Assignment No-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 libraries and create alias for Pandas, Numpy

[1]:
```
import pandas as pd
import numpy as np
```

Import the Social_Media_Adv Dataset

[2]:
```
from google.colab import files
```

[ ]:
```
files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Social_Network_Ads.csv to Social_Network_Ads (1).csv
```

[ ]:
```
{'Social_Network_Ads (1).csv': b'Age,EstimatedSalary,Purchased\r\n19,19000,0\r\n35,20000,0\r\n26,43000,0\r\n27,57000,0\r\n19,76000,0\r\n27,58000,0\r\n27,84000,0\r\n32,150000,1\r\n25,33000,0\r\n35,65000,0\r\n26,80000,0\r\n26,52000,0\r\n20,86000,0\r\n32,18000,0\r\n18,82000,0\r\n29,80000,0\r\n47,25000,1\r\n45,26000,1\r\n46,28000,1\r\n48,29000,1\r\n45,22000,1\r\n47,49000,1\r\n48,41000,1\r\n45,22000,1\r\n46,23000,1\r\n47,20000,1\r\n49,28000,1\r\n47,30000,1\r\n29,43000,0\r\n31,18000,0\r\n31,74000,0\r\n27,137000,1\r\n21,16000,0\r\n28,44000,0\r\n27,90000,0\r\n35,27000,0\r\n33,28000,0\r\n30,49000,0\r\n26,72000,0\r\n27,31000,0\r\n27,17000,0\r\n33,51000,0\r\n35,108000,0\r\n30,15000,0\r\n28,84000,0\r\n23,20000,0\r\n25,79000,0\r\n27,54000,0\r\n30,135000,1\r\n31,89000,0\r\n24,32000,0\r\n18,44000,0\r\n29,83000,0\r\n35,23000,0\r\n27,58000,0\r\n24,55000,0\r\n23,48000,0\r\n28,79000,0\r\n22,18000,0\r\n32,117000,0\r\n27,20000,0\r\n25,87000,0\r\n23,66000,0\r\n32,120000,1\r\n59,83000,0\r\n24,58000,0\r\n24,19000,0\r\n23,82000,0\r\n22,63000,0\r\n31,68000,0\r\n25,80000,0\r\n24,27000,0\r\n20,23000,0\r\n33,113000,0\r\n32,18000,0\r\n34,112000,1\r\n18,52000,0\r\n22,27000,0\r\n28,87000,0\r\n26,17000,0\r\n30,80000,0\r\n39,42000,0\r\n20,49000,0\r\n35,88000,0\r\n30,62000,0\r\n31,118000,1\r\n24,55000,0\r\n28,85000,0\r\n26,81000,0\r\n35,50000,0\r\n22,81000,0\r\n30,116000,0
```



```
\r\n26,15000,0\r\n29,28000,0\r\n29,83000,0\r\n35,44000,0\r\n35,25000,0\r\n28,123000,1\r\n35,73000,0\r\n28,37000,0\r\n27,88000,0\r\n28,59000,0\r\n32,86000,0\r\n33,149000,1\r\n19,21000,0\r\n21,72000,0\r\n26,35000,0\r\n27,89000,0\r\n26,86000,0\r\n38,80000,0\r\n39,71000,0\r\n37,71000,0\r\n38,61000,0\r\n37,55000,0\r\n42,80000,0\r\n40,57000,0\r\n35,75000,0\r\n36,52000,0\r\n40,59000,0\r\n41,59000,0\r\n36,75000,0\r\n37,72000,0\r\n40,75000,0\r\n35,53000,0\r\n41,51000,0\r\n39,61000,0\r\n42,65000,0\r\n26,32000,0\r\n30,17000,0\r\n26,84000,0\r\n31,58000,0\r\n33,31000,0\r\n30,87000,0\r\n21,68000,0\r\n28,55000,0\r\n23,63000,0\r\n20,82000,0\r\n30,107000,1\r\n28,59000,0\r\n19,25000,0\r\n19,85000,0\r\n18,68000,0\r\n35,59000,0\r\n30,89000,0\r\n34,25000,0\r\n24,89000,0\r\n27,96000,1\r\n41,30000,0\r\n29,61000,0\r\n20,74000,0\r\n26,15000,0\r\n41,45000,0\r\n31,76000,0\r\n36,50000,0\r\n40,47000,0\r\n31,15000,0\r\n46,59000,0\r\n29,75000,0\r\n26,30000,0\r\n32,135000,1\r\n32,100000,1\r\n25,90000,0\r\n37,33000,0\r\n35,38000,0\r\n33,69000,0\r\n18,86000,0\r\n22,55000,0\r\n35,71000,0\r\n29,148000,1\r\n29,47000,0\r\n21,88000,0\r\n34,115000,0\r\n26,118000,0\r\n34,43000,0\r\n34,72000,0\r\n23,28000,0\r\n35,47000,0\r\n25,22000,0\r\n24,23000,0\r\n31,34000,0\r\n26,16000,0\r\n31,71000,0\r\n32,117000,1\r\n33,43000,0\r\n33,60000,0\r\n31,66000,0\r\n20,82000,0\r\n33,41000,0\r\n35,72000,0\r\n28,32000,0\r\n24,84000,0\r\n19,26000,0\r\n29,43000,0\r\n19,70000,0\r\n28,89000,0\r\n34,43000,0\r\n30,79000,0\r\n20,36000,0\r\n26,80000,0\r\n35,22000,0\r\n35,39000,0\r\n49,74000,0\r\n39,134000,1\r\n41,71000,0\r\n58,101000,1\r\n47,47000,0\r\n55,130000,1\r\n52,114000,0\r\n40,142000,1\r\n46,22000,0\r\n48,96000,1\r\n52,150000,1\r\n59,42000,0\r\n35,58000,0\r\n47,43000,0\r\n60,108000,1\r\n49,65000,0\r\n40,78000,0\r\n46,96000,0\r\n59,143000,1\r\n41,80000,0\r\n35,91000,1\r\n37,144000,1\r\n60,102000,1\r\n35,60000,0\r\n37,53000,0\r\n36,126000,1\r\n56,133000,1\r\n40,72000,0\r\n42,80000,1\r\n35,147000,1\r\n39,42000,0\r\n40,107000,1\r\n49,86000,1\r\n38,112000,0\r\n46,79000,1\r\n40,57000,0\r\n37,80000,0\r\n46,82000,0\r\n53,143000,1\r\n42,149000,1\r\n38,59000,0\r\n50,88000,1\r\n56,104000,1\r\n41,72000,0\r\n51,146000,1\r\n35,50000,0\r\n57,122000,1\r\n41,52000,0\r\n35,97000,1\r\n44,39000,0\r\n37,52000,0\r\n48,134000,1\r\n37,146000,1\r\n50,44000,0\r\n52,90000,1\r\n41,72000,0\r\n40,57000,0\r\n58,95000,1\r\n45,131000,1\r\n35,77000,0\r\n36,144000,1\r\n55,125000,1\r\n35,72000,0\r\n48,90000,1\r\n42,108000,1\r\n40,75000,0\r\n37,74000,0\r\n47,144000,1\r\n40,61000,0\r\n43,133000,0\r\n59,76000,1\r\n60,42000,1\r\n39,106000,1\r\n57,26000,1\r\n57,74000,1\r\n38,71000,0\r\n49,88000,1\r\n52,38000,1\r\n50,36000,1\r\n59,88000,1\r\n35,61000,0\r\n37,70000,1\r\n52,21000,1\r\n48,141000,0\r\n37,93000,1\r\n37,62000,0\r\n48,138000,1\r\n41,79000,0\r\n37,78000,1\r\n39,134000,1\r\n49,89000,1\r\n55,39000,1\r\n37,77000,0\r\n35,57000,0\r\n36,63000,0\r\n42,73000,1\r\n43,112000,1\r\n45,79000,0\r\n46,117000,1\r\n58,38000,1\r\n48,74000,1\r\n37,137000,1\r\n37,79000,1\r\n40,60000,0\r\n42,54000,0\r\n51,134000,0\r\n47,113000,1\r\n36,125000,1\r\n38,50000,0\r\n42,70000,0\r\n39,96000,1\r\n38,50000,0\r\n49,141000,1\r\n39,79000,0\r\n39,75000,1\r\n54,104000,1\r\n35,55000,0\r\n45,32000,1\r\n36,60000,0\r\n52,138000,1\r\n53,82000,1\r\n41,52000,0\r\n48,30000,1\r\n48,131000,1\r\n41,60000,0\r\n41,72000,0\r\n42,75000,0\r\n36,118000,1\r\n47,107000,1\r\n38,51000,0\r\n48,119000,1\r\n42,65000,0\r\n40,65000,0\r\n57,60000,1\r\n36,54000,0\r\n58,144000,1\r\n35,79000,0\r\n38,55000,0\r\n39,122000,1\r\n53,104000,1\r\n35,75000,0\r\n38,65000,0\r\n47,51000,1\r\n47,105000,1\r\n41,63000,0\r\n53,72000,1\r\n54,108000,1\r\n39,77000,0\r\n38,61000,0\r\n38,113000,1\r\n37,75000,0\r\n42,90000,1\r\n37,57000,0\r\n36,99000,1\r\n60,34000,1\r\n
```

```
54,70000,1\r\n41,72000,0\r\n40,71000,1\r\n42,54000,0\r\n43,129000,1\r\n53,34000,
1\r\n47,50000,1\r\n42,79000,0\r\n42,104000,1\r\n59,29000,1\r\n58,47000,1\r\n46,8
8000,1\r\n38,71000,0\r\n54,26000,1\r\n60,46000,1\r\n60,83000,1\r\n39,73000,0\r\n
59,130000,1\r\n37,80000,0\r\n46,32000,1\r\n46,74000,0\r\n42,53000,0\r\n41,87000,
1\r\n58,23000,1\r\n42,64000,0\r\n48,33000,1\r\n44,139000,1\r\n49,28000,1\r\n57,3
3000,1\r\n56,60000,1\r\n49,39000,1\r\n39,71000,0\r\n47,34000,1\r\n48,35000,1\r\n
48,33000,1\r\n47,23000,1\r\n45,45000,1\r\n60,42000,1\r\n39,59000,0\r\n46,41000,1
\r\n51,23000,1\r\n50,20000,1\r\n36,33000,0\r\n49,36000,1'}
```

Initialize the data frame

[4]:
```
df=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Social_Network_Ads.csv")
```

Perform Data Preprocessing

[9]:
```
df.head()
```

[9]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

[ ]:
```
df.tail()
```

[ ]:

	Age	EstimatedSalary	Purchased
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

[10]:
```
df
```

[10]:

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

[400 rows x 3 columns]

[11]:
```
df.describe()
```

[11]:

	Age	EstimatedSalary	Purchased
count	400.000000	400.000000	400.000000
mean	37.655000	69742.500000	0.357500
std	10.482877	34096.960282	0.479864
min	18.000000	15000.000000	0.000000
25%	29.750000	43000.000000	0.000000
50%	37.000000	70000.000000	0.000000
75%	46.000000	88000.000000	1.000000
max	60.000000	150000.000000	1.000000

Identification and Handling of Null Values

[6]:
```
df.isnull()
```

[6]:

	Age	EstimatedSalary	Purchased
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
..	…	…	…
395	False	False	False
396	False	False	False
397	False	False	False
398	False	False	False
399	False	False	False

[400 rows x 3 columns]

[7]:
```
df.isna()
```

[7]:

	Age	EstimatedSalary	Purchased
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
..	…	…	…
395	False	False	False
396	False	False	False
397	False	False	False
398	False	False	False
399	False	False	False

```
[400 rows x 3 columns]
```

[8]:
```
df.isnull().sum()
```

[8]:
```
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

[12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              400 non-null    int64
 1   EstimatedSalary  400 non-null    int64
 2   Purchased        400 non-null    int64
dtypes: int64(3)
memory usage: 9.5 KB
```

[13]:
```
df.dtypes
```

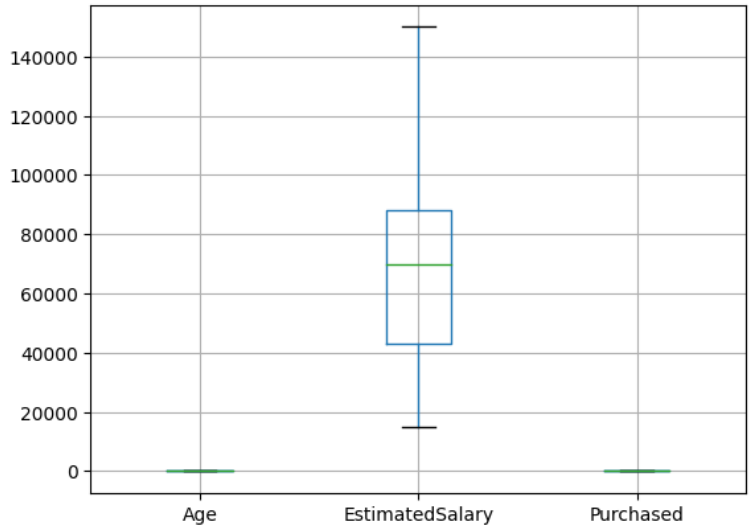
[13]:
```
Age                int64
EstimatedSalary    int64
Purchased          int64
dtype: object
```

Import Seaborn and Matplotlib

[14]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[17]:
```
df.boxplot()
```

[17]:
```
<Axes: >
```

[19]:
```
X = df.drop(['Purchased'], axis = 1)
Y = df['Purchased']
```

Use Logistic regression(Train the Machine) to Create Model:

[20]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(X, Y, test_size =0.
↪2,random_state = 0)
```

[21]:
```
ytrain
```

[21]:
```
336    1
64     0
55     0
106    0
300    1
      ..
323    1
192    0
117    0
47     0
172    0
```

```
Name: Purchased, Length: 320, dtype: int64
```

[22]:
```python
xtrain
```

[22]:
```
     Age  EstimatedSalary
336   58           144000
64    59            83000
55    24            55000
106   26            35000
300   58            38000
..    …               …
323   48            30000
192   29            43000
117   36            52000
47    27            54000
172   26           118000

[320 rows x 2 columns]
```

[23]:
```python
import sklearn
from sklearn.linear_model import LogisticRegression
```

[24]:
```python
logreg = LogisticRegression()
```

[25]:
```python
model=logreg.fit(xtrain, ytrain)
```

[26]:
```python
LogisticRegression()
```

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

Predict the y_pred

[27]:
```python
y_pred=logreg.predict(xtest)
```

[28]:
```python
y_pred
```

[28]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

[29]:
```python
y_pred_xtrain=logreg.predict(xtrain)
```

[30]:
```python
y_pred_xtrain
```

[30]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```



```
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

[31]:
```python
print(xtrain)
print("------------\n")
print(xtest)
print("------------\n")
print(ytrain)
print("------------\n")
print(ytest)
print("------------\n")
print(y_pred)
print("------------\n")
print(y_pred_xtrain)
```

```
     Age  EstimatedSalary
336   58           144000
64    59            83000
55    24            55000
106   26            35000
300   58            38000
..    …               …
323   48            30000
192   29            43000
117   36            52000
47    27            54000
172   26           118000

[320 rows x 2 columns]
------------

     Age  EstimatedSalary
132   30            87000
309   38            50000
341   35            75000
196   30            79000
```

```
246   35            50000
..    …             …
14    18            82000
363   42            79000
304   40            60000
361   53            34000
329   47           107000

[80 rows x 2 columns]
------------

336    1
64     0
55     0
106    0
300    1
      ..
323    1
192    0
117    0
47     0
172    0
Name: Purchased, Length: 320, dtype: int64
------------

132    0
309    0
341    0
196    0
246    0
      ..
14     0
363    0
304    0
361    1
329    1
Name: Purchased, Length: 80, dtype: int64
------------

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0]
------------

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```



```
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

2 Evaluation parameters

[32]:
```python
from sklearn.metrics import precision_score,confusion_matrix,accuracy_score,recall_score,classification_report
```

1.Confusion Matrix

[33]:
```python
cm= confusion_matrix(ytest, y_pred)
```

[34]:
```python
cm
```

[34]:
```
array([[58,  0],
       [22,  0]])
```

2.Accuracy_score

[35]:
```python
print ("Accuracy : ", accuracy_score(ytest, y_pred))
```

```
Accuracy :  0.725
```

3.Precision

[36]:
```python
ps = precision_score(ytest, y_pred)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

4.Recall score

[37]:
```python
rs = recall_score(ytest, y_pred)
```

[38]:
```python
rs
```

[38]:
```
0.0
```

5.Error Rate

[39]:
```python
error_rate = 1- accuracy_score(ytest, y_pred)
error_rate
```

[39]:
```
0.275
```

6.Classification Report

```
[40]: print("classification report: ",classification_report(ytest, y_pred))
```

```
classification report:                precision    recall  f1-score   support

           0       0.72      1.00      0.84        58
           1       0.00      0.00      0.00        22

    accuracy                           0.73        80
   macro avg       0.36      0.50      0.42        80
weighted avg       0.53      0.72      0.61        80
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```