# **Dry Beans Classification using Perceptron and Adaline**

## **Task Description**

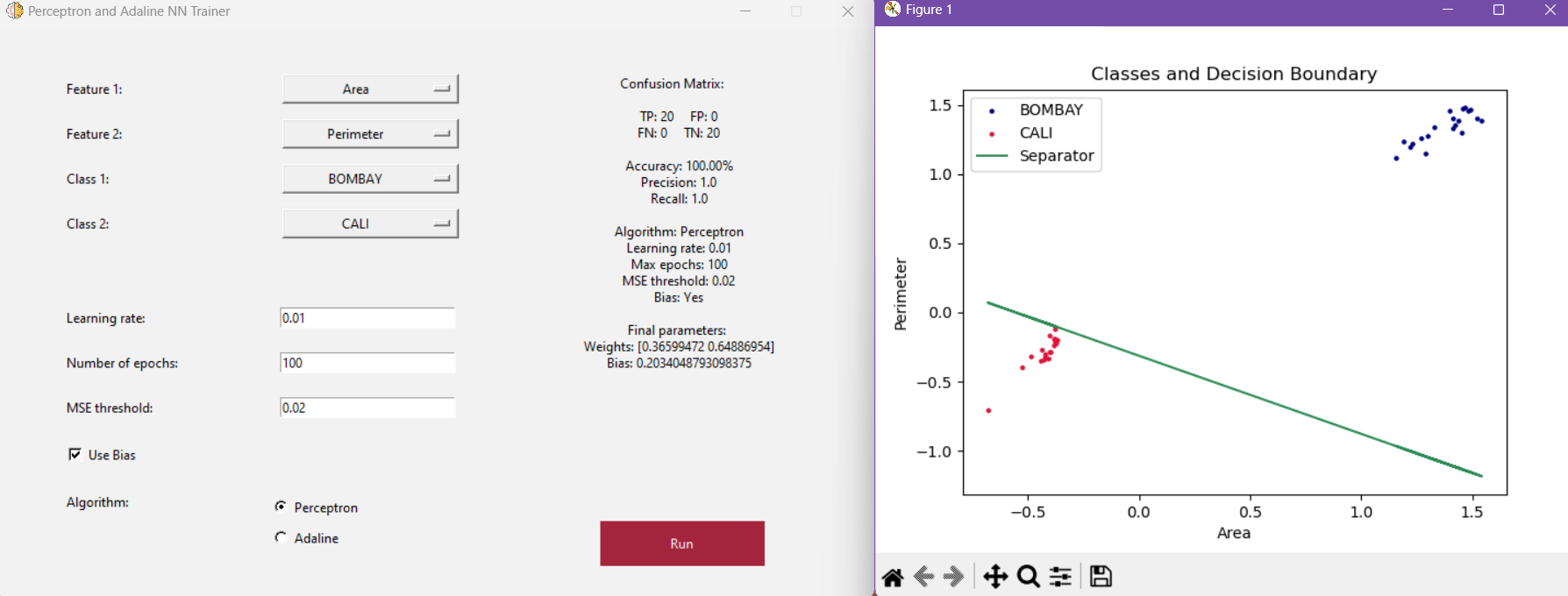
The dataset in this task contains 5 features (Area, Perimeter, MajorAxisLength, MinorAxisLength, Roundness) and 3 classes (BOMBAY, CALI, SIRA). The aim of this task is to build a simple neural network that uses either the Single Layer Perceptron algorithm or the Adaline algorithm to differentiate between 2 of the 3 classes using 2 of the 5 features.

## **Preprocessing**

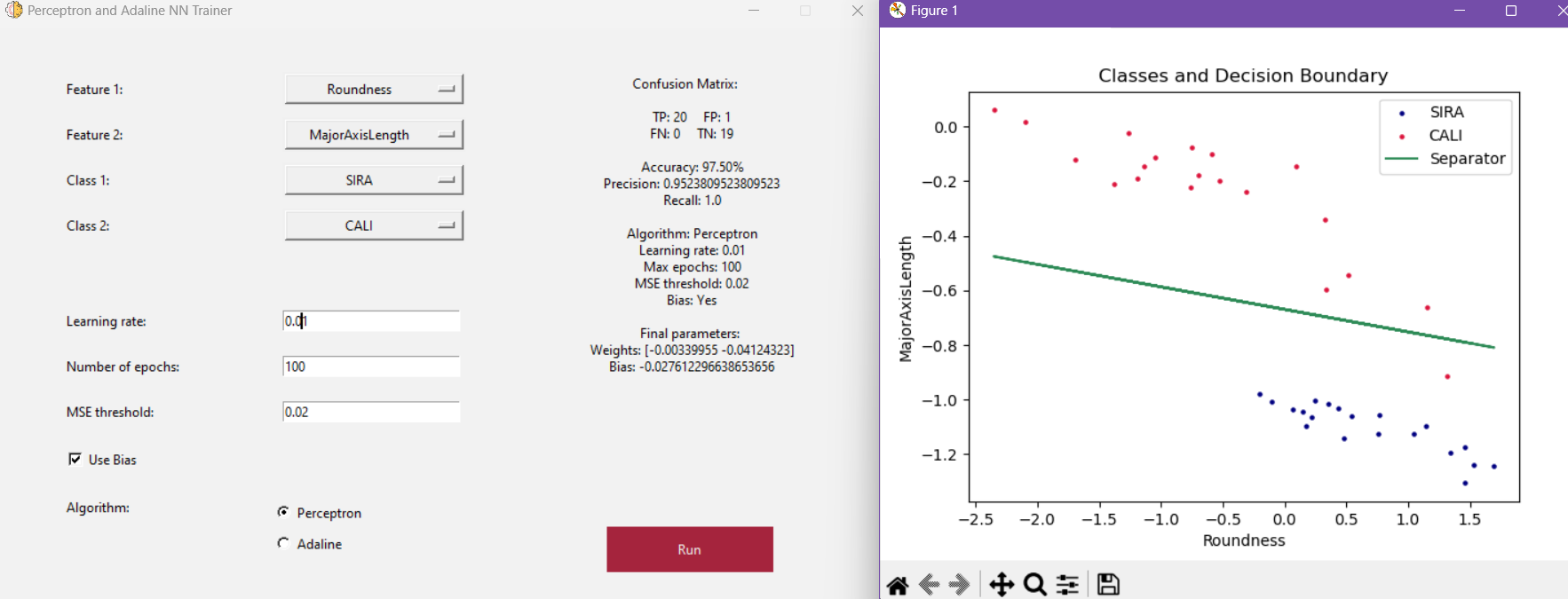
* This dataset does not require much preprocessing given the nature of the features provided.
* The features were normalized using a standard scaler to avoid overflow issues when multiplying big numbers.
* Null values were filled by the mean value of their respective column.
* Only the rows containing the user selected classes were selected for training and the rest were omitted, in addition to the only selecting the columns of the user selected features.
* The class column was encoded to be 1 for one class and -1 for the other for the signum activation function to function properly when training the models.

## **Analysis and Visualization**

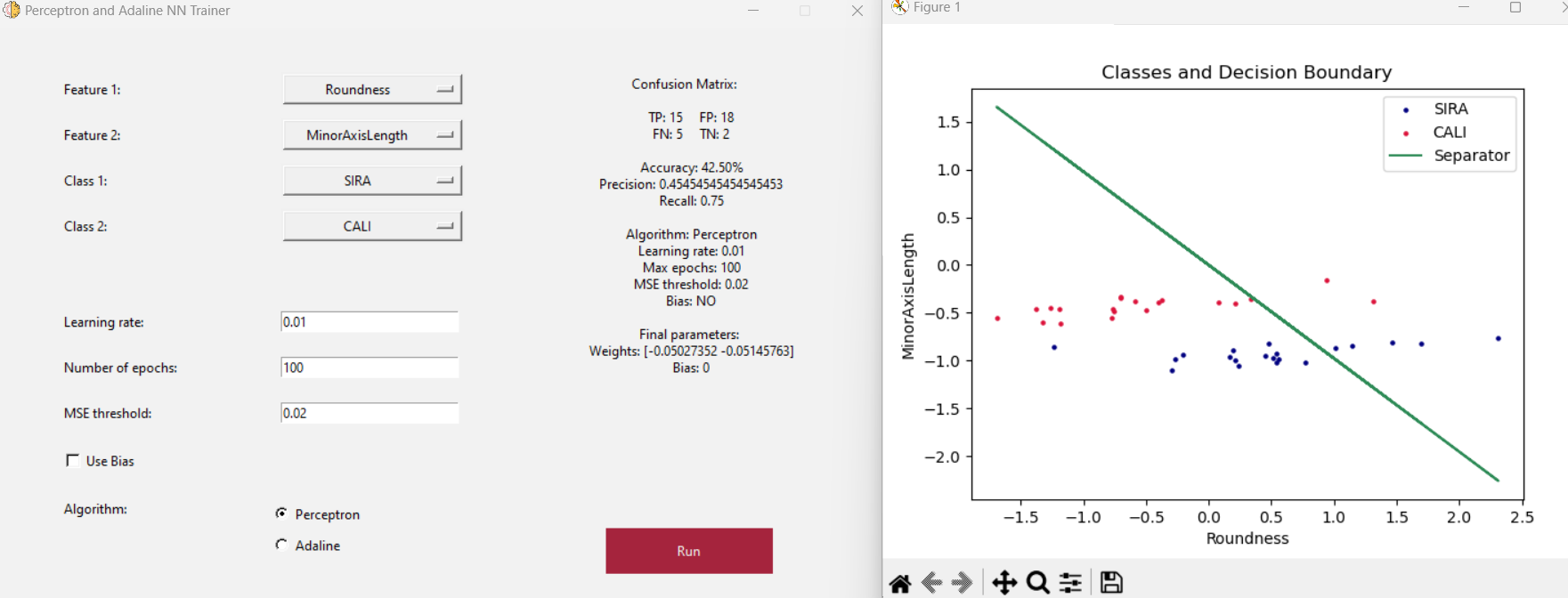
### **#1 Perceptron Algorithm**



The model achieves great accuracy and differentiates well between the two classes “BOMBAY” and “CALI” however, the decision boundary margins are very tight on one side and spacious on the other.



The accuracy is high but not perfect, the model separates well between the two classes “SIRA” and “CALI” with these features. Sometimes the model achieves 100% accuracy and sometimes not, it is most likely due to the random initialization of the weights and bias.

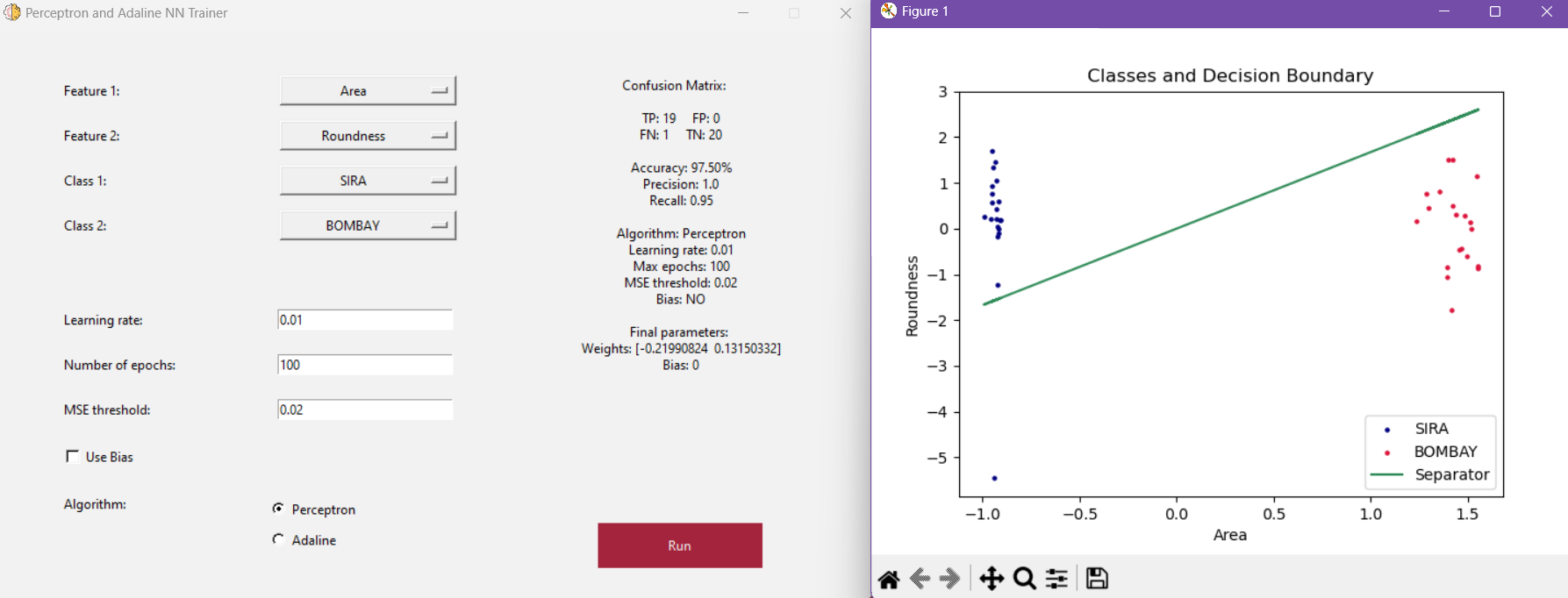


The model struggles to differentiate between “SIRA” and “CALI” with these features, the accuracy is very low, but it improves noticeably when enabling bias with these particular parameters.

A screenshot of a computer

Description automatically generated

The algorithm struggles to differentiate between the “SIRA” and “CALI” classes with these features, however adding a bias helps greatly with these settings.



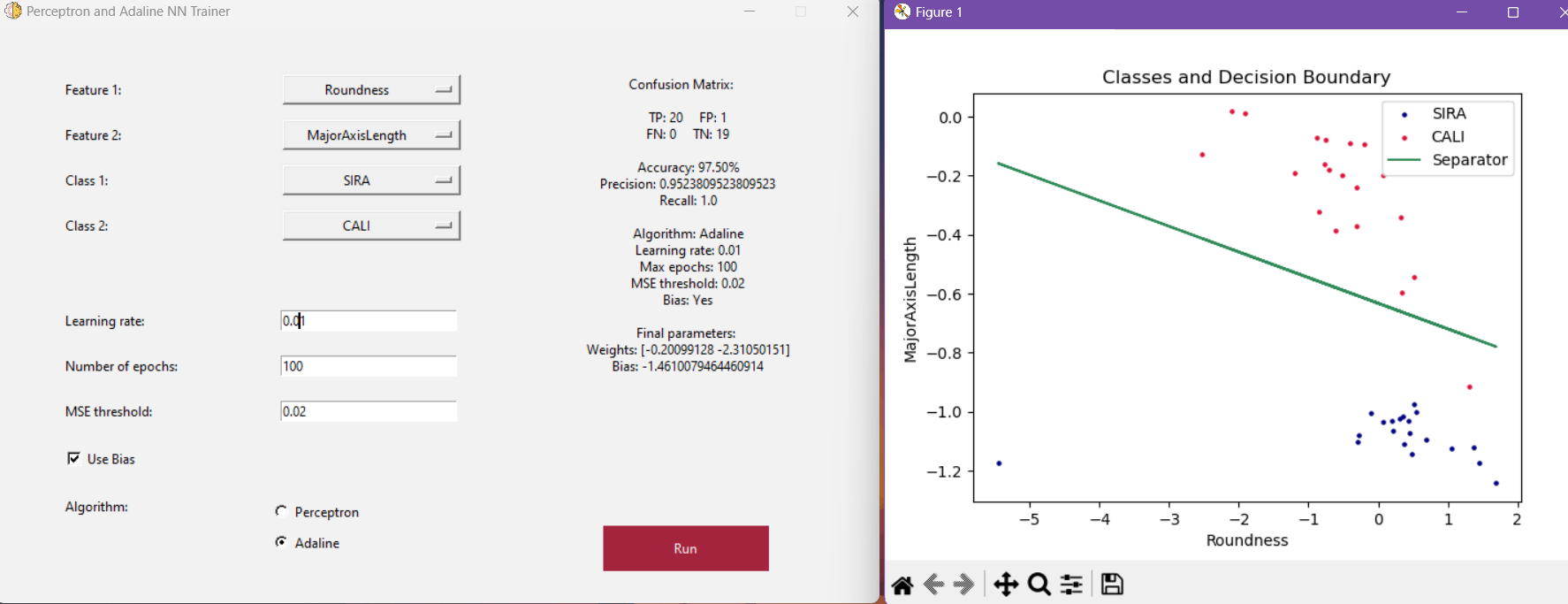
The model differentiates well between “SIRA” and “BOMBAY” beans with these features, the accuracy is quite good even without adding bias. However the separation is not so great, a near vertical line would provide better separation here given the distribution of the data.

### **#2 Adaline Algorithm**

A screenshot of a computer

Description automatically generated

The model achieves great accuracy and differentiates well between the two classes “BOMBAY” and “CALI” the separation here is better than that of the Perceptron algorithm with the same parameters.

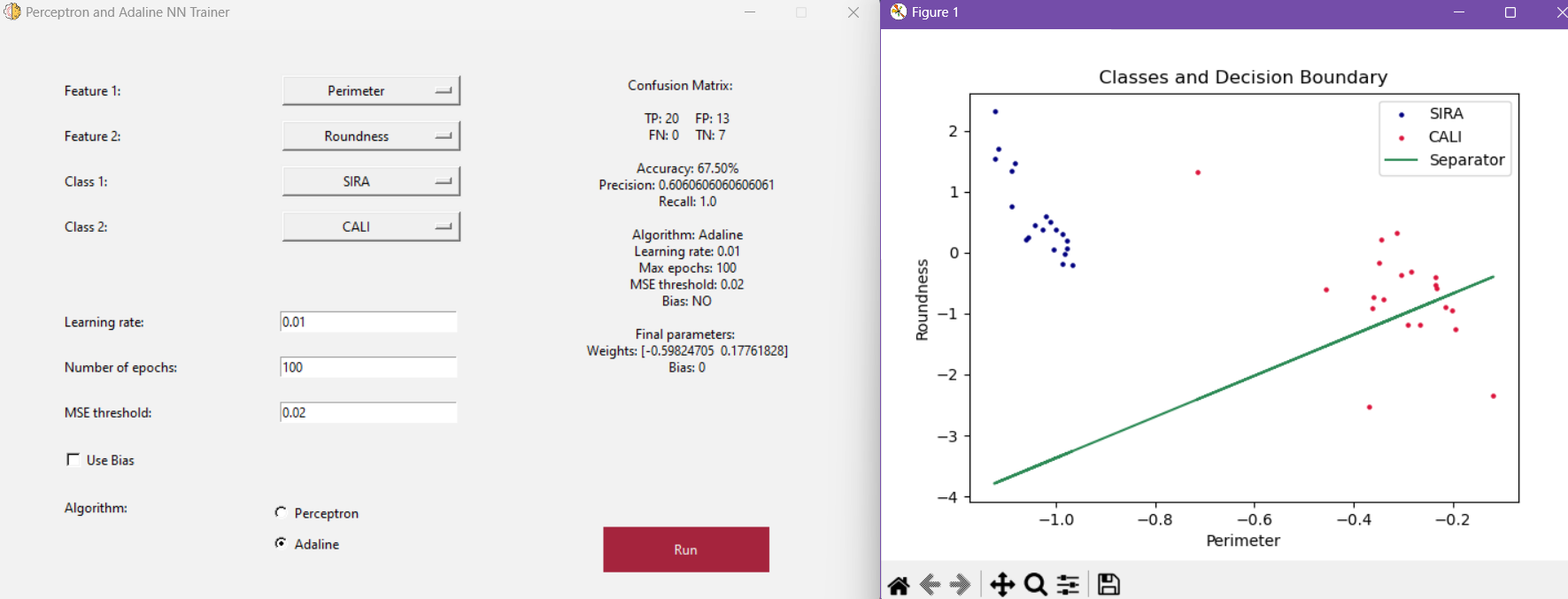


Similar to the Perceptron algorithm, the accuracy is high and the model separates well between the two classes “SIRA” and “CALI” with these features.

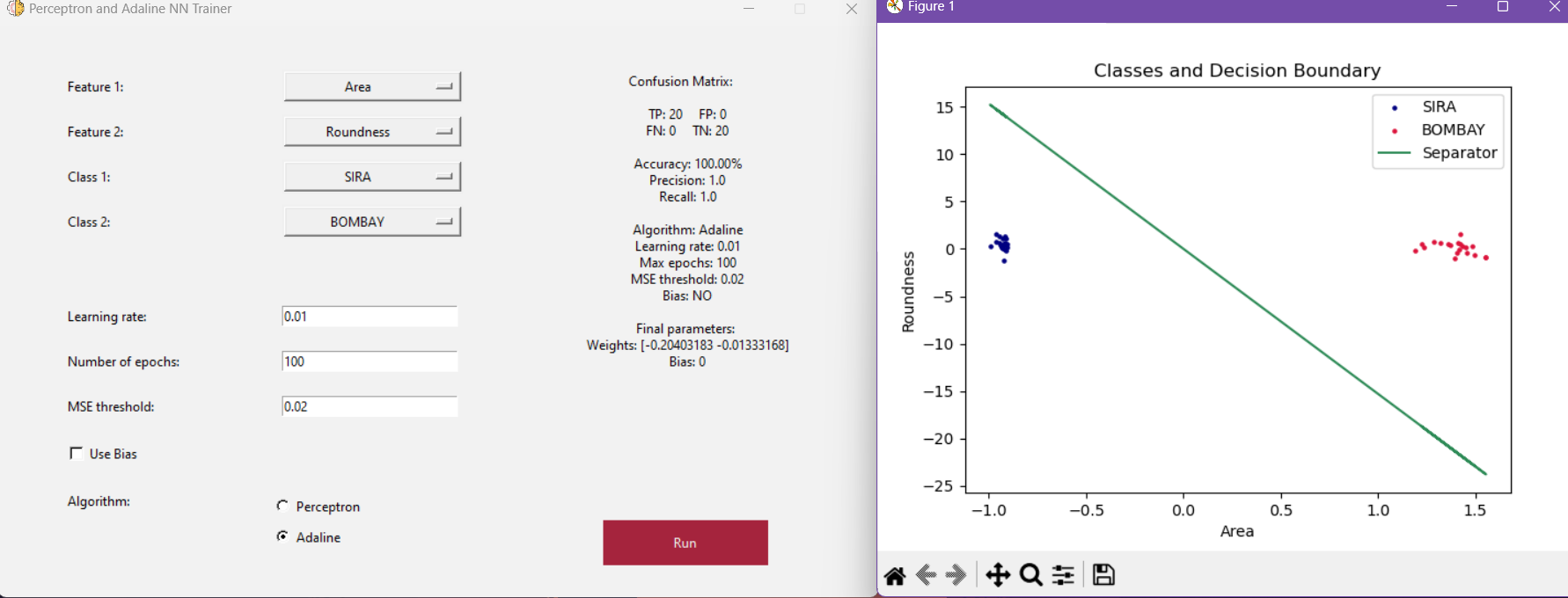
A screenshot of a computer

Description automatically generated

This Adaline model here achieves a better result when compared to the Perceptron, likely because it can pick up on detailed differences when calculating errors. The separation is very hard with these features, but enabling bias helps get a higher accuracy because the separator is horizontal so a constant bias would help achieve that.



The model performs poorly with these parameters, it does not perform well when it comes to differentiating between “CALI” and “SIRA” beans with this parameters, however it does perform a bit better than the Perceptron model and adding bias makes it significantly better with much higher accuracy.



The model performs exceptionally good here in differentiating between “SIRA” and “BOMBAY” beans, it achieves the highest accuracy with very good and clear separation and enabling or disabling bias has no effect here.

## **Conclusion**

* The data is linearly separable to a great extent, so both Perceptron and Adaline algorithms perform well on it.
* The Perceptron algorithm often converges faster than the Adaline algorithm, on average, it finishes training in less epochs.
* The Adaline algorithm is often more capable of picking up on finer details and often creates better margins with better classes separation.
* Adding bias with some features combinations can have a significant impact on the performance of the models.