A Report on

CS671: Assignment 1

Classification and Regression using Perceptron

Team Members:

1. **Ananya Angra(S22025)**
2. **Shilpa Chandra(S22004)**
3. **Pushkar Kumar(S22038)**

Course Instructor

Dr. Dileep A. D.

Indian Institute of Technology Mandi

IIT Mandi Academic Year 2022-2023

Contents

List of Figures

Chapter 1

Perceptron

Perceptron model uses a single McCulloch-Pitts (MP) neuron that can be trained using the perceptron learning algorithm. A perceptron is basically a neuron with continuous activation function whereas a MP neuron is a neuron with threshold logic activation function.

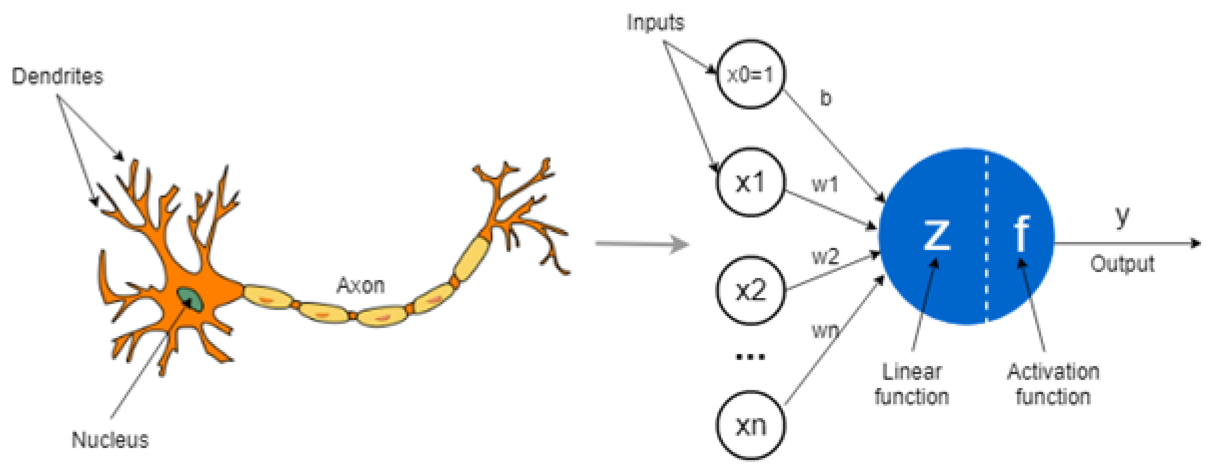


Figure 1.1: Comparing Natural Neuron and MP Neuron

The properties of the continuous activation function are:

- Monotonicity
- Continuity
- Differentiable: Differentiation operation can be performed
- Non-linearity

In Fig 1.1 perceptron is shown which is a neuron with sigmoidal activation function. There are 2 types of sigmoidal activation functions namely logistic function and tan hyperbolic function. The neuron takes a d dimensional input vector $\mathbf{x} = [\mathrm{x}_1,\mathrm{x}_2,......\mathrm{x}_\mathrm{d}]$ and weight vector $\mathbf{w} = [\mathrm{w}_1,\mathrm{w}_2,\ldots\mathrm{w}_\mathrm{d}]$. The neuron calculates the dot product of $\mathbf{w^T x}$ and sums with a bias value w_0. The sum $\mathbf{w^T x} + \mathbf{w_0}$ is given as input to the sigmoidal activation function. The sigmoidal activation function used here is a logistic activation function which produces output in the range [0,1]. Other activation functions are Binary Step, Linear Sigmoid, Tanh, ReLU, Leaky ReLU, Softmax etc.

1.1 Perceptron Learning Algorithm

The goal of the perceptron learning algorithm is to find the parameter vector $\mathbf{w} = [\mathrm{w}_1,\mathrm{w}_2,\ldots\mathrm{w}_\mathrm{d}]^\mathrm{T}$ such that the hyperplane is placed between the training data of the 2 classes so that the training error is minimum. The algorithm works by taking in a set of input features, each with an associated weight, and combining them into a weighted sum. If the weighted sum is above a threshold value, the perceptron outputs a positive label, otherwise it outputs a negative label.

The basic steps of the perceptron learning algorithm are:

1. Initialize the weights and the bias term to small random values.
2. For each input in the training set, compute the weighted sum of the input features and the bias term.
3. Apply the activation function (usually a step function or a sigmoid function) to the weighted sum to produce the output of the perceptron.
4. Compute the error between the predicted output and the actual output for the input.
5. Update the weights and bias term according to the error, using the following update rule:
    - new _weight = old_weight + learning_rate * error * input_feature where the learning rate is a hyperparameter that controls the step size of the weight and bias updates.
6. Repeat steps 2-5 for a fixed number of iterations or until the model converges on the training set.

Chapter 2

Classification Task

2.1 ONE-AGAINST-ONE APPROACH

In the context of the Perceptron algorithm, a one-against-one approach refers to the way the algorithm is used to solve a binary classification problem where there are only two classes taken at a time in consideration.

In a one-to-one approach, the Perceptron algorithm is trained to learn a single linear boundary that separates one class from the other. The algorithm is trained on pairs of examples, one from each class, and adjusts the weights of the linear boundary until it correctly classifies each example in the pair. This process is repeated for all pairs of examples until the algorithm converges and the linear boundary is found. Once the linear boundary is learned, the Perceptron algorithm can be used to classify new examples by computing the dot product of the input features with the learned weights of the linear boundary.

If the result is positive, the example is classified as belonging to one class, and if it is negative, it is classified as belonging to the other class. We are using a one against one approach to classify three classes by making a decision boundary between any two classes at a time and then merging all the boundaries to create a three class decision boundary.

2.2 Parameter Learning – Gradient Descent Method

Neuron with Sigmoidal Activation Function

1. Initialize $\mathbf{w}$ with random values
2. Choose single training example such as x_{ni}

3. Compute output of the neuron

$$s_{\mathrm{n}} = a_{\mathrm{n}} = w^{\mathrm{T}}\hat{x_{\mathrm{n}}}$$

4. Calculate instantaneous error

$$E_{\mathrm{n}} = \frac{1}{2}(y^{\mathrm{n}} - s^{\mathrm{n}})^2$$

5. Calculate change in weight

$$\Delta W = \eta\delta 0\hat{x_{\mathrm{n}}}$$

$$\delta 0 = (y^{\mathrm{n}} - s^{\mathrm{n}})\frac{df(a_{\mathrm{n}})}{d(a_{\mathrm{n}})}$$

   Here logistic neuron is used

6. Update the weight W

$$w = w + \Delta w$$

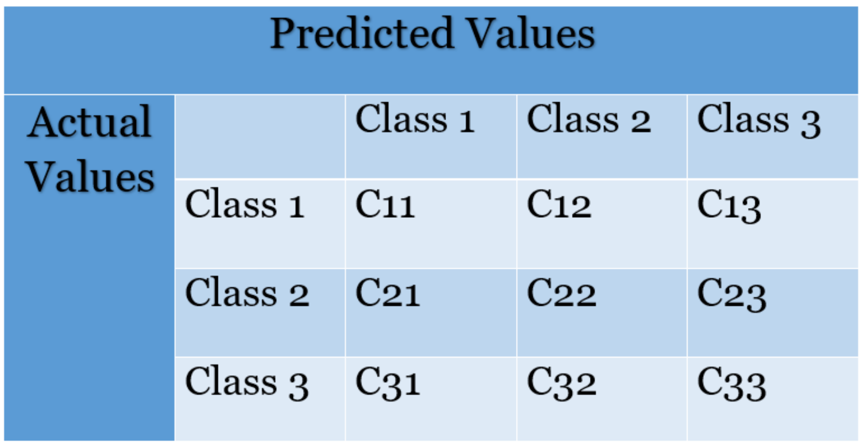
7. Repeat steps 2 and 6 till all the training examples are presented once (Epoch)

8. Compute the average error

$$E_{\mathrm{av}} = \frac{1}{N}\sum_{i=1}^{N} E_{\mathrm{n}}(w)$$

9. Repeat steps 2 to 8 till the fixed number of epochs

2.3 Confusion matrix and classification accuracy



	Predicted Values			
Actual Values		Class 1	Class 2	Class 3
	Class 1	C11	C12	C13
	Class 2	C21	C22	C23
	Class 3	C31	C32	C33

Accuracy : Accuracy is a measure of how well a machine learning model is able to correctly predict the outcome of a task. It is defined as the number of correct predictions made by the model divided by the total number of predictions. In other words, it represents the percentage of correctly predicted outcomes.

Precision: In machine learning, precision is a metric that measures the fraction of true positives among the instances that the model has classified as positive. It is a measure of the model's ability to accurately predict positive instances.

Recall: Total number of samples classified as positive class out of all the samples belonging to positive class.

F-measure: F-measure, also known as F1 score, is a measure of a test's accuracy that balances precision and recall. It is commonly used in machine learning and information retrieval to evaluate the performance of a classification model or a search algorithm. Here, c_{11} means : number of samples predicted as C_1 and actually belong to C_1

c_{12} means : number of samples predicted as C_1 and actually belong to C_2

$$Accuracy = \frac{c_{11} + c_{22} + c_{33}}{N} * 100$$

$$C_1 Precision = \frac{t_\mathrm{p}}{t_\mathrm{p} + f_\mathrm{p}}$$

Therefore,

$$Precision = \frac{c_1 pecision + c_2 pecision + c_3 pecision}{3} * 100$$

$$C_1 Recall = \frac{t_\mathrm{p}}{t_\mathrm{p} + f_\mathrm{n}}$$

where $\mathrm{t_p}$ and $\mathrm{f_n}$ are true positive and false positive for C_1

$$Recall = \frac{c_1 recall + c_2 recall + c_3 recall}{3} * 100$$

$$F1 - score = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

2.4 Linearly Separable Classes Results

The algorithm is tested for linearly separable data shown below:

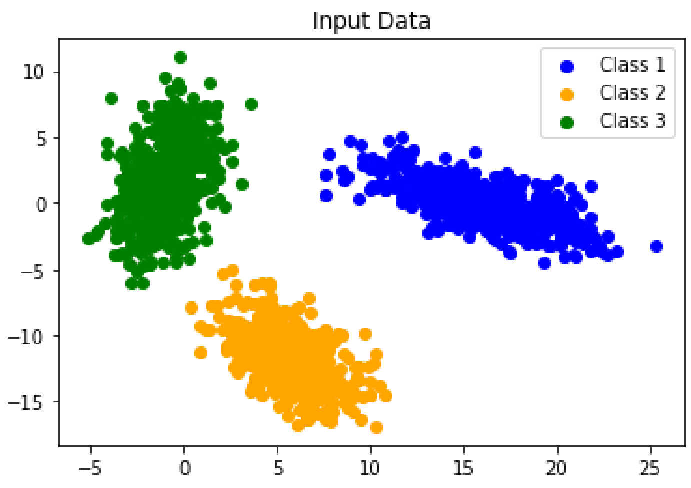


Figure 2.1: Input data for Linearly separable data

Plots of Decision boundary for one to one class at a time

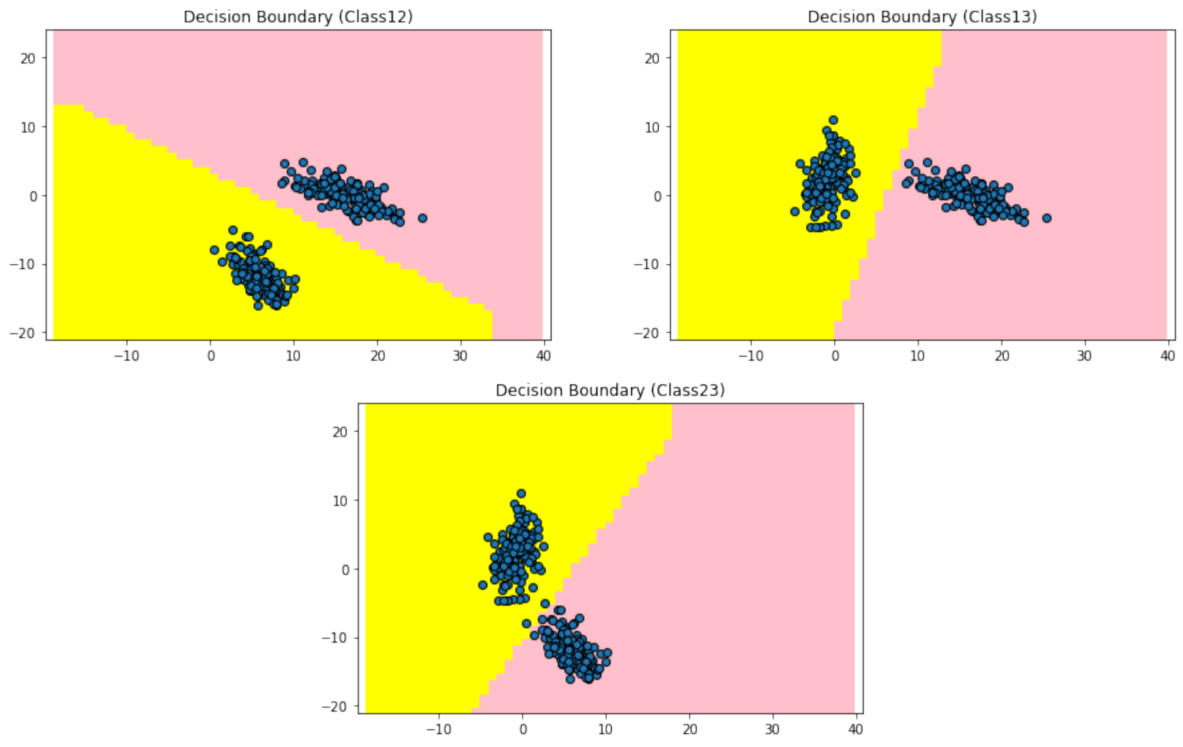


Figure 2.2: Decision boundary between 2 different classes

We have classified the data using the one against one approach and merged them together to form the 3 class classification decision boundary against the given input . Below is the merged plot.

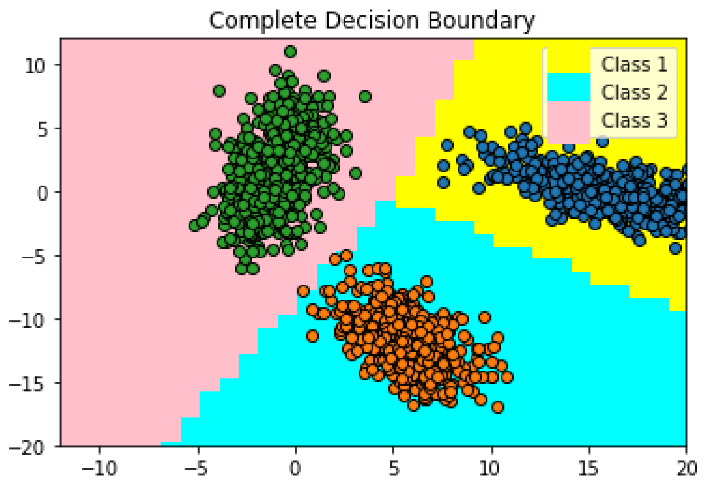


Figure 2.3: Decision boundary of all 3 classes

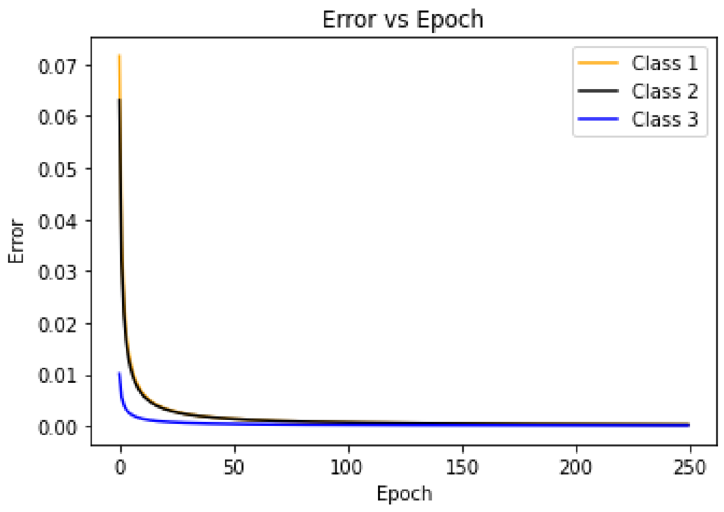
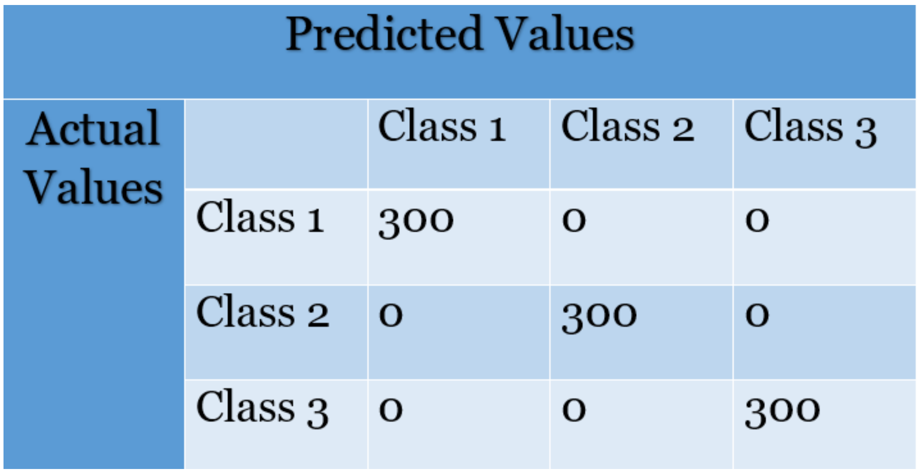


Figure 2.4: Plot of average error (y-axis) vs epochs (x-axis)

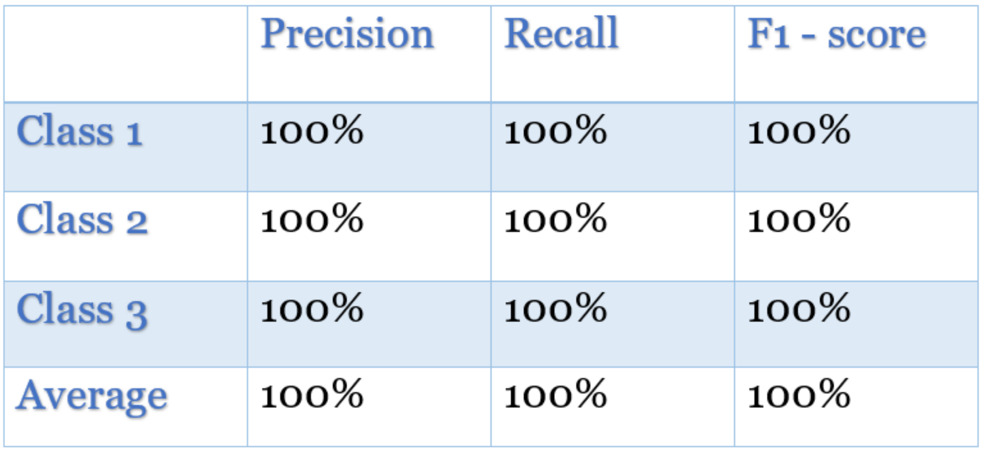
Plot of Average error vs number of epochs shows that as the number of epochs increases, the error decreases exponentially.

Results for Linearly separable data



		Predicted Values		
Actual Values		Class 1	Class 2	Class 3
	Class 1	300	0	0
	Class 2	0	300	0
	Class 3	0	0	300

Figure 2.5: Confusion Matrix of Linearly separable data



	Precision	Recall	F1 - score
Class 1	100%	100%	100%
Class 2	100%	100%	100%
Class 3	100%	100%	100%
Average	100%	100%	100%

Figure 2.6: Accuracy, Precision, Recall and F-1 score of Linearly separable data for the three classes

As the data is linearly separable one neuron is enough to train the data and get correct function approximation.

2.5 Non Linearly Separable Results

The algorithm is tested for linearly separable data shown below:

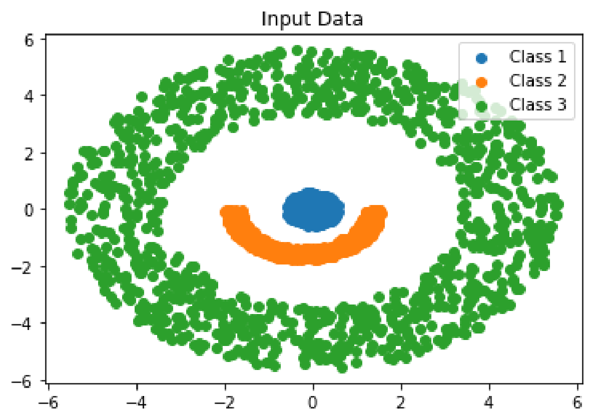


Figure 2.7: Input data for Non Linearly separable data

Plots of Decision boundary for one to one class at a time:

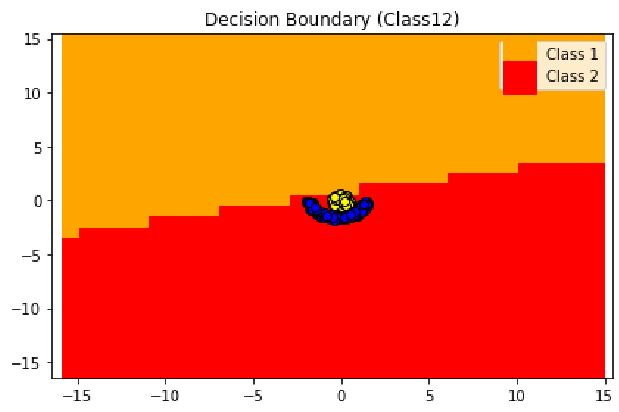


Figure 2.8: Output of neuron 1

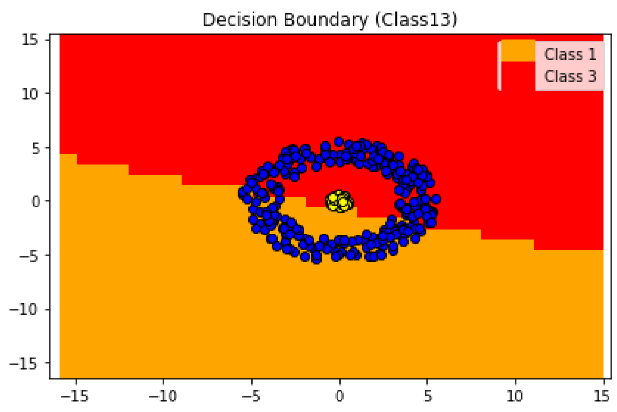


Figure 2.9: Output of neuron 2

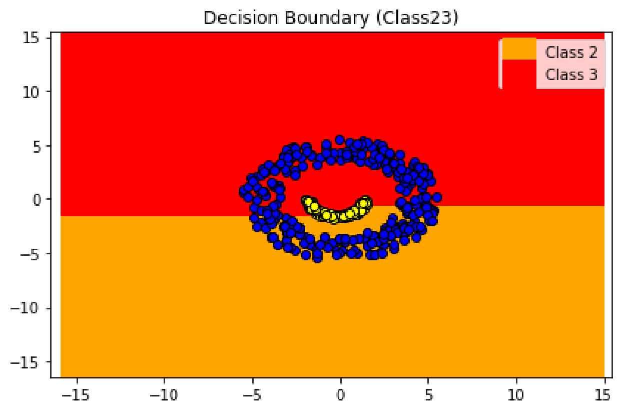


Figure 2.10: Decision boundary between 2 different classes at a time

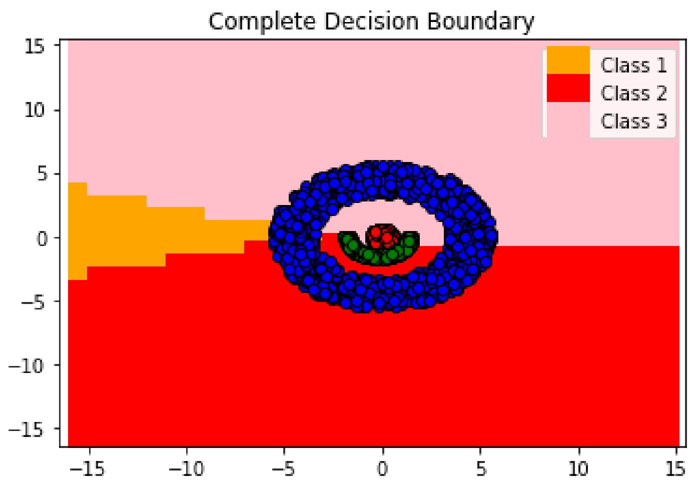


Figure 2.11: Decision boundary of all 3 classes

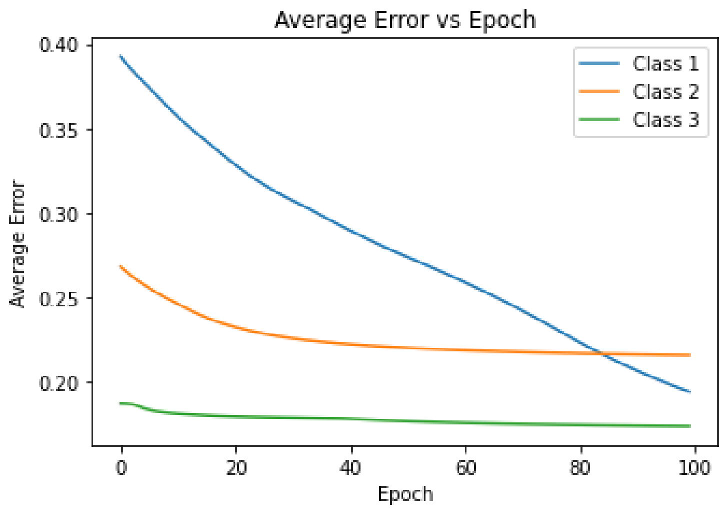
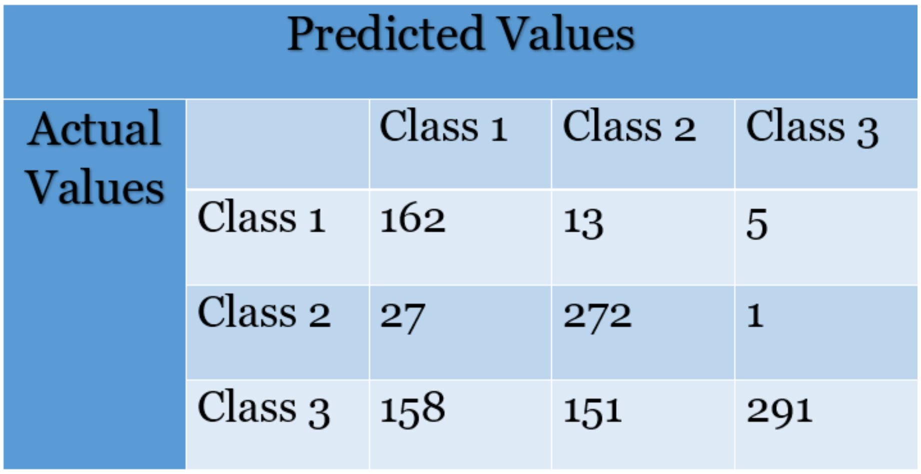


Figure 2.12: Plot of average error (y-axis) vs epochs (x-axis)

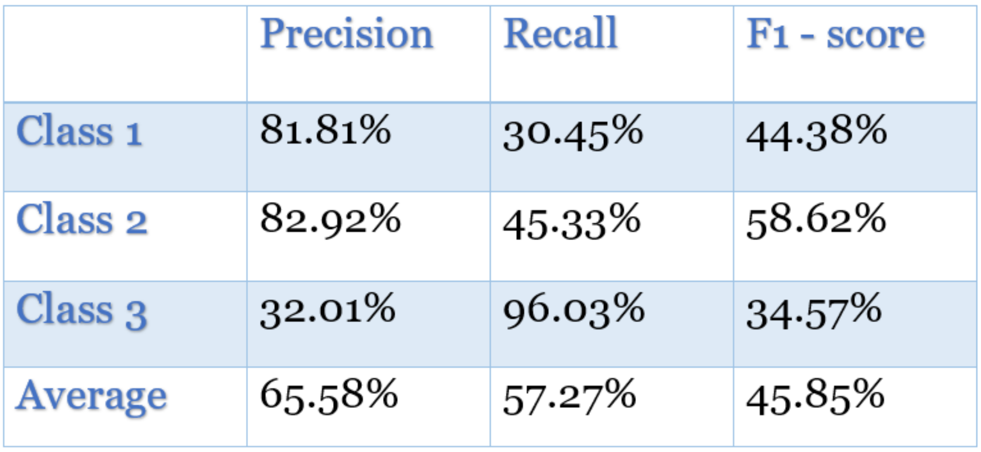
Plot of Average error vs number of epochs shows that as the number of epochs increases, the error need not necessarily decreases exponentially. This is beacause the dataset is non linear in nature and only a single neuron is used to train this dataset.
Therefore, single perceptron is not sufficient to create a boundary between any two classes due to non linear nature of the data

Results for Non Linearly separable data



	Predicted Values			
Actual Values		Class 1	Class 2	Class 3
	Class 1	162	13	5
	Class 2	27	272	1
	Class 3	158	151	291

Figure 2.13: Confusion Matrix of Non Linearly separable data



	Precision	Recall	F1 - score
Class 1	81.81%	30.45%	44.38%
Class 2	82.92%	45.33%	58.62%
Class 3	32.01%	96.03%	34.57%
Average	65.58%	57.27%	45.85%

Figure 2.14: Accuracy, Precision, Recall and F-1 score of Non linearly separable data for the three classes

As the data is non- linearly separable one perceptron is not enough to train the data and get correct function approximation. Multiple layers of neurons will help to stitch a non linear boundary required to correctly classify the non linear data

Chapter 3

Regression Task

Regression is a type of supervised learning technique that predicts values for a given set of inputs. Here, it approximates a function so that it can map an input to a specified output.
The input set of values are independent from each other, while the output values are dependent in nature. These output values lie on the same dimensional plane that also contains the set of inputs.

The activation function used here is linear activation function as the output values need not be bounded and can lie on any plane as represented by the input data.

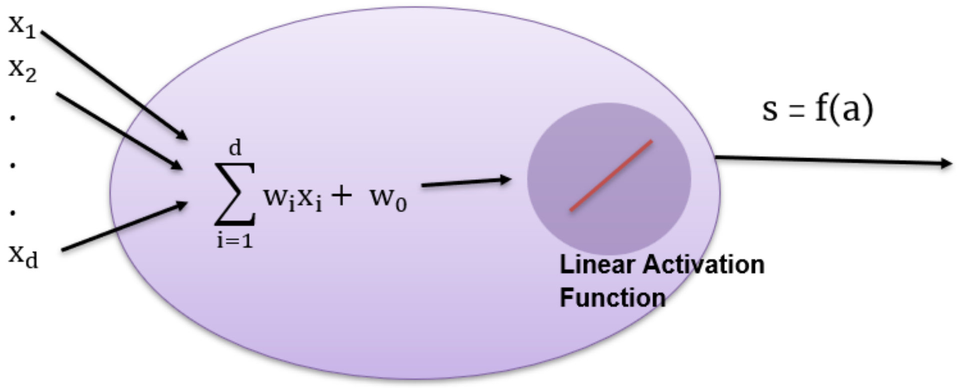


Figure 3.1: Regression using Neuron with Linear Activation Function

3.1 Parameter Learning – Gradient Descent Method

1. Initialize w with random values
2. Choose single training example such as x_{ni}

3. Compute output of the neuron

$$s_{\mathrm{n}} = a_{\mathrm{n}} = w^{\mathrm{T}}\hat{x_{\mathrm{n}}}$$

4. Calculate instantaneous error

$$E_{\mathrm{n}} = \frac{1}{2}(y^{\mathrm{n}} - s^{\mathrm{n}})^2$$

5. Calculate change in weight

$$\Delta W = \eta(y^{\mathrm{n}} - s^{\mathrm{n}})\hat{x_{\mathrm{n}}}$$

   Linear activation function is used

6. Update the weight W

$$w = w + \Delta w$$

7. Repeat steps 2 and 6 till all the training examples are presented once (Epoch)

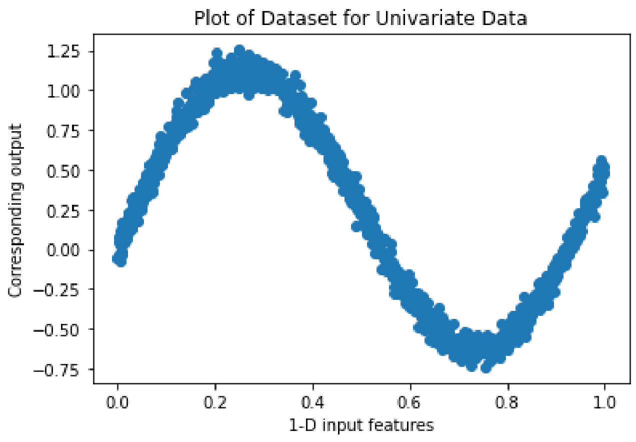
8. Compute the average error

$$E_{\mathrm{av}} = \frac{1}{N}\sum_{i=1}^{N} E_{\mathrm{n}}(w)$$

9. Repeat steps 2 to 8 till the fixed number of epochs

3.2 Univariate Data Results

The below plot show the input dataset for univariate case



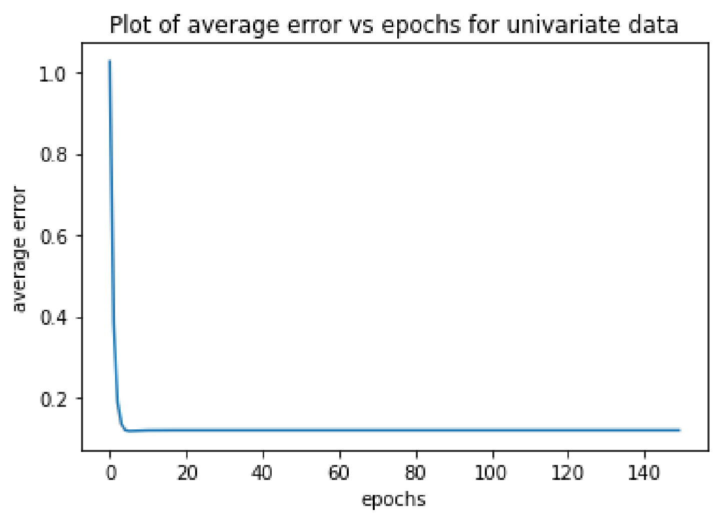


Figure 3.2: Plot of average error (y-axis) vs epochs (x-axis)

In Fig 3.2 the average error decreases exponentially as the number of epochs increases.

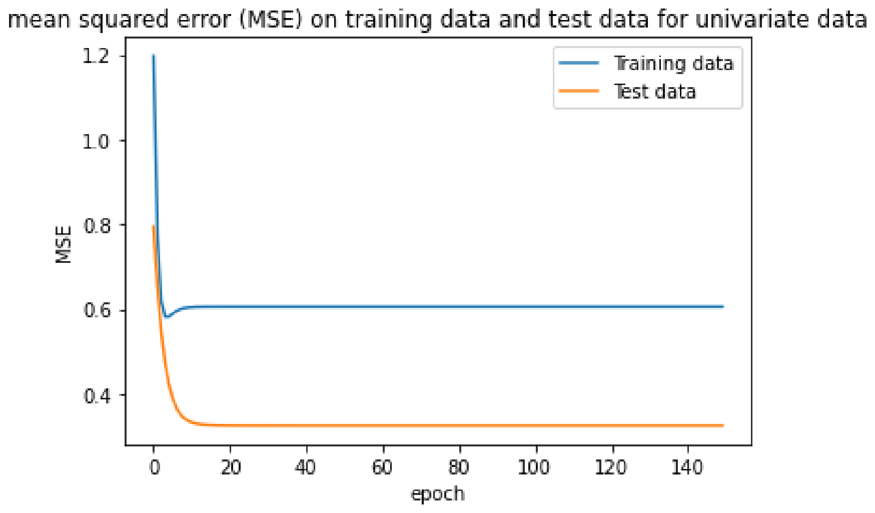


Figure 3.3: Plot of the values of mean squared error (MSE) on training data and test data

The prediction accuracy of regression model is measured in terms of root mean squared error in Fig 3.3

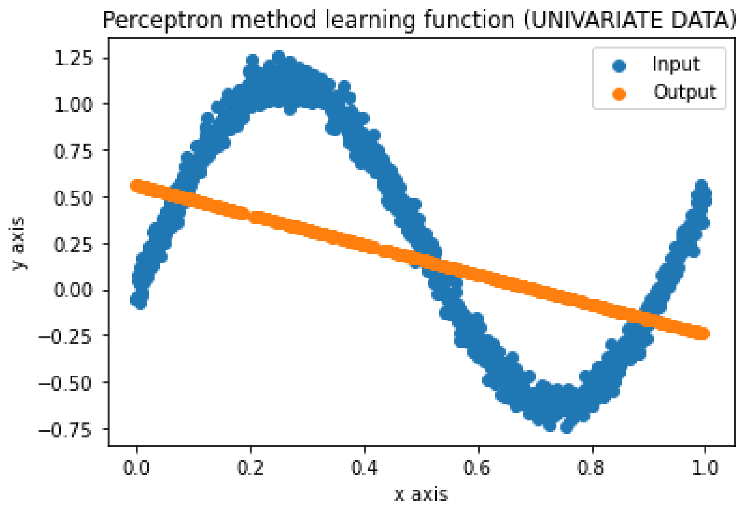


Figure 3.4: The learnt function form perceptron model

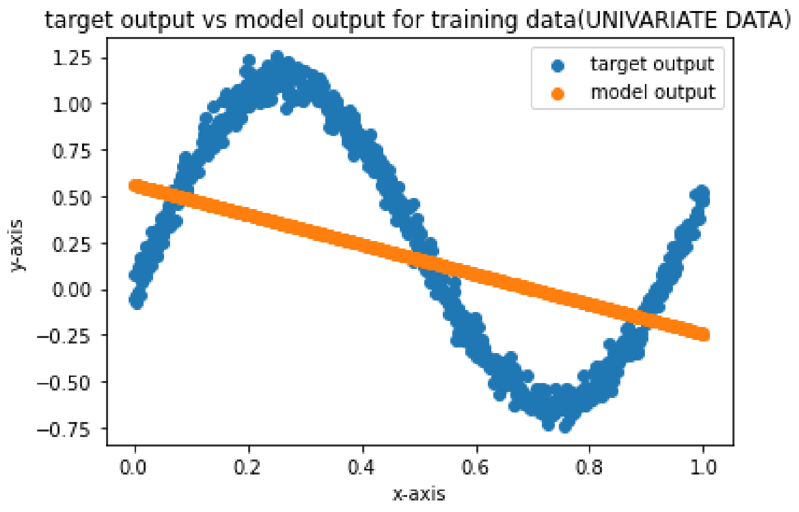


Figure 3.5: Scatter plot with target output on x-axis and model output on y-axis for training data

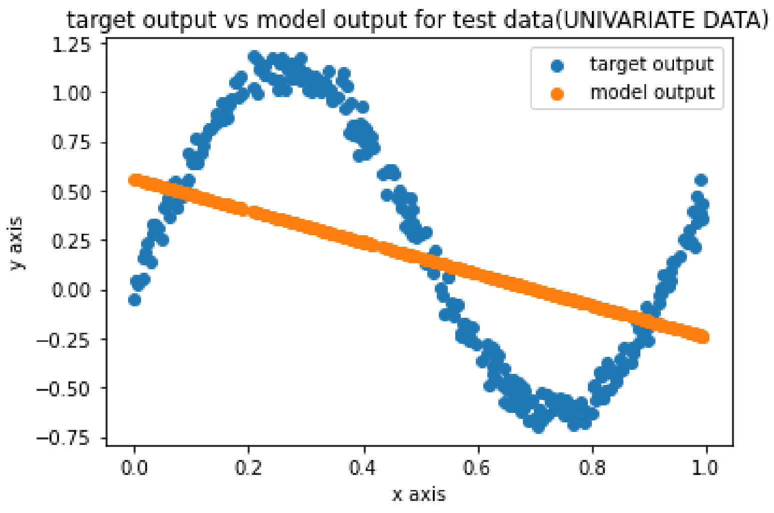


Figure 3.6: Scatter plot with target output on x-axis and model output on y-axis for test data

In Fig 3.4 the function approximates a linear line for the given nonlinear data as a single neuron is used. For smooth curved line to fit our non-linear data more neuron will be a better approach.

3.3 Bivariate Data Results

The below plot show the input dataset for Bivariate case which is nonlinear in nature

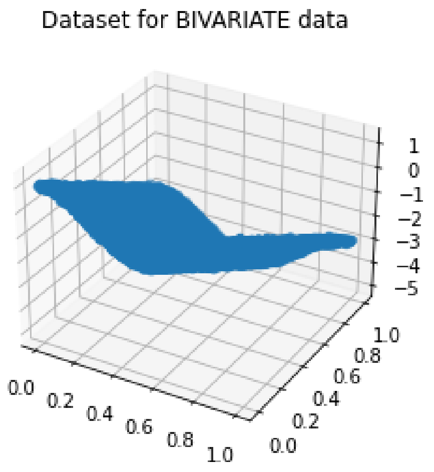


Figure 3.7: Plot of dataset for Bivariate Data

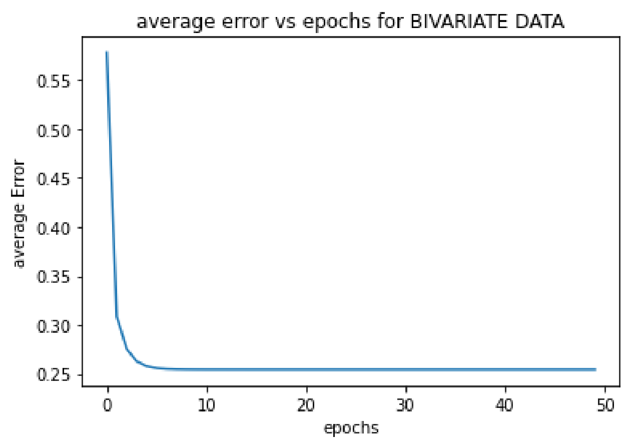


Figure 3.8: Plot of average error (y-axis) vs epochs (x-axis)

In Fig 3.8 the average error decreases exponentially as the number of epochs increases.

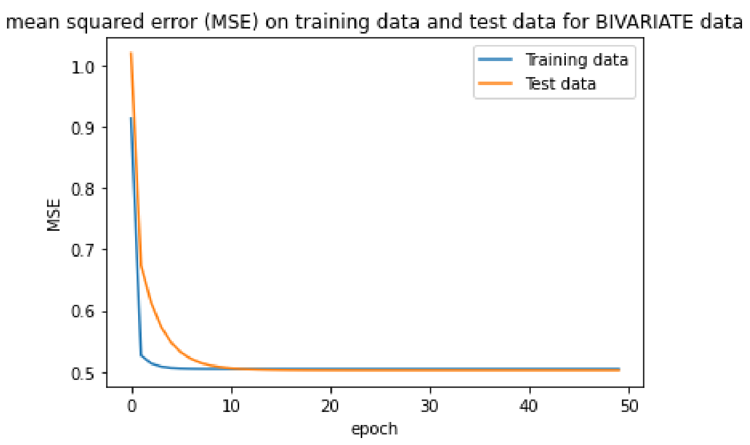


Figure 3.9: Plot of the values of mean squared error (MSE) on training data and test data

The prediction accuracy of regression model is measured in terms of root mean squared error in Fig 3.9

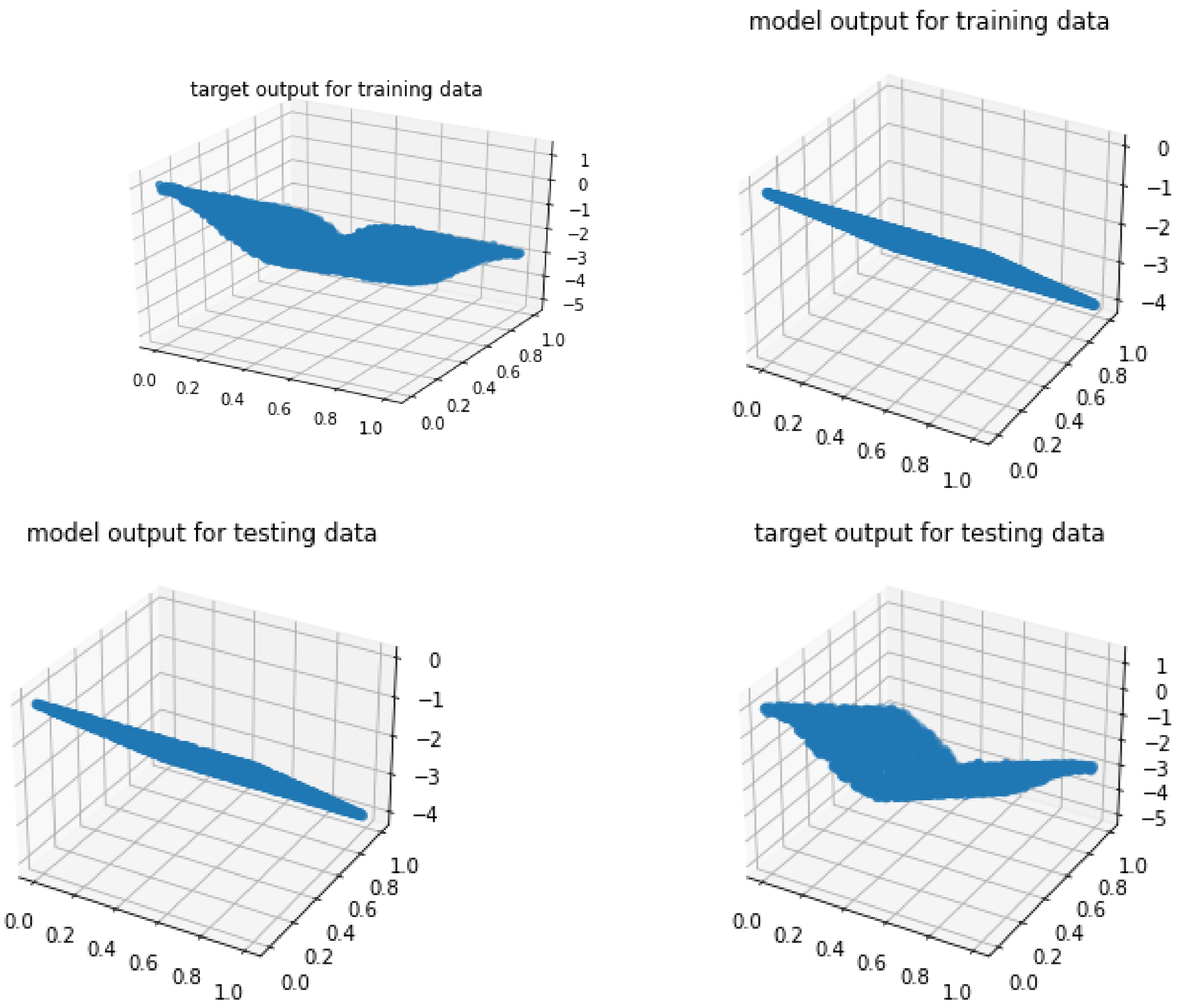


Figure 3.10: Target vs Model output of training and testing data

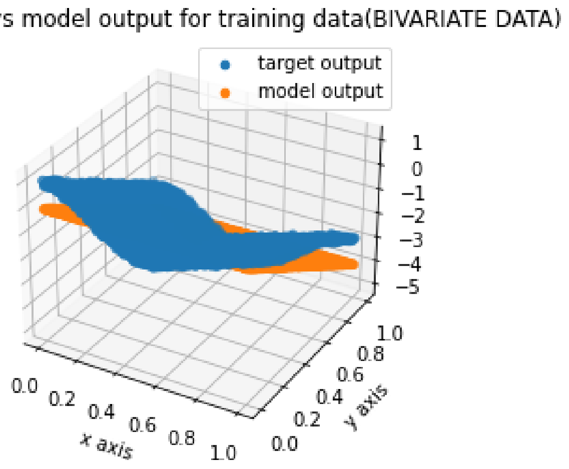


Figure 3.11: Scatter plot with target output on x-axis and model output on y-axis for training data

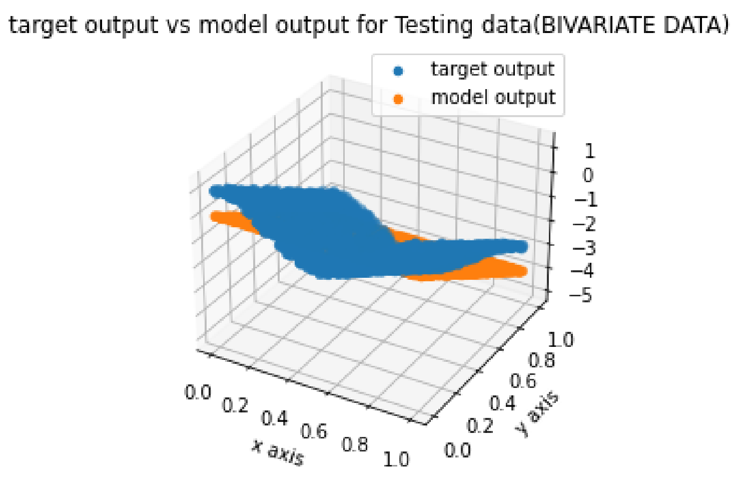


Figure 3.12: Scatter plot with target output on x-axis and model output on y-axis for test data

Again for the model output for test data and training data(as seen in figures below) gives a linear surface as our data is trained on a single neuron. For a smoother curve multiple layers of neuron are required which together stitch together to form a curved surface