Report

on

Deep Learning and Applications

(CS671)

Assignment 1

Submitted by:

GROUP - 9

Shashank Kapoor (S22022)

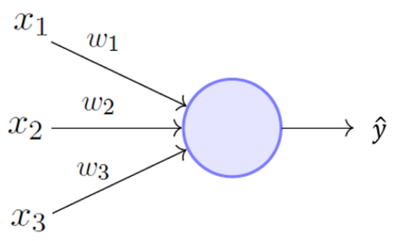
Shubham Patwar (T22108)

Syed Rizwan Ali Quadri (T22113)

Assignment Problem Statement

In this assignment , the key objective is to classify linearly separable data and non linearly separable data by training the single neuron perceptron and case study accordingly by changing the hyperparameters.

In this project we use perceptron models with gradient descent approach to update the weights for classification tasks and regression tasks.



$$a = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$\hat{y} = f(a)$$

Fig. Perceptron model

1).Classification Task

In classification, we are provided with the linearly separable data and non linearly separable data (fig.). In linearly separable data we have been provided with the three classes consisting, two features for each class respectively. In Non-linearly separable data also we have been provided with the three classes each consisting of two features.

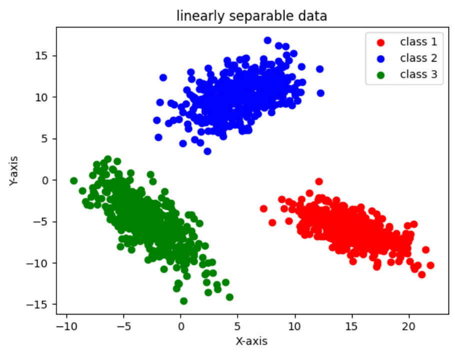


Fig. Linearly separable data for each class

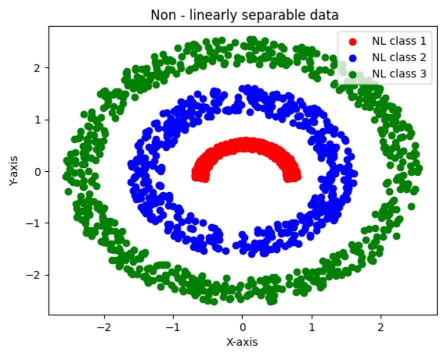


Fig. Non linearly separable data for each class

In each class of linearly separable data we have been provided with the 500 data points in which we have to take 70% of the data for training and 30% for testing and in non linearly separable data also we have been provided with the 500 data points out of which we have to take 70% of the data for training and 30% for testing.

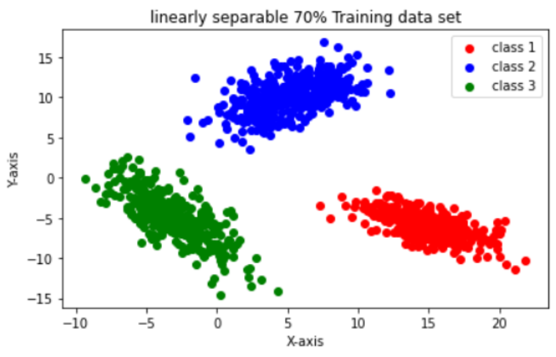


Fig. Linearly separable data for training

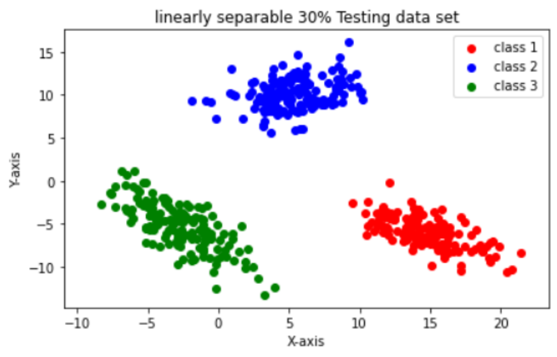


Fig. Linearly separable data for testing.

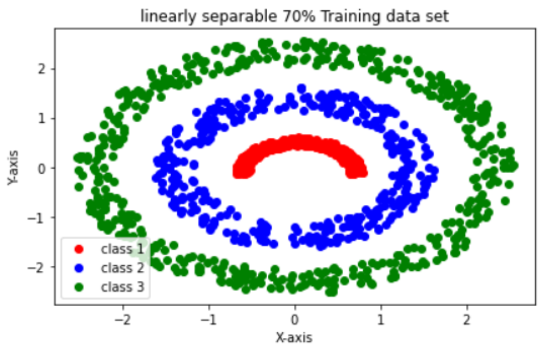


Fig. Non Linearly separable data for training

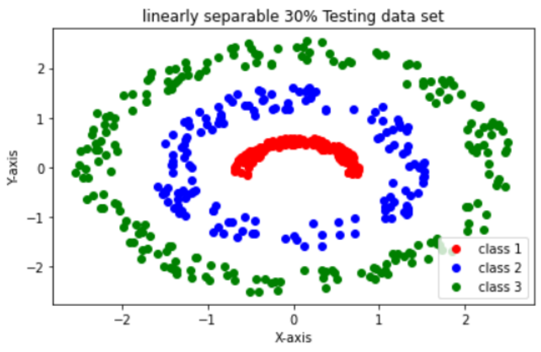


Fig. Non Linearly separable data for testing.

1.1) Training

1. Linearly separable data

After we have the data for training, Now we need to train the perceptron so that it classifies the data to their respective classes. In order to go for the "one-against-one" approach used we need to make three perceptrons for each class classification against another.(fig).

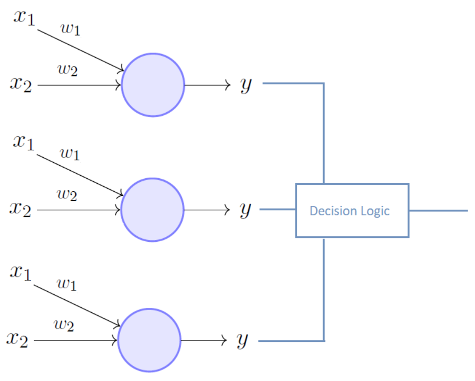


Fig. Three neurons to classify the three classes.

Logic

In the first perceptron we are able to classify between class1 vs class2 and in the second perceptron we are able to classify between class 1 vs class 3 and in the third perceptron we are able to classify between class 2 vs class 3. In each perceptron we are using a sigmoidal activation function to get the predicted output and thus calculating the instantaneous error which further helps in updating the weights and average error for each epoch. We also setted the learning rate to 0.05 to train our perceptron and perceptron to converge.

Class1 vs Class2

- No. of epochs : 50
- learning Factor : 0.05
- Weights: [0.4,0.4,0.4]
- Training Data : 70% (350 data points)
- After Weights : [0.21, -0.87,-0.05]

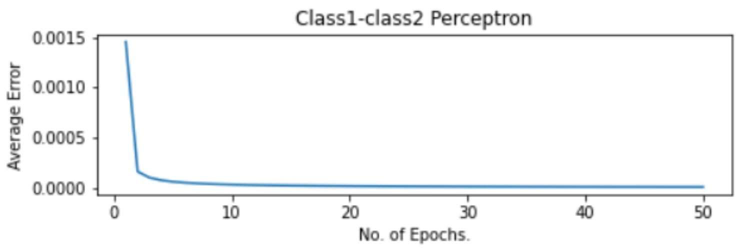


Fig. Average Error vs Epochs for class1-class2 perceptron.

Observation: In class 1 vs class 2, It is clearly seen that it converges very quickly then the other one against one. There is the steep slope from that we can infer that there is a rapid change in the weights to the optimal weights.

Class1 vs Class3

- No. of epochs : 50
- learning Factor : 0.05
- Weights: [0.4,0.4,0.4]
- Training Data : 70% (350 data points)
- After Weights : [1.04, 0.89, 0.12]

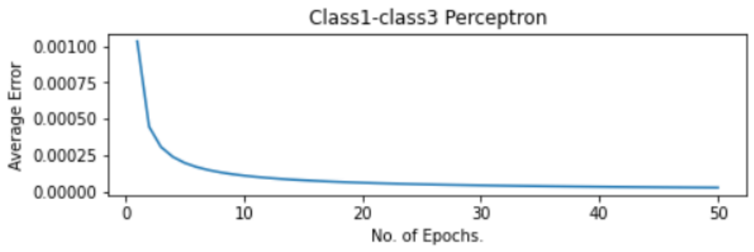


Fig. Average Error vs Epochs for class1-class3 perceptron.

Observation: In class 1 vs class 3, It is seen that it converges faster than class2 vs class3. There is the monotonically decreasing form from which we can infer that there is a change in the weights to the optimal weights.

Class2 vs Class3

- No. of epochs : 50
- learning Factor : 0.05
- Weights: [0.4,0.4,0.4]
- Training Data : 70% (350 data points)
- After Weights : [1.30, 1.14, 0.84]

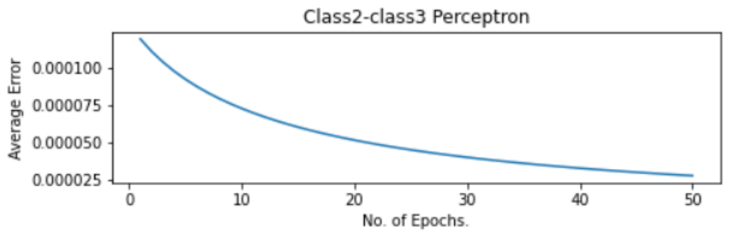


Fig. Average Error vs Epochs for class2-class3 perceptron.

Observation: In class 2 vs class 3, It is seen that it converges slowly comparatively than the others. There is the monotonically decreasing form from which we can infer that there is a change in the weights slowly to the optimal weights and hence it needs more number of epochs to converge.

Testing

After the training is done for each perceptron and getting the updated weights. Now we have test trained perceptrons for the 70 % of the trained as test data for each class.

Class1 vs Class2

- Testing Data : 70% of class 1 [350 data Points]
  70% of class 2 [350 data Points]

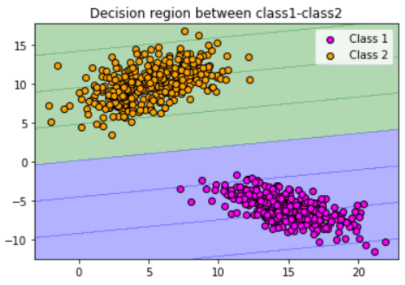


Fig. Decision boundary between Class1 vs Class2

Observation: class1 vs class 2 is clearly separable as we can see in Fig. thus the single perceptron can easily separate linearly separable data of 2 classes.

Class1 vs Class3

- Testing Data : 70% of class 1 [350 data Points]
  70% of class 3 [350 data Points]

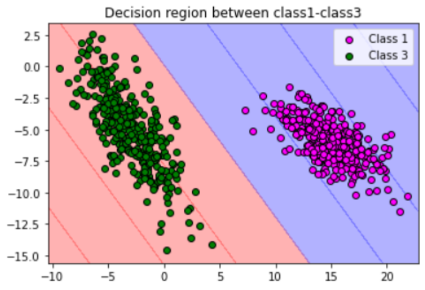


Fig. Decision boundary between Class1 vs Class3

Observation:class1 vs class 2 is clearly separable as we can see in Fig. thus the single perceptron can easily separate linearly separable data of 2 classes.

Class2 vs Class3

- Testing Data : 70% of class 2 [350 data Points]
  70% of class 3 [350 data Points]

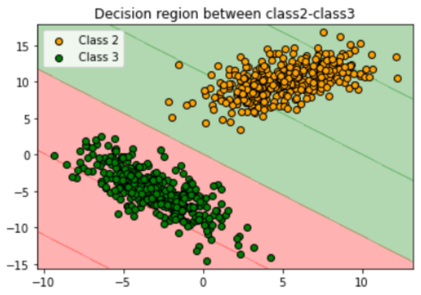


Fig. Decision boundary between Class2 vs Class3

Observation:class2 vs class 3 is clearly separable as we can see in Fig. thus the single perceptron can easily separate linearly separable data of 2 classes.

Now, we also find the decision region between each class for the by appending the 70% data for the all classes total of the 1050 data points,350 from each class.

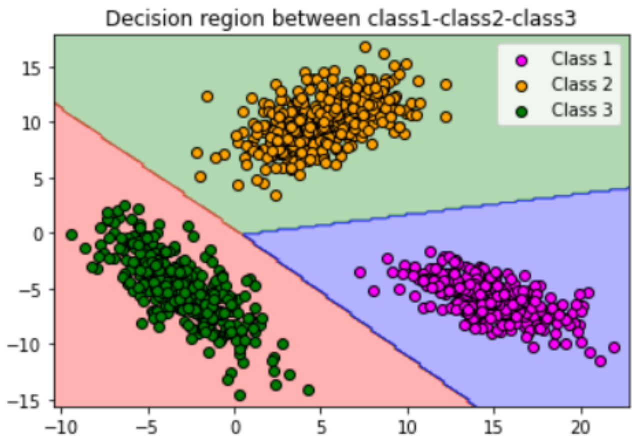


Fig. Decision boundary between Class1, Class2 and Class3

2. Non Linearly separable data

Logic

In Non Linearly separable data, in the first perceptron we wanted to classify between class1 vs class2 and in the second perceptron we wanted to classify between class 1 vs class 3 and in the third perceptron we wanted to classify between class 2 vs class 3. In each perceptron we are using a sigmoidal activation function to get the predicted output and thus calculating the instantaneous error which further helps in updating the weights and average error for each epoch.

Class1 vs Class2

● No. of epochs :	50
● learning Factor :	0.02
● Weights:	[0.4,0.4,0.4]
● Training Data :	70% (350 data points)
● After Weights	[0.12, 0.62, -0.01]

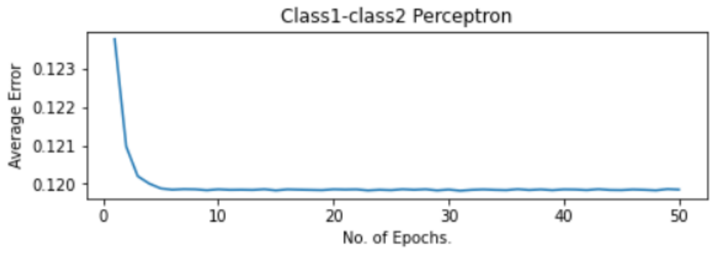


Fig. Average Error vs Epochs for class1-class2 perceptron.

Observation: In class 1 vs class 2, It is seen that it converges faster than class2 vs class3. There is the monotonically decreasing form from which we can infer that there is a change in the weights to the optimal weights.

Class1 vs Class3

● No. of epochs :	50
● learning Factor :	0.02
● Weights:	[0.4,0.4,0.4]
● Training Data :	70% (350 data points)
● After Weights :	[0.05, 0.10 , -0.35]

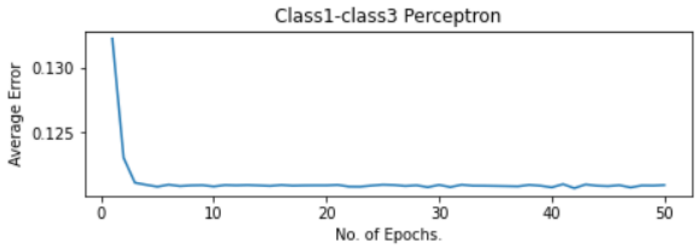


Fig. Average Error vs Epochs for class1-class3 perceptron.

Observation: In class 1 vs class 3, It is seen that it converges faster than class2 vs class3. There is the monotonically decreasing form from which we can infer that there is a change in the weights to the optimal weights.

Class2 vs Class3

- No. of epochs : 50
- learning Factor : 0.02
- Weights: [0.4,0.4,0.4]
- Training Data : 70% (350 data points)
- After Weights : [0.01, -0.01, -0.32]

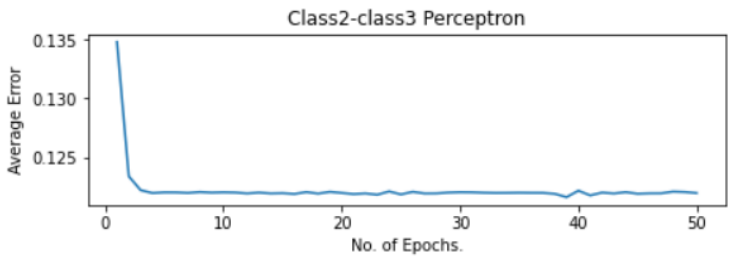


Fig. Average Error vs Epochs for class2-class3 perceptron.

Observation: In class 2 vs class 3, It is seen that it converges faster than class2 vs class3. There is the monotonically decreasing form from which we can infer that there is a change in the weights to the optimal weights.

Testing

After the training is done for each perceptron and getting the updated weights. Now we have test trained perceptron for the 70 % of the trained as test data for each class.

Class1 vs Class2

- Testing Data : 70% of class 1 [350 data Points]
  70% of class 2 [350 data Points]

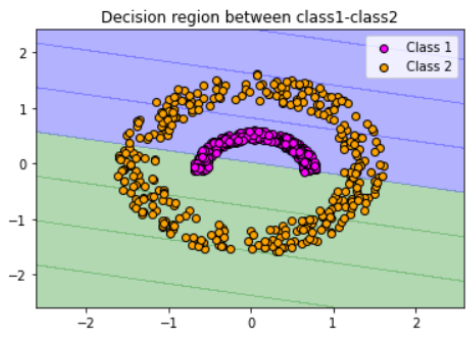


Fig. Decision boundary between Class1 vs Class2

Observation: class1 vs class 2 is non linearly separable as we can see in Fig. thus the single perceptron cannot separate non linearly separable data of 2 classes.

Class1 vs Class3

- Testing Data : 70% of class 1 [350 data Points]
  70% of class 3 [350 data Points]

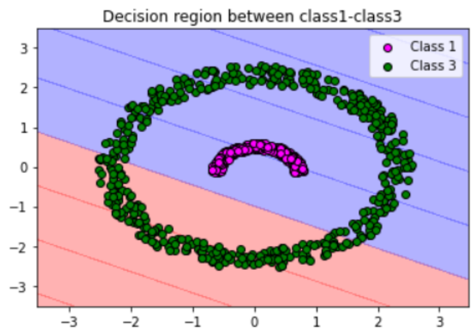


Fig. Decision boundary between Class1 vs Class3

Observation: class1 vs class 3 is non linearly separable as we can see in Fig. thus the single perceptron cannot separate non linearly separable data of 2 classes.

Class2 vs Class3

- Testing Data : 70% of class 2 [350 data Points]
  70% of class 3 [350 data Points]

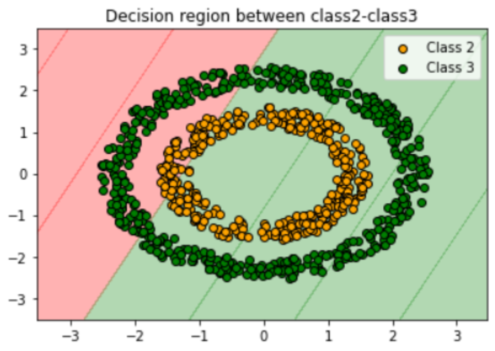


Fig. Decision boundary between Class2 vs Class3

Observation:class2 vs class 2 is non linearly separable as we can see in Fig. thus the single perceptron cannot separate non linearly separable data of 2 classes.

Now, we also find the decision region between each class for the by appending the 70% data for the all classes total of the 1190 data points, 350 from each class.

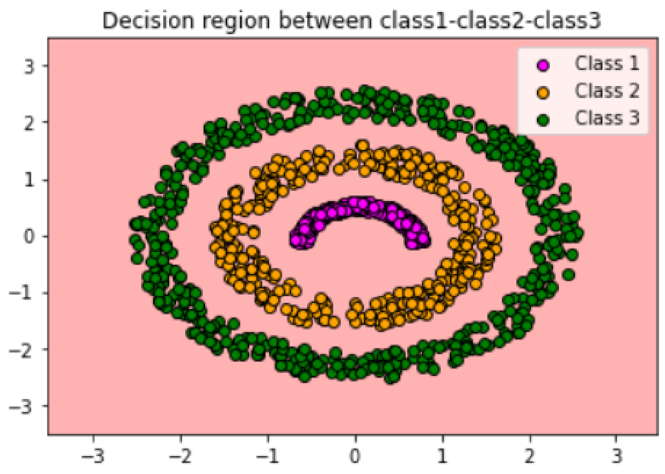


Fig. Decision boundary between Class1, Class2 and Class3

Inference Points

The main objective of this assignment is to evaluate the performance of perceptron on different types of datasets and to compare the classification accuracy achieved on linearly separable and non-linearly separable datasets. We found that the perceptron performs well on linearly separable datasets but struggles with non-linearly separable datasets.

The gradient descent method is an effective way to train the weights of the perceptron, and finding the optimal learning rate is crucial for achieving good accuracy.

2.Regression Task

In Regression, we are provided with two datasets, the 1-dimensional (Univariate) input data and 2-dimensional (Bivariate) input data. In Univariate data we have been provided with the single input and its corresponding Actual Value (ground truth). In Bivariate data also we have been provided two inputs and its corresponding value (ground truth).

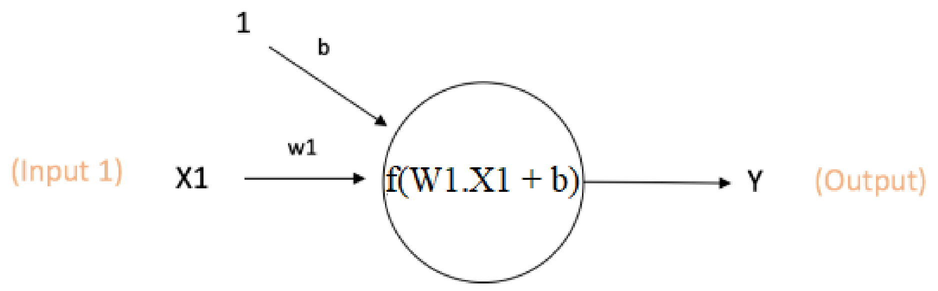
The implementation of the Perceptron with a linear activation function for regression tasks on two different datasets. We will be using the gradient descent method for the perceptron learning algorithm. The datasets are divided into 70% training data and 30% test data for both univariate and bivariate datasets.

Perceptron Model:

The Perceptron is a simple feedforward neural network. It consists of a single neuron that is connected to the input features. Neuron receives inputs from the input layer and produces an activation value based on a linear combination of these inputs with respective weights. The activation value is then passed through the linear activation function to produce the final output of the perceptron.

The Perceptron model used in this report has a linear activation function. This means that the output of the neuron is simply the linear combination of the inputs. The Perceptron learning algorithm uses gradient descent to update the weights of the respective inputs so that it can better predict the dependent variable.

2.1 Dataset 1 - Univariate Input Data:



Output of neuron = Y= f(W1.X1 + b)

Fig. Perceptron model with linear activation function (for univariate dataset)

The first dataset is a univariate dataset, meaning it only has one input feature. The dataset contains a single column of data that represents the independent variable, which will be used to predict the dependent variable. The dependent variable is continuous, meaning it can take on any value within a certain range.

The dataset is divided into training and test data, with 70% of the data used for training and the remaining 30% for testing. The perceptron model is then trained on the training data using the gradient descent method for the perceptron learning algorithm. The model is then tested on the test data to evaluate its performance.

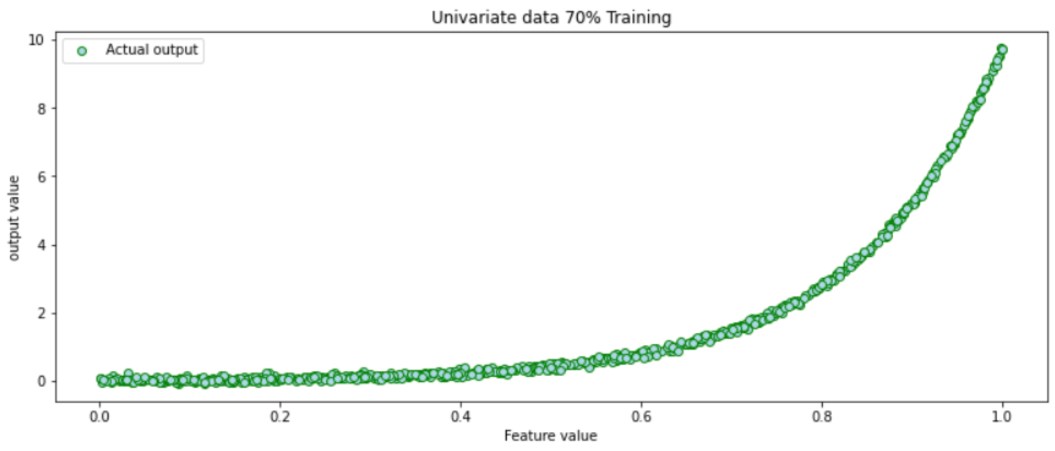


Fig. Univariate data for training

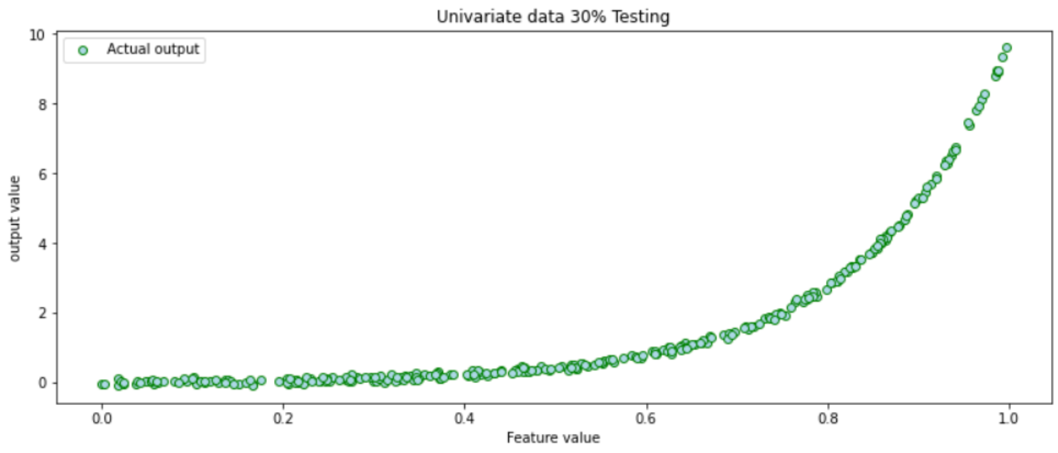


Fig. Univariate data for testing

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

In the perceptron, we are training the model for the 100 epochs for the 70 % of the given univariate data. In which first we are calculating the instantaneous error for each input in a single epoch. After each epoch the average error is calculated and is being plotted.(fig.)

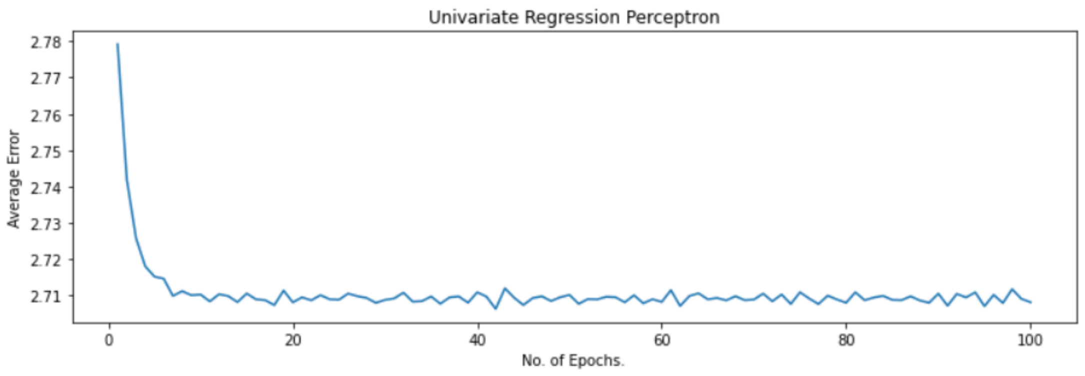
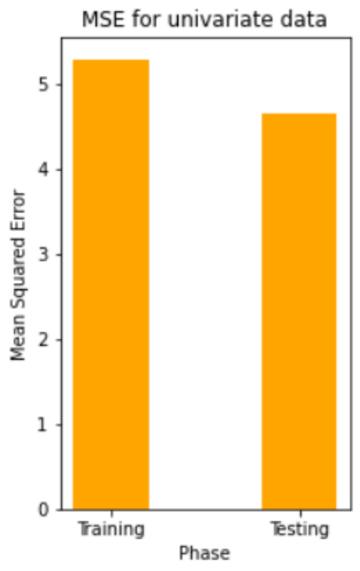


Fig. Average error for each epoch while training

2.1.2. Plot of Mean Squared Error for univariate data

In Univariate, 70% training data and the instantaneous error generated from it, is being used to calculate the Mean Squared Error. The same is also done for the remaining 30% testing data.

$$E_{\text{MSE}} = \frac{1}{N_t}\sum_{n=1}^{N_t}\left(\hat{y}_n - y_n\right)^2$$



2.1.2. Plot of Model and Actual output for univariate data

In univariate, we have plotted the actual output and model output for both training and testing, which contains 700 and 300 data points respectively.

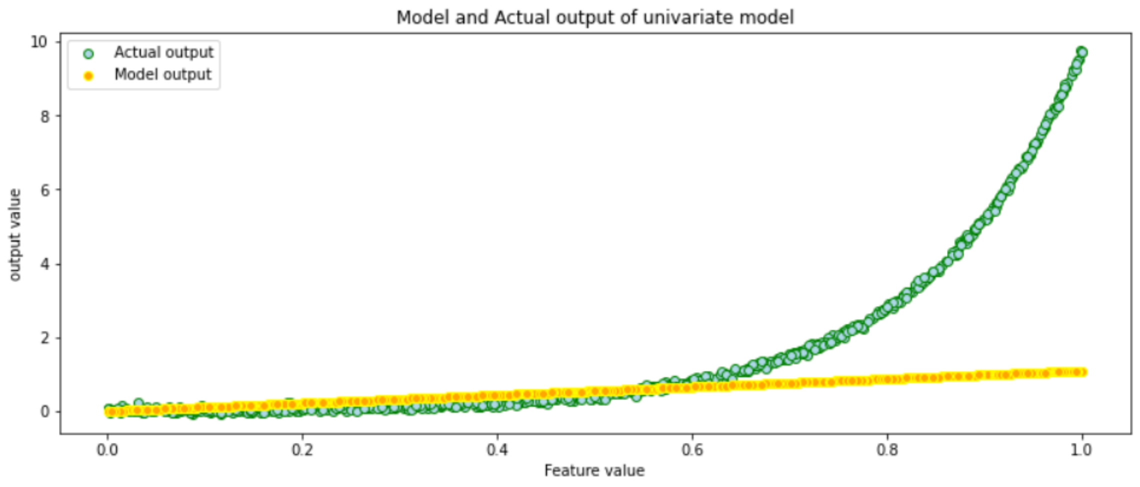


Fig. Plot between Model and Actual output in training phase

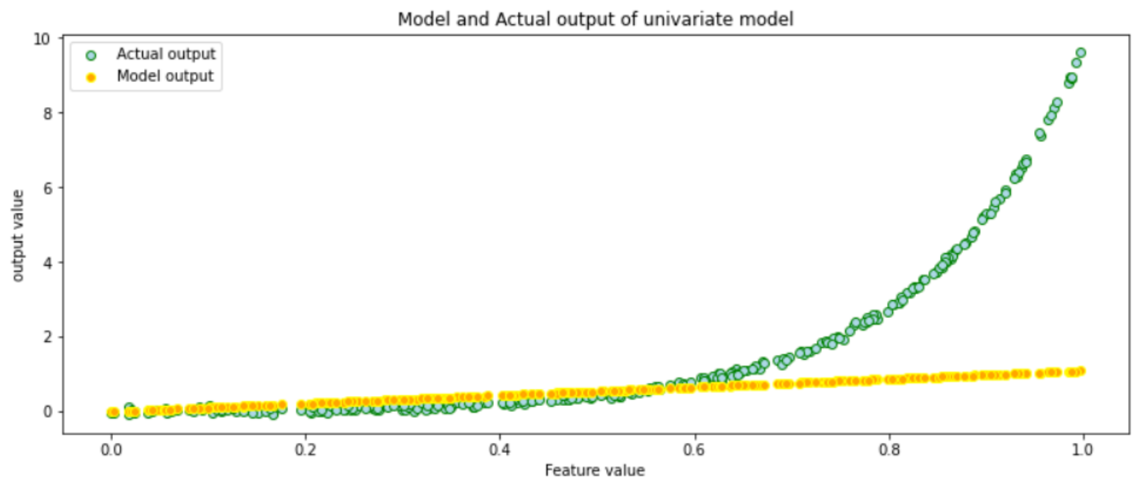


Fig. Plot between Model and Actual output in testing phase

2.1.3. Plot between Model and Actual output for univariate data

In this plot we have plotted against the training data and testing data respectively which contains 700 and 300 data points respectively.

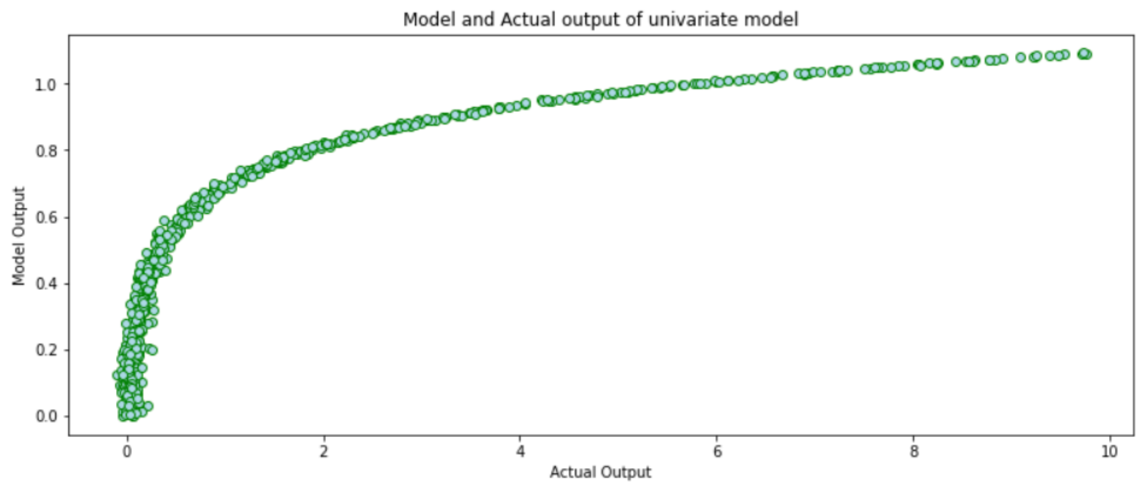


Fig. Plot between Model and Actual output in testing phase

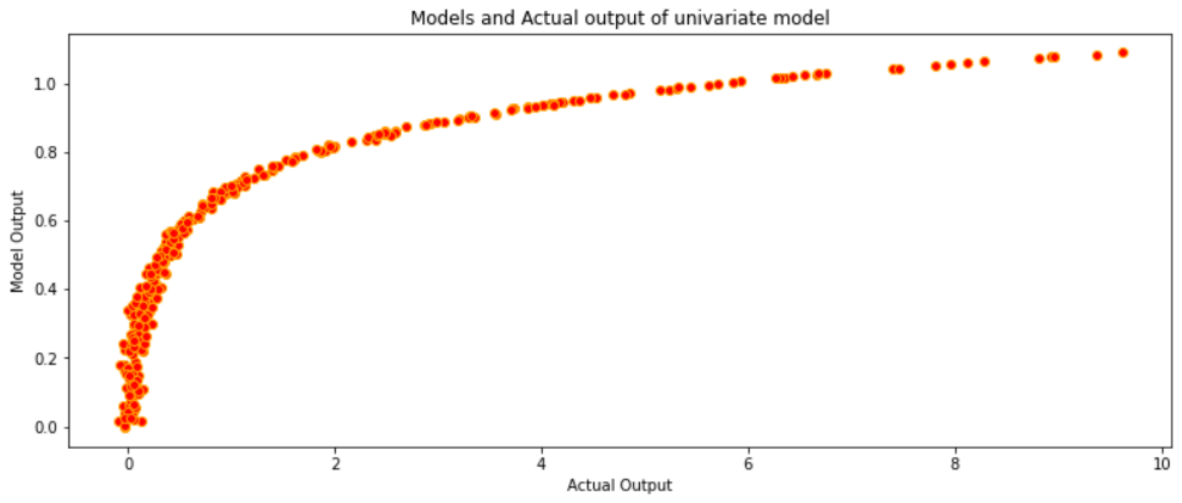


Fig. Plot between Model and Actual output in testing phase

2.2. Dataset 2 - Bivariate Input Data:

The second dataset is a bivariate dataset, meaning it has two input features. The dataset contains two columns of data that represent the independent variables, which will be used to predict the dependent variable. The dependent variable is continuous, meaning it can take on any value within a certain range.

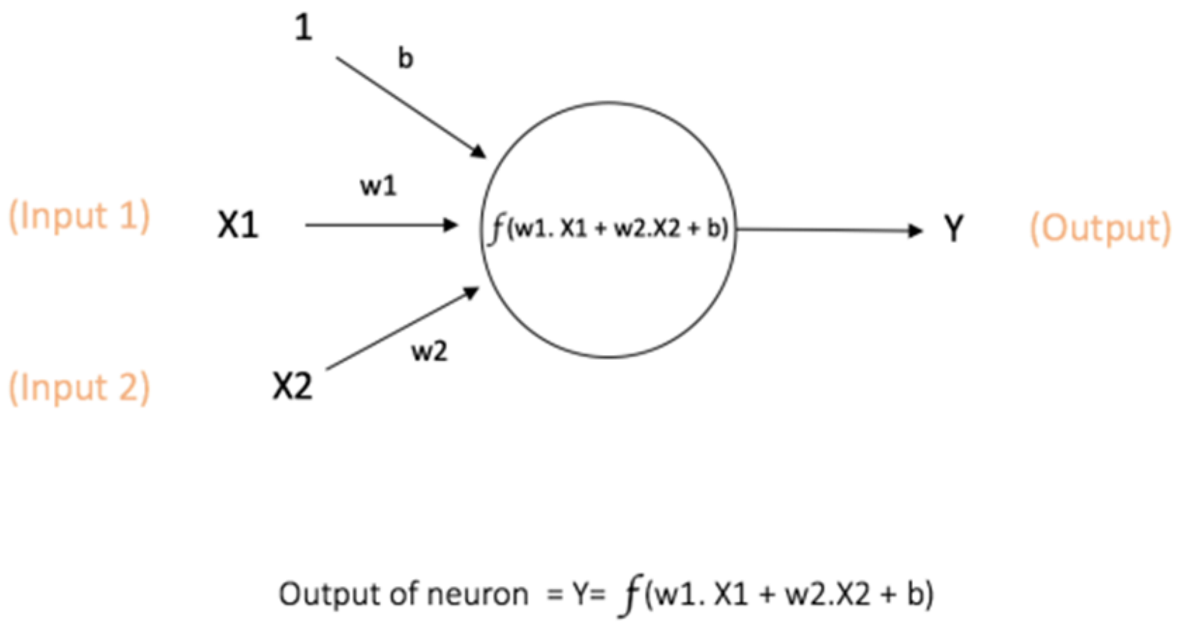


Fig. Perceptron model with linear activation function (for Bivariate dataset)

The dataset is divided into training and test data, with 70% of the data used for training and the remaining 30% for testing. The perceptron model is then trained on the training data using the gradient descent method for the perceptron learning algorithm. The model is then tested on the test data to evaluate its performance.

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

In the perceptron, we are training the model for the 100 epochs for the 70 % of the given bivariate data. In which first we are calculating the instantaneous error for each input in a single epoch. After each epoch the average error is calculated and is being plotted.(fig.)

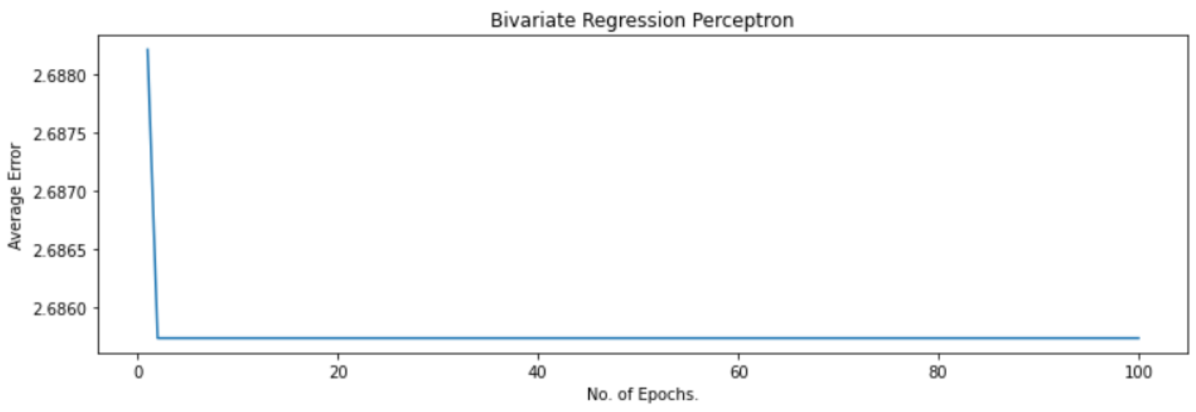
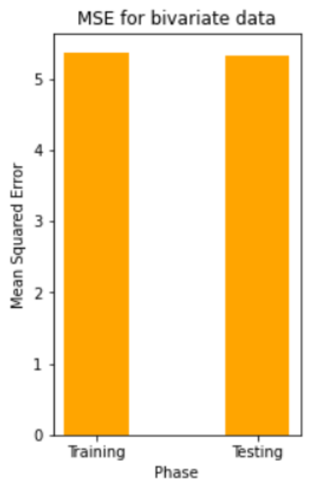


Fig. Plot between Average Error and Epochs in training phase

2.1.2. Plot of Mean Squared Error for Bivariate data

In Bivariate, 70% training data and the instantaneous error generated from it, is being used to calculate the Mean Squared Error. The same is also done for the remaining 30% testing data.

$$E_{\text{MSE}} = \frac{1}{N_t}\sum_{n=1}^{N_t}(\hat{y}_n - y_n)^2$$



2.2.2. Plot of Model and Actual output for Bivariate data

In bivariate, we have plot the actual output and model output for both training and testing, which contains 7000 and 3000 data points respectively.

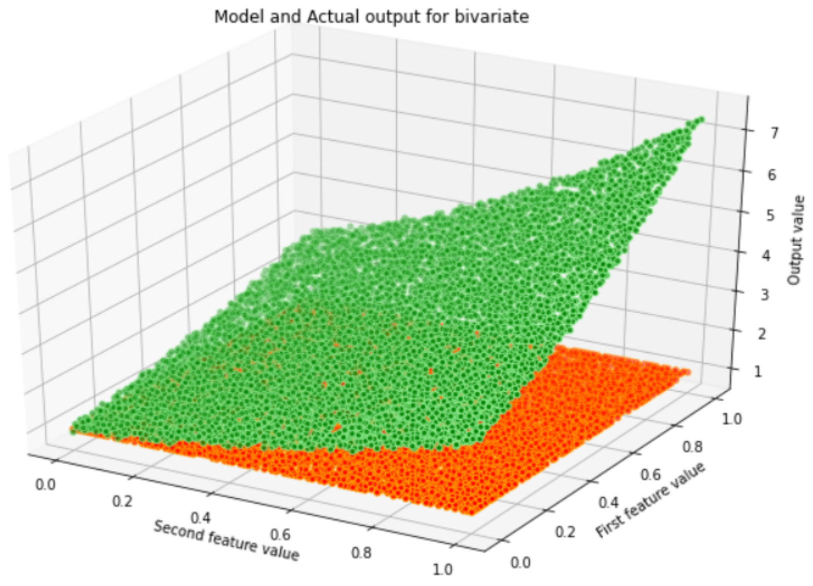


Fig. Plot between Model(red) and Actual output (green) in training phase

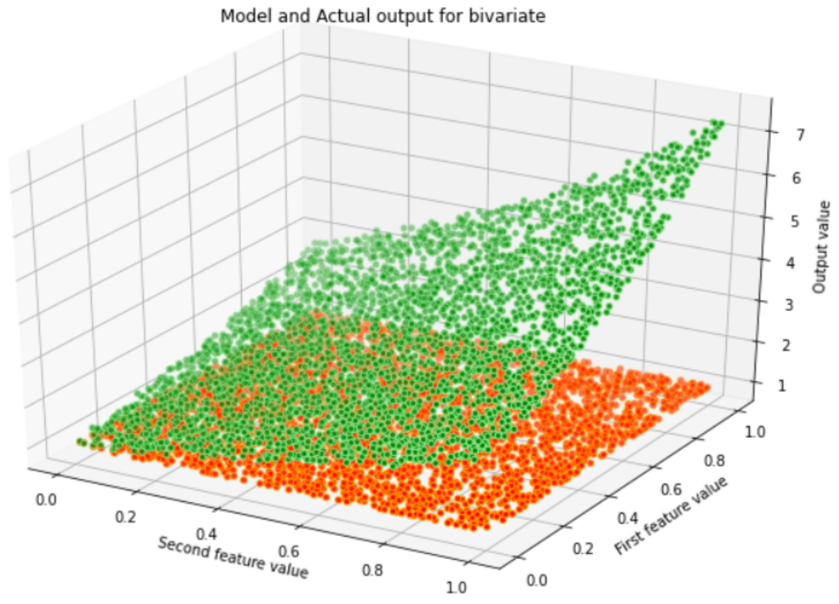


Fig. Plot between Model(red) and Actual output (green) in testing phase

2.1.3. Plot between Model and Actual output for bivariate data

In this plot we have plotted against the training data and testing data respectively which contains 7000 and 3000 data points respectively.

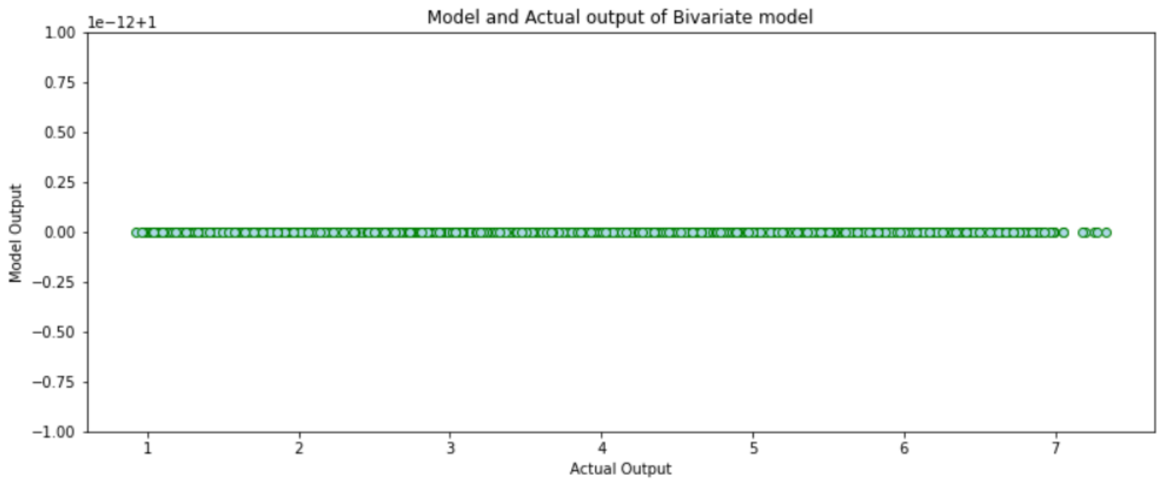


Fig. Plot between Model and Actual output in training phase

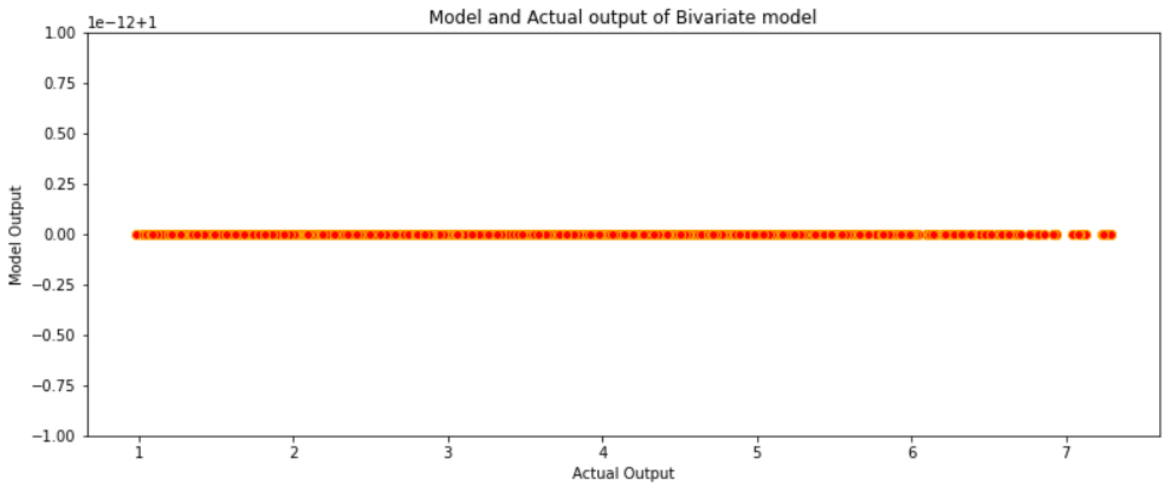


Fig. Plot between Model and Actual output in testing phase

Inference point

The main objective of this assignment is to evaluate the performance of perceptron on different types of datasets and to compare the Regression accuracy achieved on Univariate and Bivariate datasets. For regression, we used the linear activation function and MSE as the evaluation metric. Overall, our findings suggest that the perceptron algorithm can be used for regression tasks, but its performance is limited on more complex datasets. The one-against-one approach and gradient descent method have proven to be effective in improving the accuracy of the perceptron algorithm for multi-class classification. The gradient descent method is an effective way to train the perceptron algorithm, and finding the optimal learning rate is crucial for achieving good accuracy.