Deep Learning and Applications Assignment 1 Report

Group-05

Ankit Mehra(S22020)
Urvashi Goswami(S22024)
Ashish Rana(S22019)

Index

Perceptron- Introduction

Classification using a single neuron

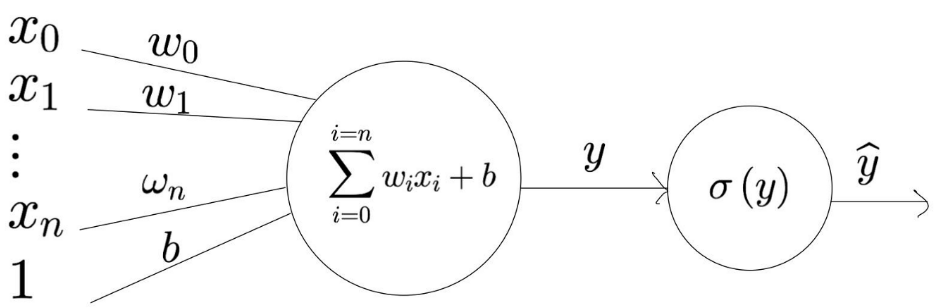


Image Source: [Article] Perceptron learning from discrete to continuous. By Vishal Jain

Basic Components of Perceptron

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

Weights and Bias: Weight parameter represents the strength of the connection between units. This is another 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: This is the final component that helps to determine whether the neuron will fire or not. Activation Function can be considered primarily as a step function.

Types of Activation functions:

- Step function
- Sigmoid function

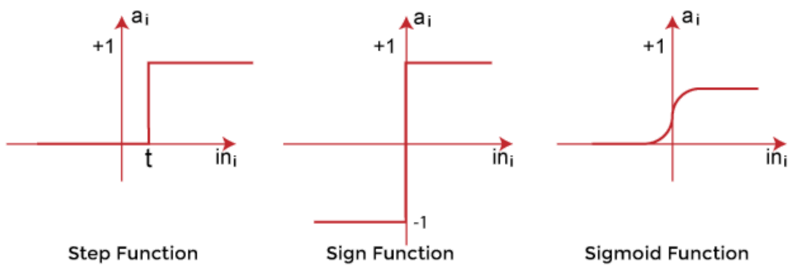


Image Source : [Article] What is Perceptron ?A beginners guide. By Mayank Banoula

The input vector (x= [x1, x2, x3, ... xn]) along with its weight(w= [w1,w2,w3, ...wn]) passed to the neuron. The neuron calculates the dot product and summed it with a bias value . The output value is then passed to the sigmoidal activation function, here we use a logistic activation function which produces the result in the range of [0, 1].

The perceptron can classify two classes ,

if (w.T). x+wo > 0 into Class 1

and

if (w.T).x+wo ≤ 0 into Class2.

We have to train the decision boundary such that it is placed between the two classes of data so that the error is minimum.

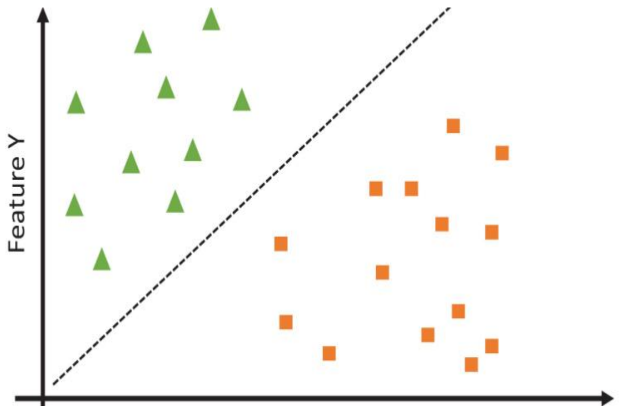


Image Source : [Article] What is a Support Vector Machine? by Spiceworks

Gradient Descent Algorithm :- Gradient descent is an optimization algorithm that is used to minimize the loss function in a machine learning model. The goal of gradient descent is to find the set of weights (or coefficients) that minimize the loss function. The algorithm works by iteratively adjusting the weights in the direction of the steepest decrease in the loss function.

Δw be the change to the direction of convergence. The learning parameter should move in a direction of Δw creating (w-new) as shown. As we know that we are going to move in Δw direction it is always better to choose a small step η at a time with the factor .

So weights, has to be updated like: $w = w + \eta\Delta$

$\Delta w \propto -\partial En/\partial w$

$\partial En/\partial w = [\ \partial En/\partial w0 \ldots \partial En/\partial wn]\ T$

$\Delta w = [\Delta wo \ldots \Delta wn]$

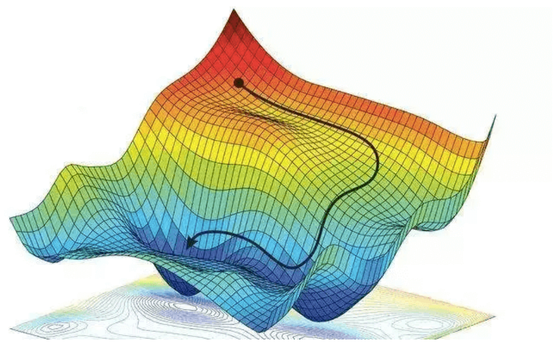


Image source : [Article] Artificial intelligence related knowledge.

$$\Delta \mathbf{w} = -\eta \frac{\partial E_n(\mathbf{w})}{\partial \mathbf{w}} \Rightarrow \Delta \mathbf{w} = \eta \delta^o \hat{\mathbf{x}}_n \quad \text{where } \delta^o = (y_n - s_n)\frac{df(a_n)}{da_n}$$

Average Error:- $$E_{av} = \frac{1}{N}\sum_{n=1}^{N} E_n(\mathbf{w})$$

Classification tasks

Training

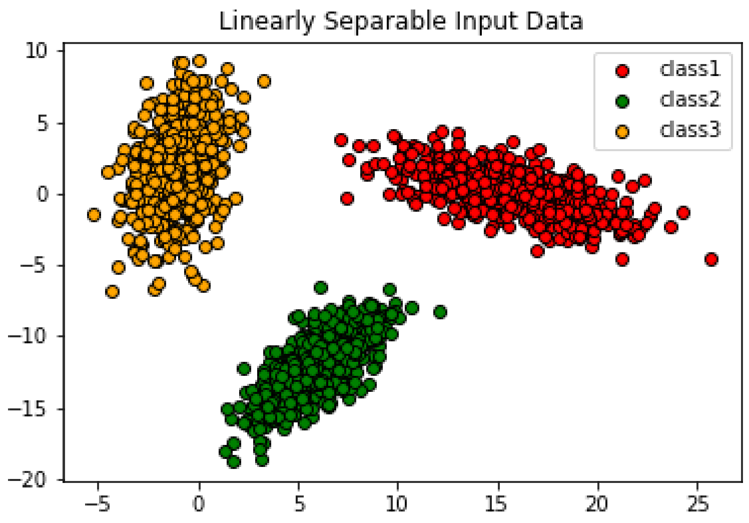
1. Random weights are initialized in w vector.
2. Xn vector is chosen for the further process.
3. Output of the neuron is calculated an = (w.T). x̂ n
4. Instantaneous error is calculated En =(½) *(yn − sn)^2
5. Weights are updated   w = w + ηΔ . where 0≤ η ≤ 1
6. Repeat steps till the defined number of epochs are completed.
7. Calculate average error (Eav).
8. Repeat the above steps until the convergence criterion is satisfied.

Testing

1. Find an = (w.T). x̂ n
2. Pass the to the logistic sigmoidal activation function
3. Then classify to class 1 if the output is less than 0.5 and to class 2 if the output is greater than or equal to 0.5.
4. Plotting of the tested data

Results

Linearly Separable data

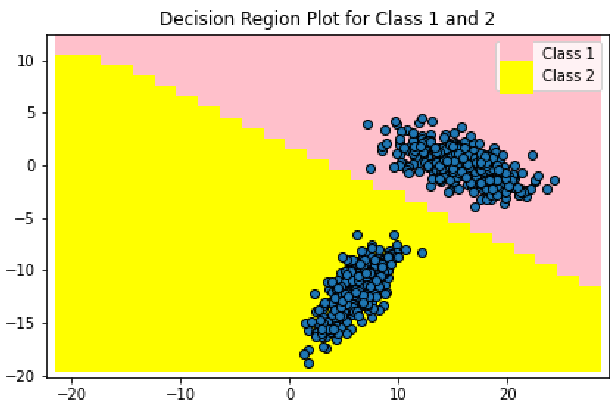


The scatter plot shows that the data is linearly separable.

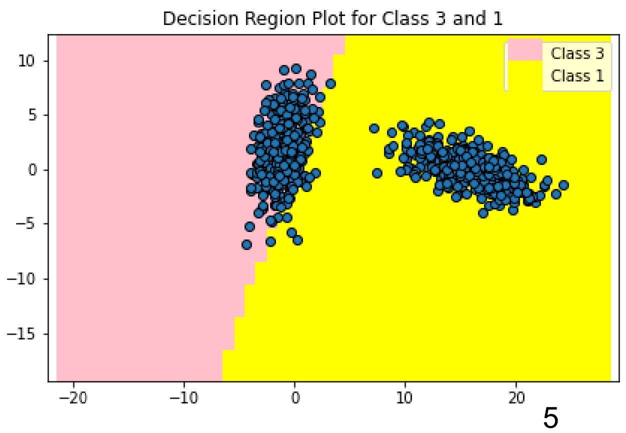
Classification of data is done by one against one method

We have used matplotlib for plotting and coloured the each pixel with size of 500.

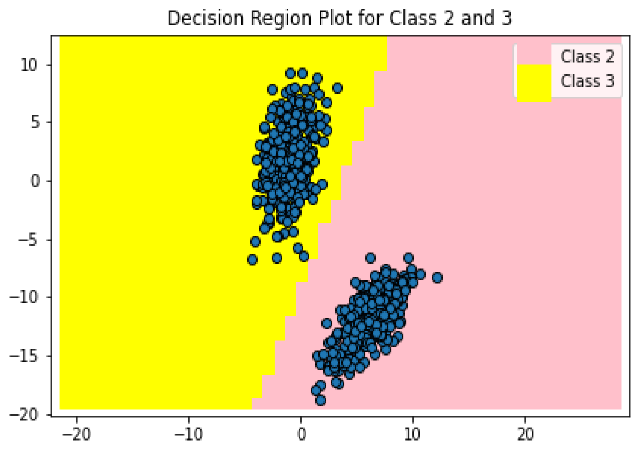
- Class1 and Class2 classification



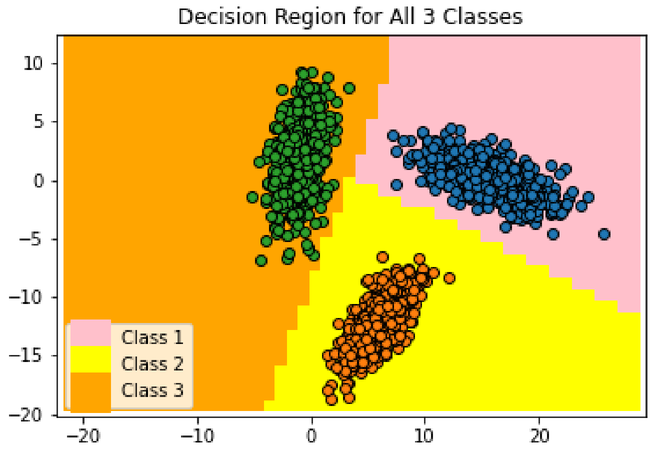
Class 3 and Class 1 classification.



- <u>Class 2 and Class 3 classification.</u>

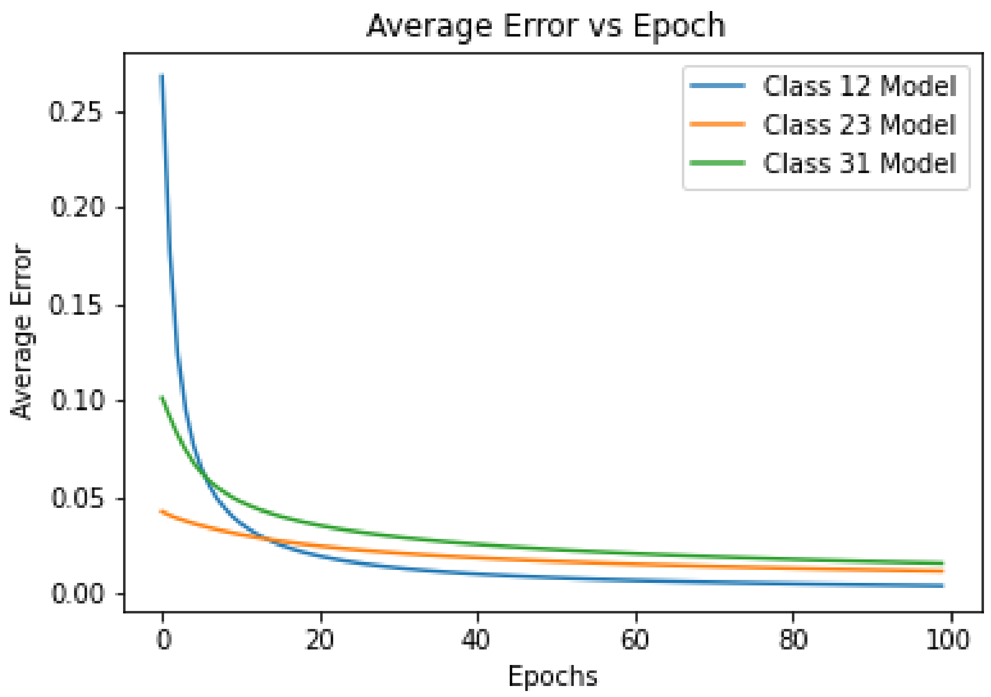


<u>Merged decision regions</u>



In all 4 cases we can see that classes are classified using a linear boundary .

Average Error v/s Epochs Graph



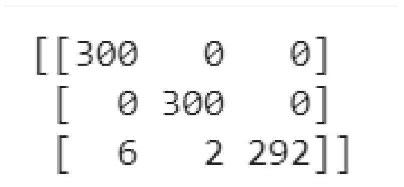
It can be inferred from the above plot that the average error is decreasing with the increase in the number of epochs.

Confusion matrix and Accuracy report.

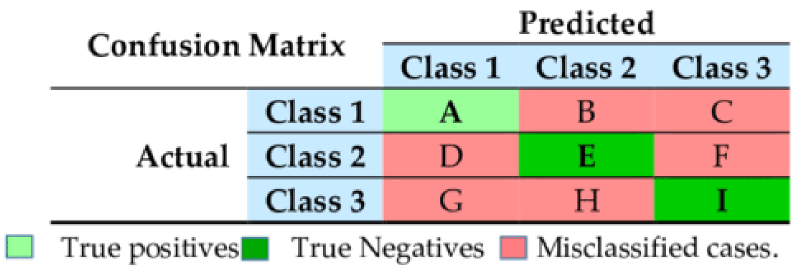
Confusion matrix.

A confusion matrix is a table that reports the counts of true positives, false positives, true negatives and false negatives which are defined as:

• True Positive (TP). The label belongs to the class, and it is correctly predicted.
• False Positive (FP). The label does not belong to the class, but the classifier predicted it as positive.
• True Negative (TN). The label does not belong to the class, and it is correctly predicted.
• False Negative (FN). The label does belong to the class but is predicted as negative.



```
[[300   0   0]
 [  0 300   0]
 [  6   2 292]]
```



Confusion Matrix		Predicted		
		Class 1	Class 2	Class 3
Actual	Class 1	A	B	C
	Class 2	D	E	F
	Class 3	G	H	I

🟩 True positives 🟩 True Negatives 🟥 Misclassified cases.

Image Source :[Article]An Accurate CT Saturation Classification Using a Deep Learning Approach Based on Unsupervised Feature Extraction and Supervised Fine-Tuning Strategy

Observations based on confusion matrix

- 300 samples of class 1 are correctly classified in class 1.
- 300 samples of class 2 are correctly classified in class 2.
- 292 samples of class 3 are correctly classified in class 3 and 8 samples are misclassified.

$$\textbf{Accuracy} = \frac{\textbf{Number of samples correctly classified}}{\textbf{Total number of samples used for testing}} * \textbf{100}$$

$$\textbf{Accuracy} = \frac{(C11+C22+C33)}{(c11+c12+c13+c21+c22+c23+c31+c32+c33)} = 99.112\%$$

Precision: "Number of samples correctly classified as positive class, out of all the examples classified as positive class"

$$\text{Class1 Precision} = \frac{\text{TP for Class 1}}{\text{TP for Class 1 + FP for class 1}} = 98.00\%$$

$$\text{Class2 Precision} = \frac{\text{TP for Class 2}}{\text{TP for Class 2 + FP for class 2}} = 99.00\%$$

$$\text{Class3 Precision} = \frac{\text{TP for Class 3}}{\text{TP for Class 3 + FP for class 3}} = 100.00\%$$

Mean Precision = ⅓(precision1+precision2+precision3) = ⅓(98+99+100) = 99%

Recall :- Number of samples correctly classified as positive class, out of all the examples belonging to positive class.

$$\text{Class1 Recall} = \frac{\text{TP for Class 1}}{\text{TP for Class 1 + FN for class 1}} = 100.00\%$$

$$\text{Class2 Recall} = \frac{\text{TP for Class 2}}{\text{TP for Class 2 + FN for class 2}} = 100.00\%$$

$$\text{Class3 Recall} = \frac{\text{TP for Class 3}}{\text{TP for Class 3 + FN for class 3}} = 97.00\%$$

Mean Recall = ⅓(recall1+recall2+recall3) = ⅓(100+100+97) = 99%

<u>F Measure</u>:- Harmonic mean of precision and recall.

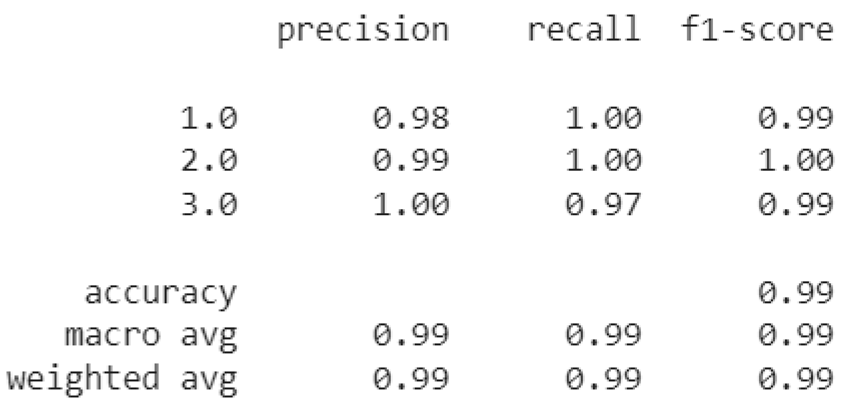
$$\text{Class 1 F-measure} = 2* \frac{\text{Precision1} * \text{Recall 1}}{\text{Precision1} + \text{Recall1}} = 99\%$$

$$\text{Class 2 F-measure} = 2* \frac{\text{Precision2} * \text{Recall 2}}{\text{Precision2} + \text{Recall2}} = 100\%$$

$$\text{Class 3 F measure} = 2* \frac{\text{Precision3} * \text{Recall 3}}{\text{Precision3} + \text{Recall3}} = 99\%$$

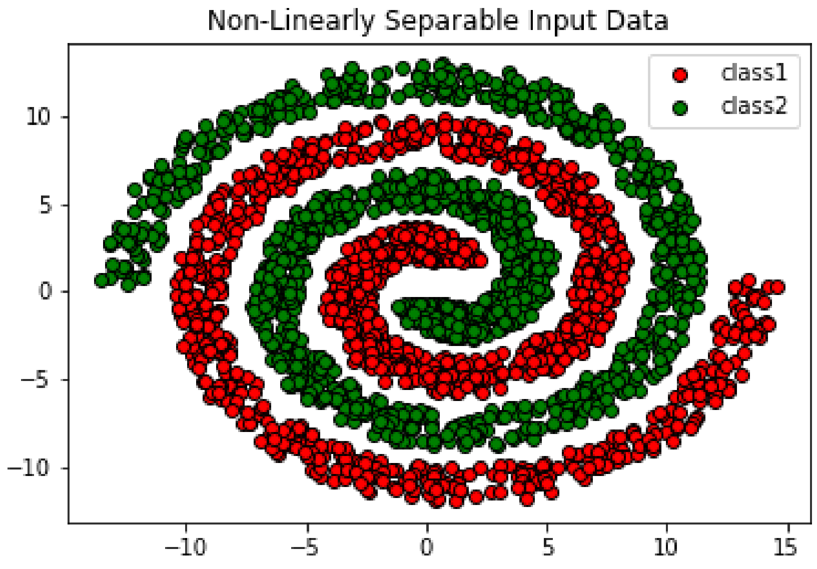
Mean F-measure = ⅓(F-m1F-m2+F-m3) = ⅓(99+100+99) = 99.33%

We have created the Confusion matrix by “**classification_report** " from"**sklearn** " and got the following results .



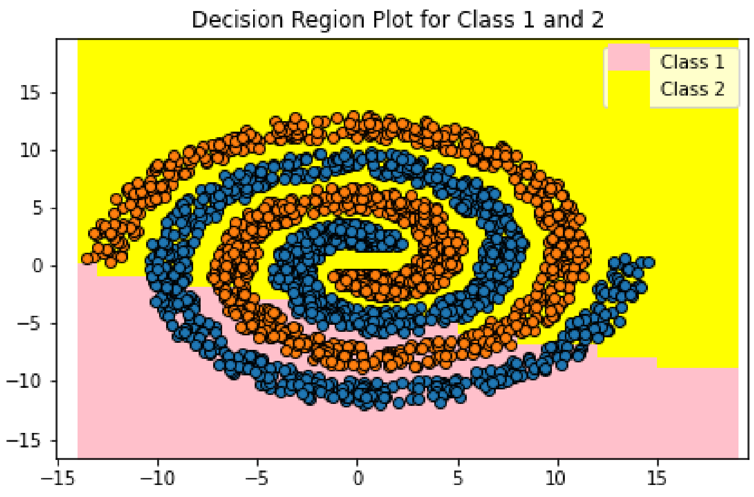
```
              precision    recall  f1-score

         1.0       0.98      1.00      0.99
         2.0       0.99      1.00      1.00
         3.0       1.00      0.97      0.99

    accuracy                           0.99
   macro avg       0.99      0.99      0.99
weighted avg       0.99      0.99      0.99
```

Non Linear data results



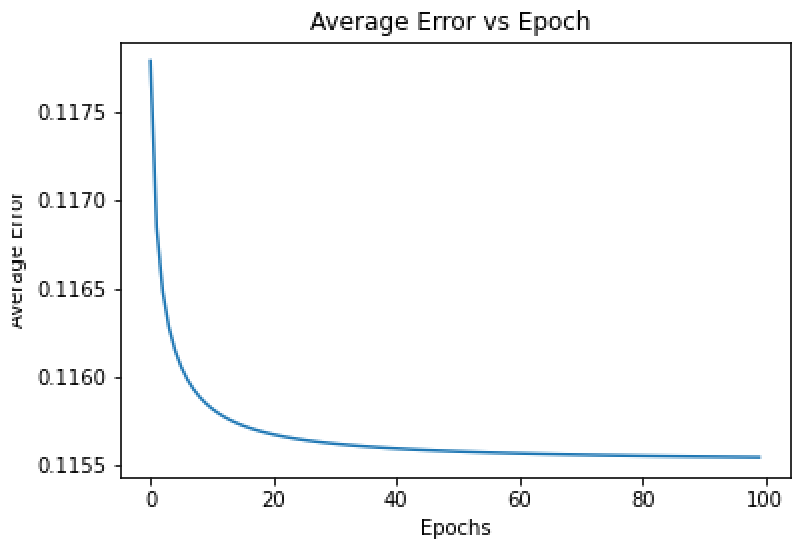
Classification of non-linearly separable data using a single neuron.

Classification Results



We can see that the perceptron model works very poorly for the non-linearly separable data. After training the perceptron is classifying pink region as class1 and yellow region as class2.

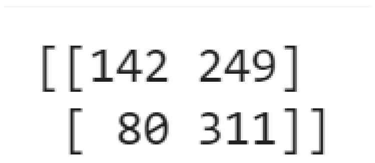
Plot of Average error V/s Epochs .



We can observe that the average error decreases with the number of epochs.

Confusion matrix and Accuracy report

Confusion matrix



```
[[142 249]
 [ 80 311]]
```

- 142 samples of class 1 are correctly classified in class 1 and 249 are misclassified in class 2.
- 311 samples of class 2 are correctly classified in class 2 and 80 are misclassified in class 1.

Accuracy = $\dfrac{\textbf{Number of samples correctly classified}}{\textbf{Total number of samples used for testing}}$ *** 100**

Accuracy = $\dfrac{(C11+C22)}{(c11+c12+c21+c22)}$ = 57.92%

Precision :-

Class1 Precision = $\dfrac{\text{TP for Class 1}}{\text{TP for Class 1 + FP for class 1}}$ = 64.00%

Class2 Precision = $\dfrac{\text{TP for Class 2}}{\text{TP for Class 2 + FP for class 2}}$ = 56.00%

Mean Precision = 1/2(precision1+precision 2) =1/2(64+56) = 60%

Recall :-

Class1 Recall = $\dfrac{\text{TP for Class 1}}{\text{TP for Class 1 + FN for class 1}}$ = 36.00%

Class2 Recall = $\dfrac{\text{TP for Class 2}}{\text{TP for Class 2 + FN for class 2}}$ = 80.00%

Mean Recall = 1/2(recall1+recall 2) = 1/2(36+80) = 58%

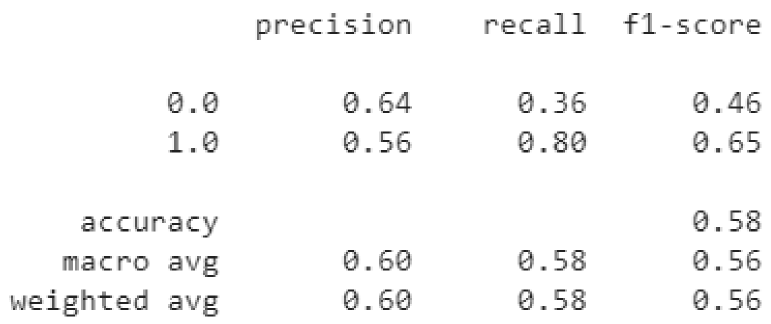
F-measure :-

Class 1 F-measure = 2* $\dfrac{\text{Precision1 * Recall 1}}{\text{Precision1 + Recall1}}$ = 46.00%

Class 2 F-measure = 2* $\dfrac{\text{Precision2 * Recall 2}}{\text{Precision2 + Recall2}}$ = 65.00%

Mean F-measure = 1/2(F-m1+F-m2) = 1/2(46+65) = 55.5%

We have created the Confusion matrix by “**classification_report** " from"**sklearn** " and got the following results .



```
              precision    recall  f1-score

         0.0       0.64      0.36      0.46
         1.0       0.56      0.80      0.65

    accuracy                           0.58
   macro avg       0.60      0.58      0.56
weighted avg       0.60      0.58      0.56
```

Regression Task

Regression is a supervised machine learning technique which is used to predict continuous values.The ultimate goal of the regression algorithm is to plot a best-fit line or a curve for univariate data and plane or hyperplane for multivariate data.

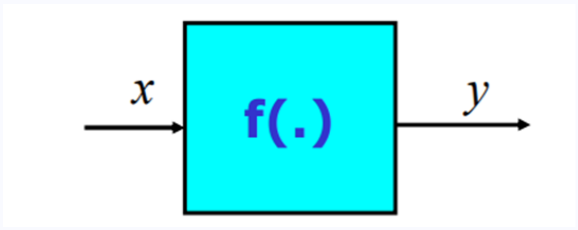


Fig- Univariate data, Image Source : Internet

Here x and y are single independent and dependent variables respectively.

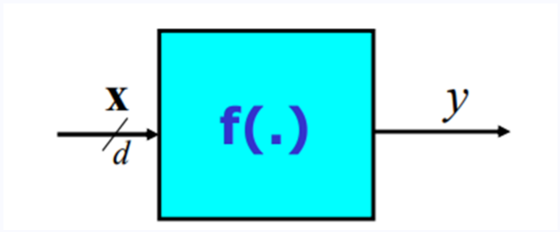
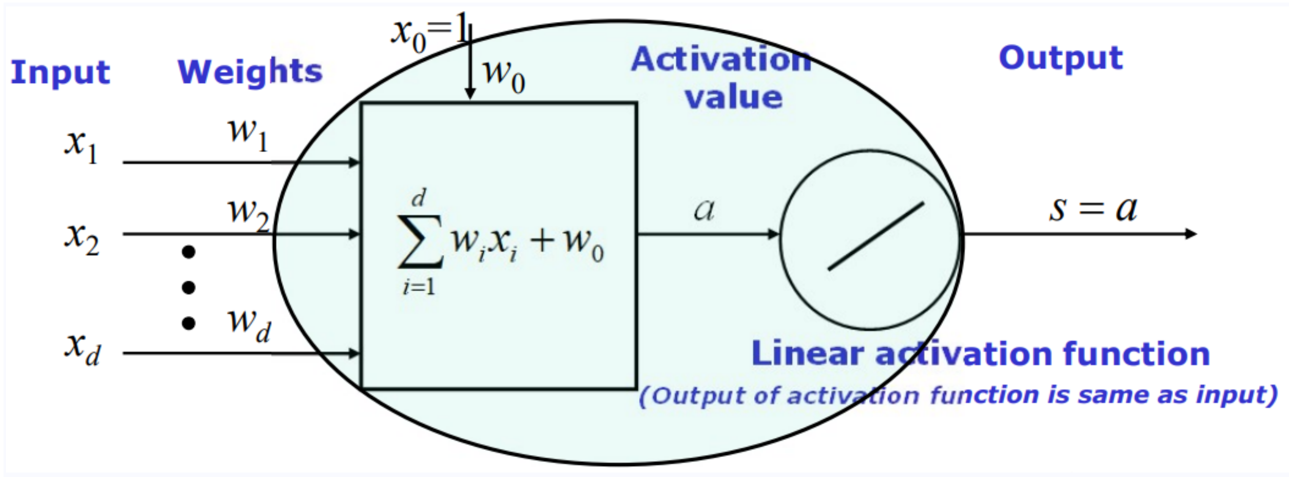


Fig- Multivariate data, Image Source : Internet

Here $x \in R^d$ are multiple independent variables and y single independent variable.

Procedure

Step1 :- We build a regression model by learning from the training data.(Training)
Step2 :- Using the regression model we predict the output variables(Testing)



Input
Weights
$x_0=1$
w_0
Activation value
Output
x_1
w_1
x_2
w_2
x_d
w_d
$\sum_{i=1}^{d} w_i x_i + w_0$
a
$s = a$
Linear activation function
(Output of activation function is same as input)

Image Source : Dr. Dileep AD's class presentation for CS671 at IIT-Mandi.

In the regression task the perceptron algorithm is the same but we use a linear activation function in place of a logistic sigmoidal activation function.

Training

1. Random weights are initialized in w vector.
2. Xn vector is chosen for the further process.
3. Output of the neuron is calculated an = (w.T). x̂ n
4. Instantaneous error is calculated En =(½) *(yn − sn)^2
5. Weights are updated w = w + ηΔw . where $0 \leq \eta \leq 1$
6. Repeat steps till the defined number of epochs are completed.
7. Calculate average error (Eav).
8. Repeat the above steps until the convergence criterion is satisfied.

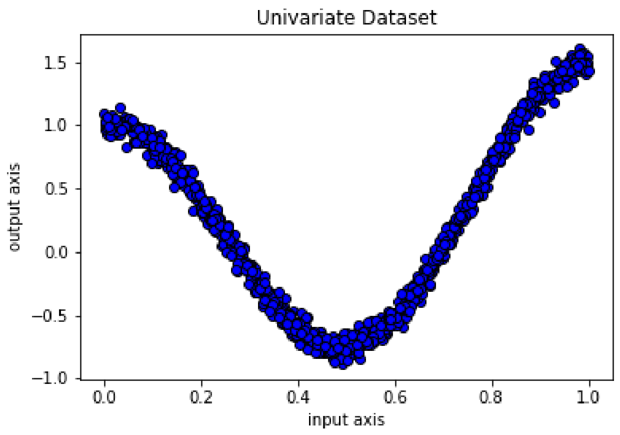
Testing

1. Find an = (w.T). x̂ n
2. Pass the output to the linear activation function

3. Predicted accuracy of the regression model is measured in terms of mean squared error. $E_{RMSE} = \sqrt{(1/N) \sum_{n=1}^{N} (\hat{y}n - yn)^2}$

4. Plotting of the tested data

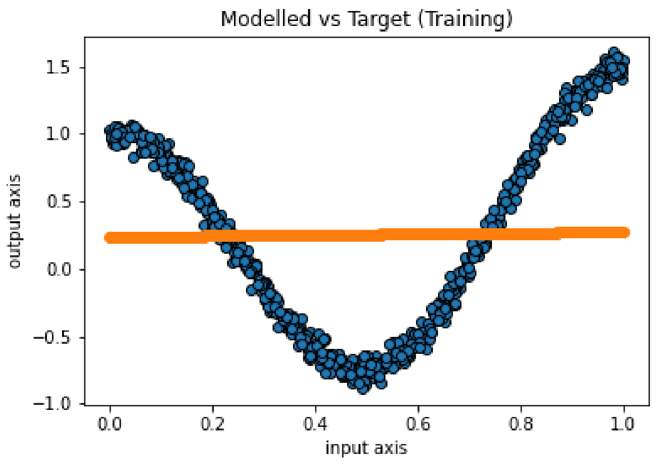
Univariate Dataset.



Using the given data we perform the regression task and obtain the following results.

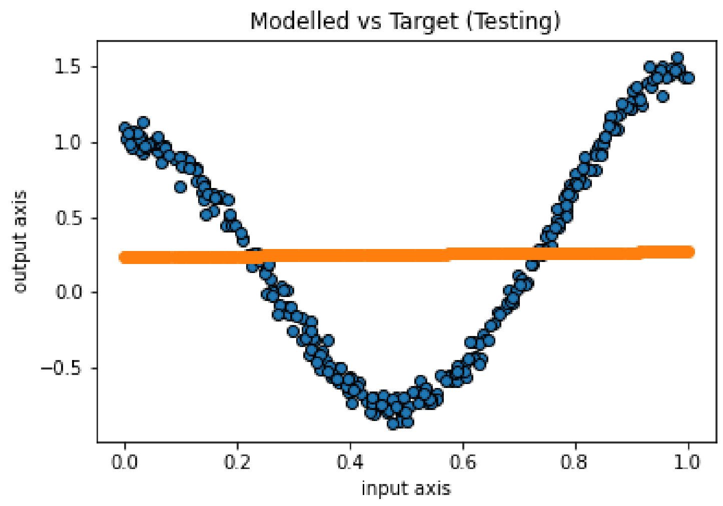
Results

- Final result on Training data



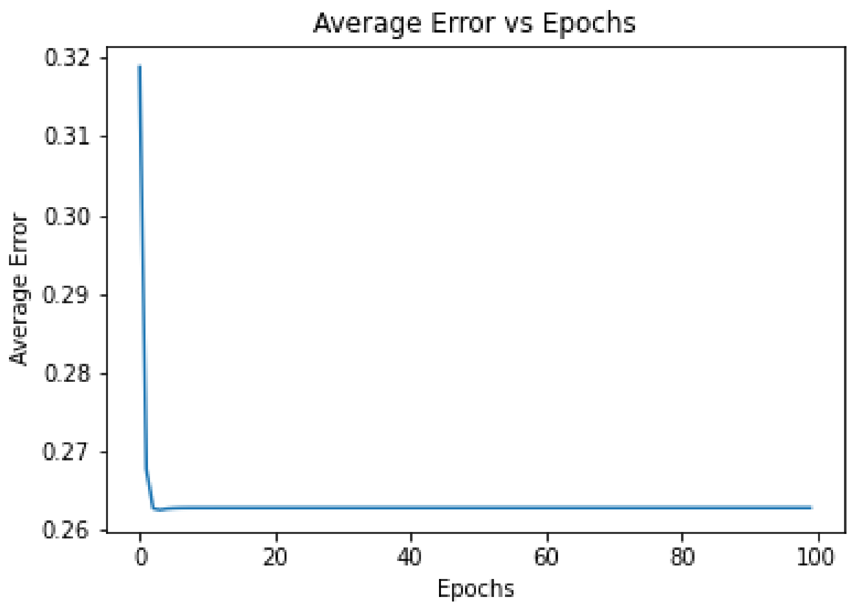
The result of the perceptron model for univariate data is a line , since the data is non-linear so a single perceptron model cannot approximate it precisely.

- Final result of Testing data.



A single neuron can not approximate the nonlinear nature of the data.

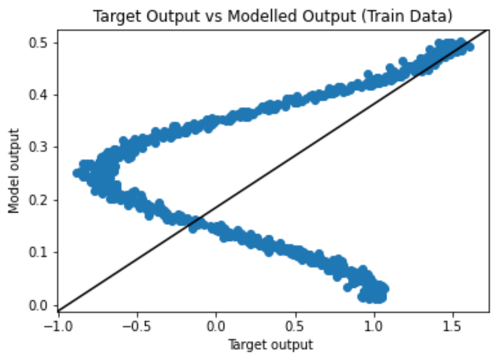
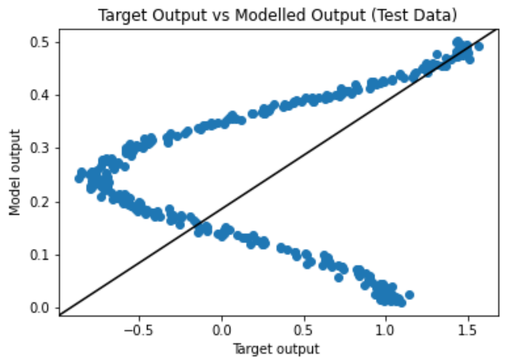
- The average error Vs epoch is plotted and observed that average error decreases exponentially with the number of epochs.



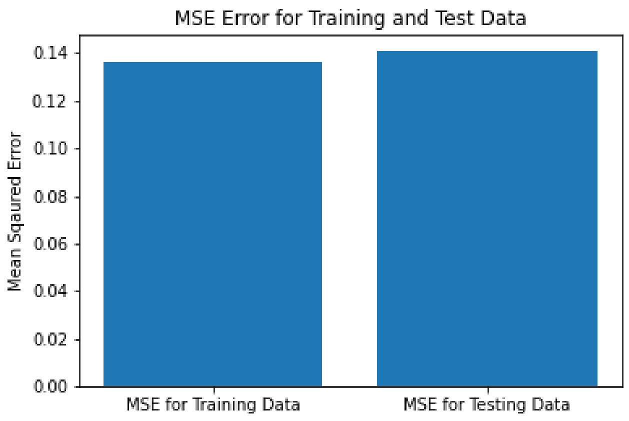
We can observe that the average error decreases with the increase in the number of epochs.

TARGET VS MODEL OUTPUT PLOT

A target against the model output plot shows the effect of the model. For a good fit, the points should be close to the fitted diagonal line. But since our data has non-linear nature , a single perceptron is not enough to capture its non-linear relation with input data and thus gives poor approximation results.

Scatter plot with target output on x-axis and model output on y-axis, for training data.	Scatter plot with target output on x-axis and model output on y-axis, for testing data.
	

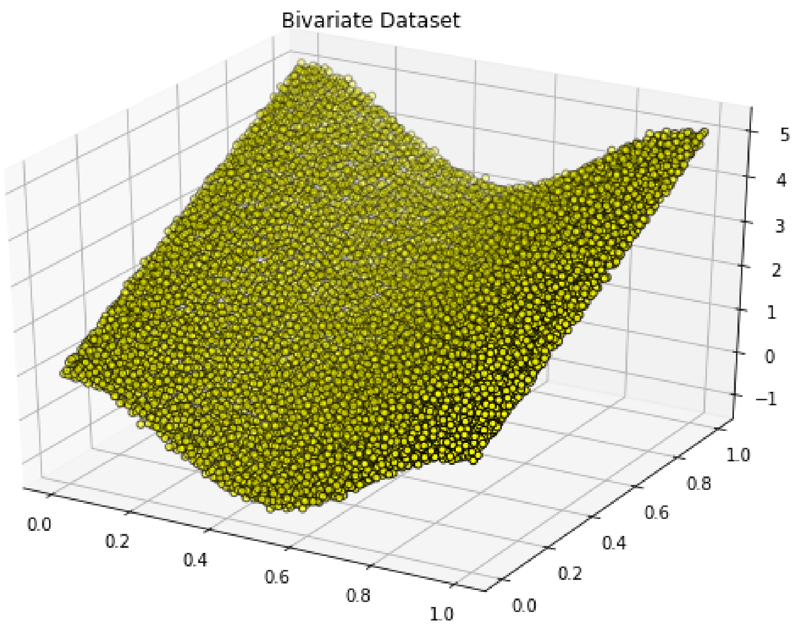
- Plot of the values of mean squared error (MSE) on training data and test data



We know that the smaller the MSE the more correct the prediction , and since the MSE for both testing and training is comparable it shows that the model works as accurately for the unseen data as it does for the training data.

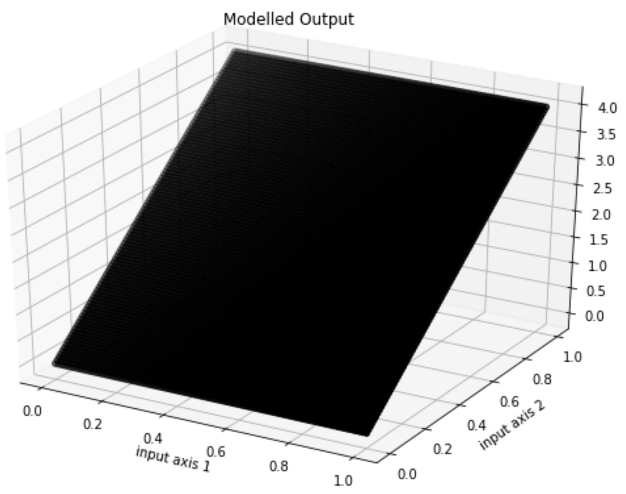
Bivariate Dataset.

- Perceptron model will test the bivariate data shown below



As observed from the bivariate data , it is a non-linear plane.

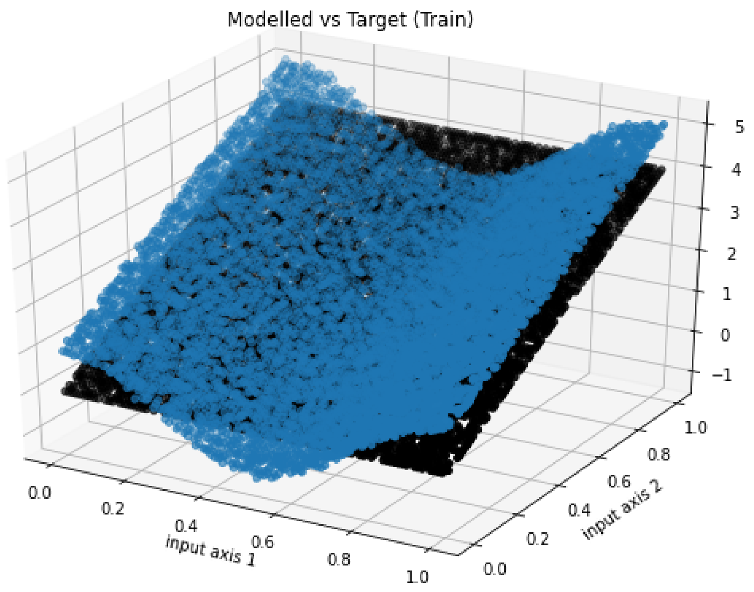
- Modeled output of the bivariate data given to us .



The single perceptron model is only capable of producing the linear plane and can not approximate the non-linear nature of the data

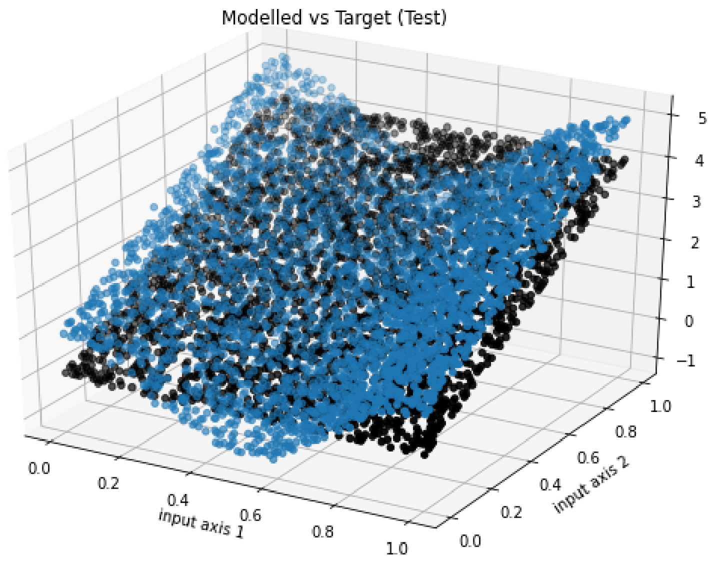
MODELED OUTPUT Vs TARGET OUTPUT

- **Training**



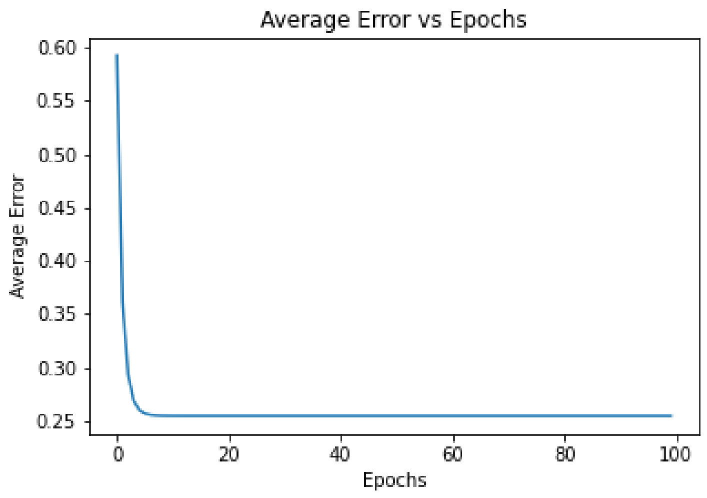
The surface in blue is the actual data and the surface in black is the modeled data For training data .

- **Testing**



The surface in blue is the actual data and the surface in black is the modeled data for testing data .

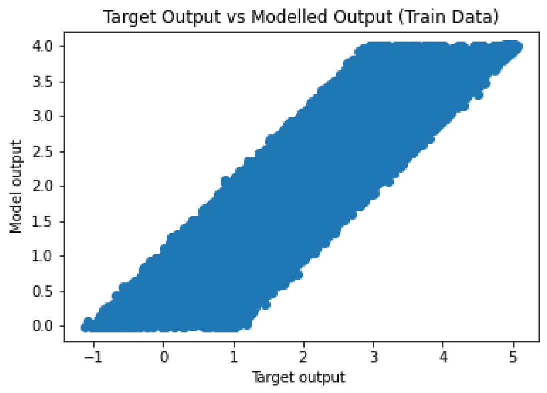
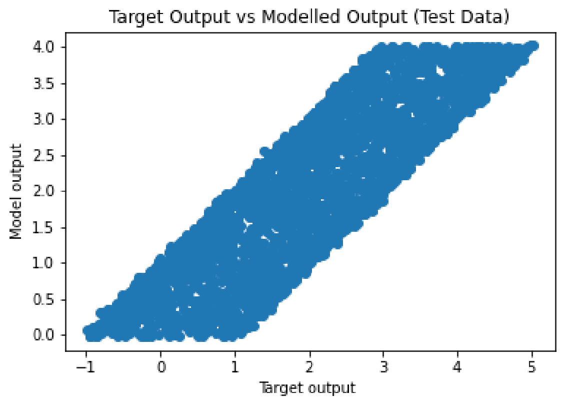
- The average error V/s Epoch plot.



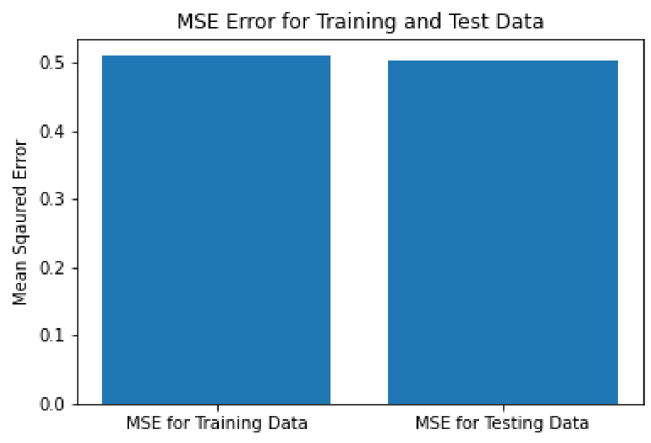
We can observe that the average error decreases with the number of epochs.

TARGET VS MODEL OUTPUT PLOT

A target against the model output plot shows the effect of the model. For a good fit, the points should be close to the fitted diagonal line. But since our data has non-linear nature , a single perceptron is not enough to capture its non-linear relation with input data and thus gives poor approximation results.

Training data.	Testing data.
	

- Plot of the values of mean squared error (MSE) on training data and test data



We know that the smaller the MSE the more correct the prediction , and since the MSE for both testing and training is comparable it shows that the model works as accurately for the unseen data as it does for the training data.