Report on Perceptron and Adaline Algorithms

Introduction

This report discusses the implementation of the Perceptron and Adaline algorithms for binary classification tasks using a dataset. The primary goal was to build and train models to classify data into two specified classes.

Perceptron Algorithm

Implementation

The Perceptron algorithm was implemented as a Python class. It takes various parameters, including features, classes, learning rate (lr), the number of epochs, mean squared error (MSE) threshold, and whether to add bias. It initializes weights and bias if required. The preprocessing method selects data of specified classes, converts the class column to numeric, handles missing values, and scales the data. It also performs train-test splitting. The **train_model** method trains the Perceptron model, and the **draw_line** method visualizes the decision boundary. The **predict** method makes predictions, and the **confusion_matrix** and **calc_accuracy** methods evaluate the model's performance.

Usage

The **workFlow** function in the main file facilitates the use of the Perceptron algorithm. It allows users to select features, classes, learning rate, epochs, MSE threshold, and bias. The algorithm is then trained and evaluated on a dataset, and the accuracy and confusion matrix are displayed.

Adaline Algorithm

Implementation

Similar to the Perceptron algorithm, the Adaline algorithm is also implemented as a Python class. It shares many features with the Perceptron, such as parameter settings, preprocessing, train-test splitting, and evaluation. However, it differs in the training process by utilizing the mean squared error to update model parameters and use linear activation function in the training, making it a more continuous learning algorithm compared to the binary classification nature of the Perceptron.

Usage

The Adaline algorithm can be employed using the same **workFlow** function, where users can select features, classes, learning rate, epochs, MSE threshold, and bias. The algorithm is trained and evaluated in a similar manner to the Perceptron, and accuracy and the confusion matrix are reported.

GUI Application

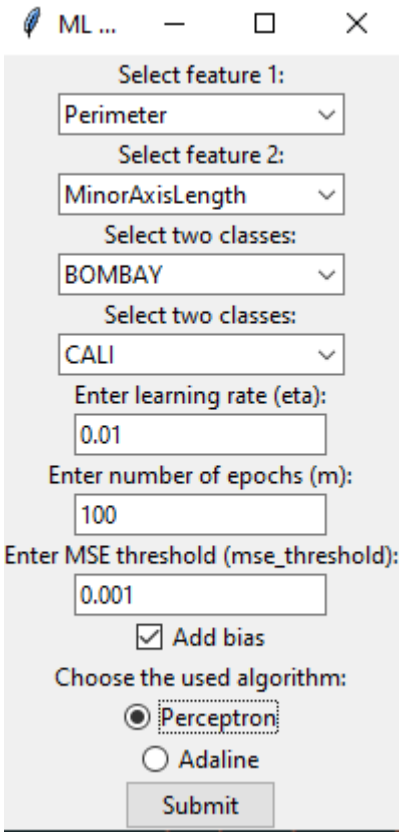
To facilitate the use of both algorithms, a GUI application has been developed using the tkinter library. Users can select features, classes, learning rate, epochs, and other parameters through the graphical interface. Upon submission, the selected algorithm (Perceptron or Adaline) is executed, and the results are displayed, including accuracy and the confusion matrix.

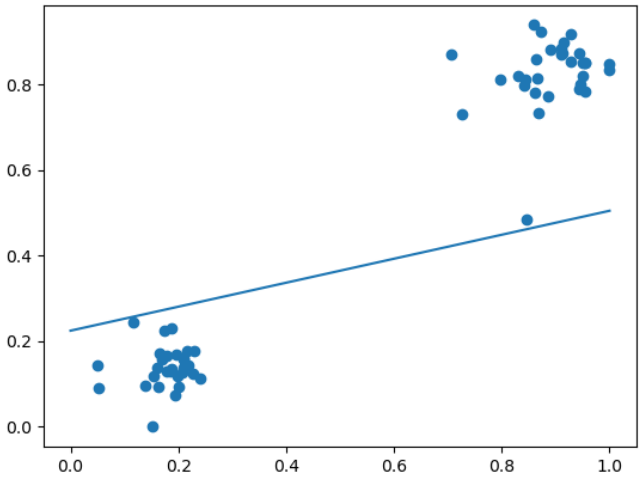
Conclusion

This project successfully implements and utilizes the Perceptron and Adaline algorithms for binary classification tasks. The graphical user interface (GUI) simplifies parameter selection and execution, making these algorithms accessible for users interested in binary classification tasks. The accuracy and confusion matrix provide insights into the models' performance.

Results

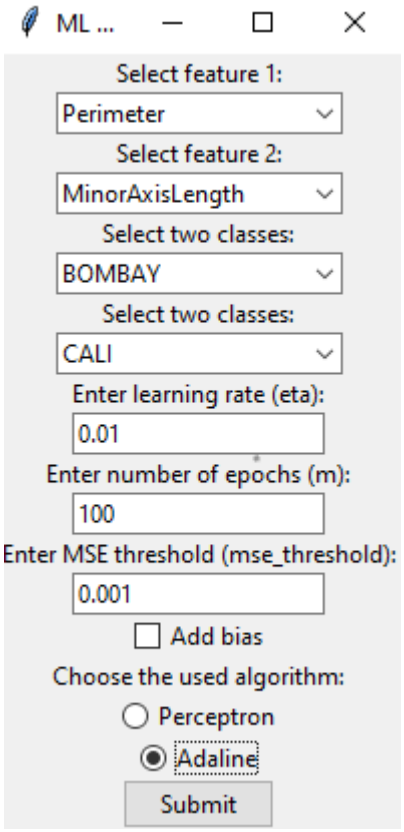
1- Perceptron with bias

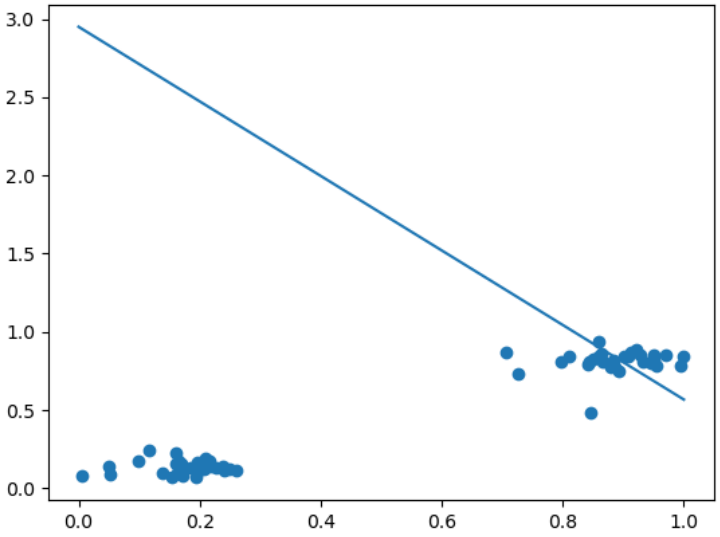





```
Accuracy =  100.0 %
Confution Matrix:
        BOMBAY  CALI
BOMBAY    19.0   0.0
CALI       0.0  21.0
```

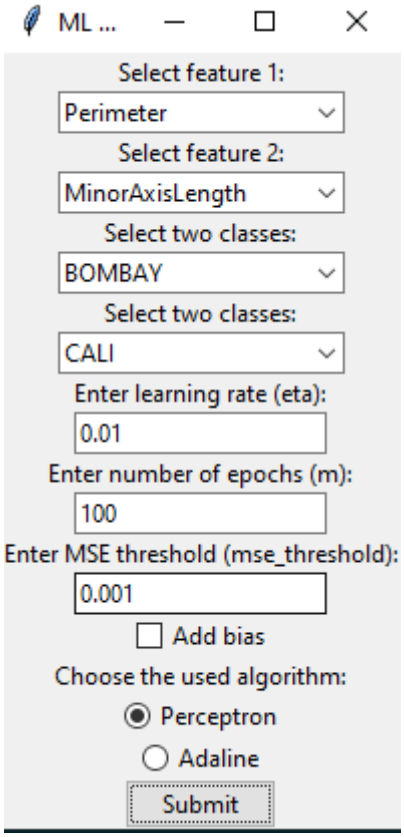
2- Adaline with bias

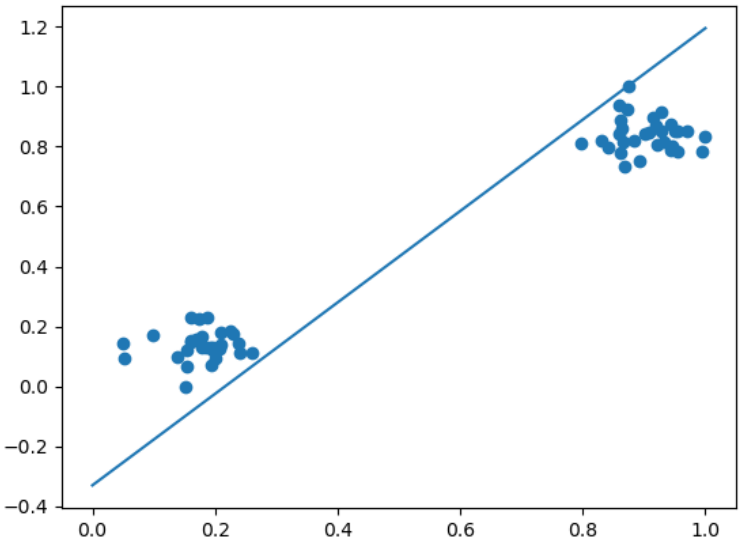





```
Accuracy =  87.5 %
Confution Matrix:
        BOMBAY  CALI
BOMBAY    15.0   5.0
CALI       0.0  20.0
```

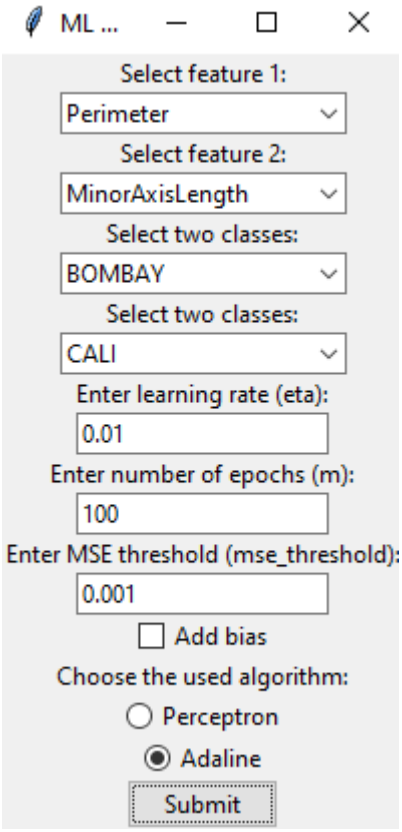
3- Perceptron without bias

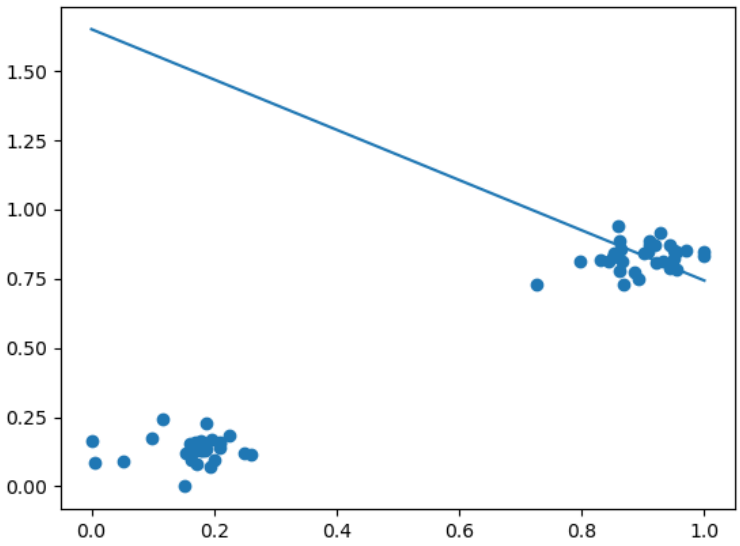





```
Accuracy =  100.0 %
Confution Matrix:
        BOMBAY  CALI
BOMBAY    19.0   0.0
CALI       0.0  21.0
```

4- Adaline without bias






```
Accuracy =  85.0 %
Confution Matrix:
        BOMBAY  CALI
BOMBAY    11.0   6.0
CALI       0.0  23.0
```