In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data = pd.read_excel("Dataset (1).xlsx")
```

In [3]:
```
categorical_col = ['job', 'marital', 'education', 'default', 'housing', 'loan',
```

In [6]:
```
data=pd.read_excel("Dataset (1).xlsx")
data
```

Out[6]:

	age	job	marital	education	default	balance	housing	loan	conta
0	58	management	married	tertiary	no	2143	yes	no	unkno
1	44	technician	single	secondary	no	29	yes	no	unkno
2	33	entrepreneur	married	secondary	no	2	yes	yes	unkno
3	47	blue-collar	married	unknown	no	1506	yes	no	unkno
4	33	unknown	single	unknown	no	1	no	no	unkno
...	...	...	...	...	...	...	...	...	
45206	51	technician	married	tertiary	no	825	no	no	cellu
45207	71	retired	divorced	primary	no	1729	no	no	cellu
45208	72	retired	married	secondary	no	5715	no	no	cellu
45209	57	blue-collar	married	secondary	no	668	no	no	telephc
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellu

45211 rows × 17 columns

In [7]:
```
print(data.columns)
```

```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'y'],
      dtype='object')
```

In [8]:
```
categorical_col = ['job', 'marital', 'education', 'default', 'housing', 'loan',
```

In [9]:
```
from sklearn.preprocessing import LabelEncoder
```

In [12]:
```
le=LabelEncoder()
for col in categorical_col:
 data[col]=le.fit_transform(data[col])
```

In [13]:
```
data
```

Out[13]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
0	58	4	1	2	0	2143	1	0	2	4	
1	44	9	2	1	0	29	1	0	2	4	
2	33	2	1	1	0	2	1	1	2	4	
3	47	1	1	3	0	1506	1	0	2	4	
4	33	11	2	3	0	1	0	0	2	4	
...	...	...	...	...	...	...	...	...	...	...	
45206	51	9	1	2	0	825	0	0	0	16	
45207	71	5	0	0	0	1729	0	0	0	16	
45208	72	5	1	1	0	5715	0	0	0	16	
45209	57	1	1	1	0	668	0	0	1	16	
45210	37	2	1	1	0	2971	0	0	0	16	

45211 rows × 17 columns

In [14]:

```
X=data.drop("y",axis=1)
Y=data["y"].apply(lambda x: 1 if x=="yes" else 0)
```

In [15]:

```
from sklearn.model_selection import train_test_split
```

In [16]:

```
xtrain,xtest,ytrain,ytest= train_test_split(X,Y
,test_size=0.3
,random_state=42)
```

In [17]:

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
```

In [18]:

```
clf = DecisionTreeClassifier(max_depth=4, min_samples_split=20,random_state=42)
```

In [19]:

```
clf.fit(xtrain,ytrain)
```

Out[19]:

```
▾ DecisionTreeClassifier
▶ Parameters
```

In [20]:

```
ypred=clf.predict(xtest)
ypred
```

Out[20]:

```
array([0, 0, 0, ..., 0, 0, 0], shape=(13564,))
```

In [21]:

```
from sklearn.metrics import accuracy_score, classification_report
```

In [22]:

```
accuracy = accuracy_score(ytest,ypred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.888897080507225
```

In [23]:

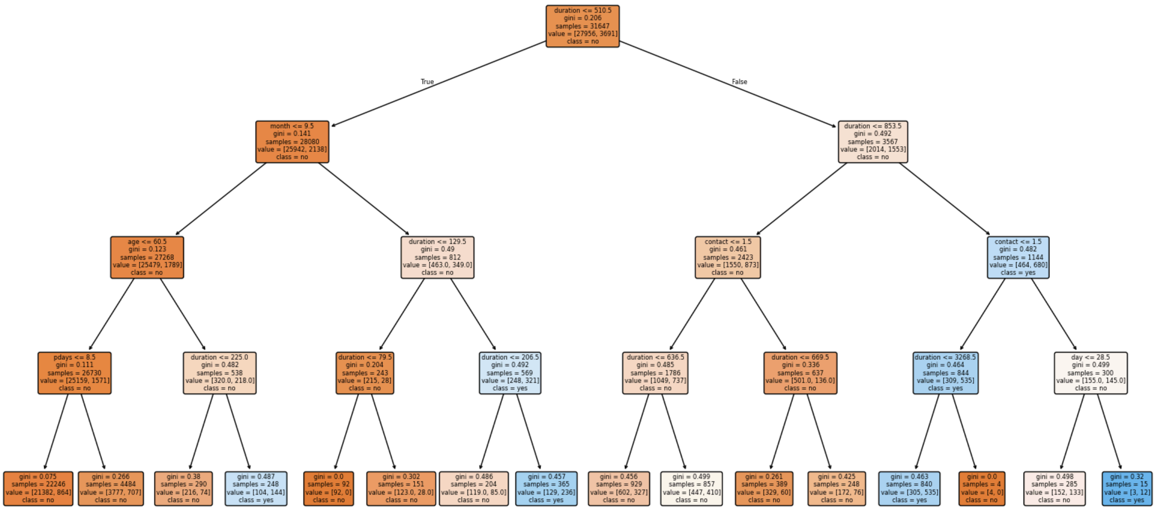
```
classification_report(ytest,ypred)
```

Out[23]:

```
'              precision    recall  f1-score   support\n\n           0       0.
90      0.98      0.94     11966\n           1      0.57      0.22      0.32
1598\n\n    accuracy                          0.89     13564\n   macro avg
0.74      0.60      0.63     13564\nweighted avg       0.87      0.89      0.87
13564\n'
```

In [24]:

```
plt.figure(figsize=(20,10))
plot_tree(clf,feature_names=X.columns, class_names=["no","yes"],filled=True,rou
plt.show()
```



In [ ]: