Step 1: Load the data set, Check if any missing vaues and split it into train & test sets.

In [1]:
```
import pandas as pd
df = pd.read_csv("Churn-Modelling.csv")
print(df.shape)
print(df.head())
pd.isnull(df).sum()
```

```
(10000, 7)
   CreditScore  Age  Tenure    Balance  HasCrCard     Salary  Exited
0          619   42       2       0.00          1  101348.88       1
1          608   41       1   83807.86          0  112542.58       0
2          502   42       8  159660.80          1  113931.57       1
3          699   39       1       0.00          0   93826.63       0
4          850   43       2  125510.82          1   79084.10       0
```

Out[1]:
```
CreditScore    0
Age            0
Tenure         0
Balance        0
HasCrCard      0
Salary         0
Exited         0
dtype: int64
```

In [2]:
```
x = df.drop("Exited", axis=1)
y = df["Exited"]
print(x.head())
print(y.head())
```

```
   CreditScore  Age  Tenure    Balance  HasCrCard     Salary
0          619   42       2       0.00          1  101348.88
1          608   41       1   83807.86          0  112542.58
2          502   42       8  159660.80          1  113931.57
3          699   39       1       0.00          0   93826.63
4          850   43       2  125510.82          1   79084.10
0    1
1    0
2    1
3    0
4    0
Name: Exited, dtype: int64
```

In [3]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.33)
print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

```
(6700, 6) (6700,)
(3300, 6) (3300,)
```

Step 2: Build the model

sklearn.linear_model.LogisticRegression class can be used to cleate model object

fit() method can be used to train the model

predict() method can be used to predict the outputs for unseen data

In [4]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(x_train, y_train)
```

Out[4]:
```
LogisticRegression()
```

Step 3: Test the model performance

we can import various metrics(confusion_matrix, accuracy_score, recall_score, precision_score, f1_score) from sklearn.metrics

In [10]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, pr
y_pred = model.predict(x_test)
print("Confuusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("F1-Score:", f1_score(y_test, y_pred))
```

```
Confuusion Matrix:
[[2587   59]
 [ 616   38]]
Accuracy: 0.7954545454545454
Recall: 0.0581039755351682
Precision: 0.3917525773195876
F1-Score: 0.10119840213049268
```

In [8]:
```
cm = confusion_matrix(y_test, y_pred)
tn = cm[0, 0]
tp = cm[1, 1]
fp = cm[0, 1]
fn = cm[1, 0]

print("Accuracy: ", (tn + tp)/(tn + tp + fp + fn))
print("Sensitivity: ", tp / (tp + fn))
print("Specificity: ", tn / (tn + fp))
print("Precision: ", tp / (tp + fp))
```

```
Accuracy:  0.7954545454545454
Sensitivity:  0.0581039755351682
Specificity:  0.9777021919879063
Precision:  0.3917525773195876
```