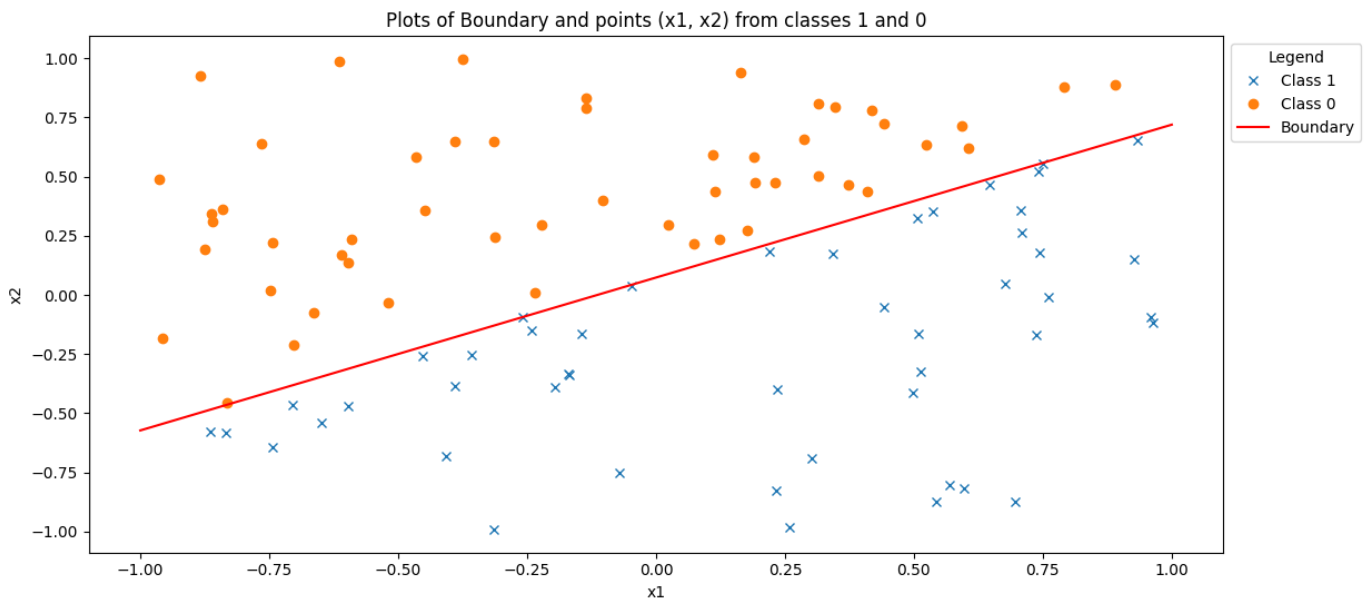
Name: Sai Anish Garapati
UIN: 650208577

Assignment 2:

-> Randomly picked Optimal weights:
Optimal W: [[0.05298914]
 [0.46673872]
 [-0.72210569]]

(1) **Experiment 1 (n = 100):**

-> Plot showing line w0 + w1x1 + w2x2 = 0 and all the points belonging to classes S1 and S2 :



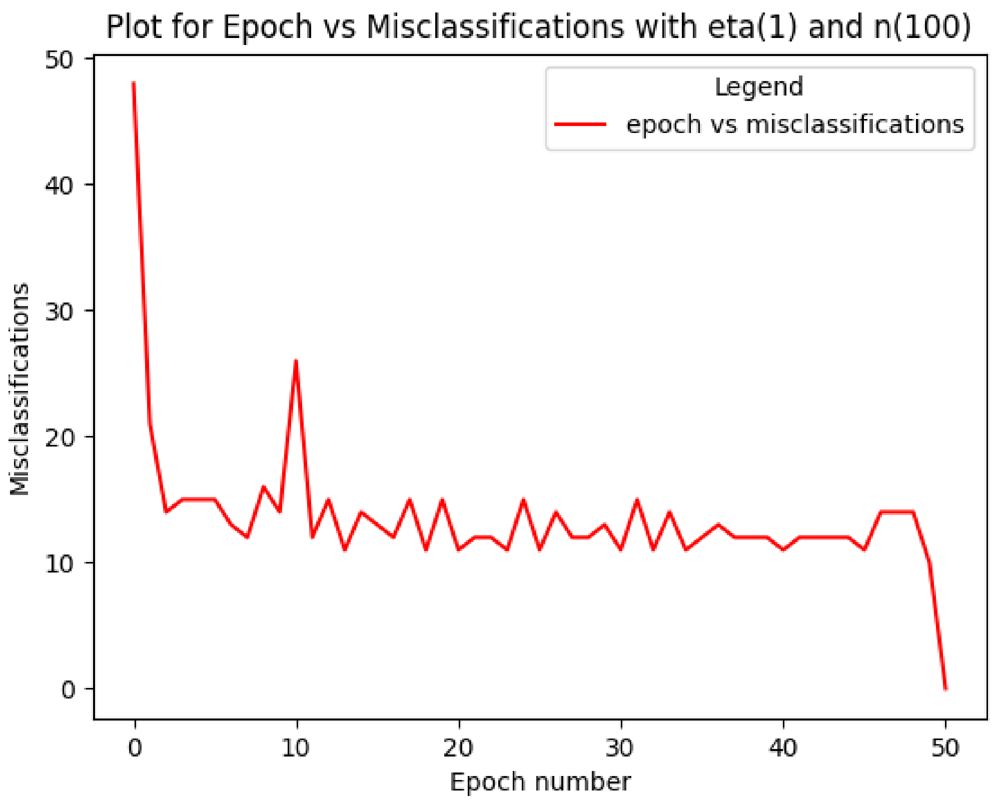
-> Initial training weights chosen to train the perception model using Perceptron Training Algorithm:
Initial W_train: [[0.86954234]
 [0.7512086]
 [-0.4745009]]

(1.1) **eta = 1**

-> Epochs taken by the algorithm to converge: 50

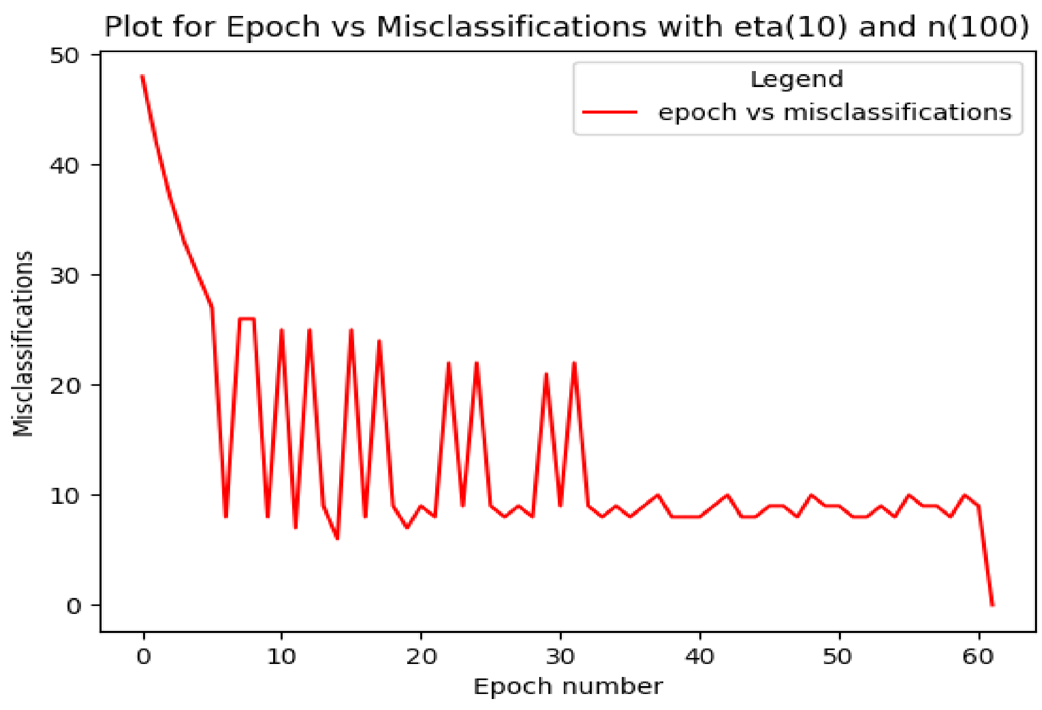
-> Final weights obtained after the completion of Perception Training Algorithm with eta = 1:
Trained W_train: [[0.86954234]
 [5.22919197]
 [-6.98240005]]

-> Weights compared to optimal weights:
The final obtained weights are different from the optimal weights, although the initial training weights are close to optimal weights as the learning rate = 1 could be updating the weights by non-optimal amounts.



(1.2) **eta = 10**

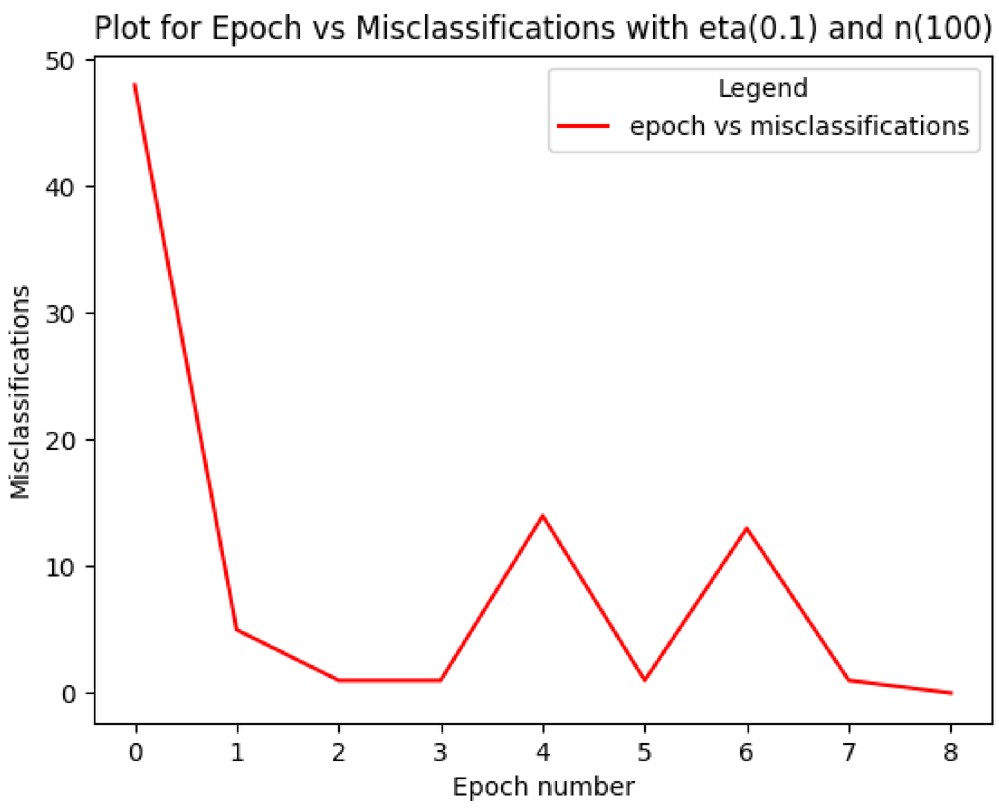
-> Epochs taken by the algorithm to converge: 61
-> Final weights obtained after the completion of Perceptron Training Algorithm with eta = 10:
Trained W_train: [[10.86954234]
 [55.66737058]
 [-77.43427285]]



(1.3) **eta = 0.1**

-> Epochs taken by the algorithm to converge: 8
-> Final weights obtained after the completion of Perceptron Training Algorithm with eta = 0.1:

```
Trained W_train: [[ 0.06954234]
 [ 0.62016663]
 [-0.9616709 ]]
```



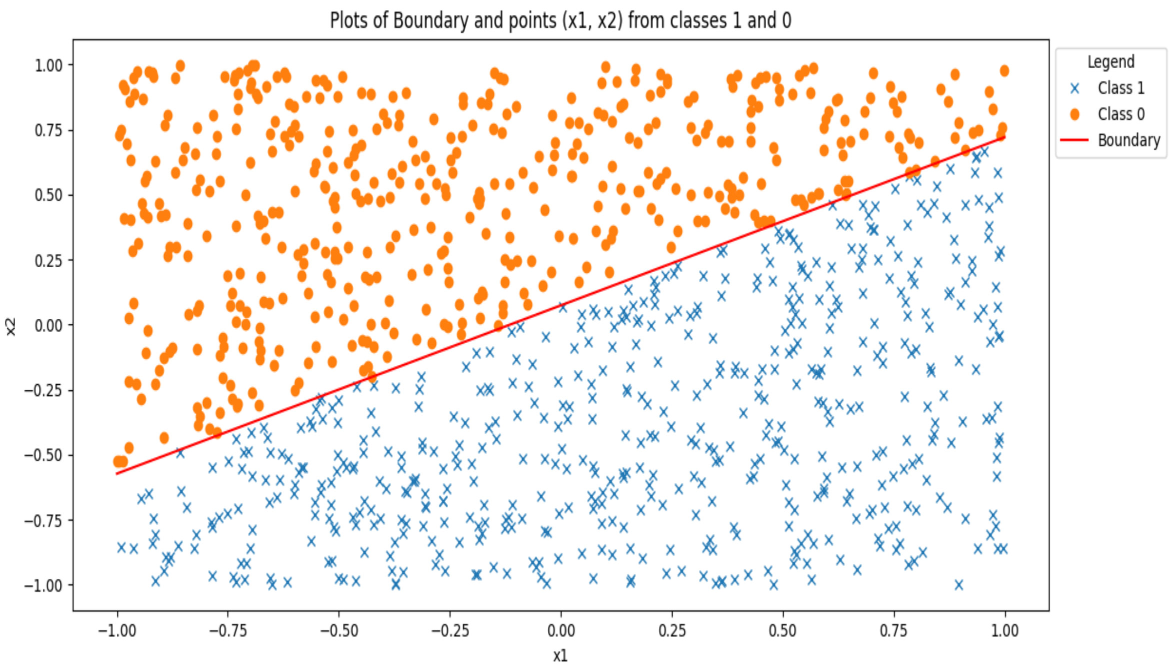
-> Effect of eta on number of epochs taken for convergence:
(1.4) Epochs taken to converge increased from 50 to 61 when eta was changed from 1 to 10. Increase in eta has resulted in more overshoots during the convergence. When eta was changed to 0, epochs taken have reduced considerably to 8 with very little overshoots during convergence.

-> Effects of eta on training performance with different weights and dataset:
(1.5) It is not guaranteed to get the same effects of eta on training performance with different choices of optimal weights, Training set and Initial weights as a change in any of these 3 choices would change the amounts by which the weights are modified during the training process which can change the rate of convergence.

---

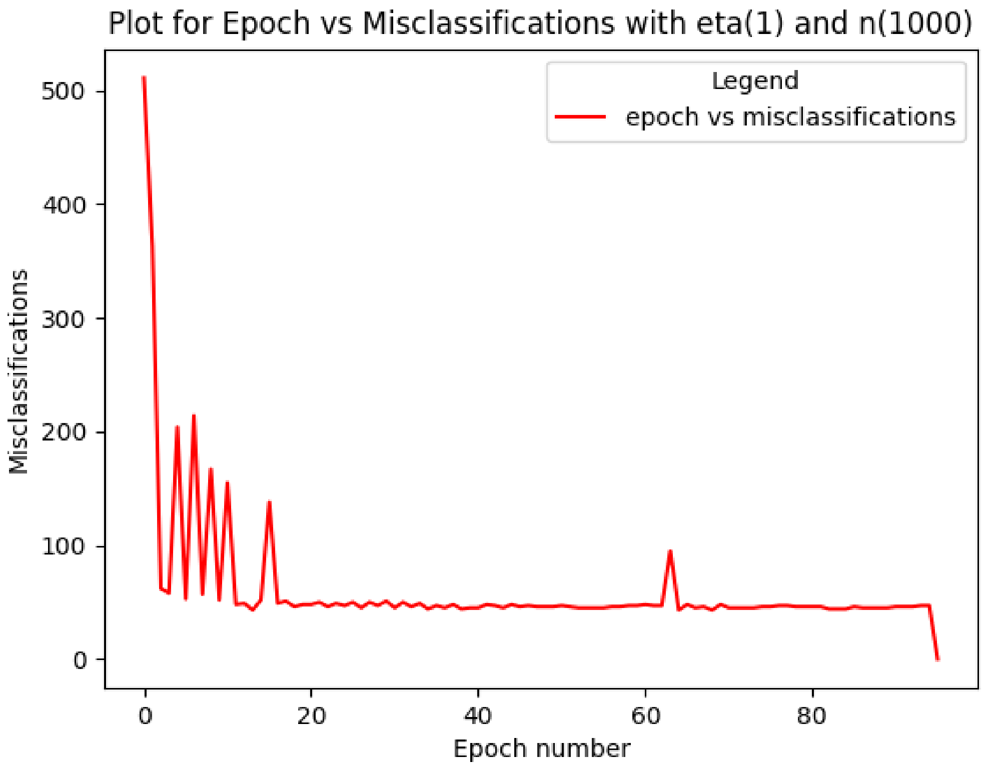
(2) **Experiment 2 (n = 1000):**

-> Plot showing line w0 + w1x1 + w2x2 = 0 and all the points belonging to classes S1 and S2 :



-> Initial training weights chosen to train the perception model using Perceptron Training Algorithm:
Initial W_train: [[0.93024056]
 [-0.57393613]
 [0.84574649]]

(2.1) **eta = 1**
-> Epochs taken by the algorithm to converge: 95
-> Final weights obtained after the completion of Perception Training Algorithm with eta = 1:
Trained W_train: [[0.93024056]
 [8.07309734]
 [-12.59425067]]

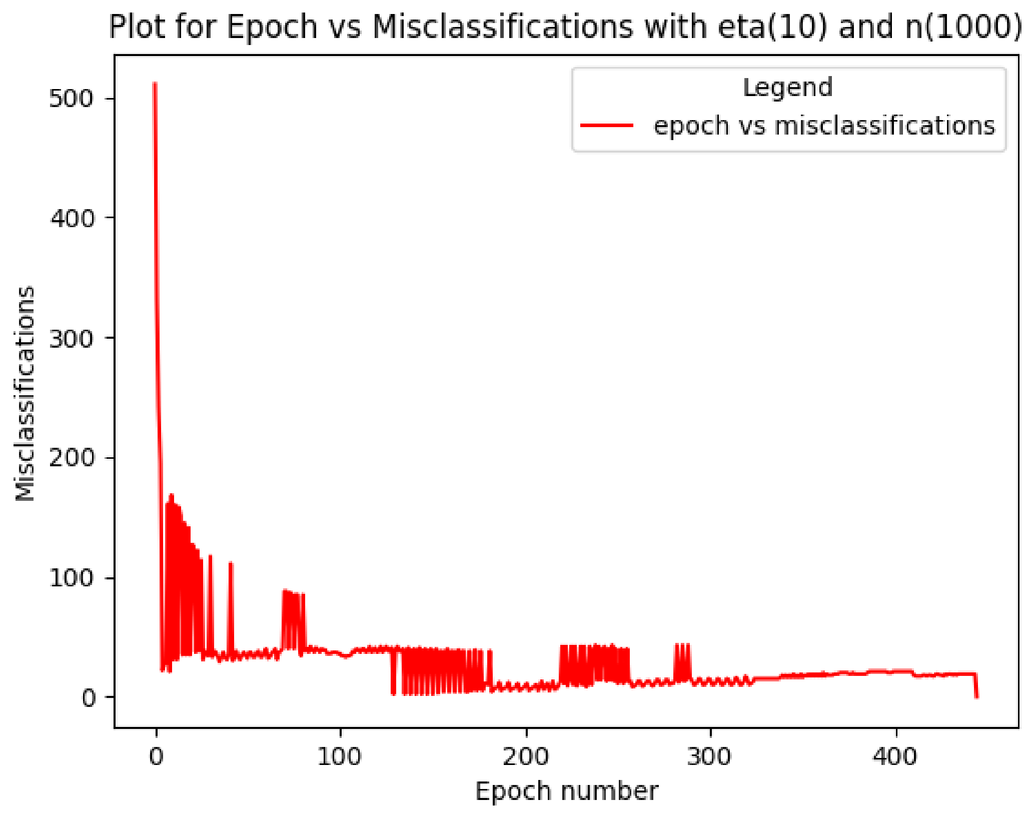


(2.2) **eta = 10**

-> Epochs taken by the algorithm to converge: 444

-> Final weights obtained after the completion of Perception Training Algorithm with eta = 10:

```
Trained W_train: [[  20.93024056]
 [ 182.79380034]
 [-286.73227752]]
```

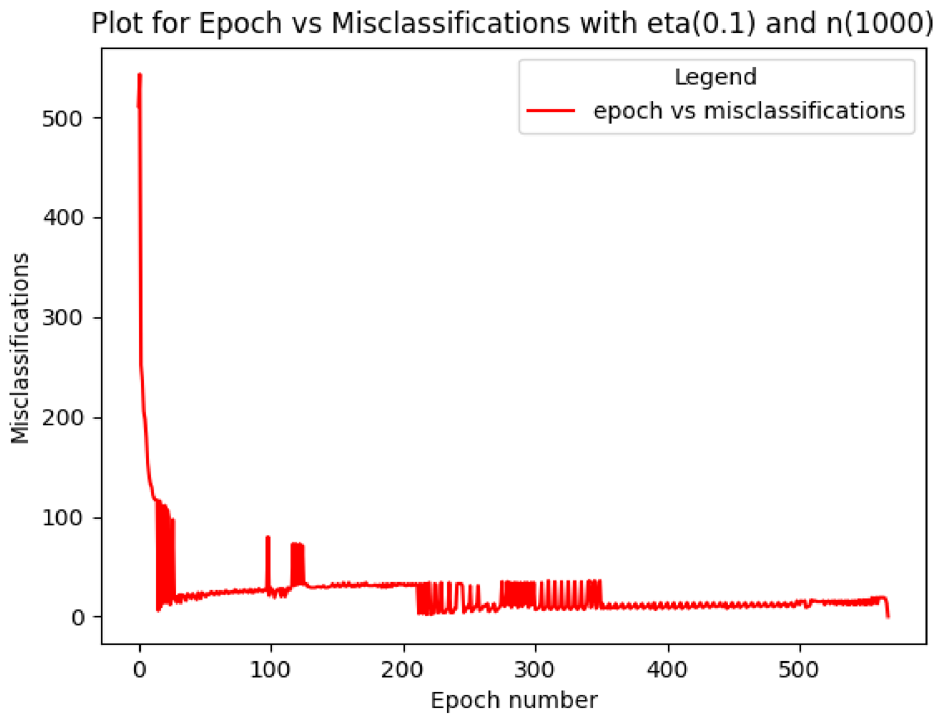


(2.3) **eta = 0.1**

-> Epochs taken by the algorithm to converge: 567

-> Final weights obtained after the completion of Perception Training Algorithm with eta = 0.1:

```
Trained W_train: [[ 0.23024056]
[ 1.93461367]
[-2.99198917]]
```



-> Effect of eta on number of epochs taken for convergence:
(2.4) Epochs taken to converge increased from 95 to 444 when eta was changed from 1 to 10. Increase in eta has resulted in more overshoots during the convergence. When eta was changed to 0, epochs taken have increased to 567. This could be because of the large size of dataset and also the dataset being evenly spread out, updating weights based on misclassifications could have resulted in new errors, and though the overshoots seem similar to the plot with eta=10, due to smaller eta(0.1), it took little long for convergence

-> Effects of eta on training performance with different weights and dataset:
(2.5) It is not guaranteed to get the same effects of eta on training performance with different choices of optimal weights, Training set and Initial weights as a change in any of these 3 choices would change the amounts by which the weights are modified during the training process which could change the rate of convergence.

-> Differences in convergence for n = 100 and n = 1000:
(3) In both cases (n = 100) and (n = 1000) an increase in epochs was seen when eta was changed from 1 to 10. The increase in the number of epochs was more in the (n = 1000) case which can be owed to the large dataset. But for eta = 0.1 in (n = 100) epochs were less as opposed to eta = 0.1 in (n = 1000) case. This could be because we are updating weights on a lot more misclassifications in the (n = 1000) case which is making curve overshoot during the convergence.

Python Code:

```python
# Name: Sai Anish Garapati
# UIN: 650208577

import numpy as np
import matplotlib.pyplot as plt

np.random.seed(2021)

def unit_activation(W, X):
    return (np.dot(np.transpose(W), X) >= 0)


def count_misclassifications(S_0, S_1, W):
    misclass = 0
    for vector in S_0:
        misclass += (unit_activation(W, vector) == 1)
    for vector in S_1:
        misclass += (unit_activation(W, vector) == 0)
    return int(misclass)


def PTA(eta, W_train_init, S_0, S_1):
    W_train = W_train_init
    epoch = 0
    misclassifications = []
    while (1):
        misclassifications.append(count_misclassifications(S_0, S_1, W_train))

        if (misclassifications[-1] == 0):
            break
        for vector in S_1:
            W_train = W_train + eta * vector * \
                (1 - unit_activation(W_train, vector))
        for vector in S_0:
            W_train = W_train + eta * vector * \
                (0 - unit_activation(W_train, vector))
        epoch += 1

    return W_train, epoch, misclassifications


def assignment_2_experiment(n, W):
    # Generating a dataset of size n
    S = []
    for _ in range(0, n):
        X = np.random.uniform(-1, 1, size=(2, 1))
        X = np.insert(X, 0, 1, axis=0)
        S.append(X)

    # Classifying dataset into classes 1 and 0
    S_1 = []
    S_0 = []
    for vector in S:
        local_field = np.dot(np.transpose(W), vector)
        if (local_field >= 0):
            S_1.append(vector)
        else:
            S_0.append(vector)

    plt.title('Plots of Boundary and points (x1, x2) from classes 1 and 0')
    plt.xlabel('x1')
    plt.ylabel('x2')
    plt.plot([x[1] for x in S_1], [x[2] for x in S_1], 'x', label='Class 1')
    plt.plot([x[1] for x in S_0], [x[2] for x in S_0], 'o', label='Class 0')

    x_1 = np.linspace(-1, 1)
    x_2 = (-W[0] - W[1]*x_1)/W[2]
    plt.plot(x_1, x_2, 'r', label='Boundary')
    plt.legend(title='Legend', bbox_to_anchor=(1.0, 1.0), loc='upper left')
    plt.show()

    print('-----------------------------------------------------------------')

    # Generating initial training weights
    W_train_init = np.random.uniform(-1, 1, size=(3, 1))

    for eta in [1, 10, 0.1]:
        # Running the PTA algorithm with a specific eta
        W_train_comp, epoch, misclassifications = PTA(eta, W_train_init, S_0, S_1)

        print('Model training completed with eta({}), n({}) in {} epochs'.format(
            eta, n, epoch))
        print('Initial W_train:', W_train_init)
        print('Trained W_train:', W_train_comp)
        print('Optimal W:', W)

        plt.title(
            'Plot for Epoch vs Misclassifications with eta({}) and n({})'.format(eta, n))
        plt.xlabel('Epoch number')
        plt.ylabel('Misclassifications')
        plt.plot([i for i in range(0, epoch + 1)], misclassifications,
                 'r', label='epoch vs misclassifications')
        plt.legend(title='Legend')
        plt.show()
        print('-----------------------------------------------------------------')

if (__name__ == '__main__'):
    # Choosing optimal weights
    w_0 = np.random.uniform(-0.25, 0.25)
    W = np.random.uniform(-1, 1, size=(2, 1))
    W = np.insert(W, 0, w_0, axis=0)

    assignment_2_experiment(n=100, W=W)
    assignment_2_experiment(n=1000, W=W)
```