In [1]:
```
import pandas as pd
```

In [2]:
```
Bank=pd.read_excel("E:\Data science training\R AND PYTHON KPMG\stat and ml\ASSIGNMENT\Bank Marketing.xlsx")
Bank.head()
```

Out[2]:

	Age	Job	Marital Status	Account Balance	Own House	Personal Loan	No of campaigns	Subscription
0	59	unemployed	married	0	0	0	0	0
1	36	Others	married	3057	0	0	0	1
2	47	blue-collar	divorced	126	1	0	0	1
3	43	management	divorced	388	1	0	0	1
4	34	self-employed	single	462	0	0	0	1

In [3]:
```
Bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Age              4521 non-null   int64 
 1   Job              4521 non-null   object
 2   Marital Status   4521 non-null   object
 3   Account Balance  4521 non-null   int64 
 4   Own House        4521 non-null   int64 
 5   Personal Loan    4521 non-null   int64 
 6   No of campaigns  4521 non-null   int64 
 7   Subscription     4521 non-null   int64 
dtypes: int64(6), object(2)
memory usage: 282.7+ KB
```

In [4]:
```
Bank.isnull().sum()
```

Out[4]:
```
Age                0
Job                0
Marital Status     0
Account Balance    0
Own House          0
Personal Loan      0
No of campaigns    0
Subscription       0
dtype: int64
```

In [5]:
```
cat_col = ['Job','Marital Status']
```

In [6]:
```
Bank_dummy = pd.get_dummies(Bank,columns=cat_col,drop_first=True)
```

In [7]:
```
Bank_dummy.shape
```

Out[7]:
```
(4521, 12)
```

In [8]:
```
Bank_dummy.head()
```

Out[8]:

	Age	Account Balance	Own House	Personal Loan	No of campaigns	Subscription	Job_blue-collar	Job_management	Job_self-employed	Job_unemployed	Marital Status_married	Marital Status_single
0	59	0	0	0	0	0	0	0	0	1	1	0
1	36	3057	0	0	0	1	0	0	0	0	1	0
2	47	126	1	0	0	1	1	0	0	0	0	0
3	43	388	1	0	0	1	0	1	0	0	0	0
4	34	462	0	0	0	1	0	0	1	0	0	1

In [9]:
```
#Identifying the input and output variables
Y = Bank_dummy[['Subscription']]
X = Bank_dummy.drop(columns=['Subscription'])
```

In [10]:
```
X.head()
```

Out[10]:

	Age	Account Balance	Own House	Personal Loan	No of campaigns	Job_blue-collar	Job_management	Job_self-employed	Job_unemployed	Marital Status_married	Marital Status_single
0	59	0	0	0	0	0	0	0	1	1	0
1	36	3057	0	0	0	0	0	0	0	1	0
2	47	126	1	0	0	1	0	0	0	0	0
3	43	388	1	0	0	0	1	0	0	0	0
4	34	462	0	0	0	0	0	1	0	0	1

In [11]:
```
##Spilliting the data into train and test

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=42)
len(x_train),len(x_test),len(y_train),len(y_test)
```

Out[11]:
```
(3616, 905, 3616, 905)
```

Building the Random Forest model:

In [12]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=1000,random_state=42)
rf = model.fit(x_train,y_train)
print('The model has been built successfully!! yeah')
```

```
C:\Users\Pratik\AppData\Local\Temp/ipykernel_5368/385801585.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  rf = model.fit(x_train,y_train)
The model has been built successfully!! yeah
```

In [13]:
```
# Predicting the model on test data
y_test
```

Out[13]:

	Subscription
2398	0
800	0
2288	0
2344	0
3615	0
...	...
3589	0
3128	0
3259	0
4239	0
2233	0

905 rows × 1 columns

In [14]:
```
y_test['Prediction'] = model.predict(x_test)
y_test
```

Out[14]:

	Subscription	Prediction
2398	0	0
800	0	0
2288	0	0
2344	0	0
3615	0	0
...	...	...
3589	0	0
3128	0	0
3259	0	0
4239	0	0
2233	0	0

905 rows × 2 columns

In [15]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
print(confusion_matrix(y_test['Subscription'],y_test['Prediction']))
```

```
[[777  26]
 [ 95   7]]
```

In [16]:
```
print(accuracy_score(y_test['Subscription'],y_test['Prediction']))
```

```
0.8662983425414365
```

In [17]:
```
## My Model Accuracy Score is 0.8662 or 86.62% correctly predicted
```

2) What is the accuracy of Support Vector Machine? How does changing the model affect the accuracy

In [18]:
```
#Identifying the input and output variables
Y1 = Bank_dummy[['Subscription']]
x1 = Bank_dummy.drop(columns=['Subscription'])
```

In [19]:
```
X1.head()
```

Out[19]:

	Age	Account Balance	Own House	Personal Loan	No of campaigns	Job_blue-collar	Job_management	Job_self-employed	Job_unemployed	Marital Status_married	Marital Status_single
0	59	0	0	0	0	0	0	0	1	1	0
1	36	3057	0	0	0	0	0	0	0	1	0
2	47	126	1	0	0	1	0	0	0	0	0
3	43	388	1	0	0	0	1	0	0	0	0
4	34	462	0	0	0	0	0	1	0	0	1

In [20]:
```
#Spilliting the data into train and test

from sklearn.model_selection import train_test_split
x1_train,x1_test,y1_train,y1_test = train_test_split(X1,Y1,test_size=0.2,random_state=42)
len(x1_train),len(x1_test),len(y1_train),len(y1_test)
```

Out[20]:
```
(3616, 905, 3616, 905)
```

In [21]:
```
from sklearn.svm import SVC
model1 = SVC(random_state=42)
```

In [22]:
```
# fitting the Model
svc_model = model1.fit(x1_train,y1_train)
print('The model has been built successfully!! yeah')
```

```
C:\Users\Pratik\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
The model has been built successfully!! yeah
```

In [23]:
```
##Prediction of test data
y1_test
```

Out[23]:

	Subscription
2398	0
800	0
2288	0
2344	0
3615	0
...	...
3589	0
3128	0
3259	0
4239	0
2233	0

905 rows × 1 columns

In [24]:
```
y1_test['Prediction'] = svc_model.predict(x1_test)
```

In [25]:
```
y1_test
```

Out[25]:

	Subscription	Prediction
2398	0	0
800	0	0
2288	0	0
2344	0	0
3615	0	0
...	...	...
3589	0	0
3128	0	0
3259	0	0
4239	0	0
2233	0	0

905 rows × 2 columns

In [26]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
print(confusion_matrix(y1_test['Subscription'],y1_test['Prediction']))
```

```
[[803   0]
 [102   0]]
```

In [27]:
```
print(accuracy_score(y1_test['Subscription'],y1_test['Prediction']))
```

```
0.8872928176795558
```

In [1]:
```
##Here we can see that using SVM accuracy of my model increase it's predict More no of Data correctly but
#  it's only Predict  one kind of Data Correctly which is those doesn't have subscription("0").
```

In [28]:
```
model.feature_importances_
```

```
array([0.2979414 , 0.47291481, 0.02827675, 0.01718792, 0.09330933,
       0.01799654, 0.0224766 , 0.00881842, 0.00651201, 0.01989496,
       0.01467124])
```

In [29]:
```
forest_importance = pd.Series(model.feature_importances_,index=X.columns)
forest_importance.sort_values(ascending=False).head(4)
```

Out[29]:
```
Account Balance    0.472915
Age                0.297941
No of campaigns    0.093309
Own House          0.028277
dtype: float64
```

In [2]:
```
## 3) above 4 columns are most importance factor in Random Forest Model.Here Customer Account Balance and Age are
#  most important Factor For getting Subscription.
```

In [ ]:
```
# 4)SVM Model  only Predict  one kind of Data Correctly which is those doesn't have subscription("0").
# so though it's have a better accuracy than Random Forest but Random Forest overall have a better Model bcz it's predict both the Data correctly
#   with good accuracy.
```