```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Load dataset
df = pd.read_csv('/content/bank-full.csv',sep=';')
df
```

b	marital	education	default	balance	housing	loan	contact	day	month	duration
nt	married	tertiary	no	2143	yes	no	unknown	5	may	261
n	single	secondary	no	29	yes	no	unknown	5	may	151
ır	married	secondary	no	2	yes	yes	unknown	5	may	76
ar	married	unknown	no	1506	yes	no	unknown	5	may	92
n	single	unknown	no	1	no	no	unknown	5	may	198
..	...	...	...	...	...	...	...	...	...	...
n	married	tertiary	no	825	no	no	cellular	17	nov	977
d	divorced	primary	no	1729	no	no	cellular	17	nov	456
d	married	secondary	no	5715	no	no	cellular	17	nov	1127
ar	married	secondary	no	668	no	no	telephone	17	nov	508
ır	married	secondary	no	2971	no	no	cellular	17	nov	361

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#Checking for null values
df.isnull().sum()
```

	0
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
y	0

dtype: int64

```
df.describe()
```

What can I help you build?

	age					
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000(
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197(

std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.1287
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000(
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000(
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000(
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000(
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000(

```
df.info()
```

```
<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
```

```
#encoding  categorical variables
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['job'] = le.fit_transform(df['job'])
df['marital'] = le.fit_transform(df['marital'])
df['education'] = le.fit_transform(df['education'])
df['default'] = le.fit_transform(df['default'])
df['housing'] = le.fit_transform(df['housing'])
df['loan'] = le.fit_transform(df['loan'])
df['contact'] = le.fit_transform(df['contact'])
df['month'] = le.fit_transform(df['month'])
df['poutcome'] = le.fit_transform(df['poutcome'])
df['y'] = le.fit_transform(df['y'])
```

```
#Splitting Data
from sklearn.model_selection import train_test_split
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)
```

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
y_pred = dt.predict(x_test)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

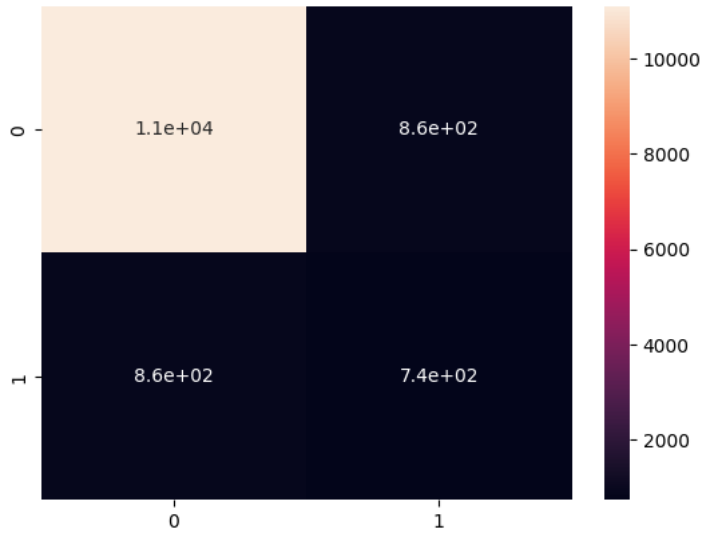
```
#Evaluating Model
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
0.8728988498967856
[[11105   861]
 [  863   735]]
              precision    recall  f1-score   support
```

```
              0       0.93      0.93      0.93     11966
              1       0.46      0.46      0.46      1598

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

```
#Creating Confusion matrix
cm = confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True)
plt.show()
```




```
#Create Decision Tree
from sklearn.tree import plot_tree
plt.figure(figsize=(15,10))
plot_tree(dt,filled=True)
plt.show()
```