Group No. 20



| Name | Roll Number |
|---|---|
| Yash Gupta | B20240 |
| Titiksha Behal | B20138 |
| Rustam Narayan | B20128 |



## Q1. Classification tasks:

### A. Dataset 1: Linearly separable classes:

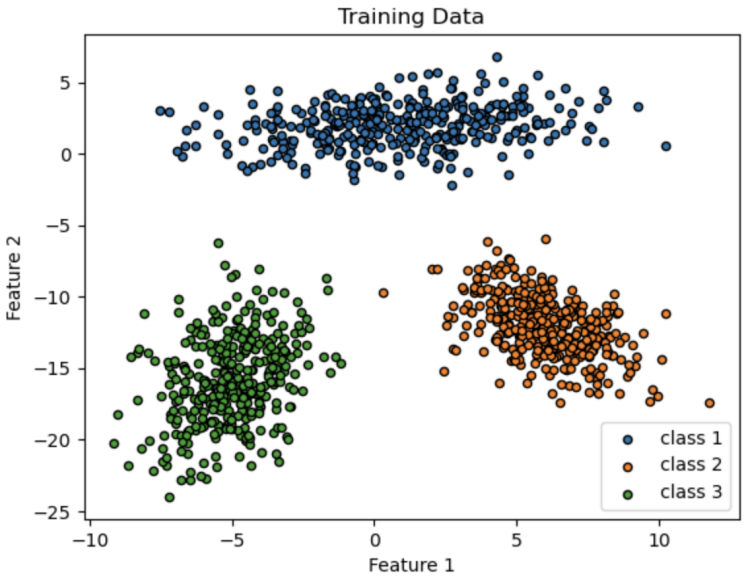



Fig.1.1 Scatter plot of input data (Linearly separable)

**Inferences:**

I. The data points in each class are highly clustered, and there is a clear separation between any two clusters. This makes it easy to distinguish between the data of any two classes.

II. All the data clusters can be separated with a linear discriminant function. For two-dimensional data, this would mean a straight line separating the data of any two classes.

III. Since the classes are clearly distinguishable and linearly separable, using a perceptron model with a sigmoidal activation function would lead to high classification accuracy.

## A.1 Plot of average error (y-axis) vs epochs (x-axis).

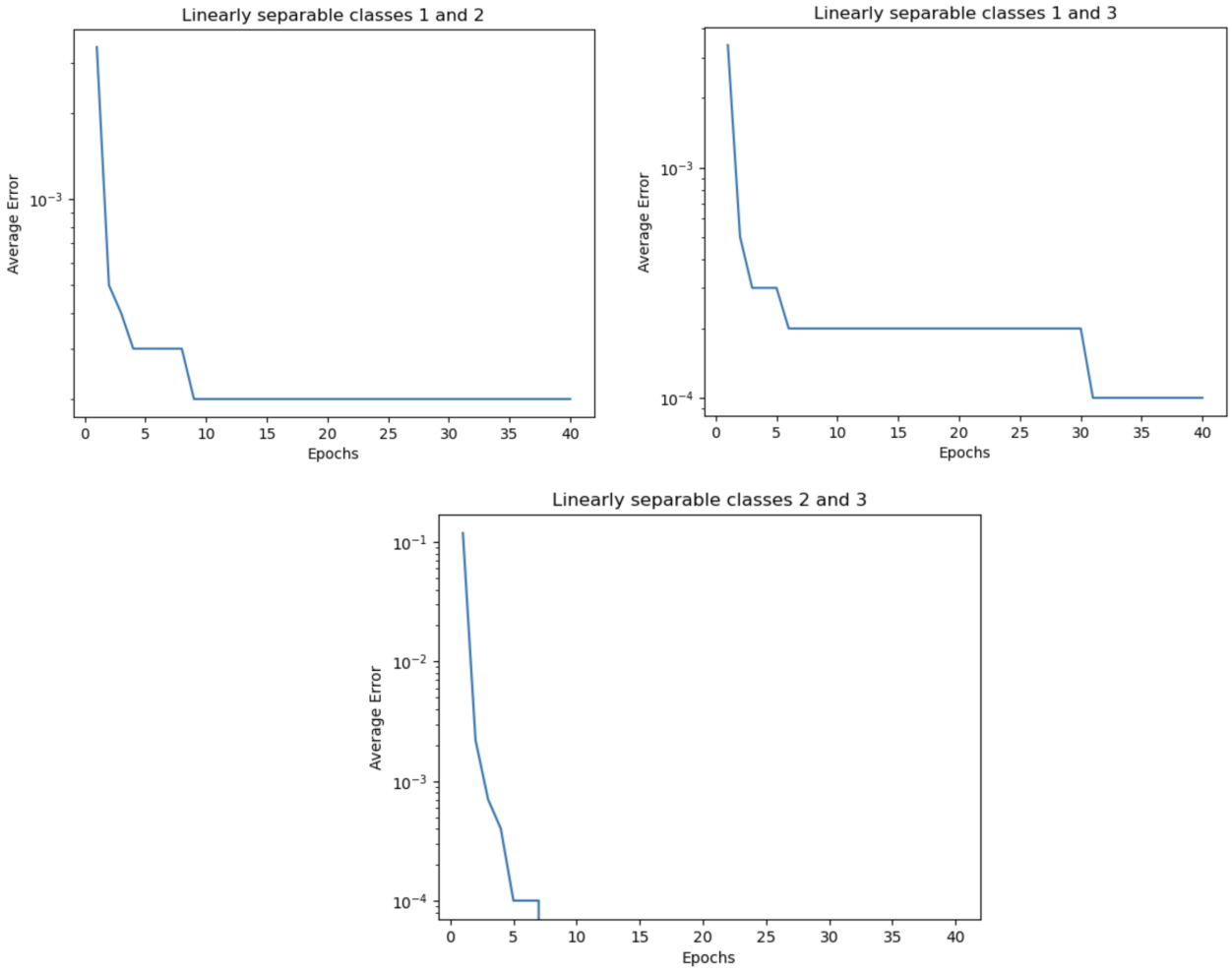



Fig 1.2: Average error v/s No. of epochs in one against-one approach

**Inferences:**

I. For all the 3 plots it is observed that the average error decreases as the number of epochs increases. This is because the gradient descent algorithm which is being used here iteratively updates its weights with each epoch to improve its classification accuracy. However, the rate of decrease in error is not consistent throughout the training process.

II. At the beginning of the training process, the error decreases rapidly as the algorithm quickly learns to identify and correct obvious errors. However, as the algorithm continues to train, the rate of decrease in error slows down. After a certain number of epochs, the average error becomes constant which means that a minimum has been achieved.

III. In some cases, the average error may even increase temporarily as the algorithm encounters a local minimum or overfits the training data.

IV. For classes 1 and 2 and classes 2 and 3, the average error for classification decreases significantly up to 10 epochs after which it becomes almost negligible and constant.

V. For classes 1 and 3 the average error for classification decreases significantly till approximately 30 epochs after which it becomes almost negligible and constant.

## A.2 Decision region plots:

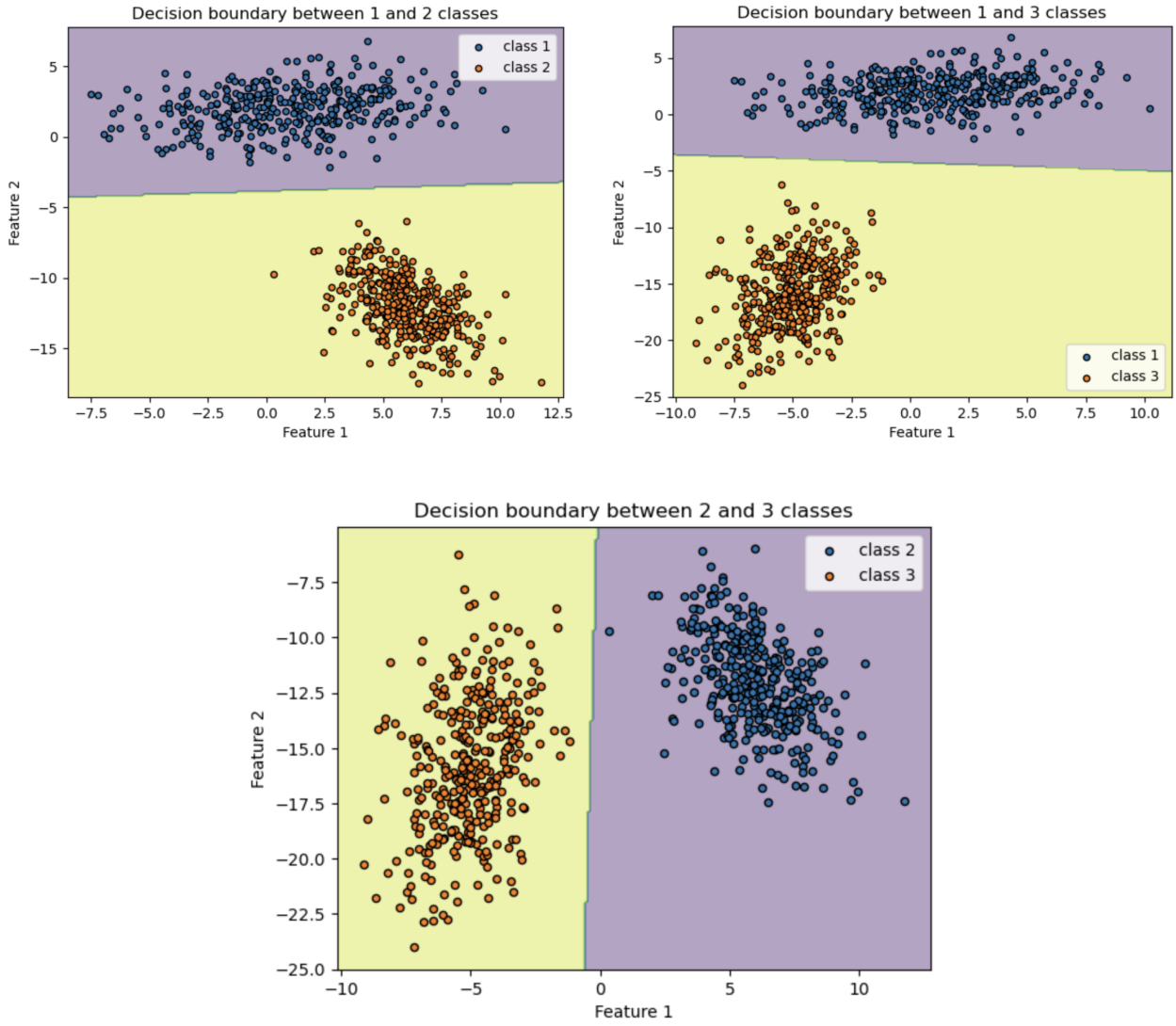



Fig 1.3: Decision boundaries between pairs of two classes each in one against one approach

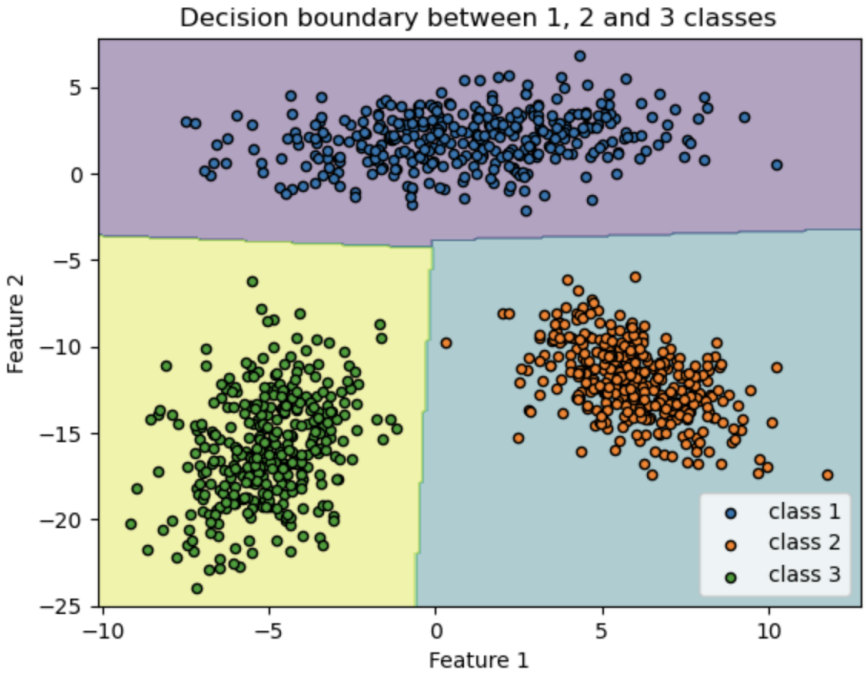



Fig 1.4: Combined decision boundary between all three classes

**Inferences:**

I. The decision boundary/ discriminant function is a linear function of the input features. This means that it is a straight line.

II. Since the classes are linearly separable, there is no overlap of classes across the decision boundary.

III. The final decision boundary for all three classes is obtained by combining the decision boundary of each pair of two classes.

IV. Any data point on one side of the decision boundary is classified as belonging to one class, and any data point on the other side of the boundary is classified as belonging to the other class. This is because the decision boundary acts as a line that separates the two classes.

### A.3 Confusion matrix and classification accuracy.

Confusion Matrix obtained after the Testing phase:

```
[[150   0   0]
 [  0 150   0]
 [  0   0 150]]
```

**Classification accuracy: 1.0**

**Inferences:**

I. From the confusion matrix, we see that all the data points in the Test data are correctly classified by the model as there is no data point that has its predicted class different from its actual class. Hence, the accuracy of the classifier is 100%.

II. The accuracy is 100% in this case because the data was linearly separable and highly clustered for each class. The perceptron model works well in such cases because it gives a linear discriminant function that can clearly distinguish the different classes.

## B. Dataset 2: Nonlinearly separable classes:

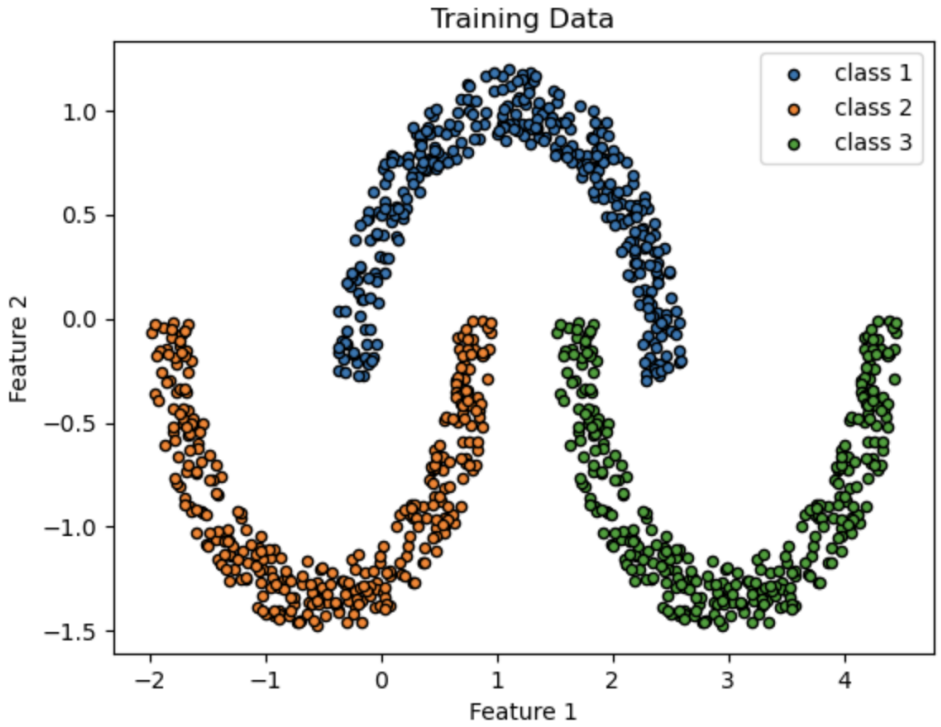



Fig.1.5 Scatter plot of input data (Linearly non-separable)

**Inferences:**

I. From the scatter plot, it is observable that the data can not be separated into different classes using a linear decision boundary. If a linear boundary is used then, the data of different classes might overlap with the boundary and some data points of the same class might even fall on different sides of the boundary.

II. Since the perceptron model gives a linear boundary between classes, in this case, we might not be able to obtain a very high accuracy as was observed in the previous case of linearly separable data.

## B.1 Plot of average error (y-axis) vs epochs (x-axis).

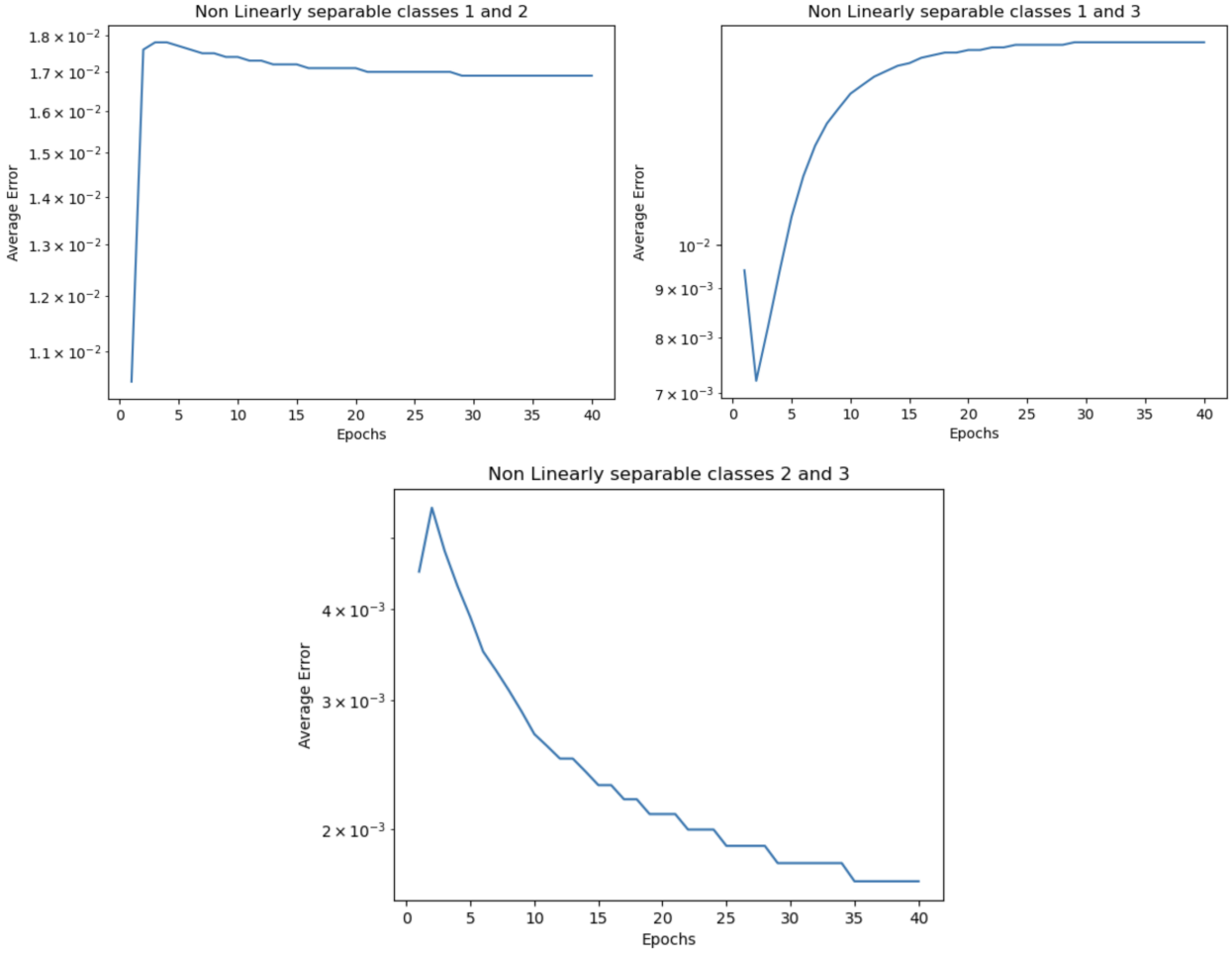



Fig 1.6: Average error v/s No. of epochs

**Inferences:**

I. In all the above average error vs epochs, the plot shows a decreasing trend as the algorithm iteratively updates its weights to improve its classification accuracy. However, the rate of decrease in error may not be consistent throughout the training process.

II. At the beginning of the training process, the error rate decreases rapidly as the algorithm quickly learns to identify and correct obvious errors. However, as the algorithm continues to train, the rate of decrease in error slows down. After a certain amount of epochs, the average error becomes constant which means a minimum has been achieved.

III. In some cases, the average error may even increase temporarily as the algorithm encounters local minima or overfits the training data.

## B.2 Decision region plots:

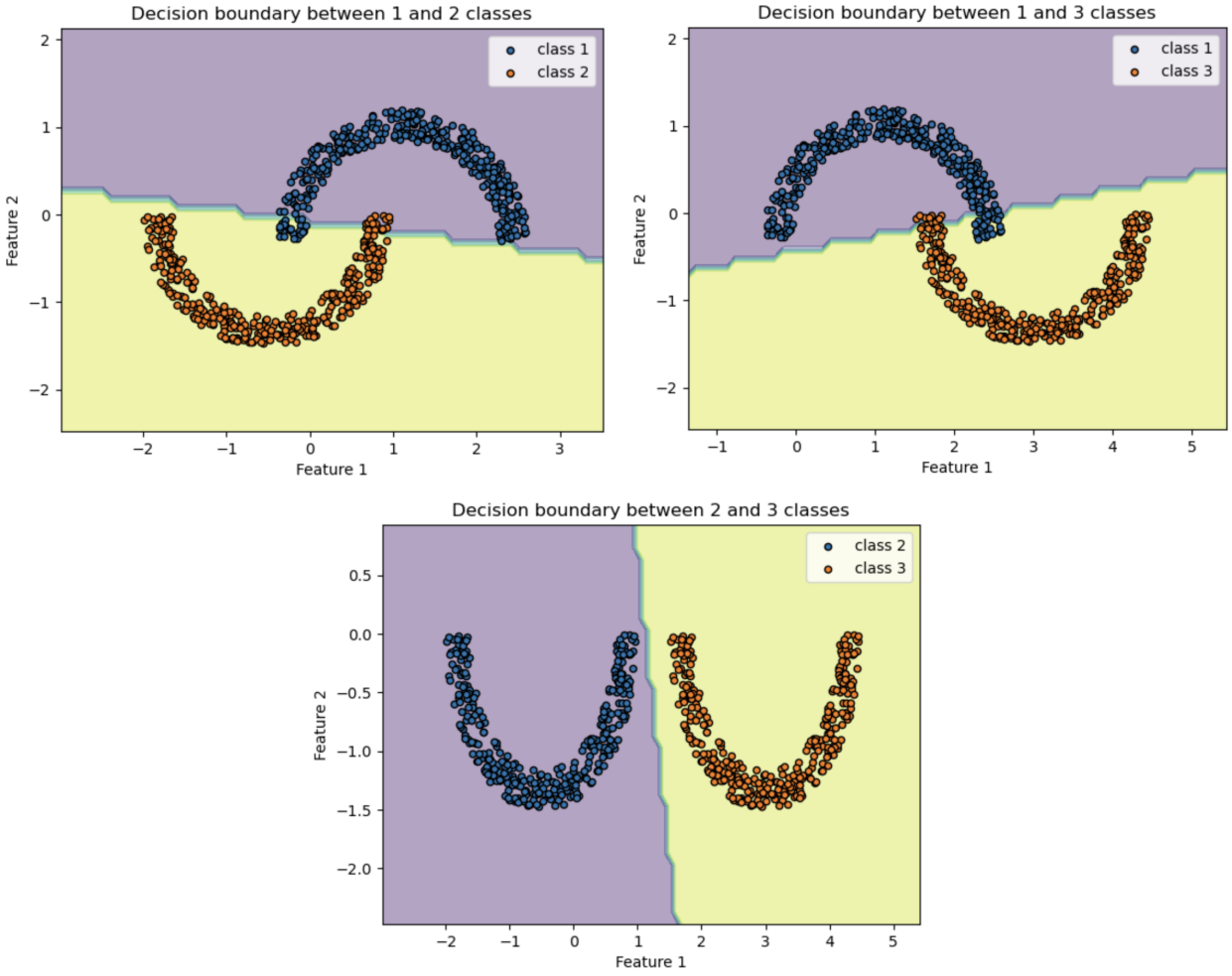



Fig 1.7: Decision boundary between pairs of two classes each - Nonlinearly separable classes

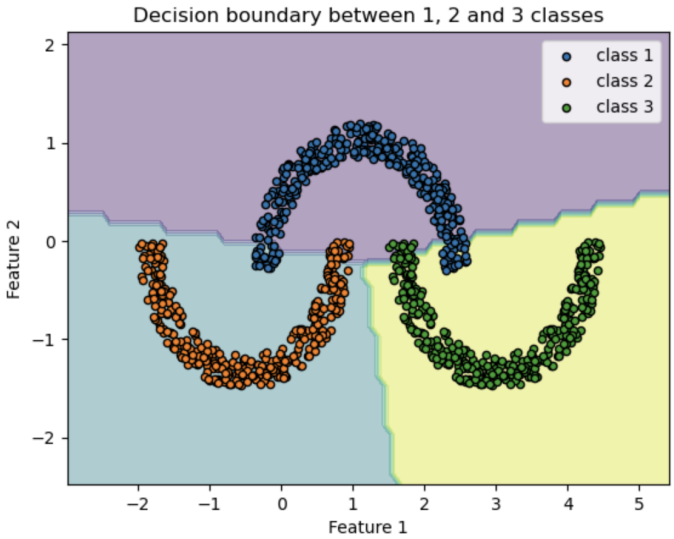



Fig 1.8: Combined decision boundary for all the three classes

**Inferences:**

I. From the graph, it is clear that a linear boundary is not able to separate the data of the three classes properly.
II. The data points are overlapping with the decision boundary and some data points of the same class are also falling on different sides of the discriminant function. This would lead to classification errors during the Test phase.
III. One common inference of decision region plots for non-linearly separable data is that the decision boundary may be highly complex or jagged, with many twists and turns in order to separate the data into different classes. This can indicate that the algorithm is struggling to find a good solution to the problem, and may be overfitting the training data.

**B.3 Confusion matrix and classification accuracy.**

Confusion Matrix obtained after the test phase :
[[134 4 12]
[ 4 146 0]
[ 7 0 143]]

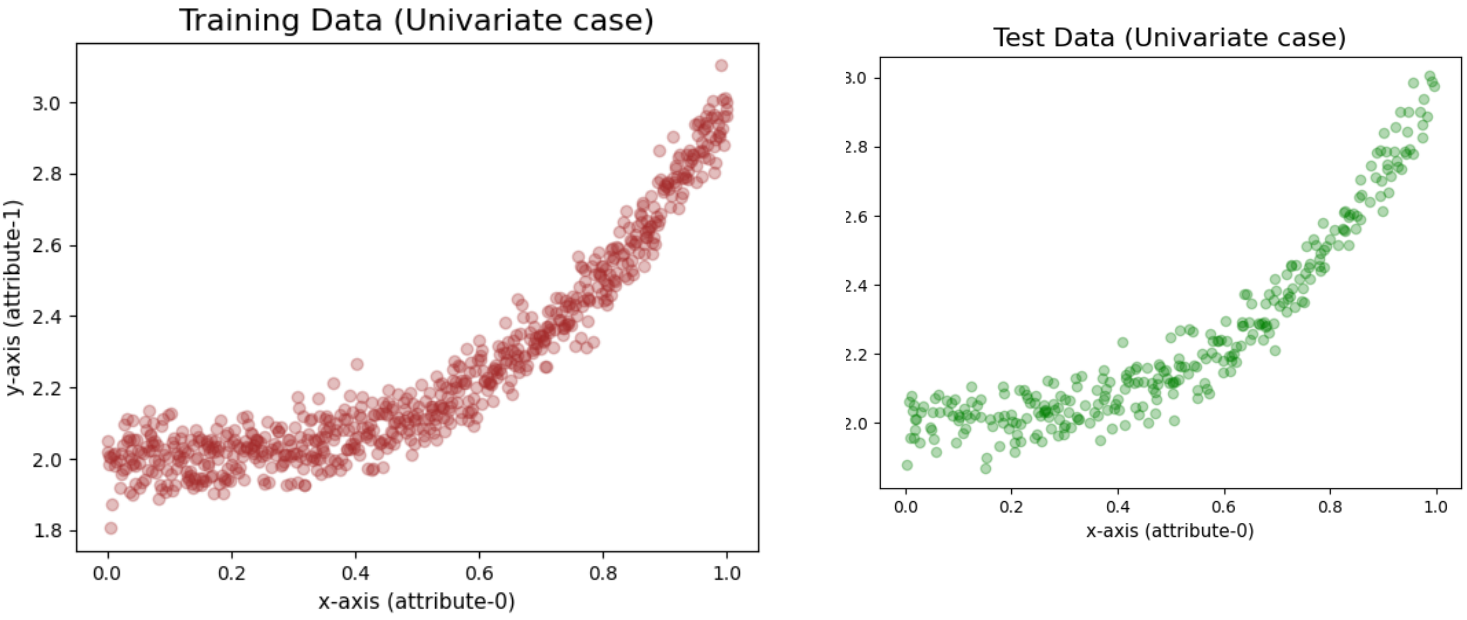
**Classification accuracy: 0.94**

**Inferences:**

I. From the above results, we can observe that in the case of non-linearly separable data, there is 94% accuracy.
II. From the confusion matrix, we can infer that all data points are not correctly classified as there are some data points for which the predicted class is not the same as the actual class.
III. This decrease in accuracy is observed because we used a perceptron model on the data which gives a linear boundary however since the data was non-linearly separable, there were some data points that were misclassified and hence, led to a decrease in accuracy.

## QII. Regression tasks:

### 1) Dataset 1: 1-dimensional (Univariate) input data:




FigII.1 scatter plot of input variable with target value

**Inferences:**

I. From the above scatter plot, we can see that the two features are strongly (+ve) correlated to each other.

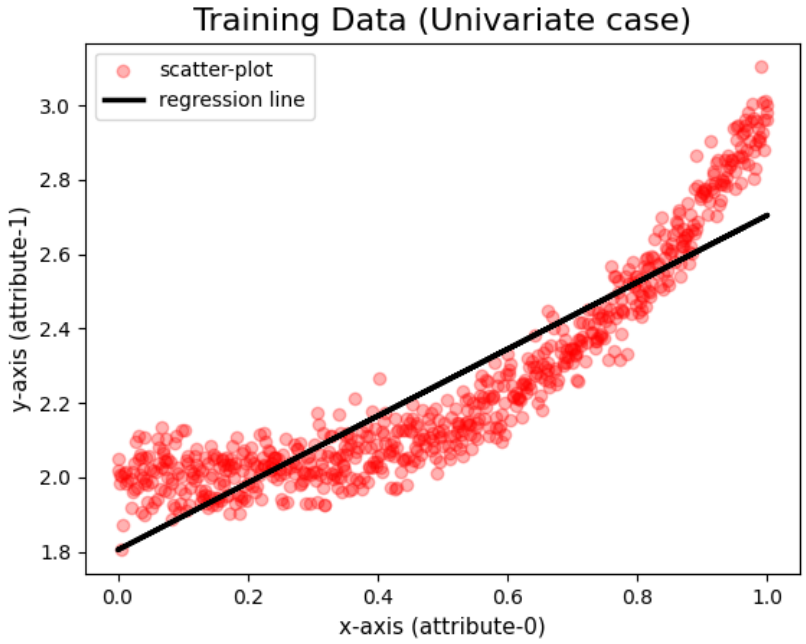
II. the correlation value is 0.9016, which indicates a very high correlation. This value shows that there is a linear relationship between the one-dimension input (attribute 0) and the corresponding target value (attribute 1).

III. This linear relationship can be modeled by a linear regression model. Here, we shall use the perceptron linear regression model.

#### 1. Linear Regression on univariate input data

The value of weights and bias for the univariate regression model run on the above training data is $\mathbf{w} = [1.80444794, 0.90023012]^T$

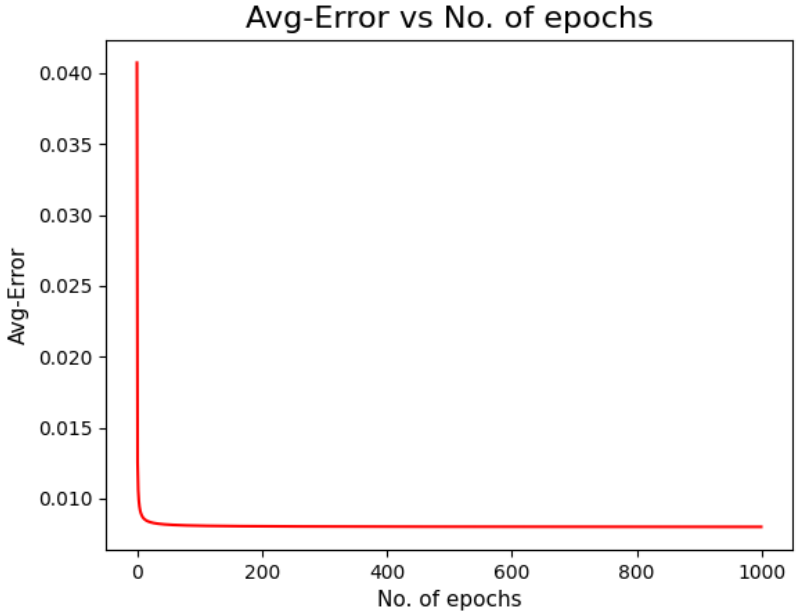
The best fitting line with respect to above **w** is as shown below:




FigII.2: Best fit line

From the above plot it can be inferred that indeed the line approximates the relationship.

2. **Average error (y-axis) vs epochs (x-axis)**




FigII.3: Average error w.r.t no. of epochs

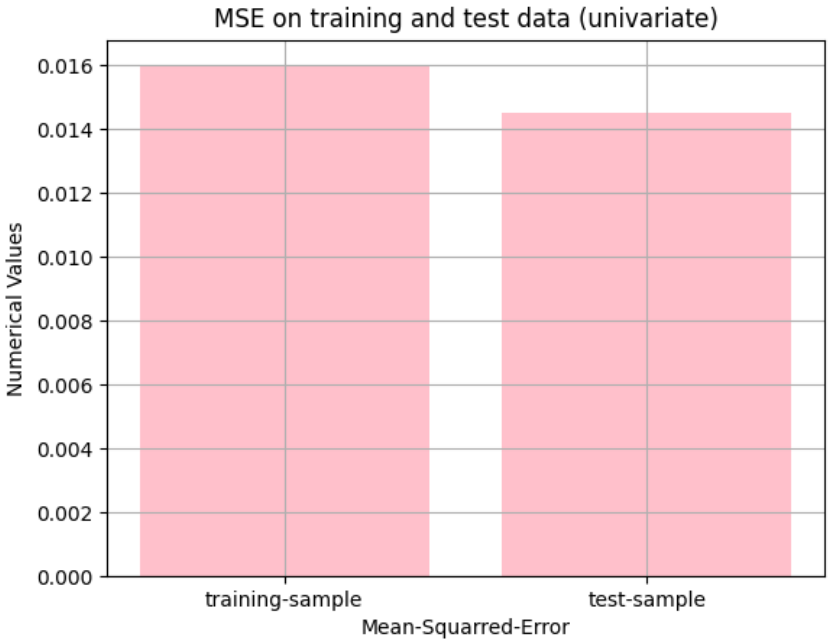
Inferences:

I. The model trains on training data for 1000 epochs and during this training it tries to learn the best weights. (using gradient descent approach)

II. From the above graph, we observe that the average error in each epoch during training decreases sharply between the first 50 epochs (approx) and then it doesn't change much during subsequent epochs.

III. It is because of our choice of learning rate which is reciprocal of no. of epochs. Thus at the start, the model learns at a greater pace than that of in later epochs and finally, it converges when avg-error (0.08) is less than 0.01.

### 3. Mean Squared Error (MSE) on training data and test data:




FigII.4: MSE on training and test data

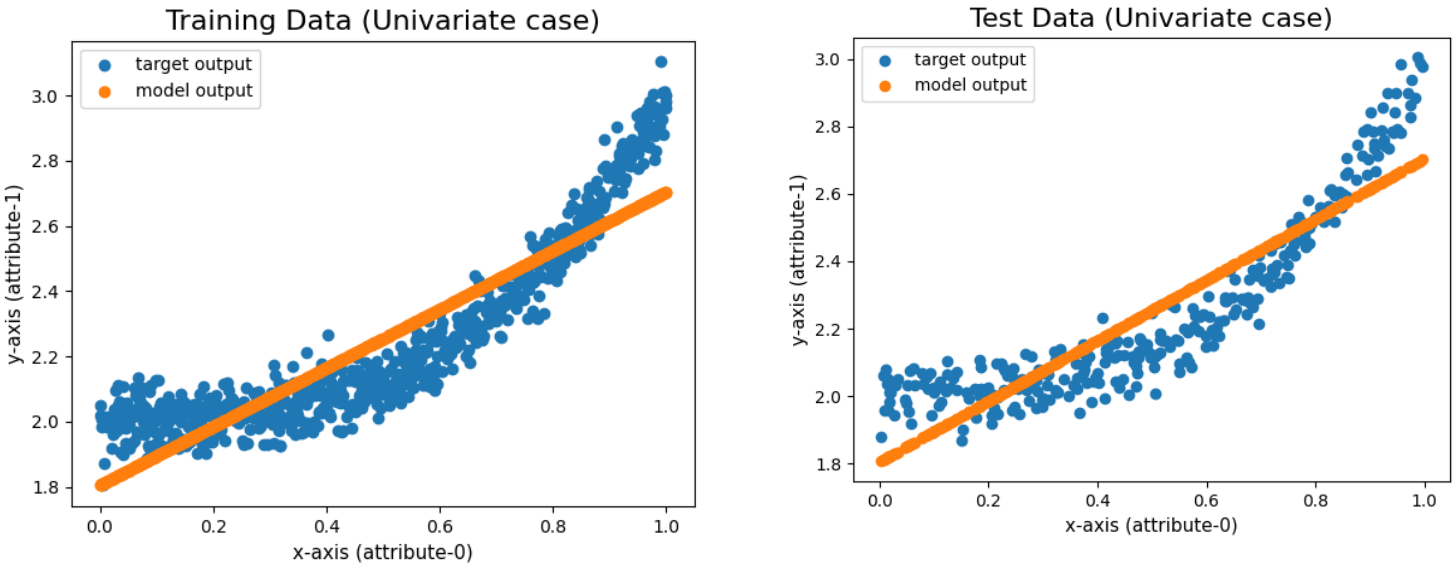
Inferences:

I. From the above bar graph, we can see that the mean squared error is very less on both training and test datasets. This shows that our model is more accurate.

II. However, the built model performs slightly better with test data than the training data.

III. The RMSE for training data = 0.126 and for test data = 0.120. (Again slightly better in the testing phase.)

## 4. Model output and target output for training data and test data




FigII.5: Model output and target output for training data

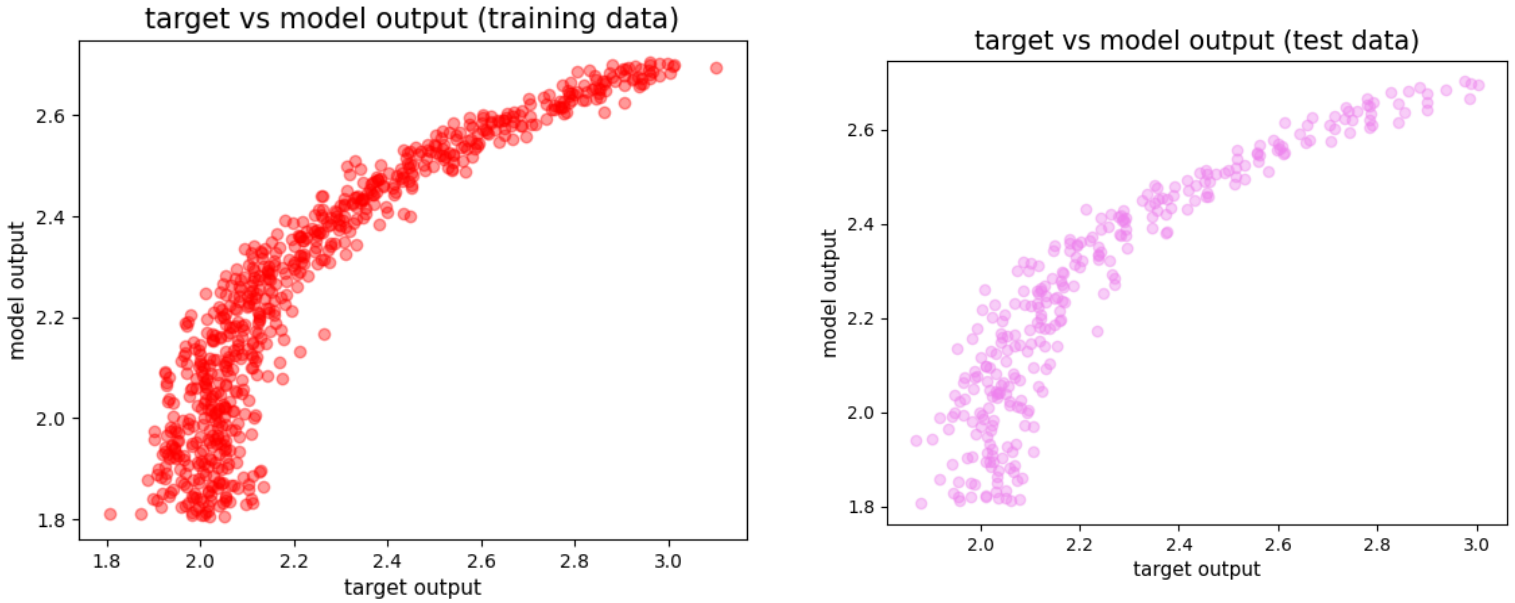
Inferences:

I. Here we observe that the model output approximately captures the results of the target output. However, it does not wholly superimpose target output because the target output is a nonlinear function whereas the model output is linear.

II. The best-fit curve (model output) is a good estimate of the target output considering the fact that it uses a single neuron to learn the relationship.

III. In the case of test data the samples are **sparsely distributed and it is the reason for their low MSE and RMSE values** when compared to training data. (MSE of predicted values by our model)

**5. Target output on the x-axis and model output on the y-axis, for training and test data**




FigII.6: Scatter plot of the target (x-axis) vs model output (y-axis)

Inference:

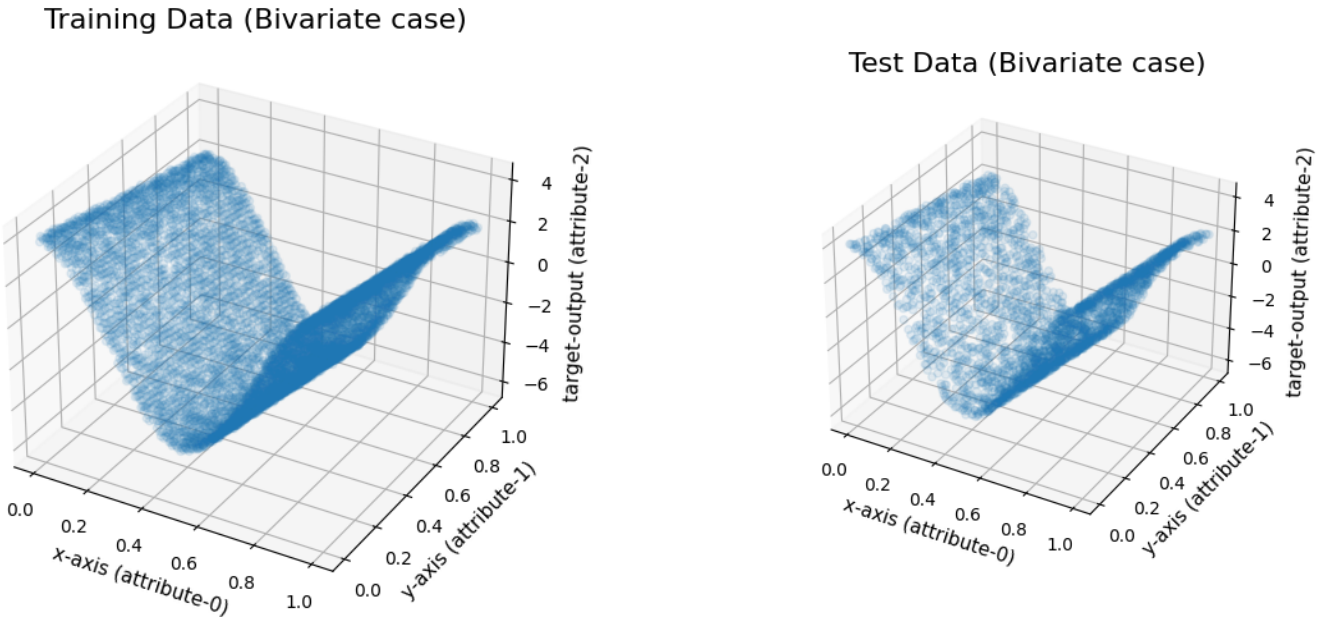
I. From the above scatter plots we observe that the graph between the predicted output and target output is not linear, instead, it is a concave curve.

II. It can be inferred that our model deviates from the expected output in terms of magnitude. And this deviation decreases at a slow rate.

III. Ideally, if the linear regression model has to be perfectly accurate it should have given a linear scatter plot with slope = 1.
But such performance is not appreciated as it leads to overfitting of the model and thus less generalization ability.

IV. Since the training and test input data is not much scattered (i.e. less variance) our model works fine and has less prediction root mean square error.

## 2.) Dataset 2: 2-dimensional (Bivariate) input data




FigII.7: scatter plot of input variable with target variable (Bivariate Case)

**Inferences:**

I. From the above scatter plot, we can see that the target output is a nonlinear surface.

II. Target output seems to be less dependent on both the features (plotted on the x-axis and y-axis). It is also evident from their correlation.

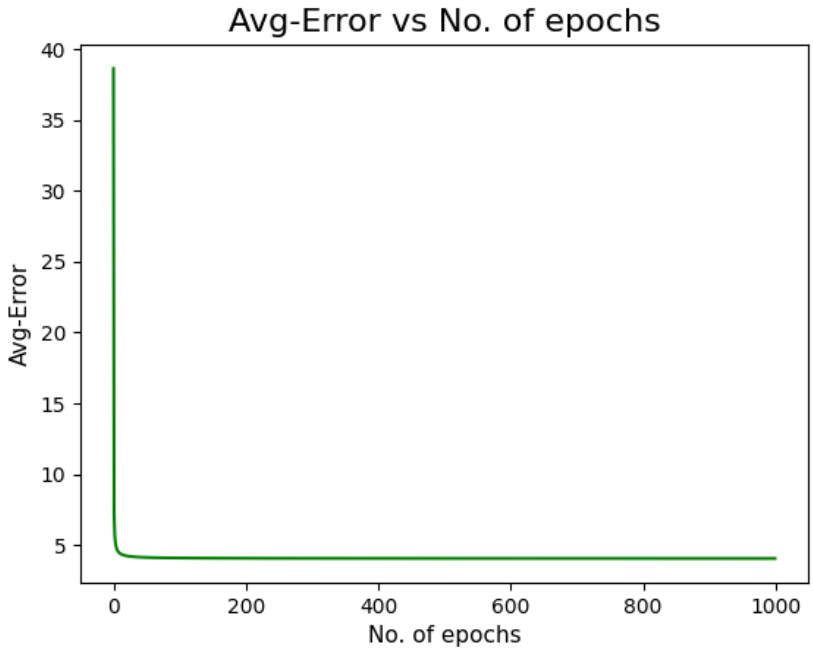
III. correlation(attribute-0, target) = 0.0034 (poorly correlated or uncorrelated)
correlation(attribute-1, target) = -0.21 (negatively associated)

IV. We want to do regression analysis on the above data and data does not have a high association with input features so the learned model would not be perfect.

### 1. Linear Regression on bivariate input data

The value of weights and bias for the bivariate regression model run on the above training data is $\mathbf{w} = [\ 0.11767659\ \ 0.04420203\ -2.05969383]^T$

2. **Average error (y-axis) vs epochs (x-axis)**




FigII.8: Average Error vs no. of epochs

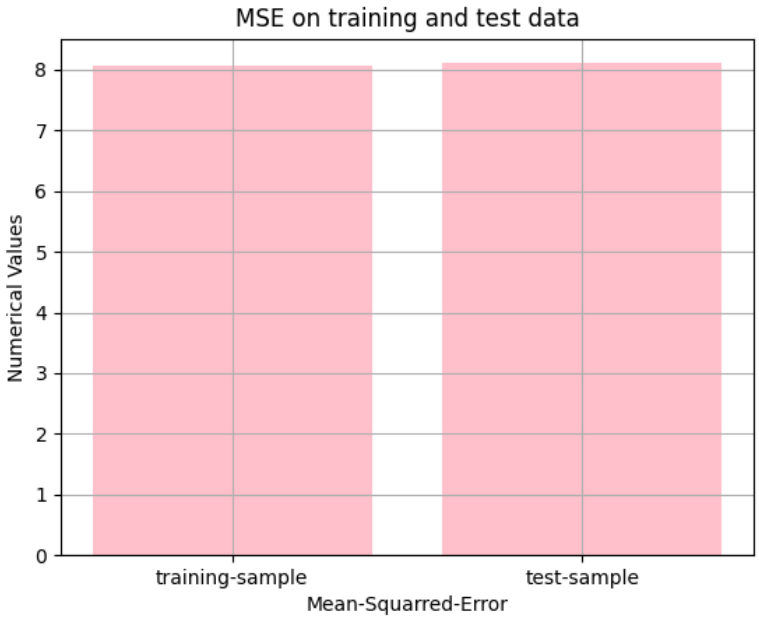
Inferences:

I. The model trains on training data for 1000 epochs and during this training it tries to learn the best weights. (using gradient descent approach)

II. From the above graph, we observe that the average error in each epoch during training decreases sharply between the first 10 epochs (approx) and then it doesn't change much during subsequent epochs.

III. It is because of our choice of learning rate which is reciprocal of no. of epochs. Thus at the start, the model learns at a greater pace than that of in later epochs and finally, it converges when avg-error (is 4.04).

### 3. Mean Squared Error (MSE) on training data and test data:



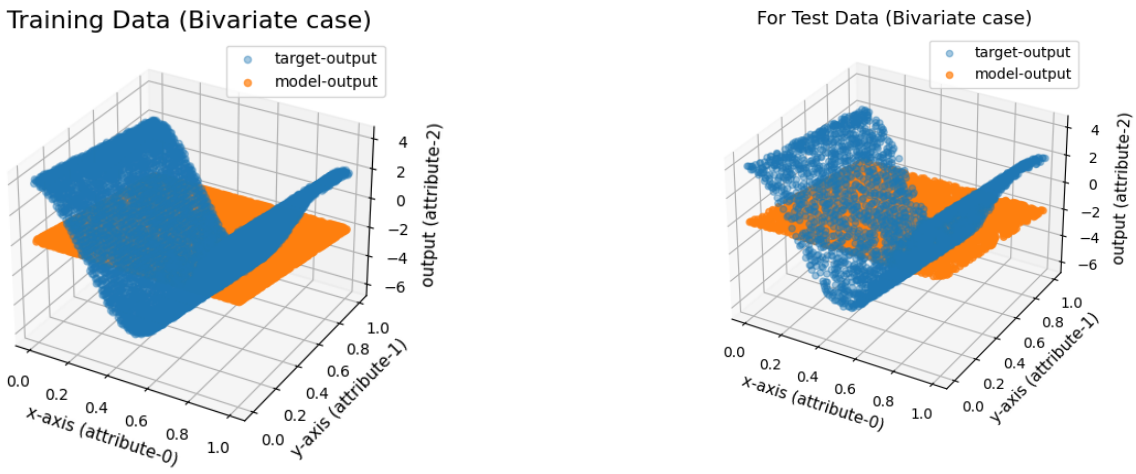

FigII.9: MSE on training and test data (Multivariate)

Inferences:

I. The mean squared error of the model on both training and test data is approximately the same.

II. The RMSE of the model on training data = 2.842 and on test data = 2.847

### 4. Model output and target output for training data and test data



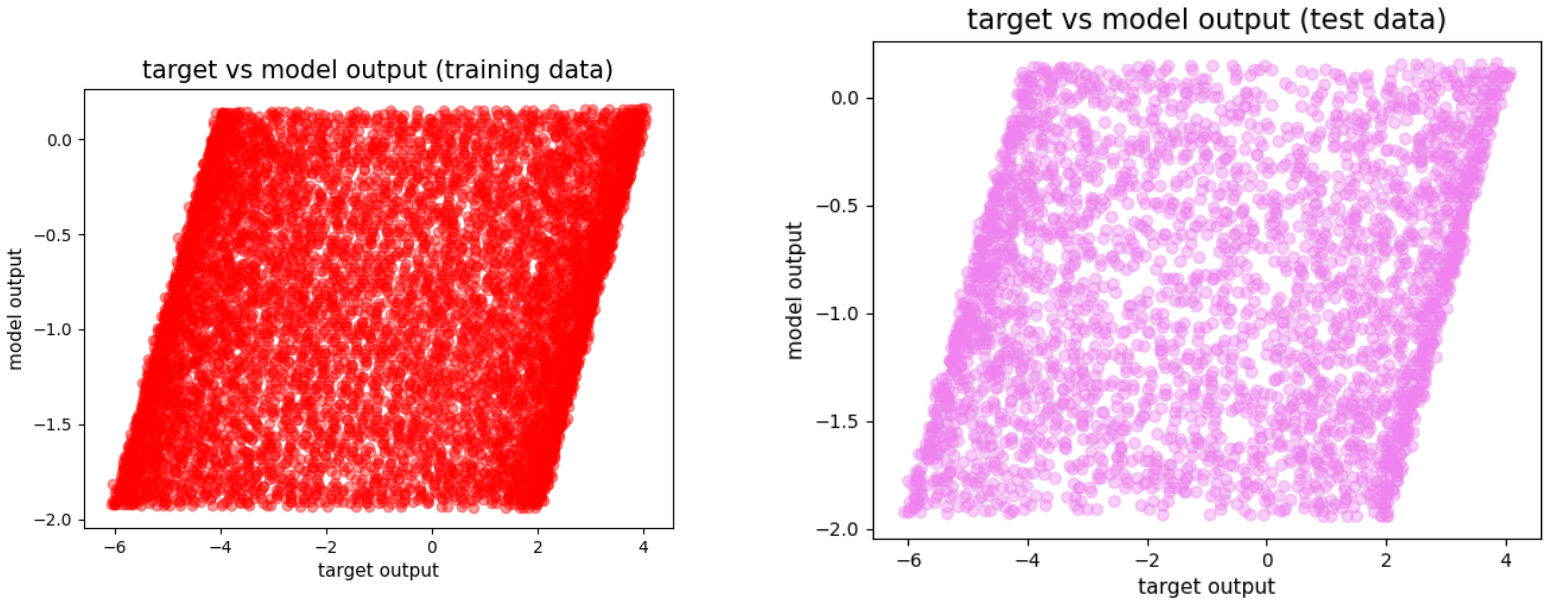

FigII.10: Model output and target output for training data

Inferences:

I. The learned relationship between the 2D input feature and output is a linear surface (a plane) and it does not capture the target output perfectly as the target output is non-linear.

II. It can be seen from the RMSE values. We can validate the same as the range of the target value is approximately [-6, 4] and our error on average is 2.84 (RMSE). It's huge.

**5. Target output on the x-axis and model output on the y-axis, for training and test data**




FigII.11: Scatter plot of the target (x-axis) vs model output (y-axis)

Inferences:

I. As the spread in the above scatter plot is very high, we can confidently say that our linear regression model didn't learn the association between the input variables well.