```
#DSBDL Assignment 5
Data Analytics II
1. Implement logistic regression using Python/R to perform classificat
Social_Network_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error ra
Precision, Recall on the given dataset.
```

DSBDL Assignment 5

Data Analytics II

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.

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

```
df = pd.read_csv('/content/drive/MyDrive/DBSDAL Assignments/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.head()
```

	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

```
df.tail()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
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

```
df.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
```

```
df.describe(include='all')
```

	User ID	Gender	Age	EstimatedSalary	Purchased
count	4.000000e+02	400	400.000000	400.000000	400.000000
unique	NaN	2	NaN	NaN	NaN
top	NaN	Female	NaN	NaN	NaN
freq	NaN	204	NaN	NaN	NaN
mean	1.569154e+07	NaN	37.655000	69742.500000	0.357500
std	7.165832e+04	NaN	10.482877	34096.960282	0.479864
min	1.556669e+07	NaN	18.000000	15000.000000	0.000000
25%	1.562676e+07	NaN	29.750000	43000.000000	0.000000
50%	1.569434e+07	NaN	37.000000	70000.000000	0.000000
75%	1.575036e+07	NaN	46.000000	88000.000000	1.000000
max	1.581524e+07	NaN	60.000000	150000.000000	1.000000

```
df.shape
```

```
(400, 5)
```

```
df.columns
```

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

```
df.isnull()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
395	False	False	False	False	False
396	False	False	False	False	False
397	False	False	False	False	False
398	False	False	False	False	False
399	False	False	False	False	False

400 rows × 5 columns

```
df.isna()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
395	False	False	False	False	False
396	False	False	False	False	False
397	False	False	False	False	False
398	False	False	False	False	False
399	False	False	False	False	False

400 rows × 5 columns

```
df.isnull().any()
```

```
User ID            False
Gender             False
Age                False
EstimatedSalary    False
Purchased          False
dtype: bool
```

converting categorical data into numerical data

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

MALE = 0, FEMALE = 1

```
df
```

	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

Model Building

```
X = df.iloc[:,[0,1,2]]
y = df.iloc[:,[3]]
```

X

	Gender	Age	EstimatedSalary
0	0	19	19000
1	0	35	20000
2	1	26	43000
3	1	27	57000
4	0	19	76000
...	...	...	...
395	1	46	41000
396	0	51	23000
397	1	50	20000
398	0	36	33000
399	1	49	36000

400 rows × 3 columns

y

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

400 rows × 1 columns

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

```
sc = StandardScaler()
X = sc.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)
```

```
X_train
```

```
array([[ 0.98019606, -1.01769239, -1.13766796],
       [ 0.98019606, -0.25358736,  0.03692631],
       [-1.02020406,  1.08359645, -1.22576253],
       [-1.02020406, -0.25358736,  0.62422344],
       [-1.02020406, -0.0625611 ,  0.15438573],
       [ 0.98019606, -0.54012675,  1.38770971],
       [ 0.98019606, -0.15807423,  0.15438573],
       [-1.02020406, -1.6862843 ,  0.47739916],
       [ 0.98019606,  0.03295203, -0.57973568],
       [ 0.98019606,  1.08359645, -0.99084367],
       [-1.02020406,  0.22397829, -0.13926283],
       [-1.02020406, -0.0625611 ,  0.21311545],
       [ 0.98019606,  2.13424088, -0.69719511],
       [ 0.98019606,  0.98808332,  2.0043717 ],
```

```
[-1.02020406,  0.22397829,  0.03692631],
[ 0.98019606, -0.15807423, -0.19799255],
[-1.02020406, -1.11320552,  0.33057487],
[-1.02020406, -0.44461362, -1.13766796],
[ 0.98019606,  0.31949142, -0.55037082],
[ 0.98019606,  0.79705706,  0.12502088],
[-1.02020406, -1.87731056,  0.35993973],
[ 0.98019606, -0.0625611 , -1.07893824],
[-1.02020406, -0.25358736, -0.49164111],
[ 0.98019606, -0.92217926, -0.31545197],
[ 0.98019606, -0.0625611 ,  0.03692631],
[ 0.98019606, -1.87731056, -0.75592482],
[-1.02020406,  0.03295203, -0.25672226],
[ 0.98019606, -0.25358736, -0.13926283],
[-1.02020406, -0.63563988, -0.10989798],
[-1.02020406,  0.89257019, -1.3138571 ],
[-1.02020406,  0.41500455,  1.00596657],
[-1.02020406, -1.78179743, -1.49004624],
[-1.02020406, -1.59077117,  0.06629116],
[-1.02020406, -0.92217926, -1.1083031 ],
[-1.02020406,  1.37013584,  2.35674998],
[ 0.98019606,  1.46564897,  1.00596657],
[-1.02020406, -1.20871865,  0.30121002],
[ 0.98019606, -1.11320552,  0.06629116],
[ 0.98019606, -1.11320552, -1.60750566],
[ 0.98019606,  0.22397829,  2.12183112],
[-1.02020406, -0.25358736, -0.93211396],
[ 0.98019606,  1.84770149,  1.53453399],
[-1.02020406,  0.31949142,  0.50676401],
[ 0.98019606, -1.30423178,  0.56549373],
[-1.02020406, -0.0625611 , -0.52100597],
[-1.02020406, -0.25358736,  1.123426  ],
[-1.02020406, -0.54012675,  0.88850715],
[ 0.98019606, -0.25358736,  2.26865541],
[ 0.98019606, -1.78179743, -1.28449224],
[ 0.98019606,  0.79705706,  0.35993973],
[-1.02020406,  0.89257019,  1.03533143],
[ 0.98019606,  0.70154394, -0.72655996],
[-1.02020406,  0.03295203, -0.55037082],
[ 0.98019606, -0.0625611 ,  0.68295315],
[-1.02020406, -0.82666613, -0.78528968],
[ 0.98019606,  0.31949142, -1.16703281],
[ 0.98019606,  0.22397829, -0.28608712],
[-1.02020406, -1.01769239,  0.53612887]
```

X_test

```
array([[-1.02020406, -0.15807423, -1.07893824],
       [ 0.98019606,  0.12846516, -0.25672226],
       [-1.02020406, -0.15807423,  1.41707457],
       [-1.02020406,  0.12846516,  1.53453399],
       [ 0.98019606, -1.11320552,  1.41707457],
       [ 0.98019606,  0.03295203, -0.13926283],
       [ 0.98019606, -1.6862843 , -0.99084367],
       [-1.02020406,  1.08359645,  0.56549373],
       [-1.02020406, -0.63563988, -1.51941109],
       [-1.02020406,  0.98808332,  2.09246627],
       [ 0.98019606, -0.34910049,  0.06629116],
       [ 0.98019606,  0.12846516,  0.09565602],
       [-1.02020406, -0.25358736,  0.06629116],
       [ 0.98019606,  0.98808332,  1.7988177 ],
       [ 0.98019606,  1.46564897,  0.35993973],
       [-1.02020406,  1.75218836,  1.85754742],
       [-1.02020406,  2.13424088,  0.38930459],
       [-1.02020406, -1.01769239, -0.34481683],
       [ 0.98019606, -0.92217926,  0.50676401],
       [-1.02020406,  2.13424088,  0.94723686],
       [ 0.98019606,  0.22397829,  0.15438573],
       [ 0.98019606,  1.17910958,  0.53612887],
       [ 0.98019606,  0.60603081,  2.03373655],
       [-1.02020406,  0.89257019, -0.78528968],
       [-1.02020406,  0.70154394, -1.28449224],
       [-1.02020406, -1.11320552, -1.60750566],
       [ 0.98019606,  1.94321462, -0.66783025],
       [-1.02020406,  1.08359645,  0.12502088],
       [ 0.98019606,  1.46564897, -1.04957339],
       [ 0.98019606,  1.37013584,  1.29961514],
       [-1.02020406,  0.12846516, -0.81465453],
       [-1.02020406, -1.78179743,  0.18375059],
       [ 0.98019606, -1.87731056,  0.47739916],
       [-1.02020406,  1.84770149,  0.12502088],
       [ 0.98019606, -1.01769239,  0.41866944],
       [-1.02020406, -0.73115301,  0.30121002],
       [-1.02020406, -1.49525804, -1.51941109],
```

```
[ 0.98019606, -0.54012675,  0.47739916],
[ 0.98019606,  1.17910958, -1.46068138],
[-1.02020406, -1.78179743, -1.3138571 ],
[ 0.98019606,  0.89257019,  2.18056084],
[ 0.98019606,  1.94321462,  0.917872  ],
[-1.02020406, -0.34910049,  1.32898   ],
[ 0.98019606, -1.39974491, -0.10989798],
[-1.02020406,  1.75218836, -0.28608712],
[ 0.98019606, -0.63563988,  1.41707457],
[ 0.98019606,  0.98808332, -1.02020853],
[ 0.98019606,  0.89257019,  1.27025028],
[ 0.98019606,  0.12846516,  0.27184516],
[ 0.98019606,  1.37013584, -0.93211396],
[-1.02020406, -1.30423178, -0.34481683],
[-1.02020406, -0.0625611 , -0.49164111],
[ 0.98019606,  0.41500455,  0.30121002],
[-1.02020406,  0.79705706, -1.22576253],
[-1.02020406,  0.41500455,  0.09565602],
[ 0.98019606, -0.0625611 , -0.2273574 ],
[-1.02020406,  2.13424088, -0.81465453],
```

```
classifier = LogisticRegression()
classifier.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversior
  y = column_or_1d(y, warn=True)
```

LogisticRegression

LogisticRegression()

```
y_test = y_test['Purchased'].to_numpy()
```

```
y_pred = classifier.predict(X_test)
result = pd.DataFrame({'Actual' : y_test, 'Predicted' : y_pred})
result
```

	Actual	Predicted
0	0	0
1	0	0
2	1	1
3	1	1
4	0	0
...	...	...
115	0	0
116	0	0
117	0	0
118	0	0
119	1	0

120 rows × 2 columns

```
y_pred
```

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

Evaluation Parameters

```
print('Accuracy of model')
print(accuracy_score(y_test,y_pred) * 100, '%')
```

```
Accuracy of model
80.83333333333333 %
```

```
cf_matrix = confusion_matrix(y_test,y_pred)
print('Confusion Matrix \n', cf_matrix)
```

```
Confusion Matrix
 [[63  9]
 [14 34]]
```

```
from sklearn.metrics import precision_score
precision = precision_score(y_test,y_pred)
print("Precision: ", precision)
```

```
Precision:  0.7906976744186046
```

```
from sklearn.metrics import recall_score
recall = recall_score(y_test,y_pred)
print("recall: ", recall)
```

```
recall:  0.7083333333333334
```

```
from sklearn.metrics import classification_report
class_report = classification_report(y_test, y_pred)

print(class_report)
```

```
                   precision    recall  f1-score   support

               0       0.82      0.88      0.85        72
               1       0.79      0.71      0.75        48

        accuracy                           0.81       120
       macro avg       0.80      0.79      0.80       120
    weighted avg       0.81      0.81      0.81       120
```