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

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

Out[2]:

	id	Gender	Age	Driving_License	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Vintage Score	Response
0	1	Male	44	1	0	> 2 Years	Yes	40454	217	1
1	2	Male	76	1	0	1-2 Year	No	33536	183	0
2	3	Male	47	1	0	> 2 Years	Yes	38294	27	1
3	4	Male	21	1	1	< 1 Year	No	28619	203	0
4	5	Female	29	1	1	< 1 Year	No	27496	39	0

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   id                   3999 non-null   int64 
 1   Gender               3999 non-null   object
 2   Age                  3999 non-null   int64 
 3   Driving_License      3999 non-null   int64 
 4   Previously_Insured   3999 non-null   int64 
 5   Vehicle_Age          3999 non-null   object
 6   Vehicle_Damage       3999 non-null   object
 7   Annual_Premium       3999 non-null   int64 
 8   Vintage Score        3999 non-null   int64 
 9   Response             3999 non-null   int64 
dtypes: int64(7), object(3)
memory usage: 312.5+ KB
```

In [4]:
```
# Checking the null value.
insurance.isnull().sum()
```

Out[4]:
```
id                   0
Gender               0
Age                  0
Driving_License      0
Previously_Insured   0
Vehicle_Age          0
Vehicle_Damage       0
Annual_Premium       0
Vintage Score        0
Response             0
dtype: int64
```

In [5]:
```
# create a seperate object column.
cat_col = ['Gender','Vehicle_Age','Vehicle_Damage']
```

In [6]:
```
## creating Dummy of the Data
insurance_dummy = pd.get_dummies(insurance,columns=cat_col,drop_first=True)
```

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

Out[7]:
```
(3999, 11)
```

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

Out[8]:

	id	Age	Driving_License	Previously_Insured	Annual_Premium	Vintage Score	Response	Gender_Male	Vehicle_Age_< 1 Year	Vehicle_Age_> 2 Years	Vehicle_Damage_Yes
0	1	44	1	0	40454	217	1	1	0	1	1
1	2	76	1	0	33536	183	0	1	0	0	0
2	3	47	1	0	38294	27	1	1	0	1	1
3	4	21	1	1	28619	203	0	1	1	0	0
4	5	29	1	1	27496	39	0	0	1	0	0

In [9]:
```
# here in this data i don't need "id" column so exclude from the Data
#Identifying the input and output variables
Y = insurance_dummy[['Response']]
X = insurance_dummy.drop(columns=['Response','id'])
```

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

Out[10]:

	Age	Driving_License	Previously_Insured	Annual_Premium	Vintage Score	Gender_Male	Vehicle_Age_< 1 Year	Vehicle_Age_> 2 Years	Vehicle_Damage_Yes
0	44	1	0	40454	217	1	0	1	1
1	76	1	0	33536	183	1	0	0	0
2	47	1	0	38294	27	1	0	1	1
3	21	1	1	28619	203	1	1	0	0
4	29	1	1	27496	39	0	1	0	0

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]:
```
(3199, 800, 3199, 800)
```

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_8264/3858015865.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]:

	Response
1760	0
3326	1
1770	0
3176	0
2099	0
...	...
2510	0
2752	0
1869	1
423	0
2990	0

800 rows × 1 columns

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

Out[14]:

	Response	Prediction
1760	0	0
3326	1	0
1770	0	0
3176	0	0
2099	0	0
...	...	...
2510	0	0
2752	0	0
1869	1	0
423	0	1
2990	0	0

800 rows × 2 columns

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

```
[[672  32]
 [ 81  15]]
```

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

```
0.85875
```

In [17]:
```
## My Model Accuracy Score is 0.85875 or 85.87% 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 = insurance_dummy[['Response']]
X1 = insurance_dummy.drop(columns=['Response','id'])
```

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

Out[19]:

	Age	Driving_License	Previously_Insured	Annual_Premium	Vintage Score	Gender_Male	Vehicle_Age_< 1 Year	Vehicle_Age_> 2 Years	Vehicle_Damage_Yes
0	44	1	0	40454	217	1	0	1	1
1	76	1	0	33536	183	1	0	0	0
2	47	1	0	38294	27	1	0	1	1
3	21	1	1	28619	203	1	1	0	0
4	29	1	1	27496	39	0	1	0	0

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]:
```
(3199, 800, 3199, 800)
```

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]:

	Response
1760	0
3326	1
1770	0
3176	0
2099	0
...	...
2510	0
2752	0
1869	1
423	0
2990	0

800 rows × 1 columns

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

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

Out[25]:

	Response	Prediction
1760	0	0
3326	1	0
1770	0	0
3176	0	0
2099	0	0
...	...	...
2510	0	0
2752	0	0
1869	1	0
423	0	0
2990	0	0

800 rows × 2 columns

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

```
[[704   0]
 [ 96   0]]
```

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

```
0.88
```

In [28]:
```
## 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 Response("0").
```

In [29]:
```
model.feature_importances_
```

Out[29]:
```
array([2.30517887e-01, 2.81301997e-04, 5.37530127e-02, 2.70065606e-01,
       3.12616760e-01, 2.27542185e-02, 2.21018753e-02, 1.23526313e-02,
       7.55575076e-02])
```

In [30]:
```
forest_importance = pd.Series(model.feature_importances_,index=X.columns)
forest_importance.sort_values(ascending=False).head(4)
```

Out[30]:
```
Vintage Score          0.312617
Annual_Premium         0.270066
Age                    0.230518
Vehicle_Damage_Yes     0.075558
dtype: float64
```

In [31]:
```
##3) above 4 columns are most importance factor in Random Forest Model.Here CVintage Score ,Annual_Premium  and Age are
#   most important Factor For getting Response.
```

In [32]:
```
# 4)SVM Model  only Predict  one kind of Data Correctly which is those doesn't have Response("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.
```

In [ ]:
```

```