	single	unknown	no	1	no	no	unknown	

In [2]:

```python
print(df.head())
print(df.info())
print(df['y'].value_counts())  # target variable distribution
```

```
   age           job  marital  education default  balance housing loan  \
0   58    management  married   tertiary      no     2143     yes   no   
1   44    technician   single  secondary      no       29     yes   no   
2   33  entrepreneur  married  secondary      no        2     yes  yes   
3   47   blue-collar  married    unknown      no     1506     yes   no   
4   33       unknown   single    unknown      no        1      no   no   

   contact  day month  duration  campaign  pdays  previous poutcome   y  
0  unknown    5   may       261         1     -1         0  unknown  no  
1  unknown    5   may       151         1     -1         0  unknown  no  
2  unknown    5   may        76         1     -1         0  unknown  no  
3  unknown    5   may        92         1     -1         0  unknown  no  
4  unknown    5   may       198         1     -1         0  unknown  no  
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
None
y
no     39922
yes     5289
Name: count, dtype: int64
```

In [3]:

```
#DATA PROPROCESSING
```

In [4]:

```
#Encode categorical variables
#Most features are categorical, so use one-hot encoding or label encoding.
```

In [5]:

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

# One-hot encode categorical variables
categorical_cols = df.select_dtypes(include=['object']).columns
df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
```

In [6]:

```
#Split data into features and target
```

In [7]:
```
X = df_encoded.drop('y', axis=1)
y = df_encoded['y']
```

In [8]:
```
# Split into train/test sets
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_
```

In [10]:
```
#DT CLASSIFIER
```

In [11]:
```
clf = DecisionTreeClassifier(random_state=42, max_depth=5)  # max_depth to preve
clf.fit(X_train, y_train)
```

Out[11]:
```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, random_state=42)
```

In [12]:
```
#EVALUATE MODEL
```

In [13]:
```
y_pred = clf.predict(X_test)

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

```
Accuracy: 0.9002506635210852

Confusion Matrix:
 [[11733   244]
 [ 1109   478]]

Classification Report:
               precision    recall  f1-score   support

           0       0.91      0.98      0.95     11977
           1       0.66      0.30      0.41      1587

    accuracy                           0.90     13564
   macro avg       0.79      0.64      0.68     13564
weighted avg       0.88      0.90      0.88     13564
```

In [14]:
```
#SEE WHAT FEATIRES MOST INFLUENCE PREDICTION
```

In [15]:
```
import numpy as np

importance = clf.feature_importances_
indices = np.argsort(importance)[::-1]
features = X.columns[indices]

for i in range(10):  # Top 10
    print(f"{features[i]}: {importance[indices[i]]:.4f}")
```

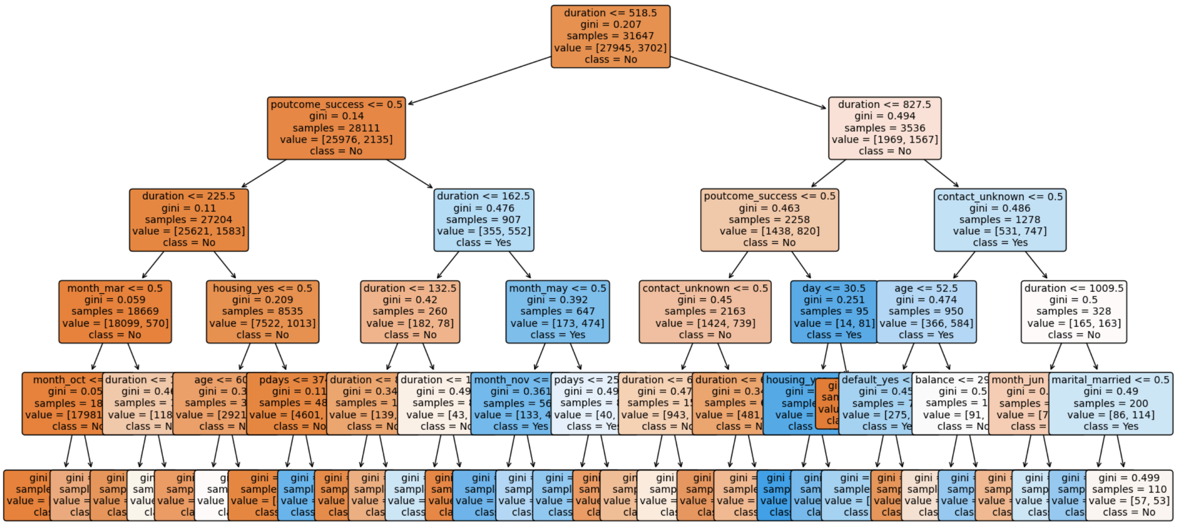
```
duration: 0.5611
poutcome_success: 0.2859
housing_yes: 0.0423
age: 0.0250
pdays: 0.0210
month_mar: 0.0208
contact_unknown: 0.0159
month_oct: 0.0137
month_may: 0.0040
marital_married: 0.0019
```

In [16]:

```
#VISUALIZE THE DT CLASSIFIER
```

In [17]:

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



In [18]:

```
#PREDICT WHETHER A NEW CLIENT WILL SUBSCRIBE OR NOT
```

In [ ]:

```
import pandas as pd

# New client data as a dictionary
new_client = {
    'age': 35,
    'job': 'technician',
    'marital': 'single',
    'education': 'tertiary',
    'default': 'no',
    'balance': 1500,
    'housing': 'yes',
    'loan': 'no',
    'contact': 'cellular',
    'day': 12,
    'month': 'may',
    'campaign': 1,
    'pdays': -1,
    'previous': 0,
    'poutcome': 'unknown'
}

# Convert to DataFrame
new_df = pd.DataFrame([new_client])
```

```
# Drop columns not used in training (e.g., duration)
# One-hot encode to match training
new_df_encoded = pd.get_dummies(new_df)

# Align columns with training set
# Add missing columns (set to 0) or drop extra columns
for col in X_train.columns:
    if col not in new_df_encoded:
        new_df_encoded[col] = 0
new_df_encoded = new_df_encoded[X_train.columns]

# Make prediction
prediction = model.predict(new_df_encoded)
probability = model.predict_proba(new_df_encoded)

# Output
print("Subscribed?" , "Yes" if prediction[0] == 1 else "No")
print("Probability of subscription:", probability[0][1])
```