2/21/2023

Deep Learning Assignment-1

Perceptron

Presented by:

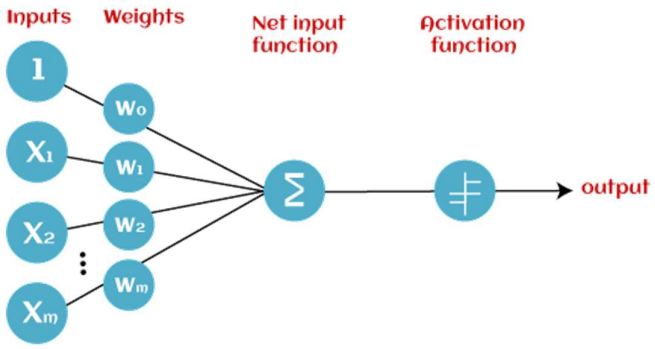
1. R. Ramakrishna Kashyap – T22101
2. Anuj Kumar Shukla – T22103
3. Nishant Gupta- T22221

Index

Perceptron

A perceptron is a type of neural network model used for binary classification tasks. It consists of a single layer of artificial neurons that receive input signals, weight them, and combine them to produce a single output. The output is then passed through an activation function, which maps the output to a binary value, usually 0 or 1.

The perceptron learning algorithm, also known as the delta rule, is used to train the perceptron by adjusting the weights assigned to each input signal in response to errors made during classification. The algorithm continues to adjust the weights until a stopping criterion is met or until the weights converge to a stable set of values.



Input Nodes or Input Layer:

This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.

Wight and Bias:

Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

Activation Function:

These are the final and important components that help to determine whether the neuron will fire or not. Activation Function can be considered primarily as a step function.

Algorithm

The Perceptron algorithm is an iterative procedure for training a binary linear classifier. Given a set of input features and corresponding binary labels, the algorithm learns a set of weights for each feature and a bias term such that the linear combination of inputs and weights plus the bias is a good predictor of the binary labels.

1) Initialize the weights and bias term to zero or small random values.
2) For each training example in the dataset, do the following:
    a) Compute the dot product of the input features and the current weights and add the bias term.
    b) Apply the activation function (usually a step function) to the result of the dot product to obtain a predicted binary label.
    c) Update the weights and bias term according to the error in the prediction, multiplied by a learning rate and the input features.
3) Repeat steps 2 until the predicted labels for all training examples are correct or a maximum number of iterations is reached.

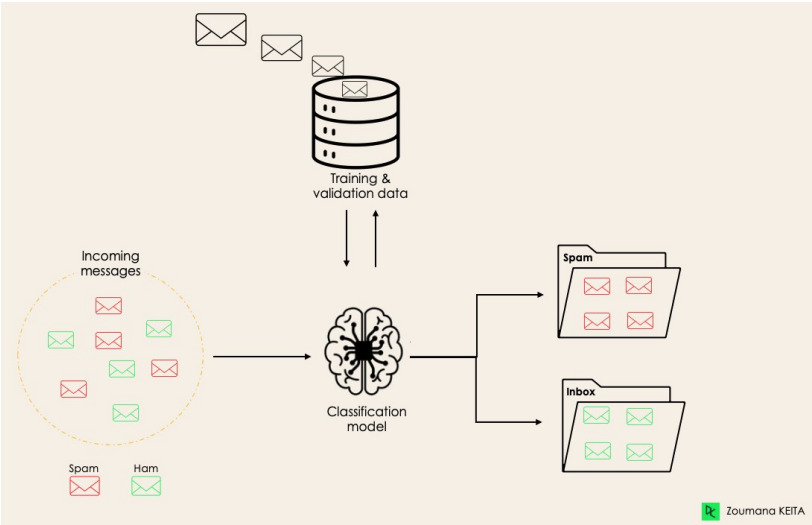
During training, the weights and bias term are updated based on the errors made in the predictions, which drive the algorithm to find the best set of weights that can accurately classify the training examples. Once trained, the weights and bias can be used to predict the binary label of new examples by computing the dot product of the input features and the weights, adding the bias term, and applying the activation function.

Classification tasks

Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.

In classification, the input data is classified into different categories or classes, based on a set of predetermined criteria. The categories or classes can be binary, meaning there are only two possible outcomes, or multi-class, meaning there are more than two possible outcomes.

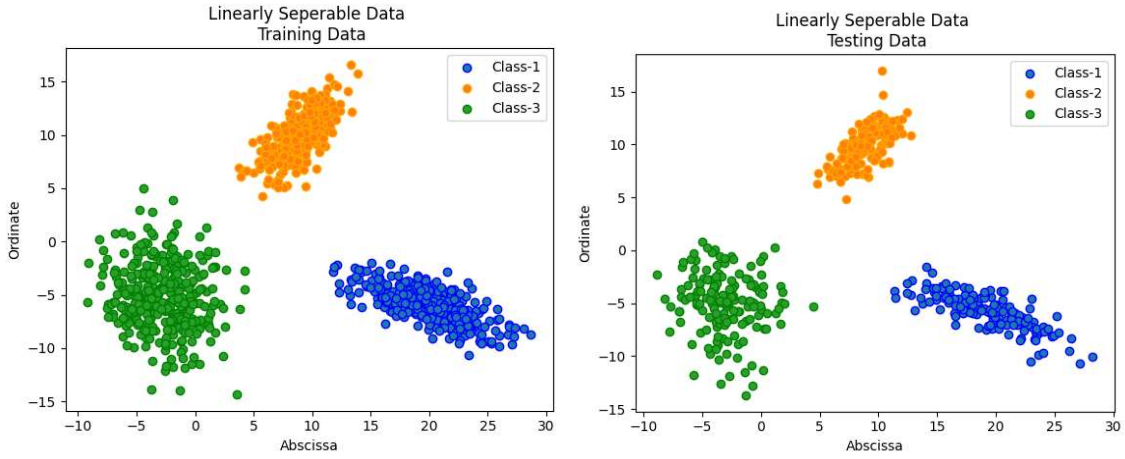
The process of classification involves representing input data as a feature vector, which comprises numerical or categorical values that describe the relevant characteristics of the data. The primary goal of a classification algorithm is to learn a function that can map these feature vectors to the target variable accurately. This learned function is then used to classify new instances of the data. The ability of the algorithm to accurately classify new instances depends on the quality of the mapping function, which should be generalizable and capable of making accurate predictions on previously unseen data.



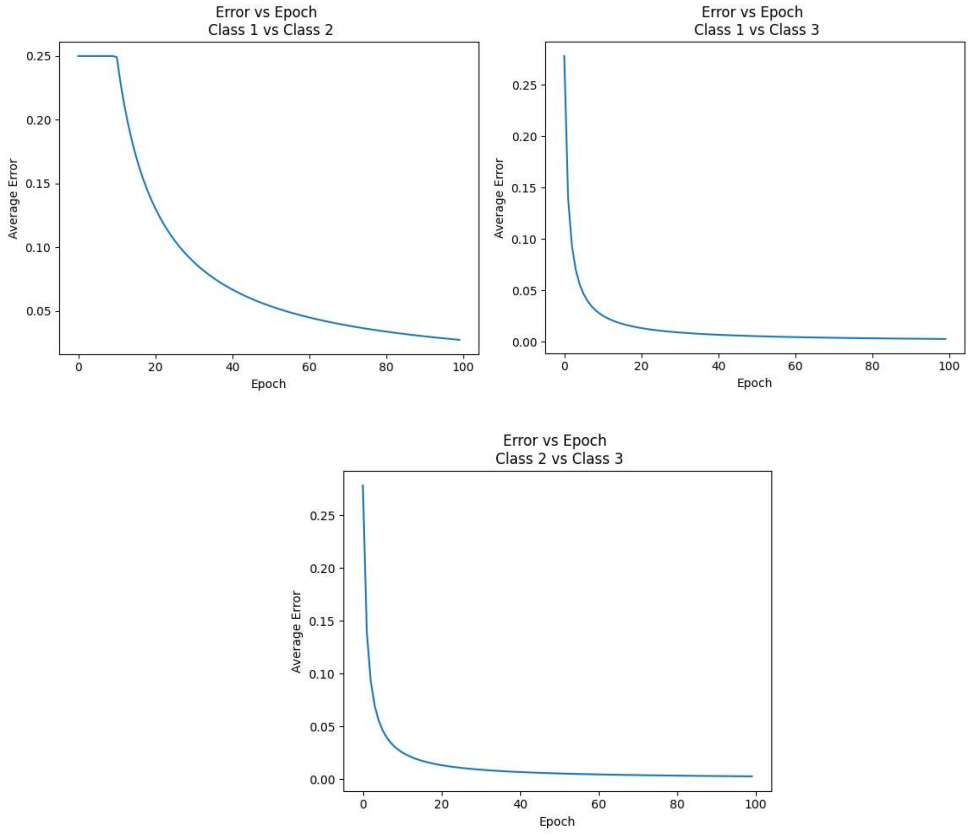
Results:

Linearly Separable Classes

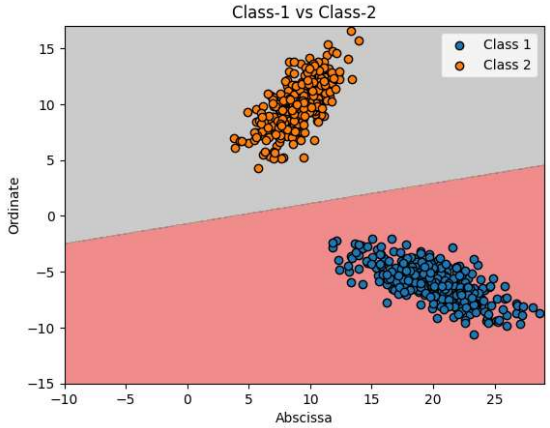
Sample Data

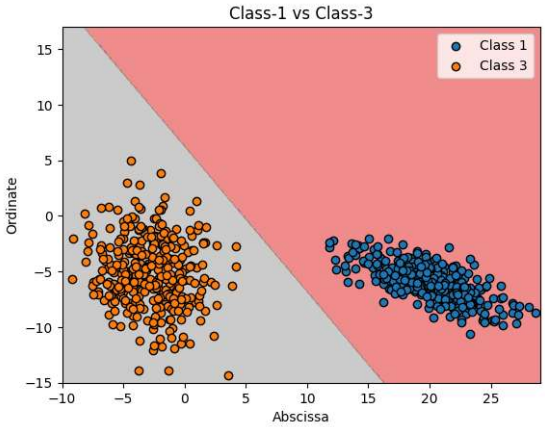


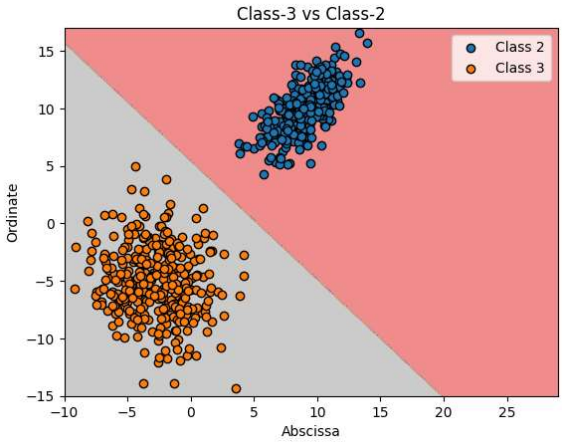
Plot of average error (y-axis) vs epochs (x-axis)

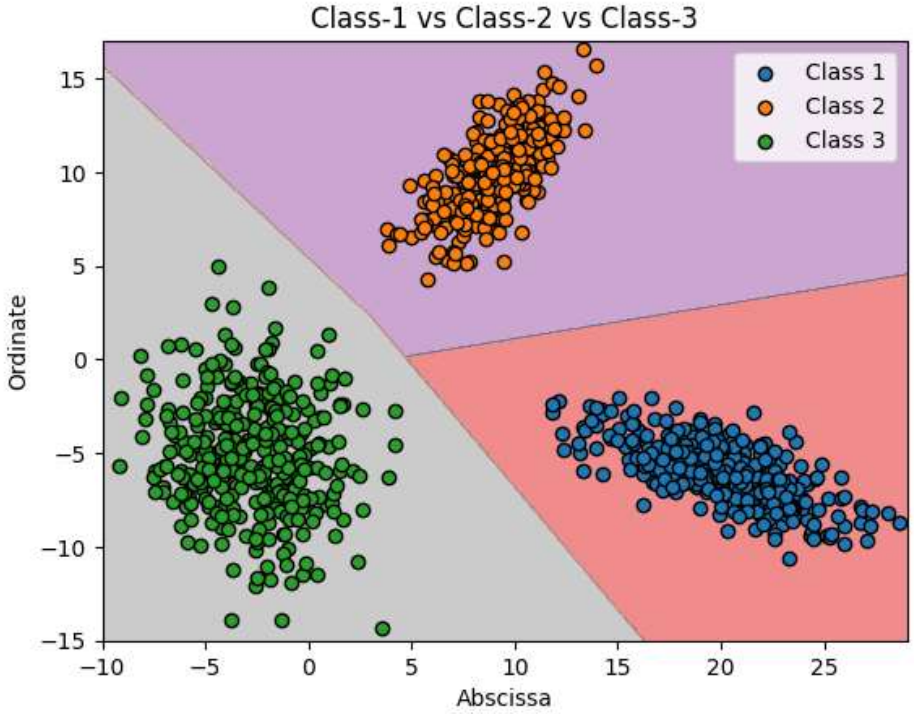


Decision Regions









Confusion Matrix:

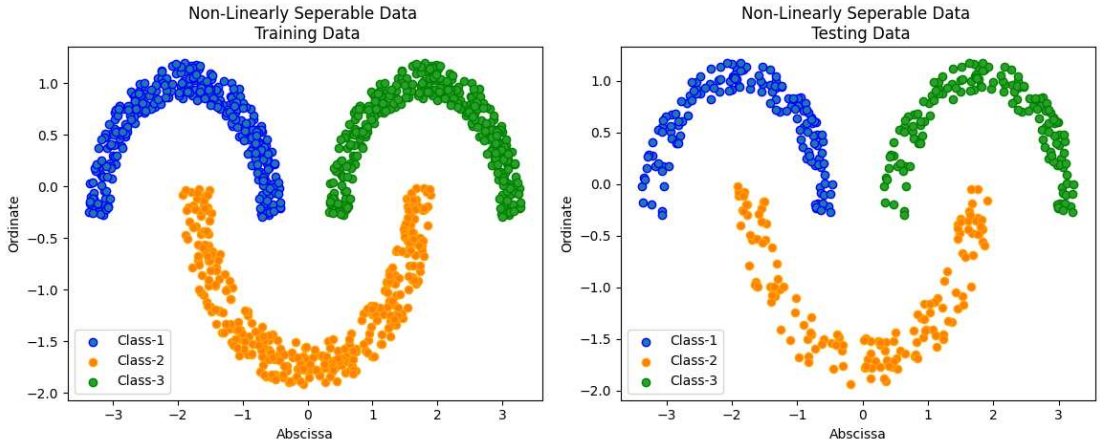
```
Confusion Matrix
 [[150   0   0]
 [  0 150   0]
 [  0   0 150]]

Accuracy : 100.0%
```

Inference:

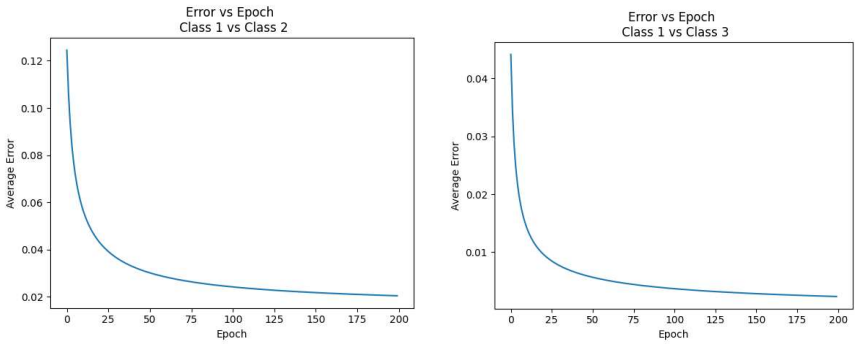
- Test samples were plotted on the decision region (Region we obtained by training the Perceptron model with train samples) and were lying accurately in their respective regions.
- Error after each epoch is decreasing and converging to zero as epochs are increased.
- Accuracy of the system is 100% as the data is linearly separable i.e. samples from different classes can be distinguished completely by a straight line. Hence all the test samples are lying strictly in their respective regions.

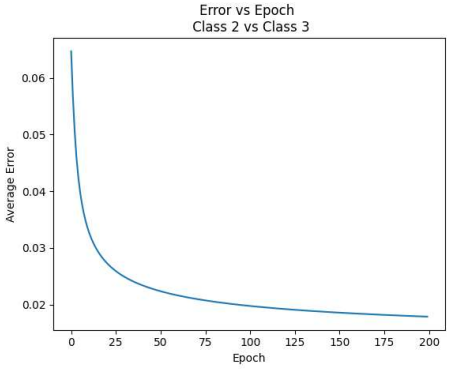
Non-Linearly Separable Classes

Sample Data

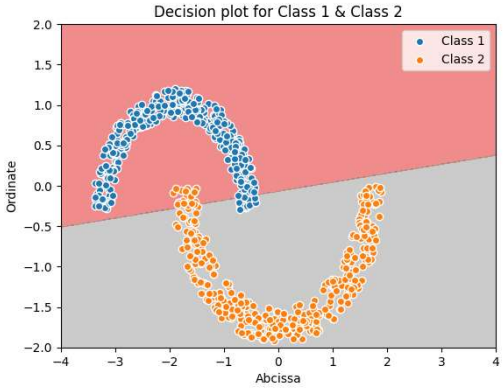


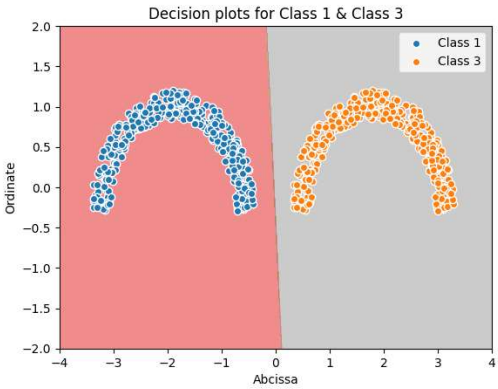
Plot of average error (y-axis) vs epochs (x-axis)

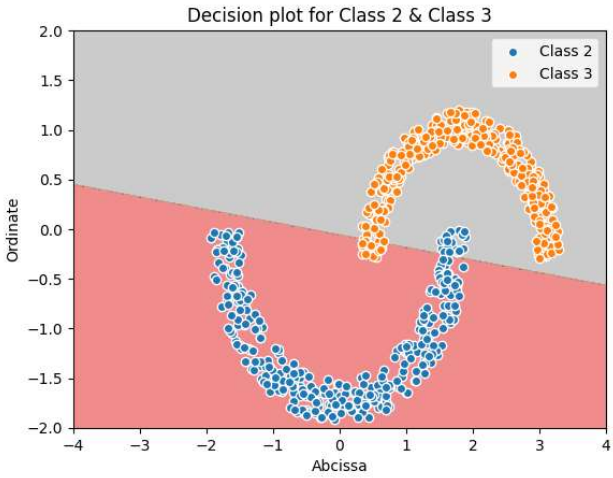


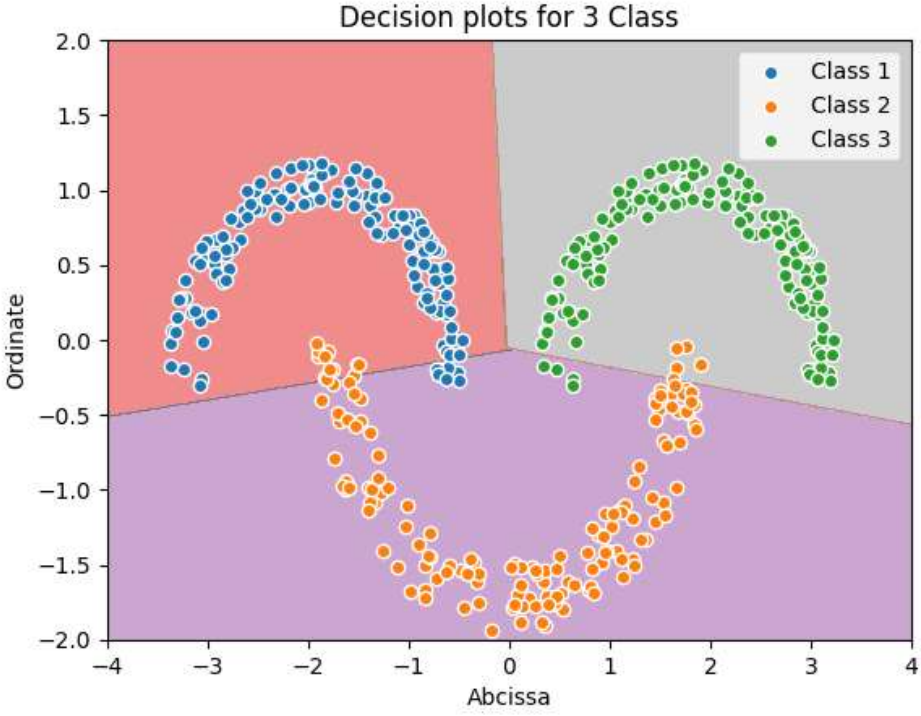


Decision Regions









Confusion Matrix:

```
Confusion Matrix
 [[142   8   0]
 [ 12 133   5]
 [  0   4 146]]

Accuracy : 94.0%
```

Inference:

- Test samples were plotted on the decision region (Region we obtained by training the Perceptron model with train samples) and were lying almost accurately in their respective regions.
- Error after each epoch is decreasing and converging to zero as epochs are increased.
- Accuracy of the system is 94% as the data is Non-linearly separable i.e. samples from different classes can't be distinguished completely by a straight line. Hence some samples might lie outside their designated regions and will be classified incorrectly.

Regression Task:

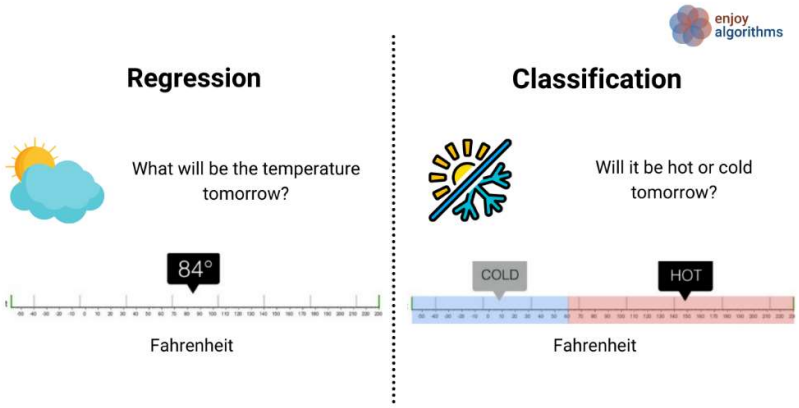
In deep learning, a regression task involves predicting a continuous output value based on a set of input features. The goal is to learn a function that maps the input features to the target output variable.

Deep learning models can be used for regression tasks by constructing a neural network architecture that is designed to learn a complex mapping between the input features and the output variable. The neural network can be trained on a dataset of labelled examples, where each example includes the input features and the corresponding output value.

During training, the neural network is adjusted to minimize the difference between its predicted output and the true output value for each example in the dataset. This is typically achieved by minimizing a loss function, such as mean squared error, which measures the difference between the predicted output and the true output.

Common deep learning techniques used for regression tasks include feedforward neural networks, convolutional neural networks, and recurrent neural networks. These models can be trained using gradient descent optimization techniques, such as stochastic gradient descent, to update the model parameters and minimize the loss function.

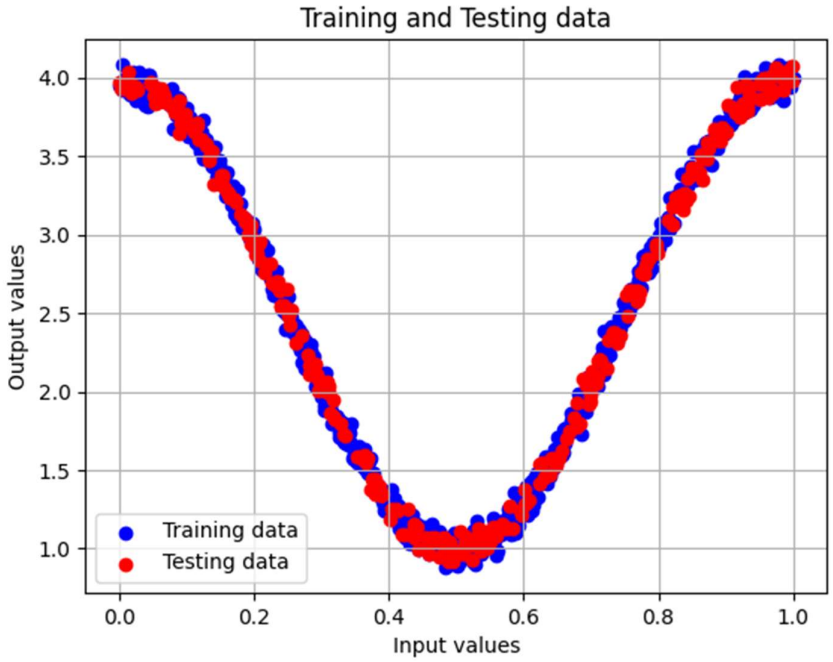
Once the model is trained, it can be used to make predictions on new, unseen data by feeding the input features through the neural network and obtaining the corresponding output value.



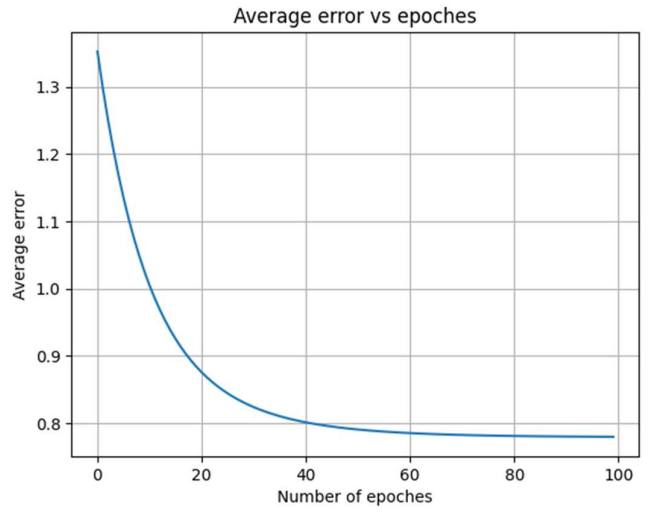
Results:

1-dimensional (Univariate) input data

Training and Testing Data

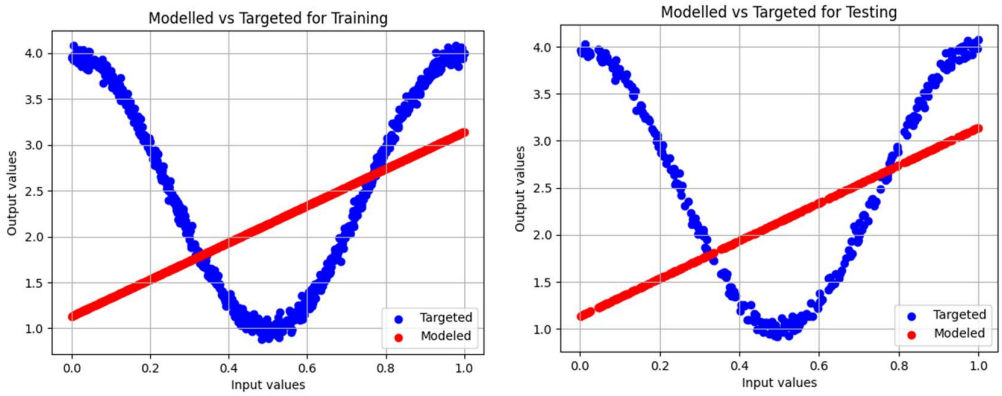


Plot of average error (y-axis) vs epochs (x-axis)

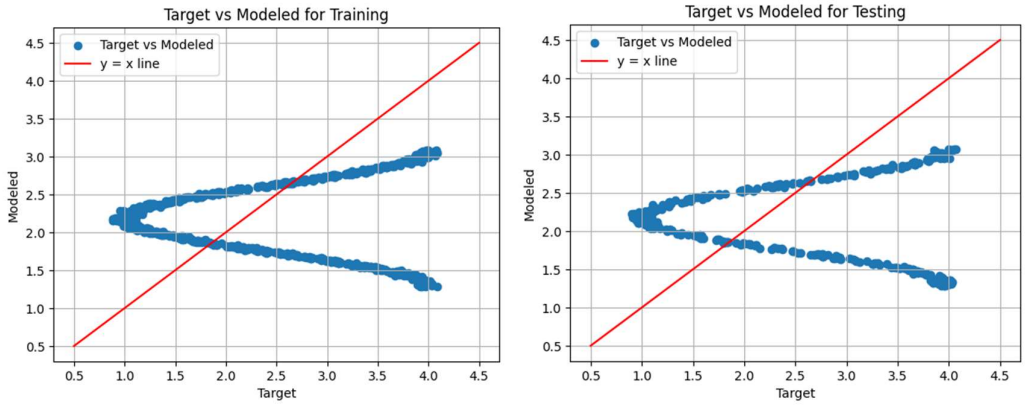


Plot of the values of mean squared error on training data and test data

Plots of model output and target output for training data and test data

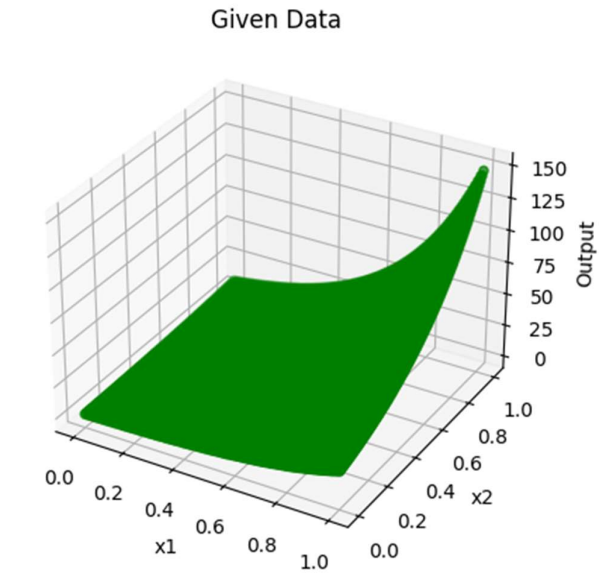


Scatter plot for Training and Testing (Target o/p on x-axis and model o/p on y-axis)

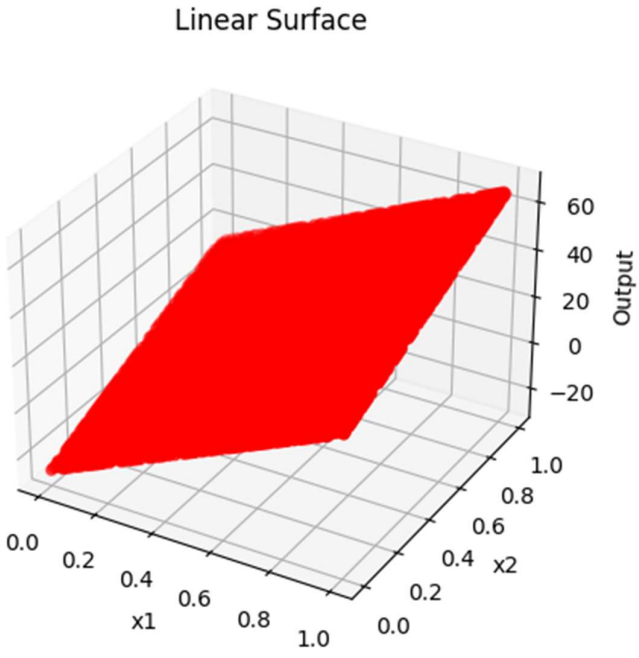


2-dimensional (Bivariate) input data

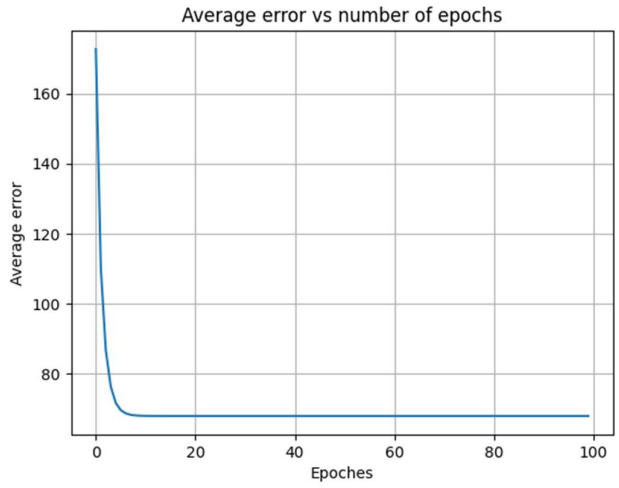
Given Data (Perceptron model is tested on this Data)



Surface which tries to fit the above data

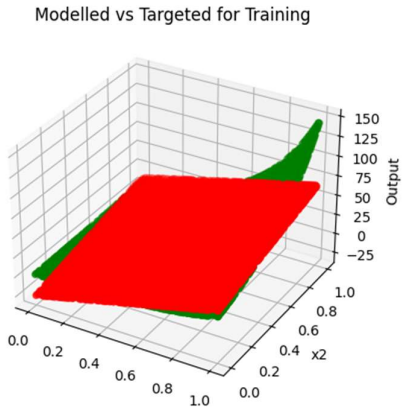


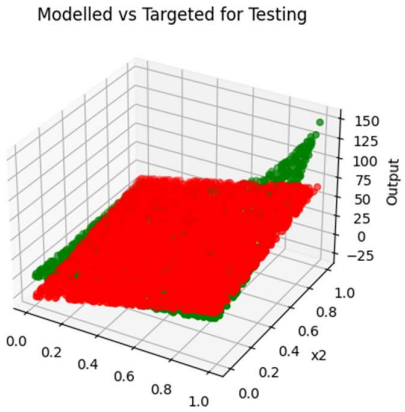
Plot of average error (y-axis) vs epochs (x-axis)



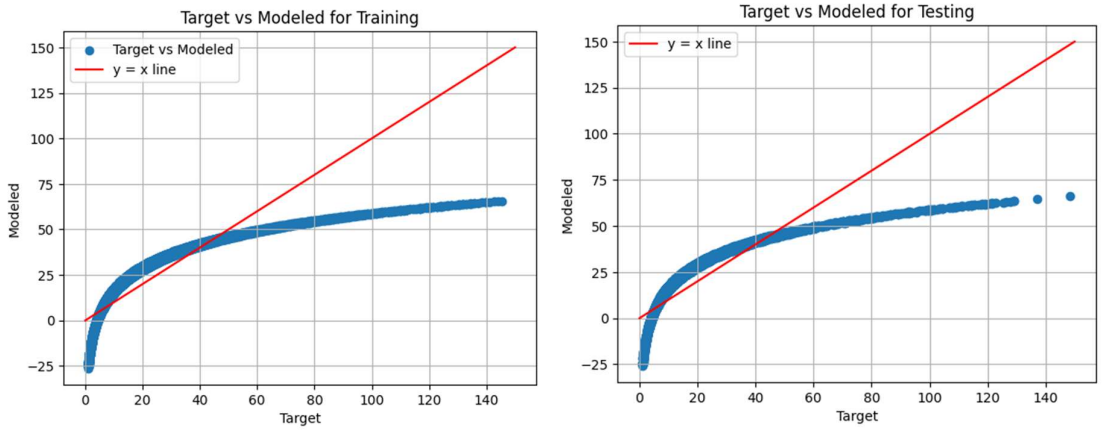
Plot of the values of mean squared error on training data and test data

Plots of model output and target output for training data and test data





Scatter plot for Training and Testing (Target o/p on x-axis and model o/p on y-axis)



Inference:

- We got a Line for 1-d data, which tries to fit the given data.
- We got a linear surface for 2-d data, which tries to fit the given data.
- Average error is plotted with respect to number of epochs and we can see that the error is getting reduced with increasing number of epochs.
- The scatter plot shows that at a target value, what is model value, a y = x line is drawn on this plot to see where Training and Modelled output values are same.
- For example (In 1-d scatter plot) , if we see the Scatter plot it is clearly visible that where the y = x line is cutting the plot , on the same values of Training or Testing and Model , just above graph is satisfying.