In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
data = pd.read_csv(r'C:\Users\Shrushti\Desktop\Social_Network_Ads.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [4]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   User ID          400 non-null    int64 
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64 
 3   EstimatedSalary  400 non-null    int64 
 4   Purchased        400 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [5]:

```
data.isnull().sum()
```

Out[5]:

```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [7]:

```
X = data.iloc[:, [2, 3]].values
y = data.iloc[:, 4].values
```

In [8]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state =
```

In [9]:

```
election import train_test_split
nodel import LogisticRegression
essing import StandardScaler
 import confusion_matrix,ConfusionMatrixDisplay,classification_report,accuracy_score, precis
```

In [11]:

```
scale = StandardScaler()
X_train = scale.fit_transform(X_train)
X_test = scale.transform(X_test)
```

In [14]:

```
lr = LogisticRegression(random_state = 0,solver = 'lbfgs')
lr.fit(X_train,y_train)
pred = lr.predict(X_test)
```

In [15]:

```
print('Expected Output:',pred[:10])
print('-'*15)
print('Predicted Output:\n',y_test[:10])
```

```
Expected Output: [0 0 0 0 0 0 0 1 0 1]
---------------
Predicted Output:
 [0 0 0 0 0 0 0 1 0 0]
```

In [16]:

```
matrix = confusion_matrix(y_test,pred,labels = lr.classes_)
print(matrix)

tp, fn, fp, tn = confusion_matrix(y_test,pred,labels=[1,0]).reshape(-1)
```

```
[[65  3]
 [ 8 24]]
```

In [17]:

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.96      0.92        68
           1       0.89      0.75      0.81        32

    accuracy                           0.89       100
   macro avg       0.89      0.85      0.87       100
weighted avg       0.89      0.89      0.89       100
```

In [18]:

```
print('\nAccuracy: {:.2f}'.format(accuracy_score(y_test,pred)))
print('Error Rate: ',(fp+fn)/(tp+tn+fn+fp))
print('Sensitivity (Recall or True positive rate) :',tp/(tp+fn))
print('Specificity (True negative rate) :',tn/(fp+tn))
print('Precision (Positive predictive value) :',tp/(tp+fp))
print('False Positive Rate :',fp/(tn+fp))
```

```

Accuracy: 0.89
Error Rate:  0.11
Sensitivity (Recall or True positive rate) : 0.75
Specificity (True negative rate) : 0.9558823529411765
Precision (Positive predictive value) : 0.8888888888888888
False Positive Rate : 0.04411764705882353
```

In [ ]:

```

```