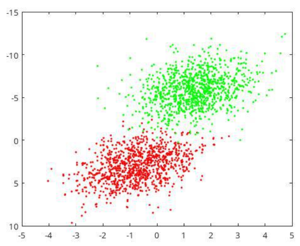
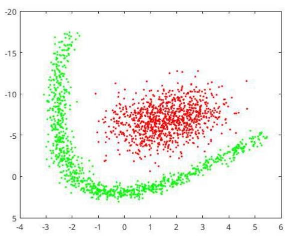
TBMI26 – Computer Assignment Report
Supervised Learning

Weng Hang Wong - wenwo535
Nour Elhouda Qweder - nouqw898

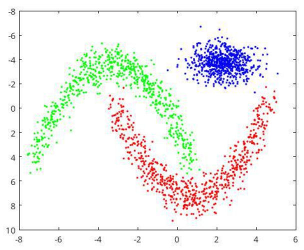
1. **Give an overview of the four datasets from a machine learning perspective. Consider if you need linear or non-linear classifiers etc.**



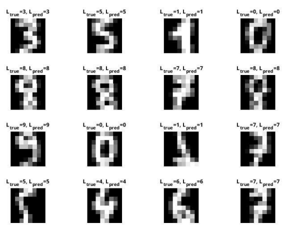
Dataset 1



Dataset 2



Dataset 3



Dateset 4

The 4 datasets given are not linearly seperable. Dataset 1 and 2 has two classes of data, Dataset 3 has 3 classes of data, and Dataset 4 is image data. In order to perfrom classification on these data non-linearly, we can use KNN algorithm on them. However, if we want to classify these data in a linear classifier, we need to transform them to a higher dimensional distubutions so that to make them linearly separable.

2. **Explain why the down sampling of the OCR data (done as pre-processing) result in a more robust feature representation. See http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits**

Down sampling means to reduce the bandwidth of the image data or to compress the data size. In this case, down sampling of the OCR data can reduce the image feature and only keep the most important features of each image. Pre-process the image data can reduce the size significantly so that the whole process can be faster. Moreover, it can increase the accuracy of classification after we down sampling the data.

3. **Give a short summary of how you implemented the kNN algorithm.**

First, we iterate the unclassified samples, and compute the distance from the each training data point towards the sample. The #k closest training data points determine the label of the unclassified samples.

4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**

One possible way to handle draws is, we first sorted the closest to farthest data point to the unclassified sample. Then the voter (data points) are the ones which having the shortest distance to the unclassified sample, e.g. if k=2 with 2 closest voters of label1, then it will be classified as label1, however, if k=2 with 1 voter of label1 and the other is label2, then it will be classified as the one which is the closest.

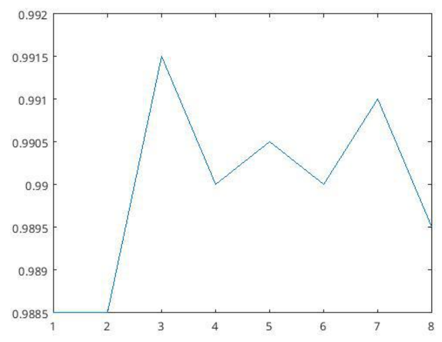
The other way to handle draw is using mode() function, if the two voters are different classes, then it will automatically classifiy it into the cluster with lower value, eg. label 1 and label 2, it will have 1 as output.

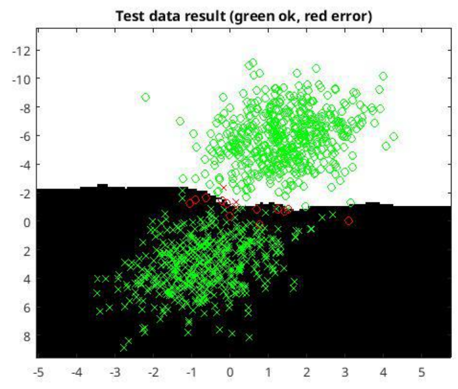
5. **Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.**

We use cross validation to select the best k for each dataset. First, we shuffle the dataset randomly, and split the datset into n folds, for each fold we hold one fold of the data as test data and take the remaining folds as training data, in each loop of the selecting the hold out data, we preform KNN model and evaluate the accuracy of each loop.

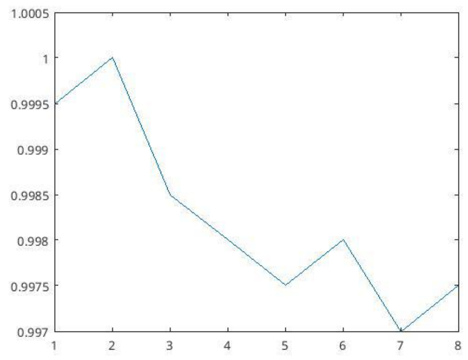
In our case, we set nfold=4, k_range as 1:8 to find out which is the best k for each dataset.

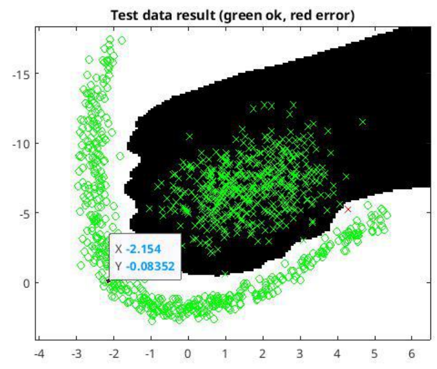
Dataset1 :
According to the plot of cross validation, we have k=3 as our best k
The accuracy of k=3 : 0.9915



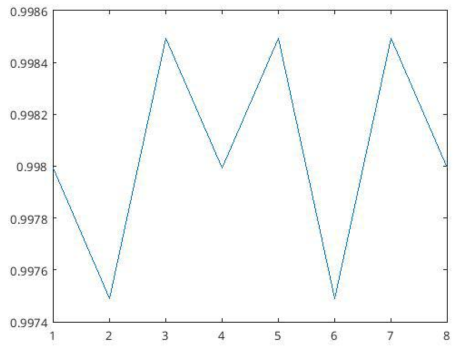


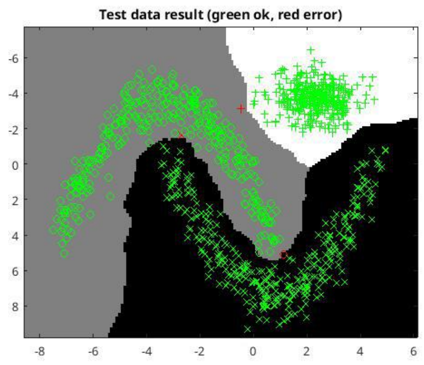
Dataset2:
According to the CV plot below, we have k=2 as best K, the accucary of k=2: 0.9980



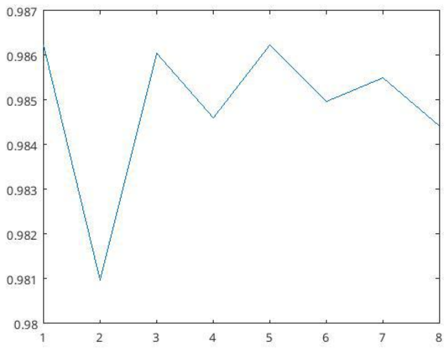


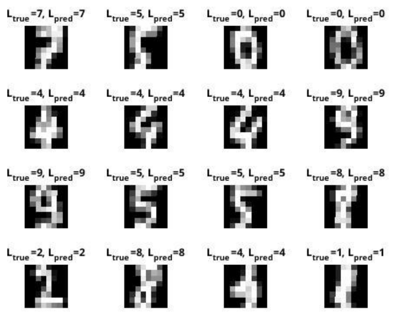
Dataset 3:
We have k=2, k=5 and k=7 as best k, so we select k=2, accuracy = 0.9970





Dateset 4 :
We have k=5 as best k, accuracy = 0.9798





6. **Give a short summary of your backprop implementations (single + multi). You do not need to derive the update rules.**

   Using back-propagation, we can find the gradient of w weight matrix (and v weight matrix in multi-layer), After that, we can use this gradient w and v to update the weight w and v by multipling learning rate. eg. w = w – learningRate*gradient_w. Using the new weight we can evaluate the errors from the training.

Parameters	**Single layer**	**Multi-layers**
Iterations	1000	30000
learningRate	0.009	0.0009
number of input and output nodes	Input : 1000 x 2 nodes Output : 1000 x 1 nodes	Input: 2770 x 65 nodes Output: 2770 x 1 nodes
number of layers	1 layer	35 layers
weight initalization	3 x 2 matrix of small random values	W matrix : 65 x 35 V matrix : 36 x 10 small random values

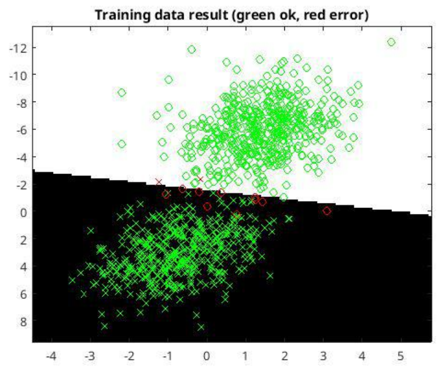
The main differences of single layer and multilayers is single layer only perform good on linearly separable data, while multilayers can work with multi-dimensional data, for example, using tanh() function when compute the gradient for the ouput layer.

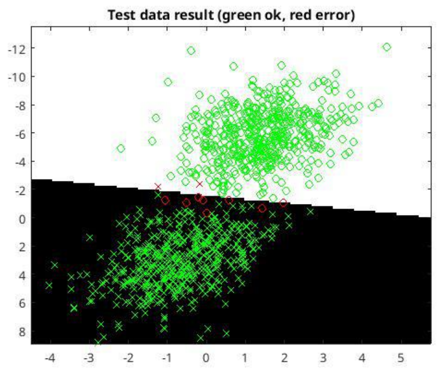
7. **Present the results from the neural network training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations and number of hidden neurons. Include images of your best result for each dataset, including parameters etc.**

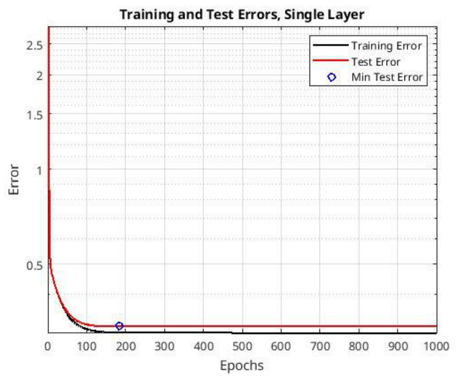
   Single layer:

   We got the result that reach the accuracy only on dataset 1. Since using single layer perceptron on neural network meaning that it's only a linear classifier. On the other datasets, they are non-linearly seperable so the single layer perceptron does not perform well on them.

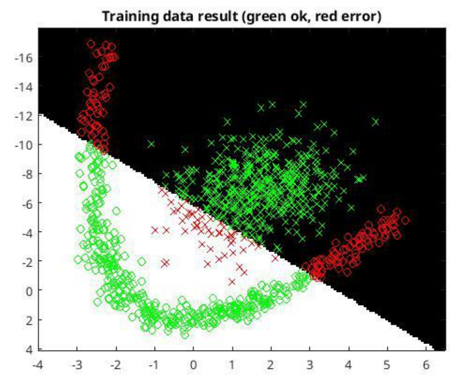
   Dataset 1:
   Iteration=1000; learningRate:0.009;
   W0=random value in a range of [-0.2:0.2].
   The train accuracy is 0.993
   The test accuracy is 0.988

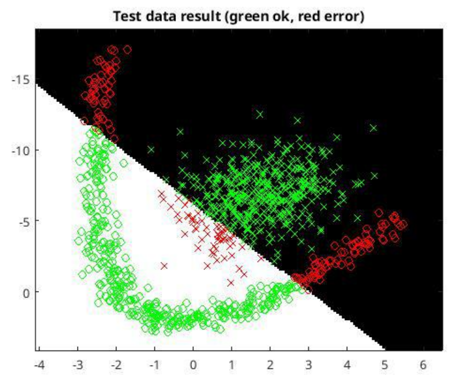


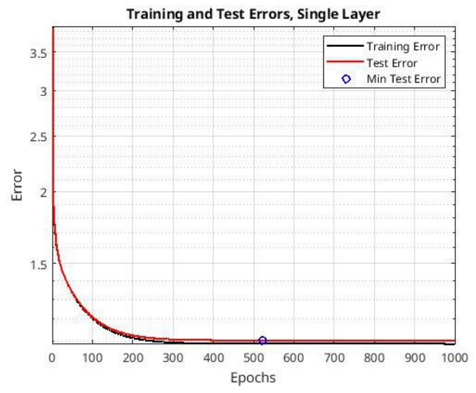




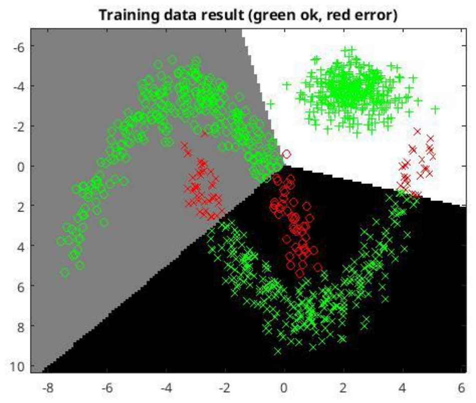
   Dateset 2:
   The train accuracy is 0.817
   The test accuracy is 0.818

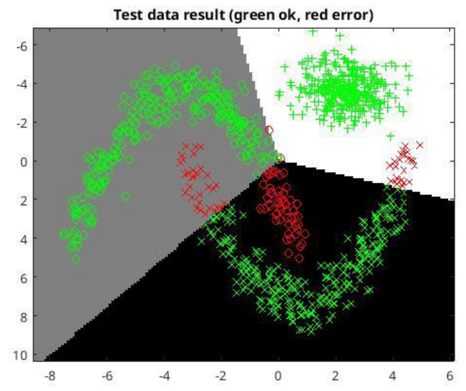


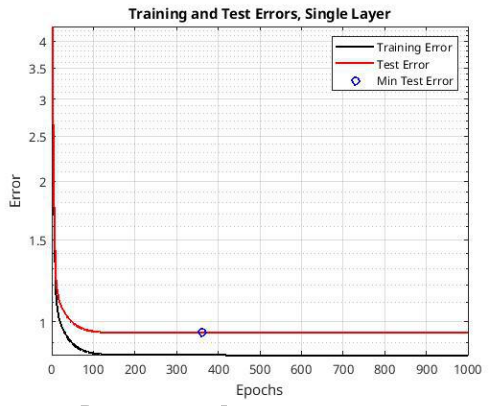




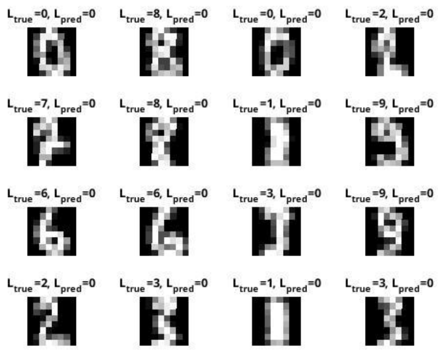
Dataset 3 :
The train accuracy is 0.88889
The test accuracy is 0.87888

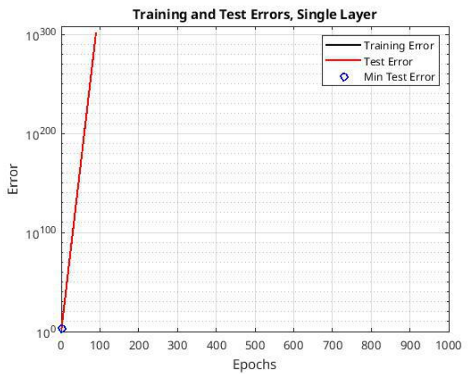






Dataset 4 :
The train and test accuracy is 0.1

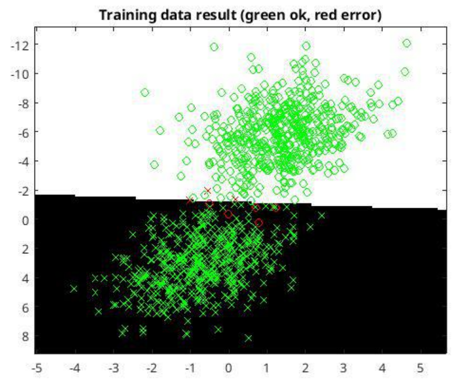


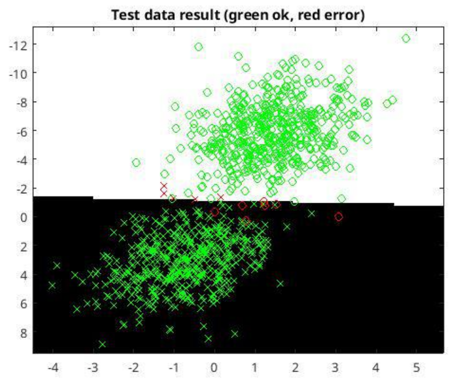


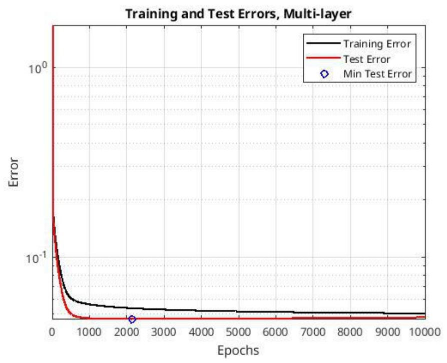
In multi-layer:

Dataset 1:
Train Accuraccy: 0.992, Test Accuracy: 0.991

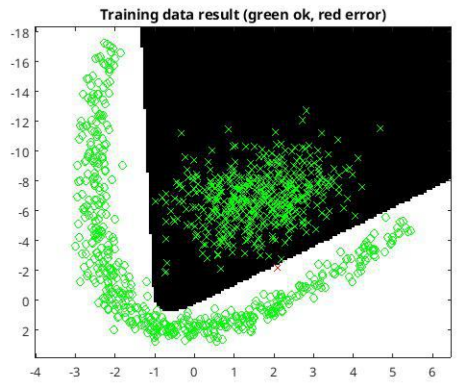
numHidden = 1; since it’s linearly separable data, so it doesn’t need a higher number of hidden layers. But we are using multi-layer network here so we have to set it up to at least 1.
numIterations = 5000; too many iterations will make the model over fitted.
learningRate = 0.05; the learning rate determine the speed of finding the local minimum, so 0.05 is a suitable value that won’t be taking too much of time but still successfully finding the minimum.

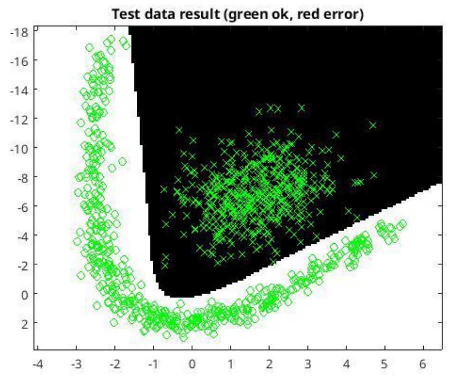


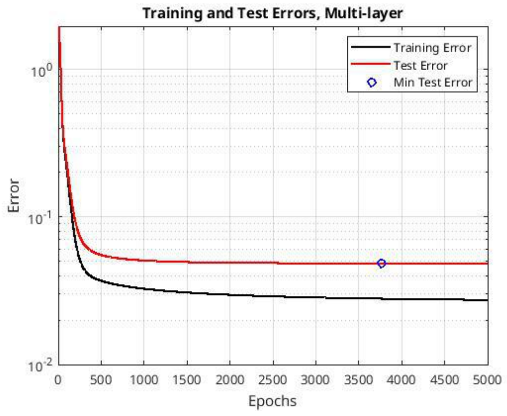




Dataset 2:
Train accuracy: 0.999 , Test accuracy: 1
numHidden = 2; this data set is not linearly separable, so we need more than 1 hidden layers on this multi-layer network.
numIterations = 5000; same reason as above
learningRate = 0.05; same reason as above

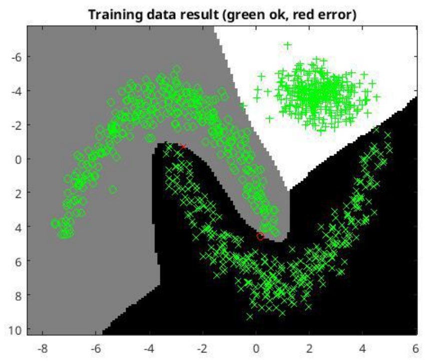


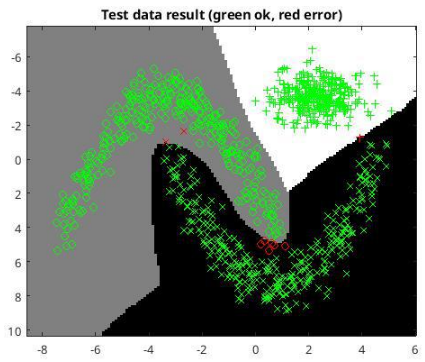


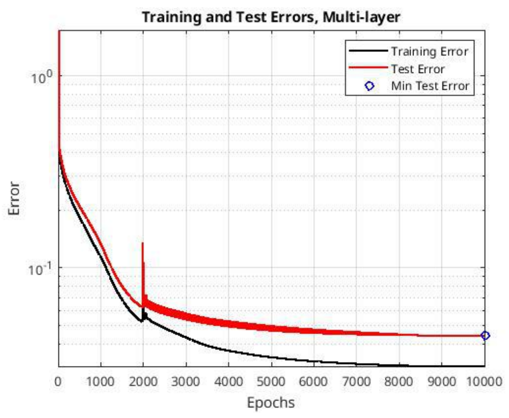


Dataset 3 :
Train Accuracy : 0.996, Test Accuracy: 0.988

numHidden = 8; This dataset has 3 classes, so we need a larger number of hidden layers in order to perform well on the classification.
numIterations = 10000; we need more iteration here to avoid the model become underfitted.
learningRate = 0.05; same as above

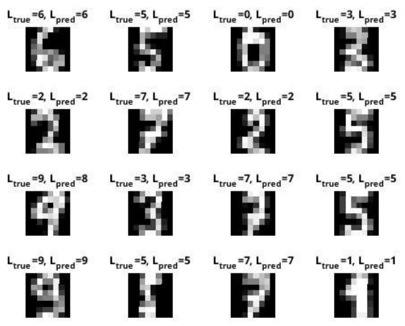
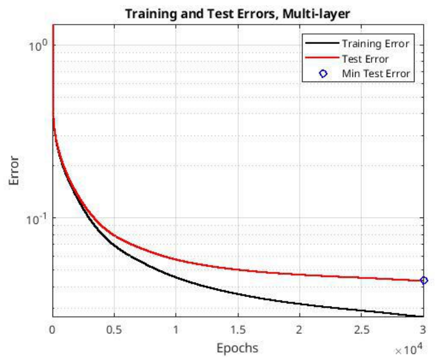






Dataset 4 :
Train Accuracy: 0.98448, Test Accuracy: 0.96005

numHidden = 35; the input layer and output layer is very large on this case, in order to train this data set, the number of hidden layers should between the number of input and output layers((65+10)/2), so we set it as 35 hidden layers here.
numIterations = 30000; in order to perform well on this large size of training data, we tried 30000 iterations to train well on this case
learningRate = 0.0009; a small value of learning rate can make the training more reliable although it's very time consuming.

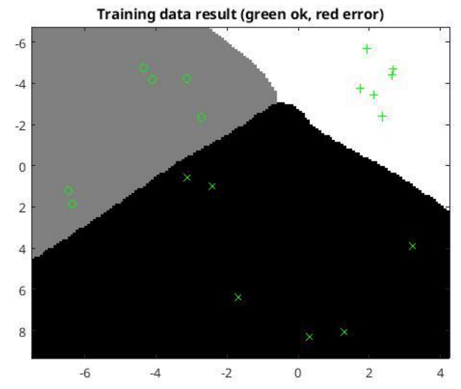

L_{true}=6, L_{pred}=6
L_{true}=5, L_{pred}=5
L_{true}=0, L_{pred}=0
L_{true}=3, L_{pred}=3
L_{true}=2, L_{pred}=2
L_{true}=7, L_{pred}=7
L_{true}=2, L_{pred}=2
L_{true}=5, L_{pred}=5
L_{true}=9, L_{pred}=8
L_{true}=3, L_{pred}=3
L_{true}=7, L_{pred}=7
L_{true}=5, L_{pred}=5
L_{true}=9, L_{pred}=9
L_{true}=5, L_{pred}=5
L_{true}=7, L_{pred}=7
L_{true}=1, L_{pred}=1


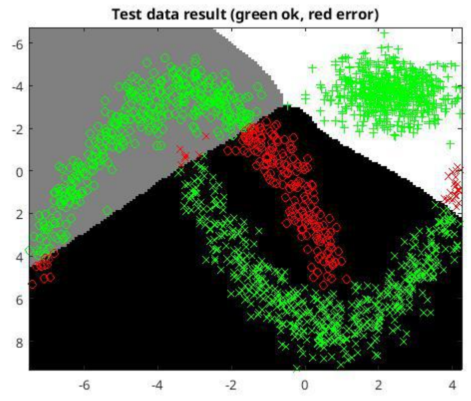
8. **Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.**

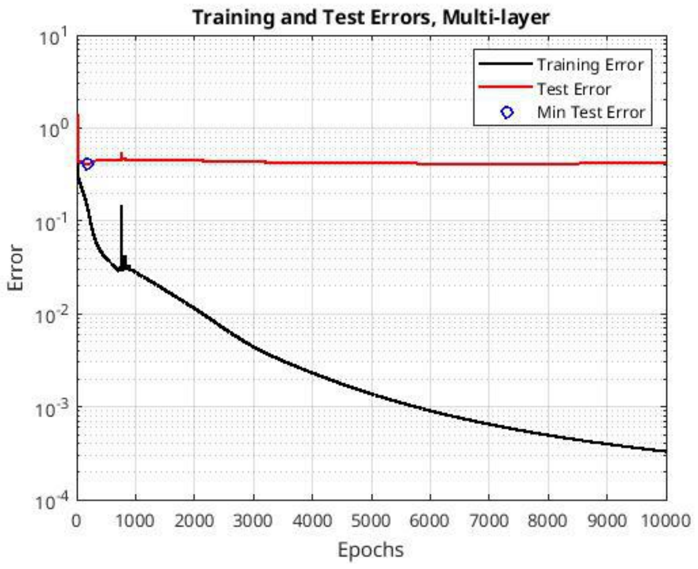
In this case, we use dataset 3 and perfrom multi-layer neural network. We splited the data into 100 bins, and use the first bin as training data and the remaining data as test data.

In non-generalizeble solution, it will be very obvious that the test error will is greater than the train error, while the training accuarcy is close to 1. It means that the model is overfitted on the training data, but it's not generalized to the test data and become underfiting on the test data.

Training accuracy : 1
Test accuracy : 0.86027







9. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**
   Pros:
   K-NN classifier is comparatively simple to implement and perform very well generally on these 4 datasets than using single/muti-layers backprop.

   Neural Networks is good with non-linear classification, and it is flexible with large amount of input and output with large amount of layers.

   Cons:
   K-NN classifier is relatively computationally as it searches every point when computing the nearest distance, so when the data is huge than it won't be the most efficient classifier.

   Neural Network is very time consuming when training higher dimensional data. Since it costs a large amount of hidden layers and number of iterations. So it doesn't fit on normal computation CPU.

10. Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.

Using K-NN algorithm has a very well performances on training and predicting the four data sets here, so we think in algorithm-wise K-NN is not a bad option. Neural Network also performs well on these cases, but it’s still a little bit worse than K-NN algorithm. So we can work on the pre-processing on Neural Network. For example, the non-linear algorithm often require more complexity data in order to get a better result, therefore, we could try to split higher portion on training data.