In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
```

In [4]:
```
dataset={'X':[9.0,8.0,9.3,9.4,3.2,2.2,3.2,1.1],
         'Y':[1,1,1,1,0,0,0,0]}
dataset
```

Out[4]:
```
{'X': [9.0, 8.0, 9.3, 9.4, 3.2, 2.2, 3.2, 1.1], 'Y': [1, 1, 1, 1, 0, 0, 0, 0]}
```

In [5]:
```
df=pd.DataFrame(dataset)
df
```

Out[5]:

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

In [6]:
```
df['XY'] = df['X'] * df['Y']
df['X2'] = df['X'] ** 2
df
```

Out[6]:

	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	3.2	0	0.0	10.24
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 [7]:
```
sum_x = df['X'].sum()
sum_y = df['Y'].sum()
sum_xy = df['XY'].sum()
sum_x2 = df['X2'].sum()
sum_X_h2 = sum_x ** 2
```

```
n=len(df)
n, sum_x, sum_y, sum_xy,sum_x2, sum_X_h2
```

Out[7]:
```
(8, 45.400000000000006, 4, 35.7, 346.38000000000005, 2061.1600000000003)
```

In [8]:
```
numerator_m = (n*(sum_xy)) - (sum_x*sum_y)
denominator_m = (n*(sum_x2)) - (sum_X_h2)
m = numerator_m / denominator_m
m
```

Out[8]:
```
0.14650363441708456
```

In [9]:
```
numerator_b = sum_y - (m * sum_x)
denominator_b = n
b = numerator_b/denominator_b
b
```

Out[9]:
```
-0.33140812531695496
```

In [10]:
```
X_cap = [m * X + b for X in df['X']]
X_cap
```

Out[10]:
```
[0.987124584436806,
 0.8406209500197215,
 1.0310756747619314,
 1.04572603820364,
 0.13740350481771563,
 -0.0091001295993689,
 0.13740350481771563,
 -0.17025412745816193]
```

In [11]:
```
def sigmoid(X_cap):
    return [(1/(1+np.exp(-X_cap))) for X_cap in X_cap]
sigmoid(X_cap)
```

Out[11]:
```
[0.7285195999613322,
 0.6985959791346004,
 0.737124384670443,
 0.7399533370174842,
 0.5342969334585324,
 0.4977249833000944,
 0.5342969334585324,
 0.45753898485684547]
```

In [12]:
```
y_pred=sigmoid(X_cap)
y_pred
```

Out[12]:
```
[0.7285195999613322,
 0.6985959791346004,
 0.737124384670443,
 0.7399533370174842,
 0.5342969334585324,
 0.4977249833000944,
 0.5342969334585324,
 0.45753898485684547]
```

In [13]:
```
def final(y_pred):
    result=[1 if y>=0.5 else 0 for y in y_pred]
```

```
    return result
final(y_pred)
```

Out[13]:
```
[1, 1, 1, 1, 1, 0, 1, 0]
```

In [17]:
```
y_pred=final(y_pred)
y_pred
```

Out[17]:
```
[1, 1, 1, 1, 1, 0, 1, 0]
```

In [18]:
```
y_true=df.Y.values
y_true
```

Out[18]:
```
array([1, 1, 1, 1, 0, 0, 0, 0], dtype=int64)
```

In [22]:
```
def accuracy(y_pred,y_true):
        correct=0
        for y_p,y_t in zip(y_pred,y_true):
            if y_p==y_t:
                correct +=1
        return correct/len(y_pred)
accuracy(y_pred,y_true)
```

Out[22]:
```
0.75
```

In [ ]:
```

```