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

In [2]:
```
X = [9.0,8.0,9.3,9.4,3.3,2.2,3.2,1.1]
Y = [1,1,1,1,0,0,0,0]

dataset = pd.DataFrame(data={
      'X'  : X,
       'Y'  : Y
})
```

In [3]:
```
dataset
```

Out[3]:

	X	Y
0	9.0	1
1	8.0	1
2	9.3	1
3	9.4	1
4	3.3	0
5	2.2	0
6	3.2	0
7	1.1	0

In [4]:
```
dataset['XY'] = dataset['X'] * dataset['Y']
dataset['X2'] = dataset['X'] ** 2
```

In [5]:
```
dataset
```

Out[5]:

	X	Y	XY	X2
0	9.0	1	9.0	81.00
1	8.0	1	8.0	64.00
2	9.3	1	9.3	86.49
3	9.4	1	9.4	88.36
4	3.3	0	0.0	10.89
5	2.2	0	0.0	4.84
6	3.2	0	0.0	10.24
7	1.1	0	0.0	1.21

In [6]:
```
n = len(dataset)
sum_x = dataset['X'].sum()
sum_y = dataset['Y'].sum()
sum_xy = dataset['XY'].sum()
sum_x2 = dataset['X2'].sum()
sum_X_h2 = sum_x ** 2
```

In [7]:
```
n, sum_x , sum_y , sum_xy , sum_x2 , sum_X_h2
```

Out[7]:
```
(8, 45.5, 4, 35.7, 347.03000000000003, 2070.25)
```

In [8]:
```
numerator_m = (n*(sum_xy)) - (sum_x*sum_y)
numerator_m
```

Out[8]:
```
103.60000000000002
```

In [9]:
```
denominator_m = (n*(sum_x2) - sum_X_h2)
denominator_m
```

Out[9]:
```
705.9900000000002
```

In [10]:
```
m = numerator_m / denominator_m
```

In [11]:
```
m
```

Out[11]:
```
0.14674428816272184
```

In [12]:
```
numerator_b = sum_y -(m * sum_x)
denominator_b =n

b = numerator_b/denominator_b
b
```

Out[12]:
```
-0.33460813892548047
```

In [13]:
```
m,b
```

Out[13]:
```
(0.14674428816272184, -0.33460813892548047)
```

In [14]:
```
X_cap = [m*X + b for X in dataset['X']]
X_cap
```

Out[14]:
```
[0.9860904545390161,
 0.8393461663762942,
 1.0301137409878327,
 1.0447881698041048,
 0.14964801201150157,
 -0.011770704967492385,
 0.13497358319522945,
 -0.17318942194648643]
```

In [15]:
```
y = [1 / (1 + np.exp(-Xcap)) for Xcap in X_cap]
```

In [16]:
```
y
```

Out[16]:
```
[0.7283150226716693,
 0.6983274932179059,
 0.7369379463021366,
 0.7397728295238472,
 0.537342340327484,
 0.4970573577331819,
 0.5336922612250837,
 0.4568105447007532]
```

In [17]:
```
 result = []
for yvals in y:
    if yvals >= 0.5:
        result.append(1)
    else:
        result.append(0)
```

In [18]:
```
result
```

Out[18]:
```
[1, 1, 1, 1, 1, 0, 1, 0]
```

In [19]:
```
result =[1 if yvals >=0.5 else 0 for yvals in y]
```

In [20]:
```
result
```

Out[20]:
```
[1, 1, 1, 1, 1, 0, 1, 0]
```

In [21]:
```
dataset['RESULT'] = result
```

In [22]:
```
dataset
```

Out[22]:

	X	Y	XY	X2	RESULT
0	9.0	1	9.0	81.00	1
1	8.0	1	8.0	64.00	1
2	9.3	1	9.3	86.49	1
3	9.4	1	9.4	88.36	1
4	3.3	0	0.0	10.89	1
5	2.2	0	0.0	4.84	0
6	3.2	0	0.0	10.24	1
7	1.1	0	0.0	1.21	0

In [23]:
```
#accuracy
```

In [24]:
```
correct = 0
for key,result in zip(dataset['Y'],dataset['RESULT']):
    if key == result:
        correct += 1
    pass
```

In [25]:
```
correct/n
```

Out[25]:
```
0.75
```

In [26]:
```
#another code in accuracy correct
correct = 0  # Inilizing the correct with 0
values = dataset[['Y','RESULT']].values
for (key,result) in values:
    if key == result:
        correct += 1
    pass
```

In [27]:
```
correct / n
```

Out[27]:
```
0.75
```

In [28]:
```
#another code in accuracy correct
correct = 0  # Inilizing the correct with 0
keys = dataset['Y']
results = dataset['RESULT']
for (key,result) in zip(keys,results):
#    print(f"key {key}, result {result}")
    if key == result:
        print(f" EQUAL key {key}, result {result}")
        correct += 1
        print(f"correct {correct}")
    pass
```

```
 EQUAL key 1, result 1
correct 1
 EQUAL key 1, result 1
correct 2
 EQUAL key 1, result 1
correct 3
 EQUAL key 1, result 1
correct 4
 EQUAL key 0, result 0
correct 5
 EQUAL key 0, result 0
correct 6
```

In [29]:
```
correct/n
```

Out[29]:
```
0.75
```

In [30]:
```
dataset
```

Out[30]:

	X	Y	XY	X2	RESULT
0	9.0	1	9.0	81.00	1
1	8.0	1	8.0	64.00	1
2	9.3	1	9.3	86.49	1
3	9.4	1	9.4	88.36	1
4	3.3	0	0.0	10.89	1
5	2.2	0	0.0	4.84	0
6	3.2	0	0.0	10.24	1
7	1.1	0	0.0	1.21	0

In [31]:
```
dataset_metrics = dataset[['Y','RESULT']]
```

In [32]:
```
from sklearn.metrics import accuracy_score,classification_report,confusion_matri
```

In [34]:
```
accuracy_score(dataset_metrics['Y'],dataset_metrics['RESULT'])
```

Out[34]:
```
0.75
```

In [39]:
```
print(classification_report(dataset_metrics['Y'],dataset_metrics['RESULT']))
```

```
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         4
           1       0.67      1.00      0.80         4

    accuracy                           0.75         8
   macro avg       0.83      0.75      0.73         8
weighted avg       0.83      0.75      0.73         8
```

In [40]:
```
confusion_matrix(dataset_metrics['Y'],dataset_metrics['RESULT'])
```

Out[40]:
```
array([[2, 2],
       [0, 4]], dtype=int64)
```

In [ ]: