In [1]:
```
#Importing Libraries:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [10]:
```
#Reading the CSV File:
df = pd.read_csv("C:\\Users\\ashwa\\Downloads\\bank-marketing\\bank.csv")
```

In [11]:
```
# View the first few rows of the DataFrame
df.head()
```

Out[11]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	ma
1	56	admin.	married	secondary	no	45	no	no	unknown	5	ma
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	ma
3	55	services	married	secondary	no	2476	yes	no	unknown	5	ma
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	ma

In [12]:
```
#Display DataFrame information:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        11162 non-null  int64 
 1   job        11162 non-null  object
 2   marital    11162 non-null  object
 3   education  11162 non-null  object
 4   default    11162 non-null  object
 5   balance    11162 non-null  int64 
 6   housing    11162 non-null  object
 7   loan       11162 non-null  object
 8   contact    11162 non-null  object
 9   day        11162 non-null  int64 
 10  month      11162 non-null  object
 11  duration   11162 non-null  int64 
 12  campaign   11162 non-null  int64 
 13  pdays      11162 non-null  int64 
 14  previous   11162 non-null  int64 
 15  poutcome   11162 non-null  object
 16  deposit    11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

In [45]:
```
# Get summary statistics
print(df.describe())
```

```
                age       balance           day      duration      campai
gn  \
count  11162.000000  11162.000000  11162.000000  11162.000000  11162.0000
00
mean      41.231948   1528.538524     15.658036    371.993818       2.5084
21
std       11.913369   3225.413326      8.420740    347.128386       2.7220
77
min       18.000000  -6847.000000      1.000000      2.000000       1.0000
00
25%       32.000000    122.000000      8.000000    138.000000       1.0000
00
50%       39.000000    550.000000     15.000000    255.000000       2.0000
00
75%       49.000000   1708.000000     22.000000    496.000000       3.0000
00
max       95.000000  81204.000000     31.000000   3881.000000      63.0000
00

              pdays      previous
count  11162.000000  11162.000000
mean      51.330407      0.832557
std      108.758282      2.292007
min       -1.000000      0.000000
25%       -1.000000      0.000000
50%       -1.000000      0.000000
75%       20.750000      1.000000
max      854.000000     58.000000
```

In [13]:
```
df.tail()
```

Out[13]:

	age	job	marital	education	default	balance	housing	loan	contact	day	n
11157	33	blue-collar	single	primary	no	1	yes	no	cellular	20	
11158	39	services	married	secondary	no	733	no	no	unknown	16	
11159	32	technician	single	secondary	no	29	no	no	cellular	19	
11160	43	technician	married	secondary	no	0	no	yes	cellular	8	
11161	34	technician	married	secondary	no	0	no	no	cellular	9	

In [46]:
```
#Check for missing values
print(df.isnull().sum())
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
deposit      0
dtype: int64
```

In [47]:
```
#Get the shape of the DataFrame:
df.shape
```

Out[47]:
```
(11162, 17)
```

In [48]:
```
#Get the column names:
df.columns
```

Out[48]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housi
ng',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pday
s',
       'previous', 'poutcome', 'deposit'],
      dtype='object')
```

In [49]:
```
#Get the data types of each column:
df.dtypes
```

Out[49]:
```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
deposit      object
dtype: object
```

In [51]:
```
#Count of each data type:
df.dtypes.value_counts()
```

Out[51]:
```
object    10
int64      7
dtype: int64
```

In [52]:
```
#Count of duplicated rows:
df.duplicated().sum()
```

Out[52]:
```
0
```

In [53]:
```
#Count of missing values in each column:
df.isna().sum()
```

Out[53]:
```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
deposit      0
dtype: int64
```

In [20]:
```
# Select columns with data type 'object'
cat_cols = df.select_dtypes(include='object').columns

# Print the categorical columns
print(cat_cols)
```

```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'cont
act',
       'month', 'poutcome', 'deposit'],
      dtype='object')
```

In [54]:
```
# Select columns with data types other than 'object'
num_cols = df.select_dtypes(exclude='object').columns

# Print the numerical columns
print(num_cols)
```

```
Index(['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previou
s'], dtype='object')
```

In [55]:
```
# Get summary statistics of the numerical columns
summary_statistics = df.describe()

# Print the summary statistics
print("Summary statistics of the numerical columns:")
print(summary_statistics)
```

```
Summary statistics of the numerical columns:
                age       balance           day      duration      campai
gn  \
count  11162.000000  11162.000000  11162.000000  11162.000000  11162.0000
00
mean      41.231948   1528.538524     15.658036    371.993818      2.5084
21
std       11.913369   3225.413326      8.420740    347.128386      2.7220
77
min       18.000000  -6847.000000      1.000000      2.000000      1.0000
00
25%       32.000000    122.000000      8.000000    138.000000      1.0000
00
50%       39.000000    550.000000     15.000000    255.000000      2.0000
00
75%       49.000000   1708.000000     22.000000    496.000000      3.0000
00
max       95.000000  81204.000000     31.000000   3881.000000     63.0000
00

              pdays      previous
count  11162.000000  11162.000000
mean      51.330407      0.832557
std      108.758282      2.292007
min       -1.000000      0.000000
25%       -1.000000      0.000000
50%       -1.000000      0.000000
75%       20.750000      1.000000
max      854.000000     58.000000
```

In [56]:
```
# Get summary statistics of the categorical columns
summary_statistics_categorical = df.describe(include='object')

# Print the summary statistics for categorical columns
print("Summary statistics of the categorical columns:")
print(summary_statistics_categorical)
```

```
Summary statistics of the categorical columns:
                job  marital  education default housing   loan   contact
\
count         11162    11162      11162   11162   11162  11162     11162
unique           12        3          4       2       2      2         3
top     management  married  secondary      no      no     no  cellular
freq           2566     6351       5476   10994    5881   9702      8042

        month poutcome deposit
count   11162    11162   11162
unique     12        4       2
top       may  unknown      no
freq     2824     8326    5873
```

In [57]:
```
#Importing LabelEncoder:
from sklearn.preprocessing import LabelEncoder

#Creating a LabelEncoder instance:
lb = LabelEncoder()

#Applying LabelEncoder to each column in the DataFrame:
df_encoded = df1.apply(lb.fit_transform)

## Print the transformed DataFrame
df_encoded
```

Out[57]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	41	0	1	1	0	2288	1	0	2	4	8
1	38	0	1	1	0	469	0	0	2	4	8
2	23	9	1	1	0	1618	1	0	2	4	8
3	37	7	1	1	0	2356	1	0	2	4	8
4	36	0	1	2	0	608	0	0	2	4	8
...	...	...	...	...	...	...	...	...	...	...	...
11157	15	1	2	0	0	425	1	0	0	19	0
11158	21	7	1	1	0	1149	0	0	2	15	6
11159	14	9	2	1	0	453	0	0	0	18	1
11160	25	9	1	1	0	424	0	1	0	7	8
11161	16	9	1	1	0	424	0	0	0	8	5

11162 rows × 17 columns

In [58]:
```
# Get the frequency of each value in the 'deposit' column
deposit_value_counts = df_encoded['deposit'].value_counts()

# Print the value counts
print("Value counts for the 'deposit' column:")
print(deposit_value_counts)
```

```
Value counts for the 'deposit' column:
0    5873
1    5289
Name: deposit, dtype: int64
```

In [59]:
```
x = df_encoded.drop('deposit',axis=1)  # independent variable
y = df_encoded['deposit']              # dependent variable

#Shape and Type Checks:
print(x.shape)
print(y.shape)
print(type(x))
print(type(y))
```

```
(11162, 16)
(11162,)
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

In [33]:
```
from sklearn.model_selection import train_test_split
```

In [60]:
```
# Split the data into training and testing sets
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_

## Print the shapes of the resulting sets
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(8371, 16)
(2791, 16)
(8371,)
(2791,)
```

In [35]:
```
from sklearn.metrics import confusion_matrix,classification_report,accurac
```

In [61]:
```
def eval_model(y_test,y_pred):
    # Calculate accuracy score
    acc = accuracy_score(y_test,y_pred)
    print('Accuracy_Score',acc)

    # Calculate confusion matrix
    cm = confusion_matrix(y_test,y_pred)
    print('Confusion Matrix\n',cm)

    # Print classification report
    print('Classification Report\n',classification_report(y_test,y_pred))
```

In [37]:
```
def mscore(model):

    # # Calculate training and testing scores
    train_score = model.score(x_train,y_train)
    test_score = model.score(x_test,y_test)

    # Print the scores
    print('Training Score',train_score)
    print('Testing Score',test_score)
```

In [38]:
```
#Importing the DecisionTreeClassifier:
from sklearn.tree import DecisionTreeClassifier

#Creating the Model:
dt = DecisionTreeClassifier(criterion='gini',max_depth=5,min_samples_split

#Fitting the Model:
dt.fit(x_train,y_train)
```

Out[38]:
```
DecisionTreeClassifier(max_depth=5, min_samples_split=10)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [39]:
```
#Evaluate Model Performance:
mscore(dt)
```

```
Training Score 0.8175845179787361
Testing Score 0.8007882479398065
```

In [40]:
```
# Make predictions on the test set
ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

```
[0 0 0 ... 0 1 0]
```

In [41]:
```
## Evaluate the predictions
eval_model(y_test,ypred_dt)
```

```
Accuracy_Score 0.8007882479398065
Confusion Matrix
 [[1078  379]
 [ 177 1157]]
Classification Report
              precision    recall  f1-score   support

           0       0.86      0.74      0.79      1457
           1       0.75      0.87      0.81      1334

    accuracy                           0.80      2791
   macro avg       0.81      0.80      0.80      2791
weighted avg       0.81      0.80      0.80      2791
```

In [42]:
```
from sklearn.tree import plot_tree
```

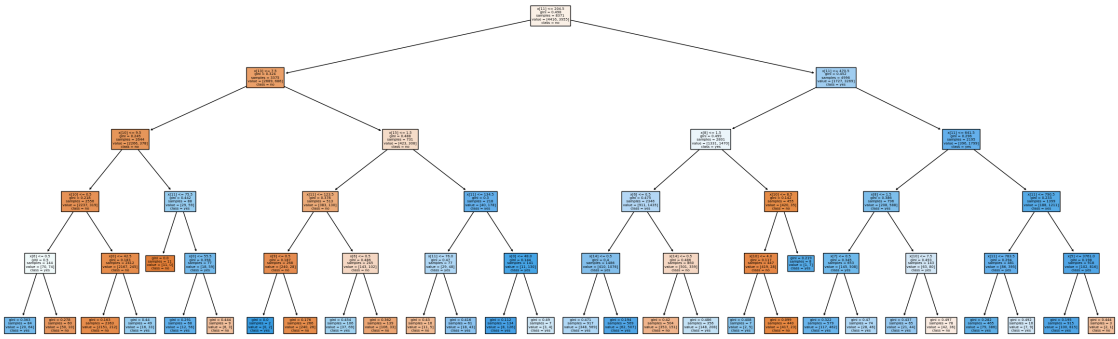
In [43]:
```
# Define class names and feature names
cn = ['no','yes']
fn = x_train.columns

# Print feature names and class names for reference
print(fn)
print(cn)
```

```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housi
ng',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pday
s',
       'previous', 'poutcome'],
      dtype='object')
['no', 'yes']
```

In [44]:
```
# Plot the decision tree
plt.figure(figsize=(30,10)) # Adjust the figure size as needed
plot_tree(dt,class_names=cn,filled=True)

# Display the plot
plt.show()
```



In [ ]: