TBMI26 – Computer Assignment Report Supervised Learning

Deadline – March 14 2021

Author/-s:
Måns Aronsson (manar189), Niclas Hansson (nicha207)

In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. Please upload the document in PDF format. **You will also need to upload all code in .m-file format**. We will correct the reports continuously so feel free to send them as soon as possible. If you meet the deadline you will have the lab part of the course reported in LADOK together with the exam. If not, you'll get the lab part reported during the re-exam period.

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

   Data set 1: The data consists of 2000 samples and each sample has 2 features. It has 2 classes, and these classes are positioned into different clusters in the feature space. The clusters are linearly separable. There is an overlap of the samples between the different clusters, so a classifier is expected to yield some false classifications.

   Data set 2: The data consists of 2000 samples and each sample has 2 features. It has 2 classes, and these classes are positioned into different clusters in the feature space. The clusters are **not** linearly separable. There is **no** overlap between the samples of the clusters, so an optimal classifier can produce a perfect classification of the data.

   Data set 3: The data consists of 2000 samples and each sample has 2 features. It has 3 classes, and these classes are positioned into different clusters in the feature space. The clusters are **not** linearly separable. There is a small overlap between the samples of the clusters, so a classifier is expected to yield a few false classifications.

   Data set 4: The data consists of 5620 samples and each sample has 64 features. It has 10 classes. Due to the high dimensionality, it is **not** 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**

   Because it reduces dimensionality and gives invariance to small distortions.

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

   A matrix containing the distance from each sample in the training data and test data was calculated. From this matrix, the k smallest distances for each test sample and their indices were saved in two separate matrices. The correct classes of these samples were then saved in another matrix. That matrix was then used to calculate the frequency of each class occurrence for each sample in the test data. These frequencies were then used to classify the test data by finding the maximum frequency for each sample and returning its index. The value of the maximum frequency was then used to check if there were any draws. These were then handled by an algorithm described in question 4).

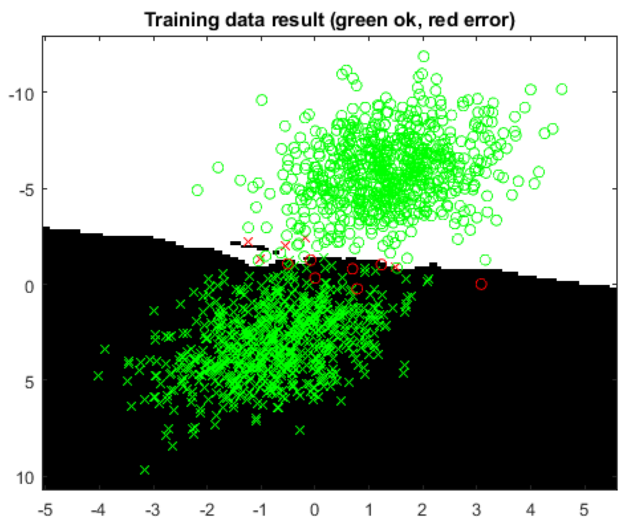
4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**

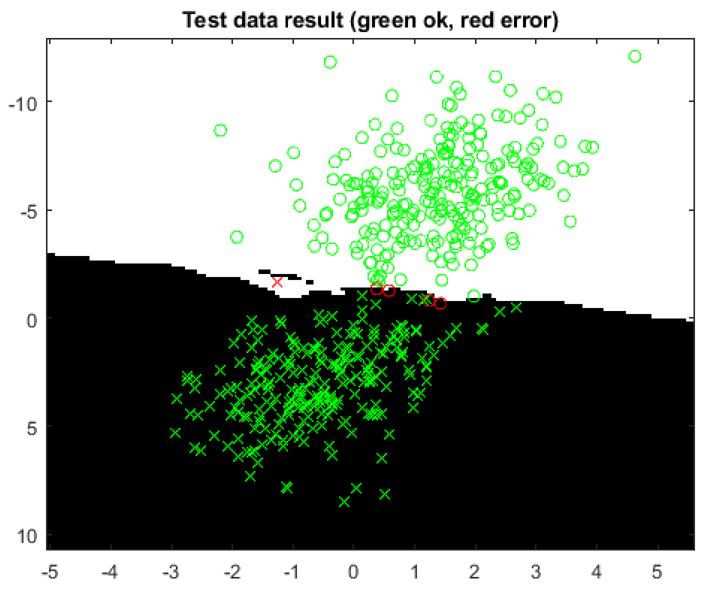
   Only the classes that are represented in the draw were considered. The mean distance of the affected train samples for each class was calculated. The test sample was then classified as the class with the minimum mean distance. If there was a draw, the class with the closest single sample wins.

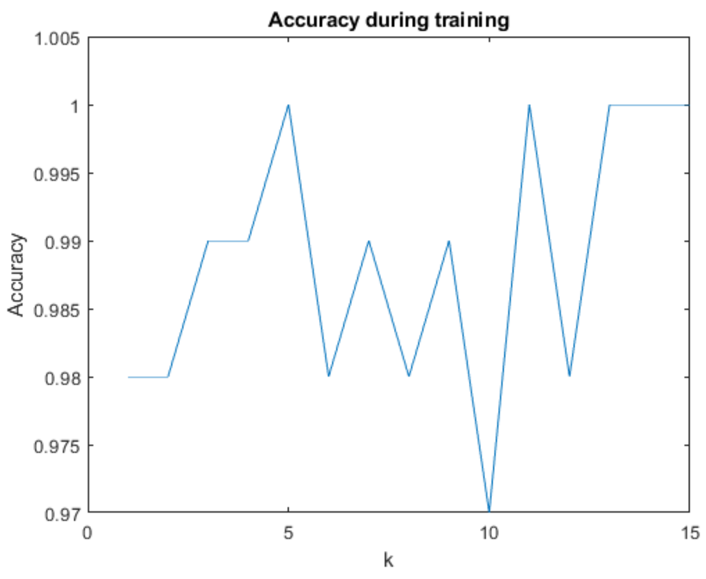
5. **Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.**

   The data was divided into 20 bins. 5 bins were used as test data, 14 bins were used as training data and 1 bin was used as validation data. Starting a k=1, the accuracy of the kNN algorithm was calculated and saved. The validation data was then swapped with one of the bins from the training data and the test was repeated for k=2. This process was then repeated until k=15. The k with the best accuracy one the validation was then used on the test data. This yielded the following results:

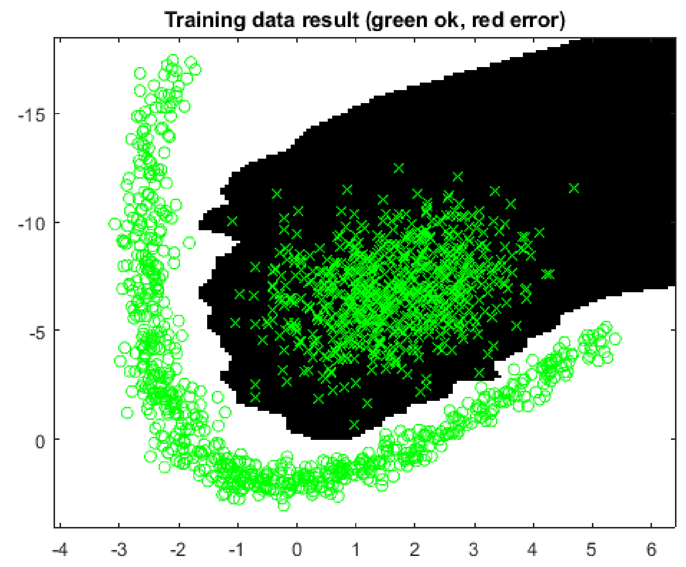
Data set 1:   99.0% for k=5

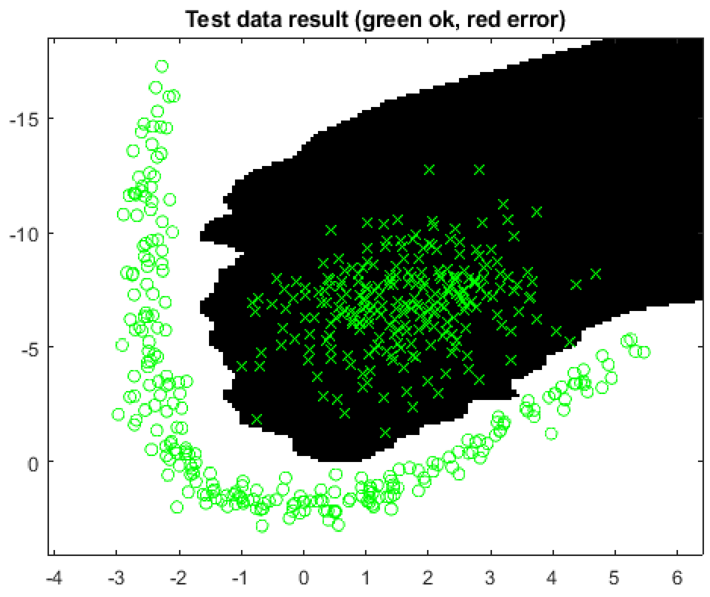


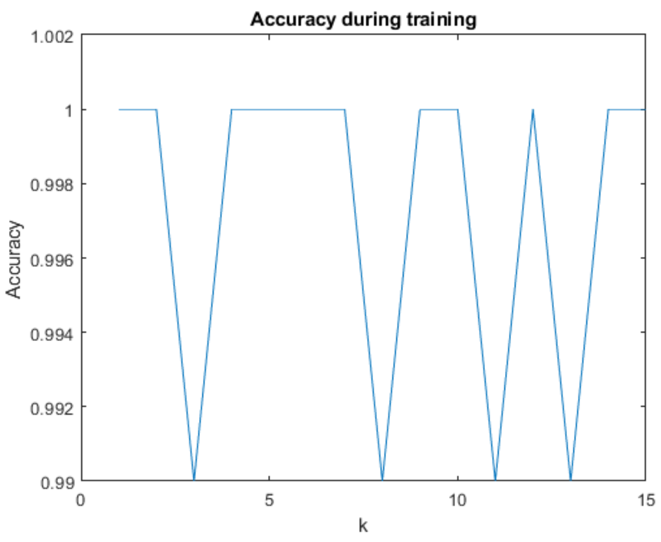




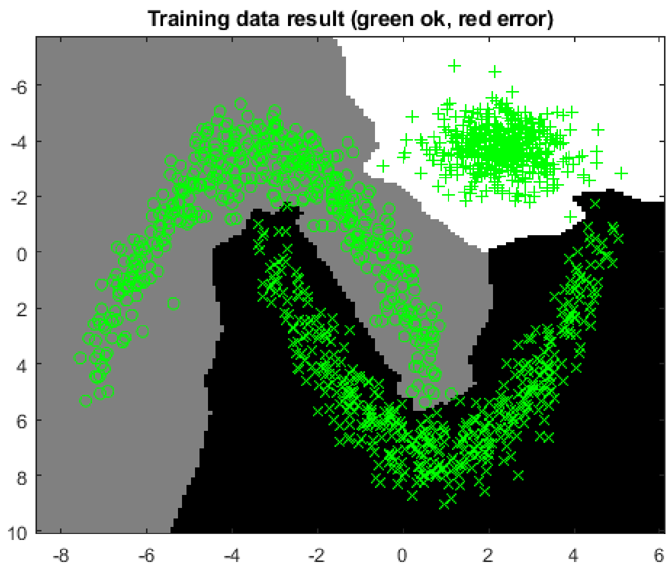
Data set 2: 100% for k=1

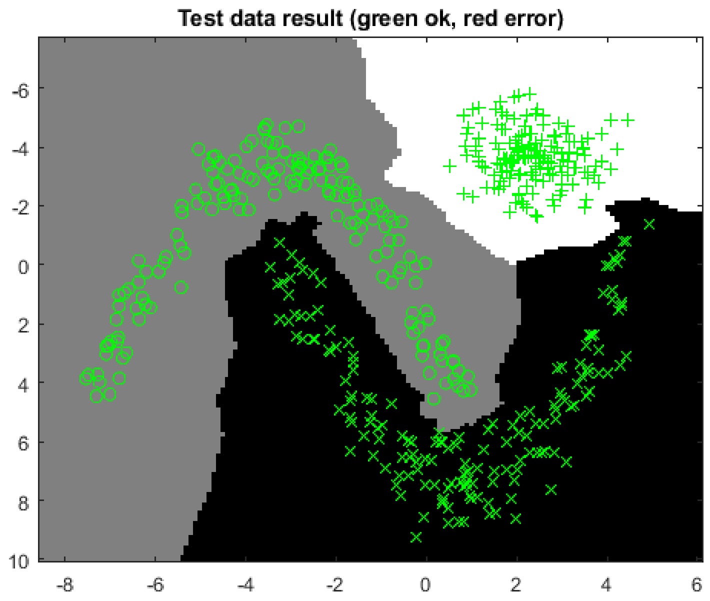


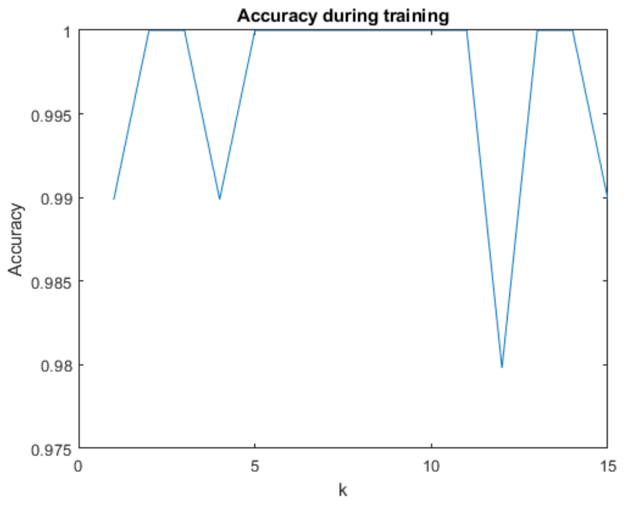




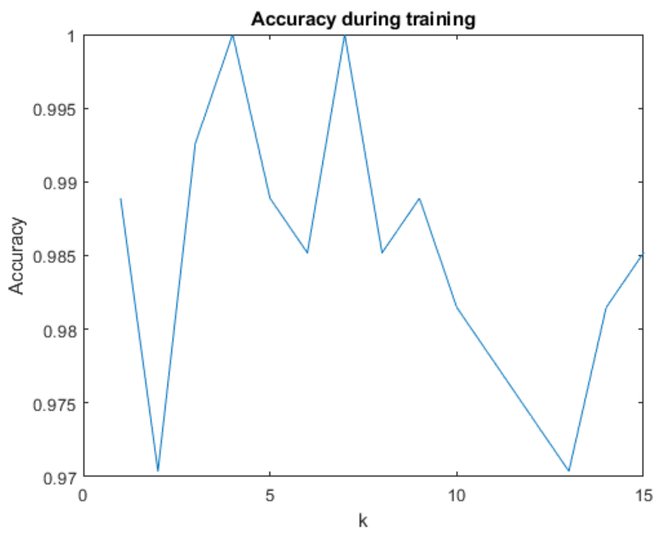
Data set 3: 100% for k=2







Data set 4: 99.04% for k=4



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

   20% of the samples were labeled as test data and the rest were labeled as training data. The training and test data was modified by adding a feature vector of ones as bias weights. The weight matrices of the networks were initialized with random weights with a normal distribution close to zero. These matrices dimensions change dynamically to fit each specific dataset.

   The weight matrices were then updated in the training function by the gradian decent method. For the multilayer network this means updating one gradian for each weight matrix.

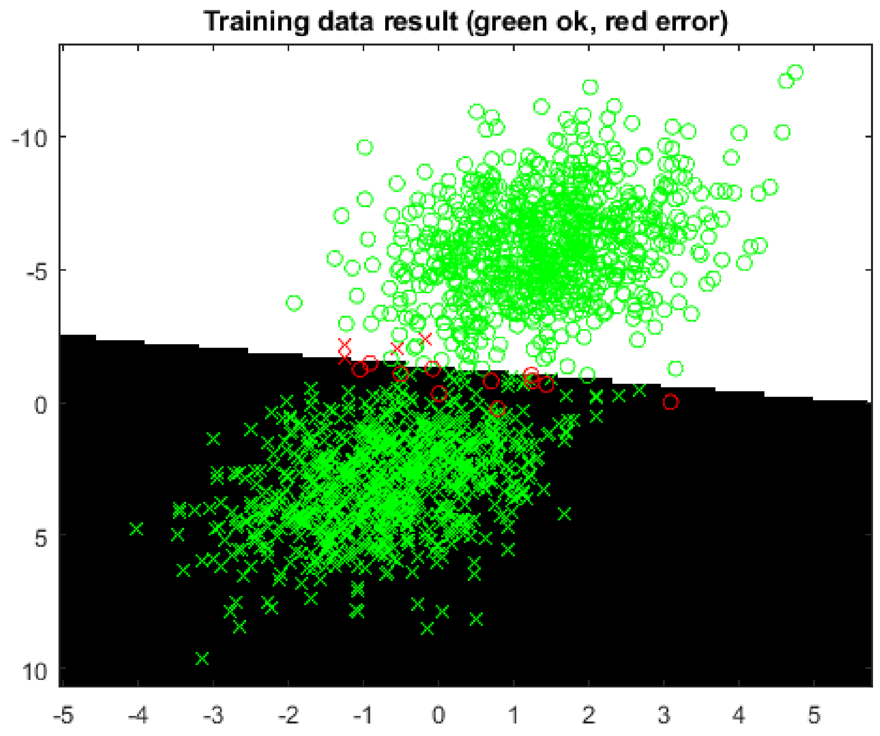
7. **Present the results from the neural network training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations and number of hidden neurons. Include images of your best result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

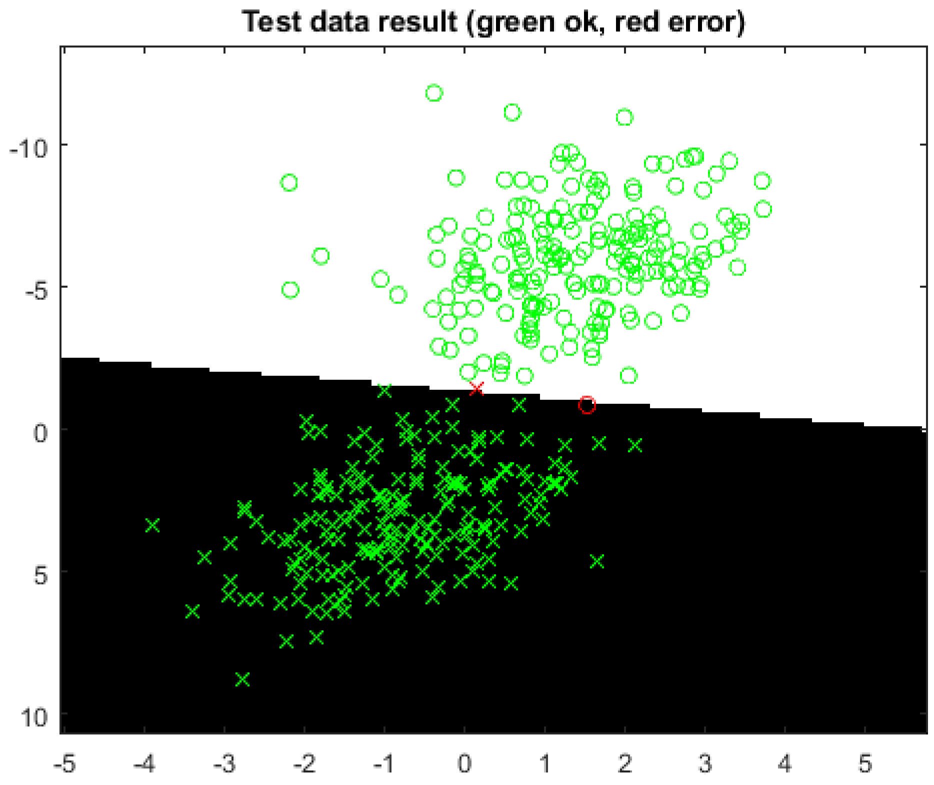
   The goal was to use as simple and fast of a network as possible that still achieved the accuracy criteria. Therefore, a single layer network should be used over a multilayered network even if both produces a good result. The same can be said about the number of hidden nodes in a multilayered network. More nodes add complexity which leads to longer training time. It also comes with increased risk of overfitting. The process of choosing parameters for each dataset were as follows:

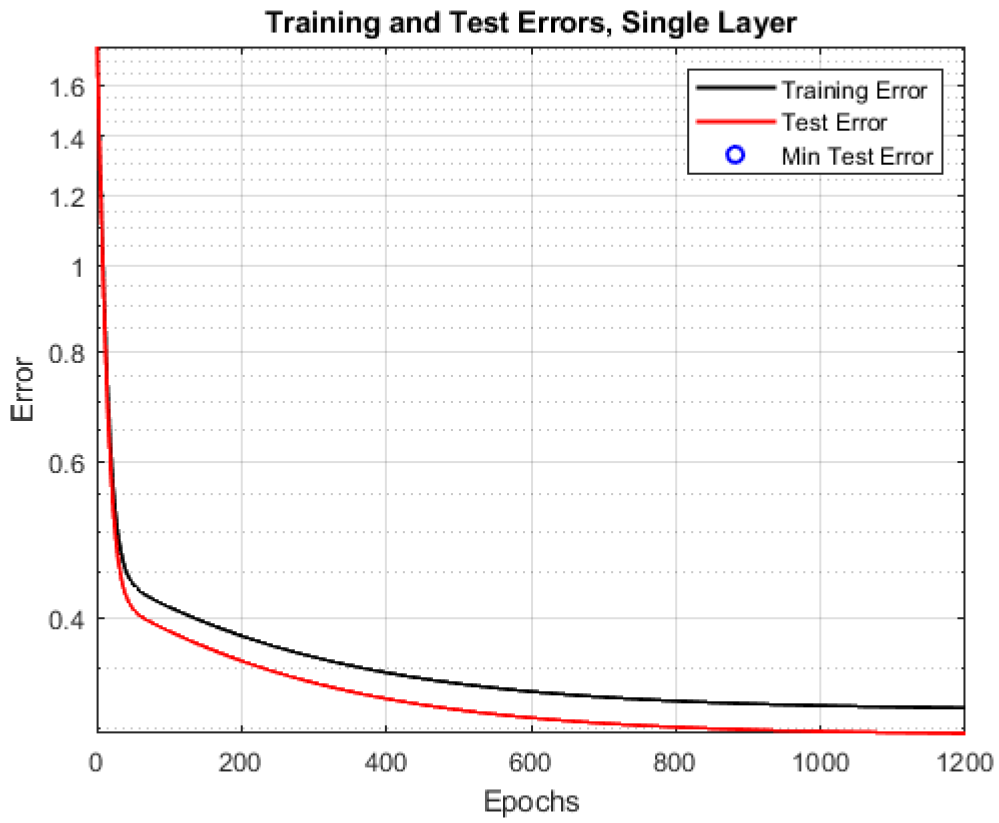
   1. Look at the data. Use a single layered network if linearly separable. Otherwise, multilayered.
   2. Run the network once with few hidden nodes and epochs.
   3. Look at the accuracy plot. If the test accuracy has not plateaued yet, increase the number of epochs, and run again. Repeat until the test accuracy plateaus.

4. If test accuracy plateaus before good result is achieved, increase number of hidden nodes.
5. Optimize the learning rate and epochs to decrease training time.

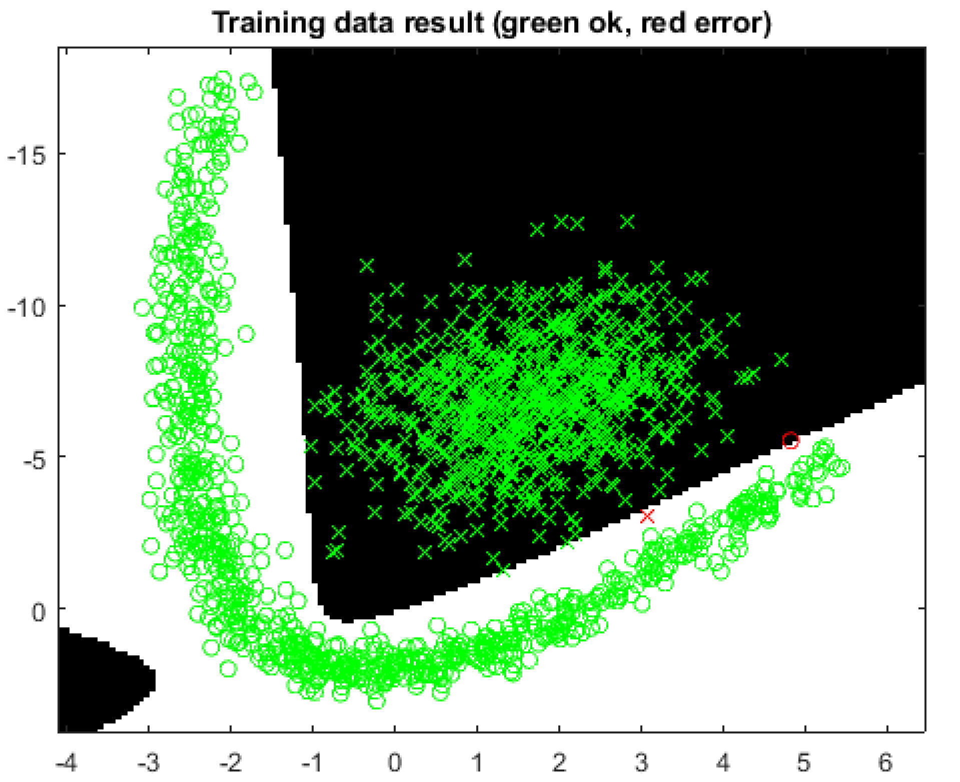
<u>Data set 1:</u> A single layer network was used as the classes are linearly separable. An accuracy of 99.5% was achieved after 1200 epochs and a learning rate of 0.001.

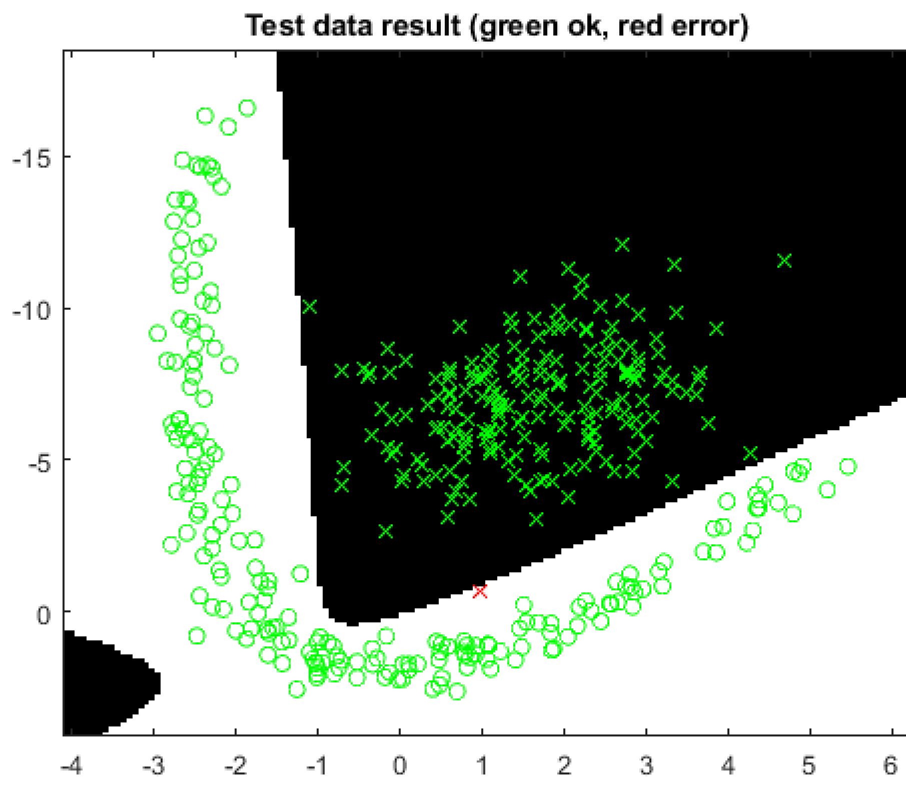


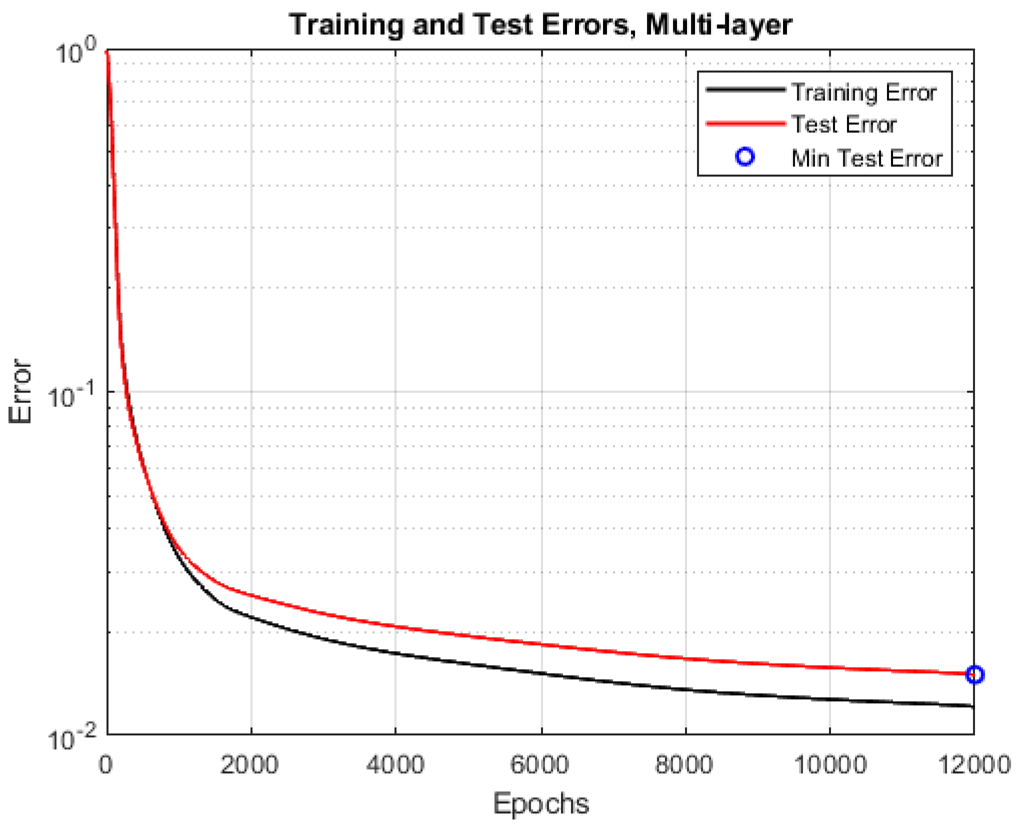




Data set 2: A multilayer network was used with 6 hidden nodes. An accuracy of 99.75% was achieved after 12000 epochs and a learning rate of 0.05.

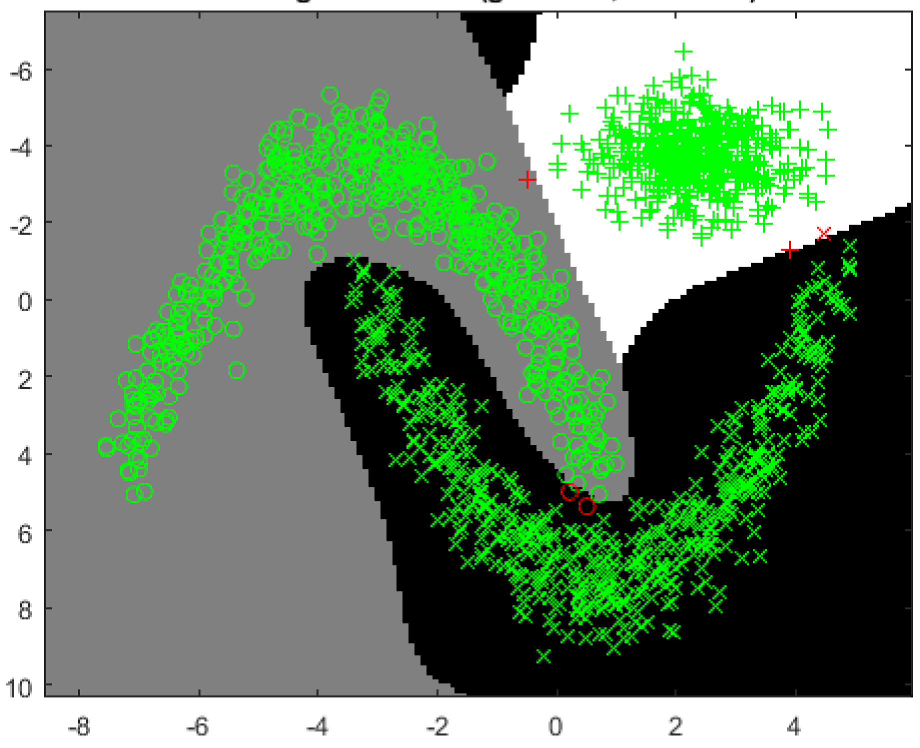




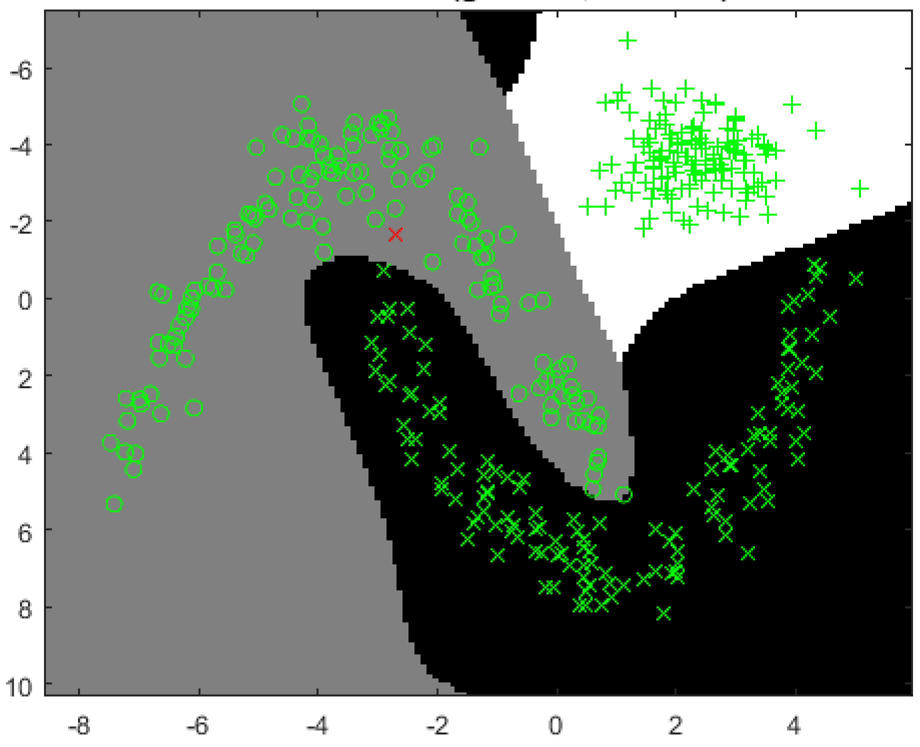


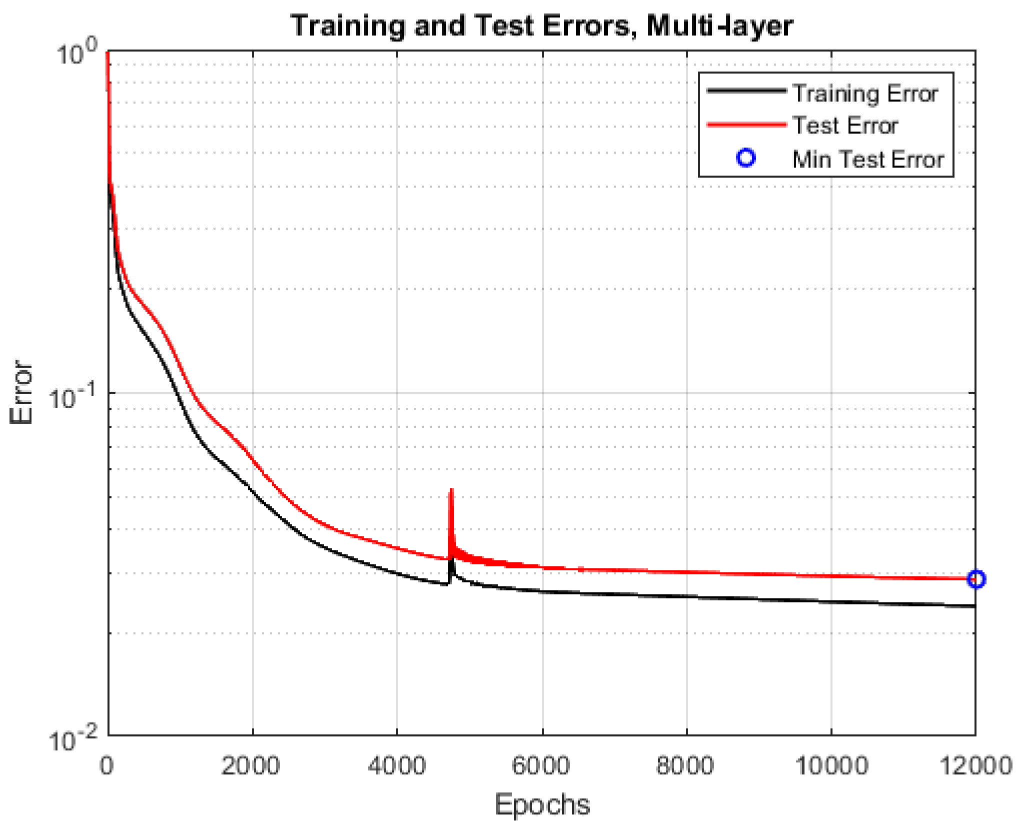
Data set 3: A multilayer network was used with 8 hidden nodes. An accuracy of 99.749% was achieved after 12000 epochs and a learning rate of 0.1.

Training data result (green ok, red error)

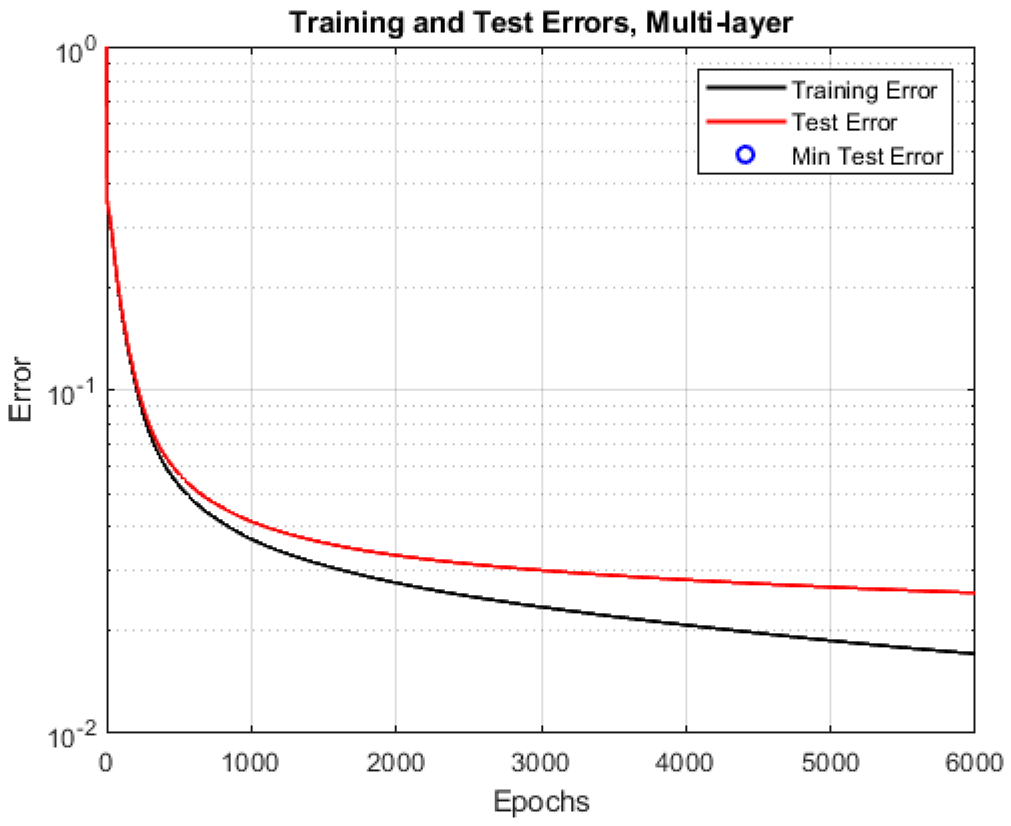


Test data result (green ok, red error)



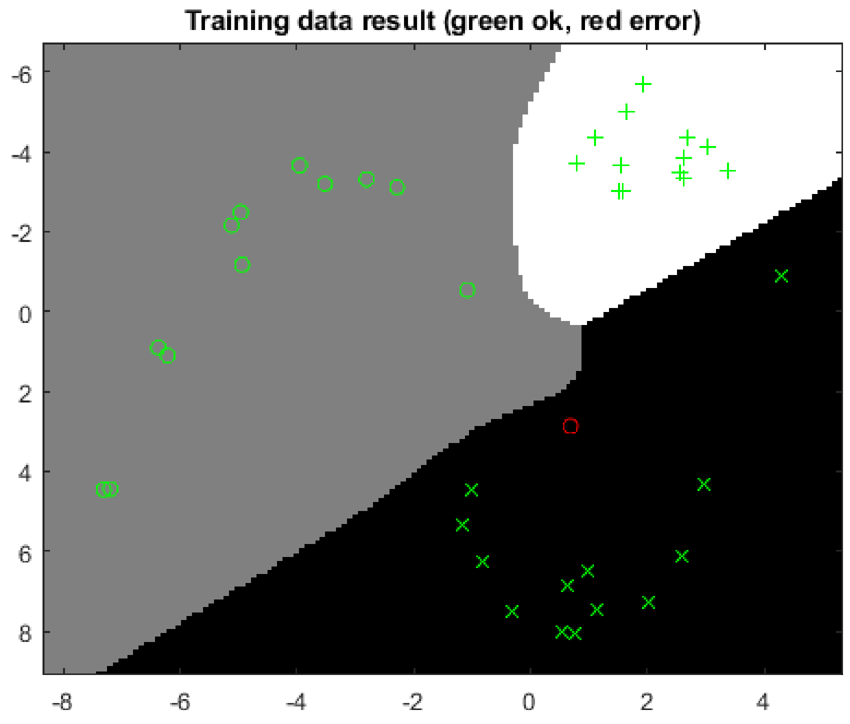


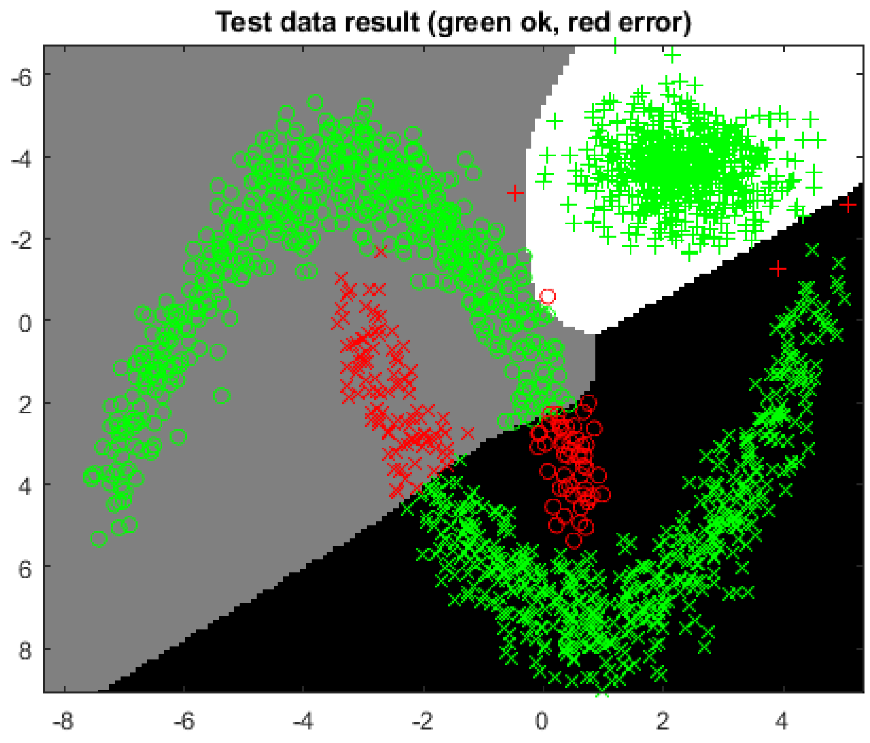
Data set 4: A multilayer network was used with 64 hidden nodes. An accuracy of 97.364% was achieved after 6000 epochs and a learning rate of 0.05.

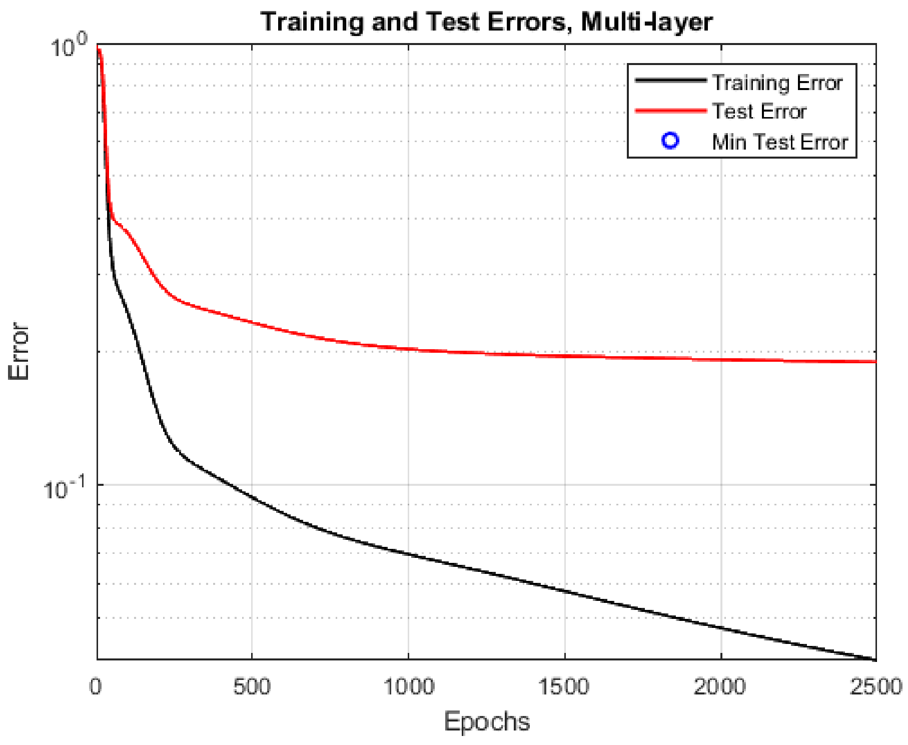


6. **Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.**

A multilayer network was used with 8 hidden nodes. An accuracy of 91.732% was achieved after 2500 epochs and a learning rate of 0.05. It was non-generalizable because the training data contained 39 samples which was too few to get a consistent result. These training samples could never achieve the accuracy criteria no matter how long it is trained.







7. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**

   kNN is easy to implement and generalizes well for different datasets, independently if the sets are non-linearly separable or linearly separable. However, it is very slow, and it needs to handle every special case for different k and number of features. For big datasets with lots of features kNN is suboptimal.

   Single layer: Harder to implement than kNN but very good for linearly separable data and bigger datasets in terms of speed and accuracy. The amount of training data to create a linear classifier can be low (compared to the multi-layer).

   Multi-layer: Very hard to implement compared to the above but also very good at classifying nonlinearly separable data. Need to have a good amount of datapoints to use as training data or else overfitting will become a problem.

   The classifier that is most suited for the task is dependent on how the data looks. For much data and non-linearly separable features the multi-layer network is a good choice. If the amount of data for the same features are sparse a kNN-classifier would suffice. For linearly separable data the single-layer network should be used.

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

   There are many datapoints that are easily classified because they have one very distinct feature. These points could be pre-processed to speed up the classifiers which only need to consider the points that are hard to classify.
   Points that are outliers could also be removed during a preprocessing step to get a more generalized solution since the network will not try to fit to the outliers.