Name :sandip gadadare. Div:A
Roll no: CO3AO9. BATCH:B1

-B2

Aim:- 1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

In [1]:
```
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
```
In [2]:
```
df=pd.read_csv('Social_Network_Ads.csv')
df
```
Out[2]:

	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

In [3]:
```
df
df["Gender"].replace({'Male':0,'Female':1},inplace=True)
```
In [4]:
```
df
```
Out[4]:

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

400 rows × 5 columns

In [5]:
```
x=df[['User ID','Gender','Age','EstimatedSalary']]
```
In [6]:
```
y=df['Purchased']
```
In [7]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=14)
```
In [8]:

```
x_train,x_test,y_train,y_test
Out[8]:
(     User ID  Gender  Age  EstimatedSalary
379  15749381       1   58            23000
369  15624755       1   54            26000
368  15779744       0   38            71000
185  15814816       0   31            66000
203  15809347       1   41            71000
..        ...     ...  ...              ...
249  15753102       1   35            97000
327  15785170       1   42            75000
268  15746203       1   47           144000
344  15729908       0   47           105000
107  15789863       0   27            89000

[300 rows x 4 columns],
User ID  Gender  Age  EstimatedSalary
85   15663939       1   31           118000
20   15649487       0   45            22000
1    15810944       0   35            20000
346  15646936       0   53            72000
69   15595324       1   31            68000
..        ...     ...  ...              ...
245  15722061       1   51           146000
343  15629739       1   47            51000
349  15721835       0   38            61000
291  15596522       0   49            89000
254  15724161       1   50            44000

[100 rows x 4 columns],
379    1
369    1
368    0
185    0
203    0
..
249    1
327    0
268    1
344    1
107    0
Name: Purchased, Length: 300, dtype: int64,
85     1
20     1
1      0
346    1
69     0
..
245    1
343    1
349    0
291    1
254    0
Name: Purchased, Length: 100, dtype: int64)
In [9]:
model=LogisticRegression()
In [10]:
model.fit(x_train,y_train)
Out[10]:
LogisticRegression()
In [11]:
```

df

Out[11]:

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

400 rows × 5 columns

In [12]:

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

In [13]:

```
y_predict
```

Out[13]:

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1,
0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0], dtype=int64)
```

In [14]:

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

Out[14]:

0.77

In [15]:

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

In [16]:

```
cm
```

Out[16]:

```
array([[55,  4],
[20, 21]], dtype=int64)
```

In [17]:

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

```
Compute confusion matrix to evaluate the accuracy of a classification.

By definition a confusion matrix :math:`C` is such that :math:`C_{i, j}`
is equal to the number of observations known to be in group :math:`i` and
predicted to be in group :math:`j`.

Thus in binary classification, the count of true negatives is
:math:`C_{0,0}`, false negatives is :math:`C_{1,0}`, true positives is
:math:`C_{1,1}` and false positives is :math:`C_{0,1}`.

Read more in the :ref:`User Guide <confusion_matrix>`.

Parameters
----------
y_true : array-like of shape (n_samples,)
Ground truth (correct) target values.

y_pred : array-like of shape (n_samples,)
Estimated targets as returned by a classifier.
```

labels : array-like of shape (n_classes), default=None
List of labels to index the matrix. This may be used to reorder
or select a subset of labels.
If ``None`` is given, those that appear at least once
in ``y_true`` or ``y_pred`` are used in sorted order.

sample_weight : array-like of shape (n_samples,), default=None
Sample weights.

.. versionadded:: 0.18

normalize : {'true', 'pred', 'all'}, default=None
Normalizes confusion matrix over the true (rows), predicted (columns)
conditions or all the population. If None, confusion matrix will not be
normalized.

Returns

C : ndarray of shape (n_classes, n_classes)
Confusion matrix whose i-th row and j-th
column entry indicates the number of
samples with true label being i-th class
and predicted label being j-th class.

See Also

ConfusionMatrixDisplay.from_estimator : Plot the confusion matrix
given an estimator, the data, and the label.
ConfusionMatrixDisplay.from_predictions : Plot the confusion matrix
given the true and predicted labels.
ConfusionMatrixDisplay : Confusion Matrix visualization.

References

.. [1] `Wikipedia entry for the Confusion matrix
<https://en.wikipedia.org/wiki/Confusion_matrix>`_
(Wikipedia and other references may use a different
convention for axes).

Examples

>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
[0, 0, 1],
[1, 0, 2]])

>>> y_true = ["cat", "ant", "cat", "cat", "ant", "bird"]
>>> y_pred = ["ant", "ant", "cat", "cat", "ant", "cat"]
>>> confusion_matrix(y_true, y_pred, labels=["ant", "bird", "cat"])
array([[2, 0, 0],
[0, 0, 1],
[1, 0, 2]])

In the binary case, we can extract true positives, etc as follows:

>>> tn, fp, fn, tp = confusion_matrix([0, 1, 0, 1], [1, 1, 1, 0]).ravel()
>>> (tn, fp, fn, tp)

(0, 2, 1, 1)

In [18]:
```
tn,fp,fn,tp=confusion_matrix(y_test,y_predict).ravel()
```
In [20]:
```
print("Tp:",tn)
print("Fp:",fp)
print("TN:",tn)
print("Fp:",fn)
```
Tp: 55

Fp: 4

TN: 55

Fp: 20

In [21]:
```
a=accuracy_score(y_test,y_predict)
```
In [22]:
```
print("Accuracy Value:",a)
```
Accuracy Value: 0.76

In [24]:
```
e=1-a
```
In [25]:
```
print("Error Rate:",e)
```
Error Rate: 0.24

In [26]:
```
precision=precision_score(y_test,y_predict)
print("precision:",precision)
```
precision: 0.84

In [27]:
```
Recall=recall_score(y_test,y_predict)
print("Recall value:",Recall)
```
Recall value: 0.5121951219512195

In [ ]: