In [11]:

```
%matplotlib inline
```

IRIS DATASET LINK UCI

https://archive.ics.uci.edu/ml/datasets/Iris#targetText=UCI%20Machine%20Learnir
(https://archive.ics.uci.edu/ml/datasets/Iris#targetText=UCI%20Machine%20Learni

In [12]:

```
from sklearn import datasets
import numpy as np
import csv
import pandas as pd
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
```

In [13]:

```
iris = datasets.load_iris()
x = iris.data[:, [0,2]]
y = iris.target
x_moded=np.delete(x,[100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,11
y_moded=np.delete(y,[100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,11

# splitting data into test and train data set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_moded,y_moded, test_size=0.3, random_
#print(x_train)
#print(y_train)


weights = np.zeros((1, 2))
```

In [14]:

```
#creating activation function sigmoid function
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

In [ ]:

```

```

Typesetting math: 0%

In [15]:

```
# training set
n = [0 for i in range(len(x_test))]
p=0
for k in range(0,1):

    for i in range (0,len(x_train)):                                    #training data

        actual_result = weights[0][0]*x_train[i][0] + weights[0][1]*x_train[i][1]
        if actual_result > 0:
            fx = 1
               # print("greater than 0")
        if actual_result <= 0:
            fx = 0
                #print("less than 0")


        if fx != y_train[i]:
            error = y_train[i] - fx
            weights[0][0] = weights[0][0] + error*x_train[i][0]
            weights[0][1] = weights[0][1] + error*x_train[i][1]
                #print(weights[:])

            p=p+1
            print(p)
            print("learning")
            #plt.plot(p)
        else:
            weights[0][0] = weights[0][0]
            weights[0][1] = weights[0][1]

            print("no error")
    #print(weights[0][i])

        for i in range (0,len(x_test)):                #testing data included in the loop

            actual_result = weights[0][0]*x_test[i][0] + weights[0][1]*x_test[i][1]
            if actual_result > 0:
                fx = 1
               # print("greater than 0")
            if actual_result <= 0:
                fx = 0
                #print("less than 0")


            n[i]=fx
print ("Accuracy",+accuracy_score(y_test[:],n[:])) #finding accuracy
print('\033[1m',  (accuracy_score(y_test[:],n[:]) *100),'%' , '\033[0m')
print("the final results are\n")
```

```
no error
1
learning
2
```

```
learning
3
learning
no error
no error
4
learning
5
learning
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
```

Typesetting math: 0%

```
no error
no error
no error
no error
no error
no error
no error
no error
no error
no error
Accuracy 1.0
 100.0 %
the final results are
```

In [ ]:

In [ ]:

In [ ]: