```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

df=pd.read_csv(r"C:\Users\ASUS\Documents\pythonStack\DS_PR\
Social_Network_Ads.csv")

df.head()
```

```
    User ID  Gender   Age  EstimatedSalary  Purchased
0  15624510    Male  19.0          19000.0          0
1  15810944    Male  35.0          20000.0          0
2  15668575  Female  26.0          43000.0          0
3  15603246  Female  27.0          57000.0          0
4  15804002    Male  19.0          76000.0          0
```

```python
xtrain=df[['Age','EstimatedSalary']]
ytrain=df['Purchased']

xtrain,xtest,ytrain,ytest=train_test_split(xtrain,ytrain,test_size=0.2
)

xtrain
```

```
      Age  EstimatedSalary
30   31.0          74000.0
300  58.0          38000.0
88   26.0          81000.0
282  37.0          70000.0
181  31.0          71000.0
..    ...              ...
64   59.0          83000.0
230  35.0         147000.0
49   31.0          89000.0
227  56.0         133000.0
183  33.0          43000.0

[304 rows x 2 columns]
```

```python
model=LogisticRegression()

model.fit(xtrain,ytrain)
```

```
LogisticRegression()
```

```python
yprediction=model.predict(xtest)

ytest
```

```
244    0
377    0
```

```
166    0
110    0
94     0
      ..
233    1
170    0
123    0
386    1
138    0
Name: Purchased, Length: 76, dtype: int64
```

```
CM=confusion_matrix(ytest,yprediction)
```

```
CM
```

```
array([[48,  4],
       [ 1, 23]], dtype=int64)
```

```
# Confusion Matrix:
#                 Predicted: 0    Predicted: 1
# Actual: 0      48 (TN)         4 (FP)
# Actual: 1      1 (FN)          23 (TP)
#
# TN = True Negative  → predicted 0, actually 0
# FP = False Positive → predicted 1, actually 0
# FN = False Negative → predicted 0, actually 1
# TP = True Positive  → predicted 1, actually 1
```

```
tn, fp, fn, tp = CM.ravel()
#unpacks the values from the confusion matrix into individual
variables
#.ravel() flattens a 2D array into a 1D list[0,0,0,0]
```

```
tn,fp,fn,tp
```

```
(48, 4, 1, 23)
```

```
#Purpose: Measures how often the model predicted correctly overall.
accuracy = (tp + tn) / (tp + tn + fp + fn)
```

```
#Because 1 means 100% of predictions, subtracting accuracy gives
incorrect percentage.
#Shows how often the model made wrong predictions.
error_rate = 1 - accuracy
```

```
accuracy.round(),error_rate.round()
```

```
(1.0, 0.0)
```

```
Psion= tp / (tp + fp)
```

```
#Purpose: Of all predicted 1s, how many are actually 1?
```

```
#Numerator: tp → Correctly predicted positives.
#Denominator: tp + fp
#fp: False Positives (predicted 1, but actually 0)
```

```
Psion
```

```
0.8518518518518519
```

```
recall = tp / (tp + fn)
# Recall tells how many actual positives the model correctly
predicted.
# Formula: Recall = TP / (TP + FN)
# Used when missing positive cases is risky (e.g., disease detection,
fraud).
# High recall means fewer false negatives (model didn’t miss many real
positives).
# Example: If 10 people actually bought and model found 8 → Recall =
8/10 = 0.8
#80 percentr real buyer found by model
```

```
recall
```

```
0.9583333333333334
```