```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
```

```
df=pd.read_csv('/content/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

```
df['Gender'].replace({"Male":1, "Female":0}, inplace=True)
```

```
<ipython-input-9-261c738c1a19>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df['Gender'].replace({"Male":1, "Female":0}, inplace=True)
<ipython-input-9-261c738c1a19>:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version
  df['Gender'].replace({"Male":1, "Female":0}, inplace=True)
```

```
df.columns
```

```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

```
x = df[['User ID', 'Gender', 'Age', 'EstimatedSalary']]
y=df['Purchased']
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.25, random_state=22)
```

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

```
▾ LogisticRegression
LogisticRegression()
```

```
y_pred= model.predict(x_test)
y_pred
```

```
array([1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0])
```

```
model.score(x_train, y_train)
```

```
0.8433333333333334
```

```
model.score(x, y)
```

```
0.855
```

```
print(confusion_matrix.__doc__)
```

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

```
array([[60,  5],
       [ 6, 29]])
```

```
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
```

```
print(tn, fp, fn, tp)
```

```
60 5 6 29
```

```
a = accuracy_score(y_test, y_pred)
p = precision_score(y_test, y_pred)
r = recall_score(y_test, y_pred)
```

```
print(a, p, r)
```

```
0.89 0.8529411764705882 0.8285714285714286
```