```
                    Name : Suryal D . Khirade
                    Roll NO: T190424399
                    Assignment No : 05
                        Data Analytics II
                            1. Implement logistic regression using Pyt
hon/R to perform classification on
                            Social_Network_Ads.csv dataset.
                            2. Compute Confusion matrix to find TP, F
P, TN, FN, Accuracy, Error rate, Precision, Recall
                            on the given dataset.
```

STEP 1

INCLUDE NECESSORY LIBRARIES

In [3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

STEP 2

upload dataset

In [5]:
```
data=pd.read_csv("C:\\Users\\alisu\\Downloads\\Social_Network_Ads.csv")
```

In [6]:
```
data
```

Out[6]:

	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

STEP 3

EXPLORATORY DATA ANALYSIS

In [8]:
```
data.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
```

In [9]:
```
data.describe()
```

Out[9]:

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

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [11]:
```
data.duplicated()
```

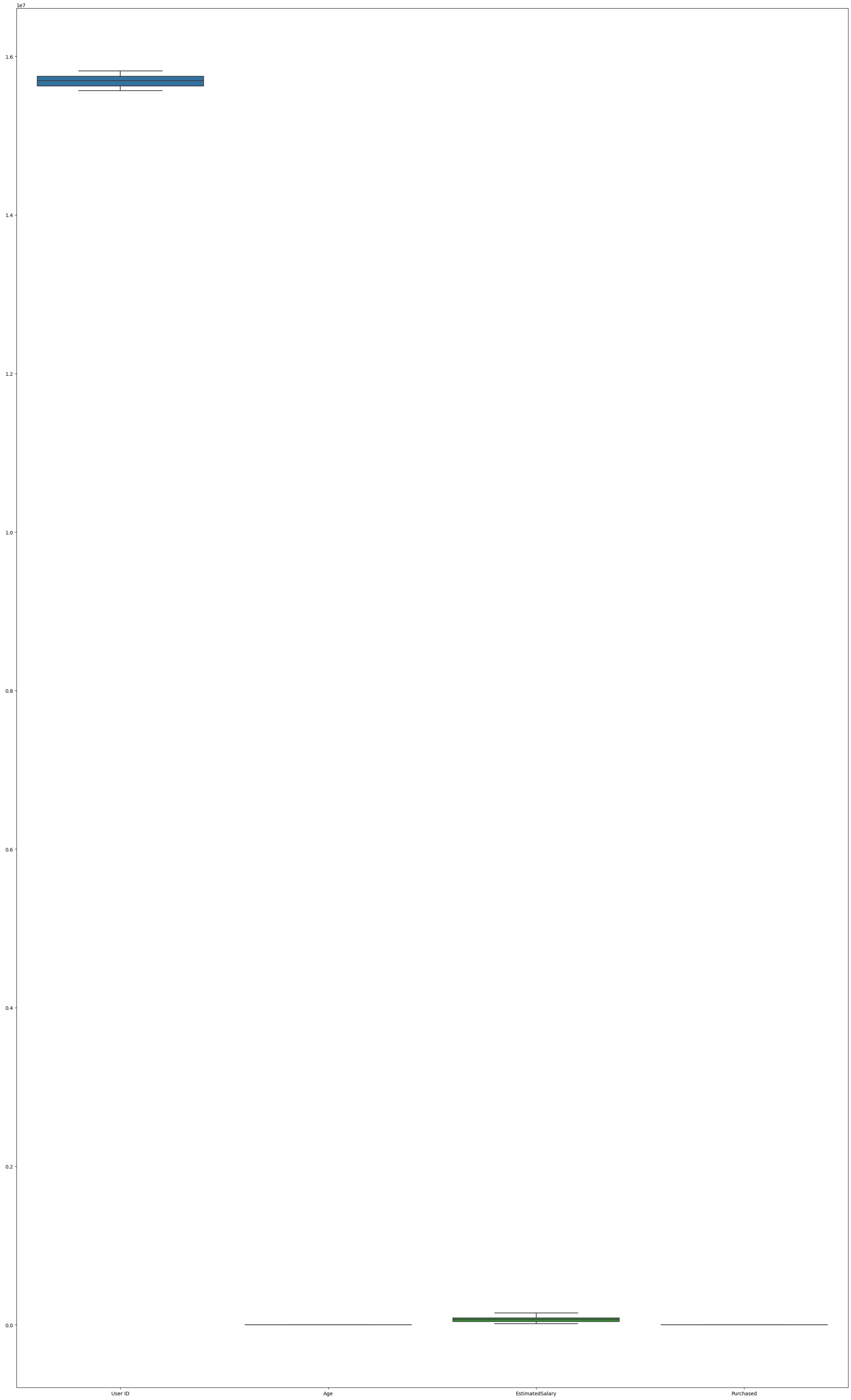
Out[11]:
```
0      False
1      False
2      False
3      False
4      False
       ...
395    False
396    False
397    False
398    False
399    False
Length: 400, dtype: bool
```

STEP 4

To find outliers

In [13]:
```
plt.figure(figsize=(30,50))
sns.boxplot(data)
```

Out[13]: `<Axes: >`


1e7
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
User ID
Age
EstimatedSalary
Purchased

STEP 5

Conversion of categorical variable to numerical variable

In [14]:
```
data.Gender=data.Gender.replace({"Male":1,"Female":0})
```

In [15]:
```
data
```

Out[15]:

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

400 rows × 5 columns

Step 6

Splitting dependent and independent variables

In [16]:
```
x=data.drop( "Purchased",axis="columns")
```

In [17]:
```
x
```

Out[17]:

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

400 rows × 4 columns

In [18]:
```
y=data.Purchased
```

In [19]:
```
y
```

Out[19]:
```
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
```

STEP 7

Split the dataset

In [20]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=
```

In [21]:
```
xtrain
```

Out[21]:

	User ID	Gender	Age	EstimatedSalary
239	15772073	0	53	143000
188	15674206	1	35	72000
240	15701537	1	42	149000
23	15599081	0	45	22000
343	15629739	0	47	51000
...	...	...	...	...
256	15609637	0	41	72000
131	15801247	1	33	31000
249	15753102	0	35	97000
152	15699247	1	31	76000
362	15768072	0	47	50000

320 rows × 4 columns

In [22]:
```
ytrain
```

Out[22]:
```
239    1
188    0
240    1
23     1
343    1
      ..
256    0
131    0
249    1
152    0
362    1
Name: Purchased, Length: 320, dtype: int64
```

In [23]:
```
xtest
```

Out[23]:

	User ID	Gender	Age	EstimatedSalary
376	15596984	0	46	74000
16	15733883	1	47	25000
365	15807525	0	59	29000
82	15709476	1	20	49000
107	15789863	1	27	89000
...	...	...	...	...
246	15638003	0	35	50000
10	15570769	0	26	80000
115	15689237	1	40	57000
74	15592877	1	32	18000
194	15689751	1	28	89000

80 rows × 4 columns

In [24]:
```
ytest
```

Out[24]:
```
376    0
16     1
365    1
82     0
107    0
      ..
246    0
10     0
115    0
74     0
194    0
Name: Purchased, Length: 80, dtype: int64
```

STEP 8

Use of logistic regression

In [25]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [26]:
```
model
```

Out[26]:
```
▾ LogisticRegression
LogisticRegression()
```

In [27]:
```
model.fit(xtrain,ytrain)
```

Out[27]:
```
▾ LogisticRegression
LogisticRegression()
```

In [28]:
```
y_predict=model.predict(xtest)
```

In [29]:
```
y_predict
```

Out[29]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [30]:
```
compare_result= pd.DataFrame(y_predict,ytest)
```

In [31]:
```
compare_result.head(50)
```

Out[31]:

	0
Purchased	
0	0
1	0
1	0
0	0
0	0
0	0
1	0
0	0
0	0
0	0
0	0
0	0
1	0
0	0
0	0
0	0
0	0
0	0
1	1
0	0
0	0
0	0
0	0
0	0
0	1
0	0
0	0
1	1
0	0
1	0
1	1
0	0
1	1
0	1
0	0
0	0
1	1

	0
Purchased	
0	0
1	0
0	0
1	1
1	1
1	1
0	0
1	0
0	0
0	0
1	1
0	0
0	0

In [46]:
```
from sklearn.metrics import confusion_matrix , classification_report ,preci
from sklearn.metrics import accuracy_score ,roc_curve , erroer_rate
```

In [38]:
```
accuracy=accuracy_score(ytest,y_predict)
```

In [39]:
```
accuracy
```

Out[39]:
```
0.8
```

In [40]:
```
confusion_matrix(ytest,y_predict)
```

Out[40]:
```
array([[52,  3],
       [13, 12]], dtype=int64)
```

In [42]:
```
precision_score(ytest,y_predict)
```

Out[42]:
```
0.8
```

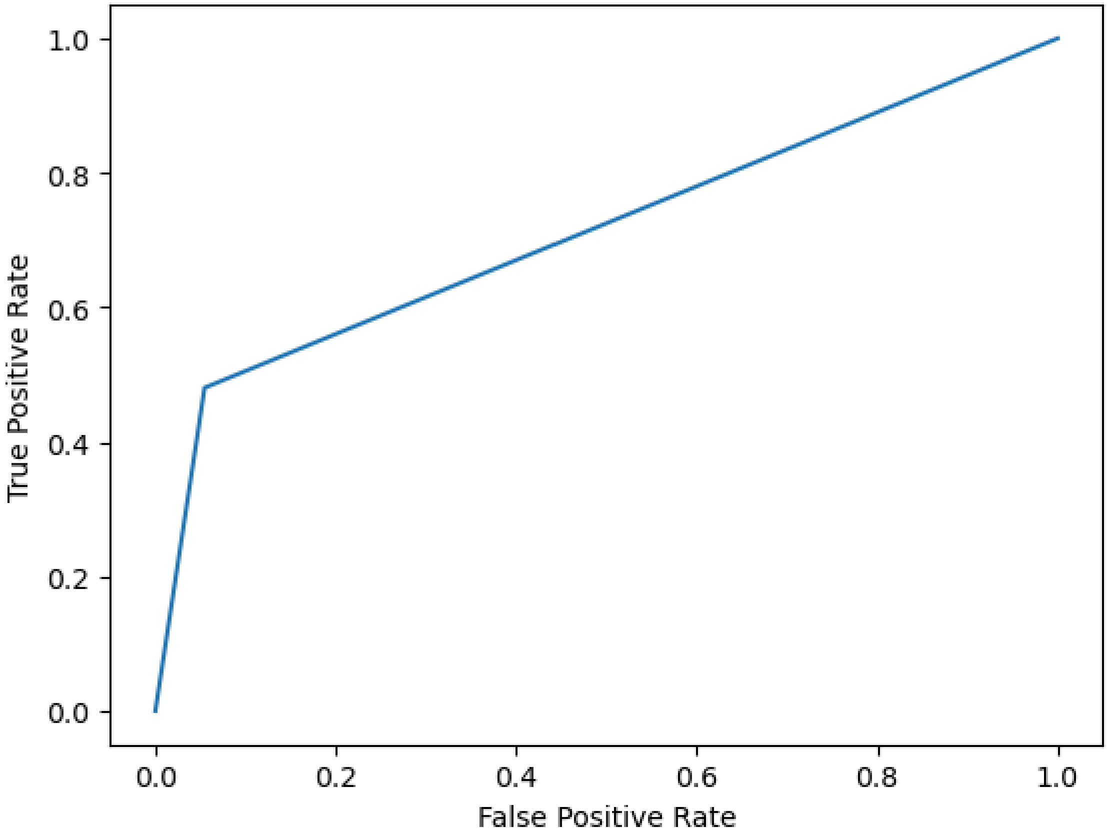
In [43]:
```
f1_score(ytest,y_predict)
```

Out[43]:
```
0.6
```

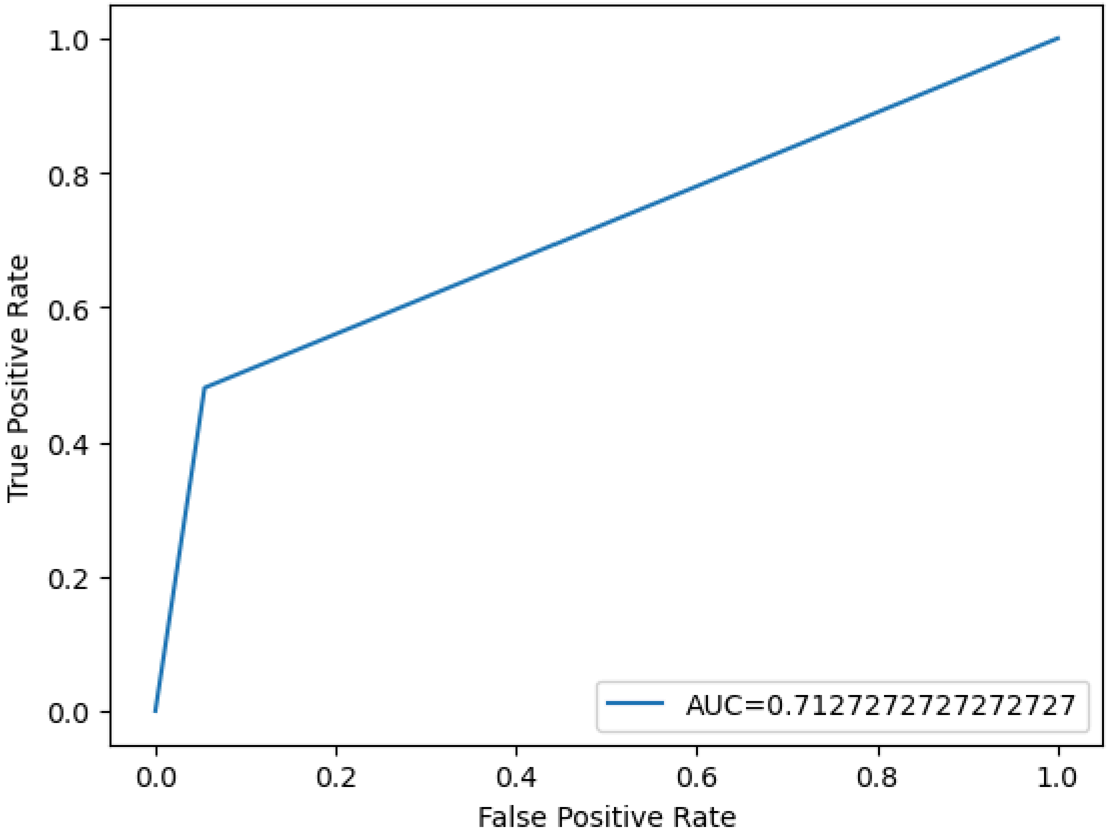
In [44]:
```
roc_auc_score(ytest,y_predict)
```

Out[44]:
```
0.7127272727272727
```

In [47]:
```
fpr, tpr, _ = roc_curve(ytest, y_predict)
#create ROC curve
plt.plot(fpr,tpr)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [48]:
```
auc= roc_auc_score(ytest,y_predict)
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
```



In [52]:
```
cr= classification_report(ytest,y_predict)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.80      0.95      0.87        55
           1       0.80      0.48      0.60        25

    accuracy                           0.80        80
   macro avg       0.80      0.71      0.73        80
weighted avg       0.80      0.80      0.78        80
```

In [55]:
```
y_predict= model.predict(xtest)
print("Testing accuracy:", model.score(xtest,ytest)*100)
```

```
Testing accuracy: 80.0
```