concept of classification

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [5]:
```
x  =np.array([0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0])
```

In [6]:
```
y = np.round(x)
```

In [7]:
```
y
```

Out[7]:
```
array([0., 0., 0., 0., 0., 1., 1., 1., 1., 1.])
```

In [8]:
```
# and that is defined the concepts of classification
```

Sigmoid curve

Sigmoid Function acts as an activation function in machine learning which is used to add non-linearity in a machine learning model, in simple words it decides which value to pass as output and what not to pass, there are mainly 7 types of Activation Functions which are used in machine learning and deep learning.

In [13]:
```
data= pd.DataFrame({'Actual':x, 'pred':y})
```

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

Out[15]:

	Actual	pred
0	0.1	0.0
1	0.2	0.0
2	0.3	0.0
3	0.4	0.0
4	0.5	0.0
5	0.6	1.0
6	0.7	1.0
7	0.8	1.0
8	0.9	1.0
9	1.0	1.0

In [28]:
```
outcome = 1/(1+np.exp(-data))
```

In [29]:
```
outcome
```

Out[29]:

	Actual	pred
0	0.524979	0.500000
1	0.549834	0.500000
2	0.574443	0.500000
3	0.598688	0.500000
4	0.622459	0.500000
5	0.645656	0.731059
6	0.668188	0.731059
7	0.689974	0.731059
8	0.710950	0.731059
9	0.731059	0.731059

In [30]:
```
data1 = np.linspace(-8,8,1000)
```

In [31]:
```
data1
```

Out[31]:
```
array([-8.        , -7.98398398, -7.96796797, -7.95195195, -7.93593594,
       -7.91991992, -7.9039039 , -7.88788789, -7.87187187, -7.85585586,
       -7.83983984, -7.82382382, -7.80780781, -7.79179179, -7.77577578,
       -7.75975976, -7.74374374, -7.72772773, -7.71171171, -7.6956957 ,
       -7.67967968, -7.66366366, -7.64764765, -7.63163163, -7.61561562,
       -7.5995996 , -7.58358358, -7.56756757, -7.55155155, -7.53553554,
       -7.51951952, -7.5035035 , -7.48748749, -7.47147147, -7.45545546,
       -7.43943944, -7.42342342, -7.40740741, -7.39139139, -7.37537538,
       -7.35935936, -7.34334334, -7.32732733, -7.31131131, -7.2952953 ,
       -7.27927928, -7.26326326, -7.24724725, -7.23123123, -7.21521522,
       -7.1991992 , -7.18318318, -7.16716717, -7.15115115, -7.13513514,
       -7.11911912, -7.1031031 , -7.08708709, -7.07107107, -7.05505506,
       -7.03903904, -7.02302302, -7.00700701, -6.99099099, -6.97497497,
       -6.95895896, -6.94294294, -6.92692693, -6.91091091, -6.89489489,
       -6.87887888, -6.86286286, -6.84684685, -6.83083083, -6.81481481,
       -6.7987988 , -6.78278278, -6.76676677, -6.75075075, -6.73473473,
       -6.71871872, -6.7027027 , -6.68668669, -6.67067067, -6.65465465,
       -6.63863864, -6.62262262, -6.60660661, -6.59059059, -6.57457457,
       -6.55855856, -6.54254254, -6.52652653, -6.51051051, -6.49449449,
```

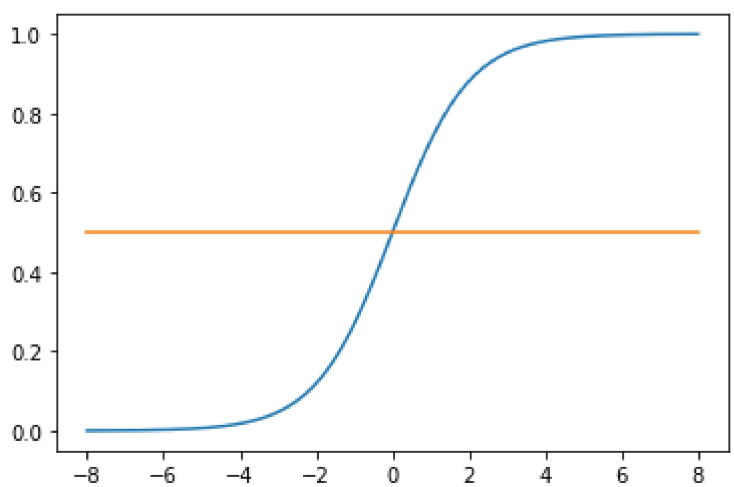
In [35]:
```
outcome =  1/(1+np.exp(-data1))
```

In [36]:
```
outcome
```

Out[36]:
```
array([3.35350130e-04, 3.40762500e-04, 3.46262192e-04, 3.51850614e-04,
       3.57529197e-04, 3.63299395e-04, 3.69162683e-04, 3.75120564e-04,
       3.81174562e-04, 3.87326227e-04, 3.93577132e-04, 3.99928878e-04,
       4.06383089e-04, 4.12941419e-04, 4.19605544e-04, 4.26377170e-04,
       4.33258030e-04, 4.40249884e-04, 4.47354521e-04, 4.54573758e-04,
       4.61909443e-04, 4.69363453e-04, 4.76937693e-04, 4.84634101e-04,
       4.92454646e-04, 5.00401328e-04, 5.08476179e-04, 5.16681265e-04,
       5.25018684e-04, 5.33490567e-04, 5.42099082e-04, 5.50846429e-04,
       5.59734844e-04, 5.68766601e-04, 5.77944008e-04, 5.87269411e-04,
       5.96745194e-04, 6.06373779e-04, 6.16157627e-04, 6.26099239e-04,
       6.36201155e-04, 6.46465957e-04, 6.56896268e-04, 6.67494753e-04,
       6.78264119e-04, 6.89207119e-04, 7.00326548e-04, 7.11625245e-04,
       7.23106099e-04, 7.34772039e-04, 7.46626047e-04, 7.58671150e-04,
       7.70910422e-04, 7.83346990e-04, 7.95984029e-04, 8.08824765e-04,
       8.21872476e-04, 8.35130493e-04, 8.48602199e-04, 8.62291034e-04,
       8.76200490e-04, 8.90334117e-04, 9.04695521e-04, 9.19288366e-04,
       9.34116376e-04, 9.49183333e-04, 9.64493079e-04, 9.80049520e-04,
       9.95856623e-04, 1.01191842e-03, 1.02823900e-03, 1.04482253e-03,
       1.06167324e-03, 1.07879542e-03, 1.09619344e-03, 1.11387173e-03,
```

In [71]:
```
plt.plot(data1,outcome)
plt.plot([-8,8],[0.5,0.5
                ])
```

Out[71]:
```
[<matplotlib.lines.Line2D at 0x2a5ad128430>]
```



Confusion matrix

confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data forwhich the true (actual) values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can beconfusing

##Let's now define the most basic terms, which are whole numbers (not rates):

true positives

(TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.true negatives (TN): We predicted no, and they don't have the disease.false positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")false negatives (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

In [ ]:

```

```

In [ ]:

```

```

In [39]:

```
actual= [1,0,1,1,0,0,0,0,1,1]
pred= [0,1,1,0,0,1,0,1,0,1]
```

In [42]:

```
dataf = pd.DataFrame({'Actual':actual,'predicted':pred})
```

In [43]:

```
dataf
```

Out[43]:

	Actual	predicted
0	1	0
1	0	1
2	1	1
3	1	0
4	0	0
5	0	1
6	0	0
7	0	1
8	1	0
9	1	1

In [48]:

```
from sklearn  import metrics
```

In [51]:

```
from sklearn import metrics
```

In [53]:

```
metrics.confusion_matrix(actual,pred)
```

Out[53]:

```
array([[2, 3],
       [3, 2]], dtype=int64)
```

In [54]:

```
metrics.confusion_matrix(actual,pred)
```

Out[54]:

```
array([[2, 3],
       [3, 2]], dtype=int64)
```

In [58]:
```
metrics.accuracy_score(actual,pred)*100
```
Out[58]:
```
40.0
```

In [60]:
```
from sklearn import metrics
```

In [62]:
```
metrics.confusion_matrix(actual,pred)
```
Out[62]:
```
array([[2, 3],
       [3, 2]], dtype=int64)
```

In [63]:
```
from sklearn import metrics
```

In [65]:
```
metrics.confusion_matrix(actual,pred)
```
Out[65]:
```
array([[2, 3],
       [3, 2]], dtype=int64)
```

In [66]:
```
from sklearn import metrics
```

In [67]:
```
metrics.confusion_matrix(actual,pred)
```
Out[67]:
```
array([[2, 3],
       [3, 2]], dtype=int64)
```

In [70]:
```
metrics.accuracy_score(actual,pred)*100
```
Out[70]:
```
40.0
```