```
import pandas as pd
```

```
df=pd.read_csv("Social_Network_Ads.csv")
df
```

	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
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

```
# x=df.drop(columns='Purchased')
x=df.drop('Purchased',axis=1)
y=df['Purchased']
print("\n",x)
print("\n",y)
x=x.drop(columns='Gender')
print("\n",x)
```

```
        User ID  Gender  Age  EstimatedSalary
0    15624510    Male   19            19000
1    15810944    Male   35            20000
2    15668575  Female   26            43000
3    15603246  Female   27            57000
4    15804002    Male   19            76000
..        ...     ...  ...              ...
395  15691863  Female   46            41000
396  15706071    Male   51            23000
397  15654296  Female   50            20000
398  15755018    Male   36            33000
399  15594041  Female   49            36000

[400 rows x 4 columns]

 0      0
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
```

```
Name: Purchased, Length: 400, dtype: int64

      User ID  Age  EstimatedSalary
0    15624510   19            19000
1    15810944   35            20000
2    15668575   26            43000
3    15603246   27            57000
4    15804002   19            76000
..        ...  ...              ...
395  15691863   46            41000
396  15706071   51            23000
397  15654296   50            20000
398  15755018   36            33000
399  15594041   49            36000

[400 rows x 3 columns]
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
print(x_train)
print(x_test)
print(y_train)
print(y_test)
```

```
      User ID  Age  EstimatedSalary
39   15782806   27            31000
167  15614827   35            71000
383  15707634   49            28000
221  15663161   35            91000
351  15591279   37            75000
..        ...  ...              ...
255  15750056   52            90000
72   15595228   20            23000
396  15706071   51            23000
235  15646227   46            79000
37   15689425   30            49000

[280 rows x 3 columns]
      User ID  Age  EstimatedSalary
398  15755018   36            33000
125  15697020   39            61000
328  15796351   36           118000
339  15665760   39           122000
172  15794661   26           118000
..        ...  ...              ...
91   15636428   30           116000
322  15674331   41            52000
248  15730688   41            52000
186  15724402   20            82000
395  15691863   46            41000

[120 rows x 3 columns]
39     0
167    0
383    1
221    1
351    0
      ..
255    1
72     0
396    1
235    1
37     0
Name: Purchased, Length: 280, dtype: int64
```

```
     398    0
     125    0
     328    1
     339    1
     172    0
           ..
     91     0
     322    0
     248    0
     186    0
     395    1
     Name: Purchased, Length: 120, dtype: int64
```

```
from sklearn.linear_model import LogisticRegression
```

```
model1=LogisticRegression()
model1.fit(x_train,y_train)
Pred_y=model1.predict(x_test)
print(Pred_y)
```

```
     [0 0 1 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0
      0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0
      0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
      0 0 0 1 1 0 0 0 0]
```

```
from sklearn.metrics import accuracy_score
```

```
acc=accuracy_score(y_test,Pred_y)
print(acc)
```

```
     0.6916666666666667
```

```
from sklearn.preprocessing import StandardScaler
```

```
std=StandardScaler()
x_test=std.fit_transform(x_test)
x_train=std.fit_transform(x_train)
print(x_test)
print(x_train)
```

```
     [[ 1.06520773e+00 -2.75451436e-01 -1.14462515e+00]
      [ 2.16654071e-01  4.66866840e-03 -3.08732842e-01]
      [ 1.66994011e+00 -2.75451436e-01  1.39290506e+00]
      [-2.40702862e-01  4.66866840e-03  1.51231825e+00]
      [ 1.64521416e+00 -1.20918512e+00  1.39290506e+00]
      [ 5.17169471e-01 -8.87046996e-02 -1.89319656e-01]
      [-2.88058544e-02 -1.76942532e+00 -1.05506526e+00]
      [-1.25370604e+00  9.38402348e-01  5.27159462e-01]
      [-1.83701499e-01 -7.42318275e-01 -1.59242460e+00]
      [-6.37400748e-01  8.45028980e-01  2.07953089e+00]
      [-5.54737066e-01 -4.62198171e-01  1.96534203e-02]
      [-7.65580661e-01  4.66866840e-03  4.95067168e-02]
      [-1.17131632e-01 -3.68824803e-01  1.96534203e-02]
      [-1.56497503e+00  8.45028980e-01  1.78099792e+00]
      [-1.03908772e+00  1.31189582e+00  3.18186386e-01]
      [ 4.58548120e-02  1.59201592e+00  1.84070451e+00]
      [ 1.35381371e+00  1.96550940e+00  3.48039683e-01]
      [ 6.81223959e-01 -1.11581175e+00 -3.98292732e-01]
      [-2.18683599e-01 -1.02243838e+00  4.67452869e-01]
      [-1.29477453e+00  1.96550940e+00  9.15252318e-01]
      [ 6.17580240e-01  9.80420364e-02  1.09213310e-01]
      [ 1.43908166e+00  1.03177572e+00  4.97306166e-01]
```

```
 [ 4.88815097e-01  4.71535508e-01  2.01982429e+00]
 [-8.73950552e-01  7.51655612e-01 -8.46092181e-01]
 [-9.47045730e-01  5.64908876e-01 -1.35359822e+00]
 [ 1.86704544e+00 -1.20918512e+00 -1.68198449e+00]
 [-1.57777693e+00  1.77876266e+00 -7.26678995e-01]
 [ 3.80167222e-01  9.38402348e-01  7.93600134e-02]
 [ 1.41359491e+00  1.31189582e+00 -1.11477185e+00]
 [ 1.64382424e+00  1.21852245e+00  1.27349188e+00]
 [-2.02297170e-01  4.66866840e-03 -8.75945477e-01]
 [ 1.78187990e+00 -1.86279869e+00  1.39066607e-01]
 [-1.51389912e+00 -1.95617206e+00  4.37599573e-01]
 [ 6.62116213e-01  1.68538929e+00  7.93600134e-02]
 [-1.23143805e+00 -1.11581175e+00  3.77892979e-01]
 [-1.26255764e+00 -8.35691643e-01  2.58479793e-01]
 [-5.75381041e-01 -1.58267859e+00 -1.59242460e+00]
 [-1.42893841e+00 -6.48944907e-01  4.37599573e-01]
 [-4.08429666e-01  1.03177572e+00 -1.53271800e+00]
 [ 8.61489307e-01 -1.86279869e+00 -1.38345152e+00]
 [ 9.36237758e-01  7.51655612e-01  2.16909077e+00]
 [-3.12305704e-01  1.77876266e+00  8.85399021e-01]
 [-1.62810668e+00 -4.62198171e-01  1.30334517e+00]
 [-1.27399888e+00 -1.48930522e+00 -1.59466359e-01]
 [ 1.36246048e+00  1.59201592e+00 -3.38586139e-01]
 [-2.67345440e-01 -7.42318275e-01  1.39290506e+00]
 [-2.00307389e-01  8.45028980e-01 -1.08491855e+00]
 [ 7.57303808e-01  7.51655612e-01  1.24363858e+00]
 [ 9.49317640e-01  4.66866840e-03  2.28626496e-01]
 [-1.01241588e+00  1.21852245e+00 -9.95358664e-01]
 [ 5.24645780e-01 -1.39593185e+00 -3.98292732e-01]
 [-8.78442190e-01 -1.82078068e-01 -5.47559215e-01]
 [ 5.67601632e-01  2.84788772e-01  2.58479793e-01]
 [ 3.27306358e-01  6.58282244e-01 -1.29389163e+00]
 [-1.26890336e-01  2.84788772e-01  4.95067168e-02]
 [-7.48243234e-01 -1.82078068e-01 -2.78879546e-01]
 [ 1.59710829e+00  1.96550940e+00 -8.75945477e-01]
 [-1.32816188e+00 -2.75451436e-01 -5.77412512e-01]
```

```
model2=LogisticRegression()
model2.fit(x_train,y_train)
Pred_y=model2.predict(x_test)
print(Pred_y)
```

```
[0 0 1 1 0 0 0 1 0 1 0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0
 0 0 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0
 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0
 0 0 0 0 0 0 0 0 0]
```

```
acc2=accuracy_score(y_test,Pred_y)
print(acc2)
```

```
0.8
```

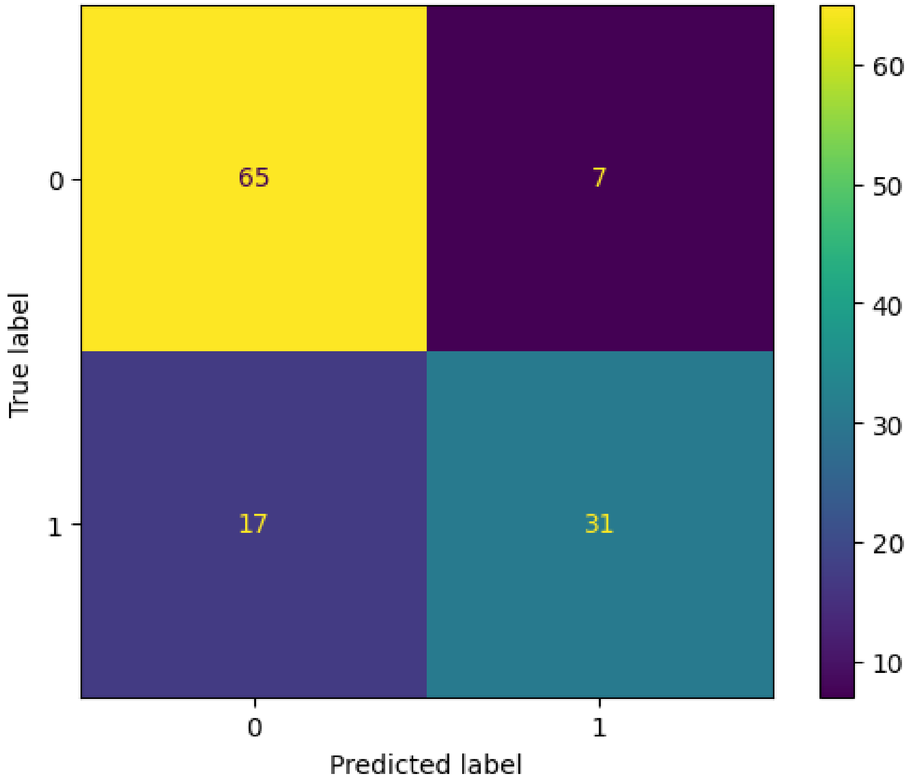
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
# in order to evaluate model when algorithm is classification based
```

```
cm=confusion_matrix(y_test,Pred_y)
cm
```

```
array([[65,  7],
       [17, 31]], dtype=int64)
```

```
ConfusionMatrixDisplay(cm).plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x125323e3250>
```



```
from sklearn.metrics import recall_score,precision_score
```

```
print('recall = ',recall_score(y_test,Pred_y))
```

```
recall =  0.6458333333333334
```