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.

By,

Vinayak Jalan

TE B 74

1. Import Libraries

In [21]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

2. Import Dataset

In [22]:

```
df = pd.read_csv("/content/Social_Network_Ads.csv")
df.head(10)
```

Out[22]:

	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
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0

In [23]:

```
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
```

In [24]:

```
df.describe()
```

Out[24]:

	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 [25]:

```
X = df.iloc[:,[2,3]].values
y = df.iloc[:,4].values
```

In [ ]:

```
X
```

In [27]:

```
y
```

Out[27]:

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

3. Split the dataset into train and test

In [28]:

```
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size = 0.25,random_stat
e=0)
```

4. Preprocessing

Standard Scalar

In [29]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [ ]:

```
X_train
```

In [31]:

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train,y_train)
```

Out[31]:

```
LogisticRegression(random_state=0)
```

6. Prediction

In [32]:

```
y_pred = classifier.predict(X_test)
```

In [33]:

```
y_pred
```

Out[33]:

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

Confusion Matrix

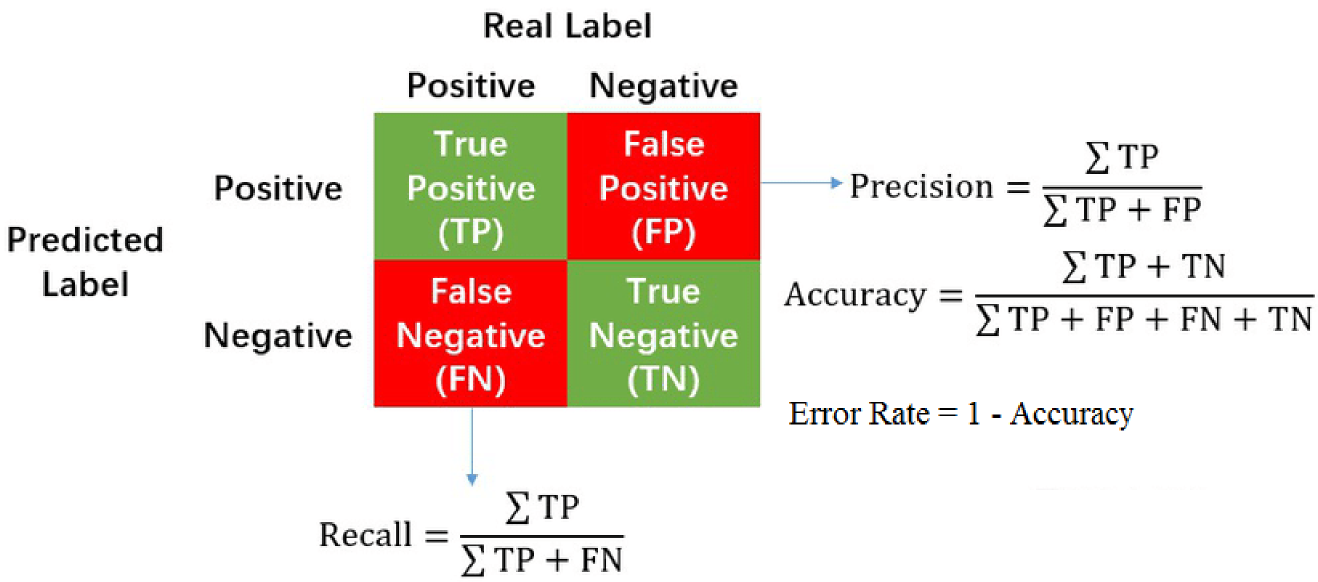
We can deduce from the confusion matrix that:

True positive: 65 (upper-left) – Number of positives we predicted correctly

True negative: 24 (lower-right) – Number of negatives we predicted correctly

False positive: 3 (top-right) – Number of positives we predicted wrongly

False negative: 8 (lower-left) – Number of negatives we predicted wrongly



In [34]:

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

In [35]:

```
cm
```

Out[35]:

```
array([[65,  3],
       [ 8, 24]])
```

In [36]:

```
c1_report = classification_report(y_test,y_pred)
```

In [37]:

```
c1_report
```

Out[37]:

```
'              precision    recall  f1-score   support\n\n           0
0.89      0.96      0.92        68\n           1       0.89      0.75
0.81        32\n\n    accuracy                           0.89       100\n
macro avg       0.89      0.85      0.87       100\nweighted avg       0.8
9      0.89      0.89       100\n'
```

In [38]:

```
tp , fn ,fp , tn = confusion_matrix(y_test,y_pred,labels=[0,1]).reshape(-1)
print('Outcome values : \n' , tp , fn , fp ,tn)
```

```
Outcome values :
 65 3 8 24
```

In [39]:

```
accuracy_cm = (tp+tn)/(tp+fp+tn+fn)
precision_cm = tp/(tp+fp)
recall_cm = tp/(tp+fn)
f1_score = 2/((1/recall_cm)+(1/precision_cm))
```

In [40]:

```
print("Accuracy   : ",accuracy_cm)
print("Precision  : ",precision_cm)
print("Recall     : ",recall_cm)
print("F1-Score   : ",f1_score)
```

```
Accuracy   :  0.89
Precision  :  0.8904109589041096
Recall     :  0.9558823529411765
F1-Score   :  0.9219858156028368
```