In [1]:
```
# Step 1: import neccessary libraries and load the data set
import pandas as pd
import numpy as np

bank_data=pd.read_csv("C:/1562_AIML/banking.csv")
bank_data.head()
```

Out[1]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_w
0	44	blue-collar	married	basic.4y	unknown	yes	no	cellular	aug	
1	53	technician	married	unknown	no	no	no	cellular	nov	
2	28	management	single	university.degree	no	yes	no	cellular	jun	
3	39	services	married	high.school	no	no	no	cellular	apr	
4	55	retired	married	basic.4y	no	yes	no	cellular	aug	

5 rows × 21 columns

In [2]:
```
bank_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp_var_rate    41188 non-null  float64
 16  cons_price_idx  41188 non-null  float64
 17  cons_conf_idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr_employed     41188 non-null  float64
 20  y               41188 non-null  int64
dtypes: float64(5), int64(6), object(10)
memory usage: 6.6+ MB
```

In [3]:
```
bank_data.y.value_counts()
```

Out[3]:
```
0    36548
1     4640
Name: y, dtype: int64
```

In [4]:
```
# find unique values of default
bank_data['default'].unique()
```

Out[4]:
```
array(['unknown', 'no', 'yes'], dtype=object)
```

In [6]:
```
bank_data.default.value_counts()
```

Out[6]:
```
no         32588
unknown     8597
yes            3
Name: default, dtype: int64
```

In [7]:
```
#convert default column to numeric
bank_data['default']=bank_data['default'].map({'unknown':0,'no':0,'yes':1})
```

In [8]:
```
bank_data.default.value_counts()
```

Out[8]:
```
0    41185
1        3
Name: default, dtype: int64
```

In [9]:
```
#consider fdatures
features=['age','default','cons_price_idx','cons_conf_idx']
x=bank_data[features]
y=bank_data.y
```

In [10]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=
```

In [13]:
```
#Build the random forest model
from sklearn.ensemble import RandomForestClassifier
#create an instance of Random Forest Classifier
rf=RandomForestClassifier(n_estimators=1000,criterion='entropy',max_depth=5,max
```

In [14]:
```
# train the model
rf.fit(x_train,y_train)
```

Out[14]:
```
RandomForestClassifier(criterion='entropy', max_depth=5, max_features='sqrt',
                       n_estimators=1000, random_state=45)
```

In [15]:
```
# test the model
y_pred=rf.predict(x_test)
```

In [16]:
```
# compute the accuracy of the model
from sklearn.metrics import accuracy_score
print ("Accuracy:",accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.8866229667394999
```

In [*]:
```
#Perform hyper-parameter tuning
from sklearn.model_selection import GridSearchCV

#Create a paramter grid
param_grid = {
    'n_estimators' : [100, 500, 1000, 2000],
    'criterion' : ['gini', 'entropy'],
    'max_depth' : [None, 5, 10],
    'max_features' : ['sqrt', 'log2'],
    'bootstrap' : [True, False]
}
#Create RandomForest
rfc = RandomForestClassifier(random_state=45)

grid_s = GridSearchCV(estimator=rfc, param_grid=param_grid, scoring='accuracy',
grid_s.fit(x_train, y_train)
```

In [ ]:
```
```