Task-03 Build a decision tree classifier to predict whether a customer will purchase a product or service based on their demographic and behavioral data. Use a dataset such as the Bank Marketing dataset from the UCI Machine Learning Repository. Sample Dateset :- https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

Decision Tree Terminologies

Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

**Leaf Node:* *Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

Branch/Sub Tree: A tree formed by splitting the tree.

Pruning: Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

IMPORT THE REQUIRED LIBRARIES

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

Bringing data to on-board using pandas library

```
from google.colab import files
data=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving bank full.csv to bank full.csv
```

Reading the csv file

```
rawfile=pd.read_csv('bank full.csv')
```

Taking shallow copy of the original dataset

```
df=rawfile.copy()
```

```
df
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previou
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	

45211 rows × 17 columns

EDA

viewing the counts of record and feature

```
df.shape
```

```
(45211, 17)
```

viewing the first five feature and record using head function

```
df.head(10)
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	p
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	u
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	u
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	u
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	u
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	u
5	35	management	married	tertiary	no	231	yes	no	unknown	5	may	139	1	-1	0	u
6	28	management	single	tertiary	no	447	yes	yes	unknown	5	may	217	1	-1	0	u
7	42	entrepreneur	divorced	tertiary	yes	2	yes	no	unknown	5	may	380	1	-1	0	u
8	58	retired	married	primary	no	121	yes	no	unknown	5	may	50	1	-1	0	u
9	43	technician	single	secondary	no	593	yes	no	unknown	5	may	55	1	-1	0	u

viewing the last five feature and record using tail function

```
df.tail(10)
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previou
45201	53	management	married	tertiary	no	583	no	no	cellular	17	nov	226	1	184	
45202	34	admin.	single	secondary	no	557	no	no	cellular	17	nov	224	1	-1	
45203	23	student	single	tertiary	no	113	no	no	cellular	17	nov	266	1	-1	
45204	73	retired	married	secondary	no	2850	no	no	cellular	17	nov	300	1	40	
45205	25	technician	single	secondary	no	505	no	yes	cellular	17	nov	386	2	-1	
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	

The info function highlights the total number of rows in the dataset, names of the columns, their data type, and any missing value. It is used to print the summary of a data frame. It is very important to know the data types of variables that aides in understanding the nature of data

```
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
```

viewing the over all columns in a dataset

```
df.columns
```

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

The describe() method is used for calculating some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame. It analyzes both numeric and object series and also the DataFrame column sets of mixed data types.

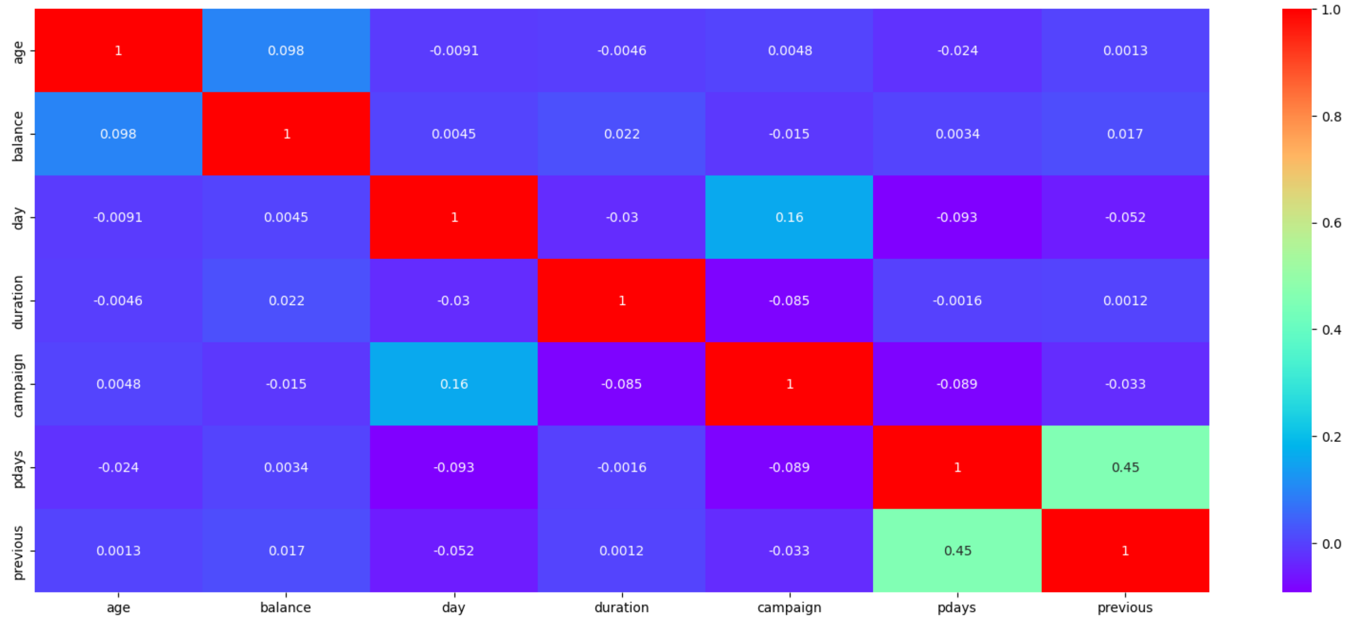
```
df.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

Here heatmap is used to see the correlation

```
plt.figure(figsize=(20,8))
sns.heatmap(data=df.corr(), annot=True, cmap='rainbow')
```

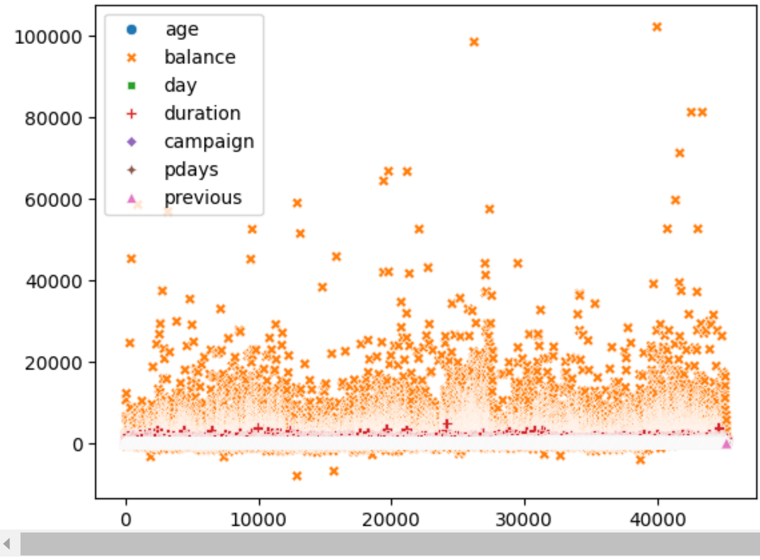
```
<ipython-input-15-bd315876de51>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  sns.heatmap(data=df.corr(), annot=True, cmap='rainbow')
<Axes: >
```



Data visualization

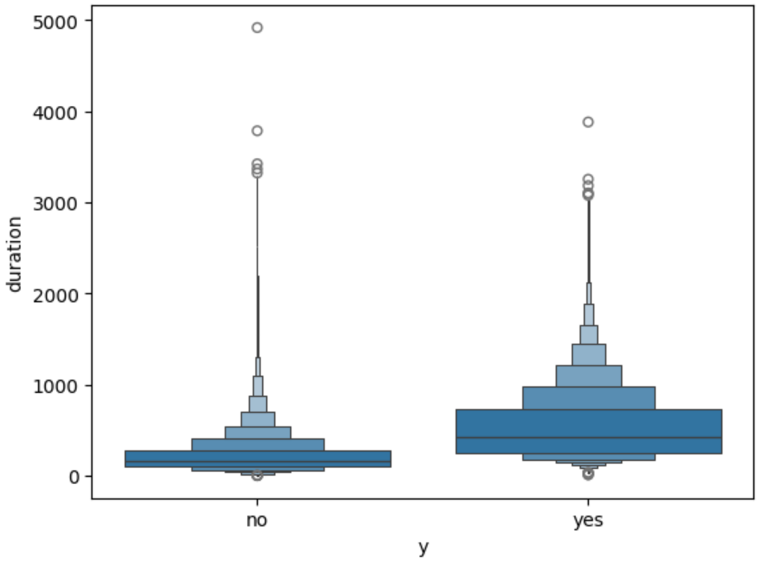
```
sns.scatterplot(df)
```

```
<Axes: >
/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: UserWarning: Creating legend with loc="best" can be slow with lar
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Creating legend with loc="best" can be slow wit
  fig.canvas.print_figure(bytes_io, **kw)
```



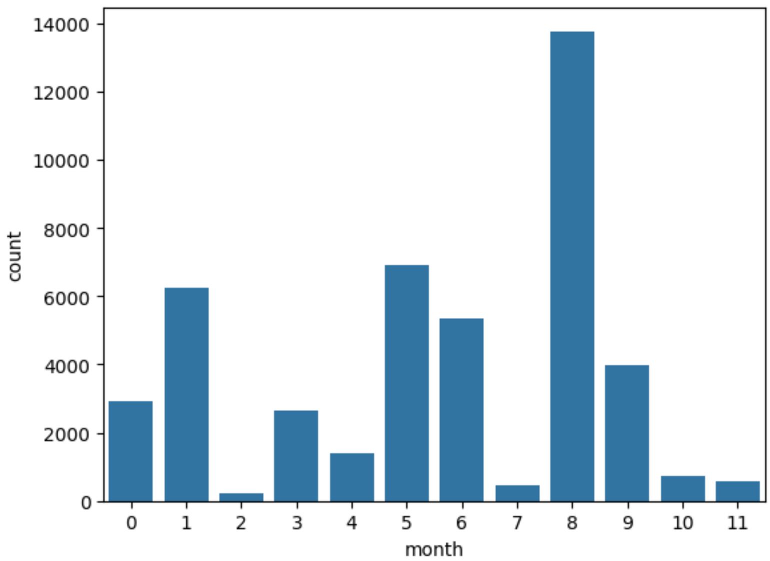

```
sns.boxenplot(y=df['duration'],x=df['y'])
```

```
<Axes: xlabel='y', ylabel='duration'>
```




```
sns.countplot(x=df['month'])
```

```
<Axes: xlabel='month', ylabel='count'>
```



Data cleaning

```
df.isnull().sum(axis=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
```

Label encoding the object into integer

```
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'])
```

```
df
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutc
0	58	4	1	2	0	2143	1	0	2	5	8	261	1	-1	0	
1	44	9	2	1	0	29	1	0	2	5	8	151	1	-1	0	
2	33	2	1	1	0	2	1	1	2	5	8	76	1	-1	0	
3	47	1	1	3	0	1506	1	0	2	5	8	92	1	-1	0	
4	33	11	2	3	0	1	0	0	2	5	8	198	1	-1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	9	1	2	0	825	0	0	0	17	9	977	3	-1	0	
45207	71	5	0	0	0	1729	0	0	0	17	9	456	2	-1	0	
45208	72	5	1	1	0	5715	0	0	0	17	9	1127	5	184	3	
45209	57	1	1	1	0	668	0	0	1	17	9	508	4	-1	0	
45210	37	2	1	1	0	2971	0	0	0	17	9	361	2	188	11	

45211 rows × 17 columns

x and y split

```
x = df.drop('y', axis=1)
y = df['y']
```

```
x
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutc
0	58	4	1	2	0	2143	1	0	2	5	8	261	1	-1	0	
1	44	9	2	1	0	29	1	0	2	5	8	151	1	-1	0	
2	33	2	1	1	0	2	1	1	2	5	8	76	1	-1	0	
3	47	1	1	3	0	1506	1	0	2	5	8	92	1	-1	0	
4	33	11	2	3	0	1	0	0	2	5	8	198	1	-1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	9	1	2	0	825	0	0	0	17	9	977	3	-1	0	
45207	71	5	0	0	0	1729	0	0	0	17	9	456	2	-1	0	
45208	72	5	1	1	0	5715	0	0	0	17	9	1127	5	184	3	
45209	57	1	1	1	0	668	0	0	1	17	9	508	4	-1	0	
45210	37	2	1	1	0	2971	0	0	0	17	9	361	2	188	11	

45211 rows × 16 columns

```
y
```

```
0        0
1        0
2        0
3        0
4        0
        ..
45206    1
45207    1
45208    1
45209    0
45210    0
Name: y, Length: 45211, dtype: int64
```

Performing standard scaler method for better understanding of the machine to imporve and learn

```
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
scale.fit_transform(x)
```

```
array([[ 1.60696496, -0.10381968, -0.27576178, ..., -0.41145311,
        -0.25194037,  0.44489814],
       [ 0.28852927,  1.42400783,  1.3683719 , ..., -0.41145311,
        -0.25194037,  0.44489814],
       [-0.74738448, -0.71495069, -0.27576178, ..., -0.41145311,
        -0.25194037,  0.44489814],
       ...,
       [ 2.92540065,  0.20174582, -0.27576178, ...,  1.43618859,
         1.05047333, -0.56617504],
       [ 1.51279098, -1.02051619, -0.27576178, ..., -0.41145311,
        -0.25194037,  0.44489814],
       [-0.37068857, -0.71495069, -0.27576178, ...,  1.4761376 ,
         4.52357654, -1.57724822]])
```

Train test split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=174)
```

Import and build 'DECISION TREE' Algorithm

Step-1: Begin the tree with the root node, says S, which contains the complete dataset. Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM). Step-3: Divide the S into subsets that contains possible values for the best attributes. Step-4: Generate the decision tree node, which contains the best attribute. Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(random_state=174)
dtc.fit(x_train,y_train)
y_pred=dtc.predict(x_test)
```

```
y_pred
```

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

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
score = dtc.score(x_test, y_test)
```

```
cm
```

```
array([[7364,  594],
       [ 587,  498]])
```

viewing the accuracy

```
score
```

```
0.8694017472077851
```

```
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='coolwarm', xticklabels=['class_0', 'class_1'], yticklabels=['class_0', 'class_1'])
```