# 1. Introduction

**Student Number: 219060**

This report is about a binary classification task with missing data. In it is an explanation of the method used, and the results gathered, as well as an analysis of these results.

## 2. Method

## 2.1. Handling the data

The data from the two datasets provided was first separated into three datasets. Each of the provided datasets were shuffled and split into three equal sections. This was done evenly with an aim to avoid bias, with a randomly selected third of dataset 1 being paired with a randomly selected third of dataset 2. This way, a combination of two of the datasets can be used to train the model and the third can be used to test it. Because of this, the model was able to be trained and tested reliably and was able to avoid being overfitted to the training data. The data and its labels and confidences were split into separate sets.

## 2.2. NaN Value Handling

The original aim was to create a mask of every position of NaN values in the data. The NaN values would then be replaced with a randomised float (0-1) in case the mask failed to detect one of the NaN values or was ignored.

The mask would then be applied to the data by creating another dimension to the data array and setting it to be the mask. This would work fairly well as the network could then learn to only take into account the values that the mask says to while ignoring the random values generated.

The RAM limitations of Google Colab have meant that this was not possible. Instead, the random numbers (0-1) were used without a mask. The data was normalised after the random numbers had been added. This was to ensure that all the data is between the values 0 and 1; due to the normalization, any larger random values would be reduced meaning that the randomly generated values are likely to be closer to the values provided in the original datasets.

## 2.3. Neural Network

The neural network was originally created with 3 layers of 128 nodes each. After some testing, it was found that 4 layers resulted in reduced the chance of overfitting while still having a high accuracy (above 85%). This too was the reason 128 nodes were used. Having 128 nodes for 4000+ features meant that the chance of overfitting was low.

A variant was also used with 8 layers of 1080 nodes each. This was done with an aim of reducing the level of fluctuation between training iterations (figures 1 & 4).

# 3. Results

As shown (figures 1 & 4) there are heavy fluctuations between the accuracies for both the same set of training and test data and between different sets of training and test data. The distribution of the accuracies for all three sets of training and testing data are shown in figures 2 & 5. The median accuracy (figure 2) is approx. 95% for all three sets of data. Furthermore, the upper and lower quartiles have accuracies ranging from 92.5% (3.s.f) to 98.0% (3.s.f) with interquartile ranges between 2.0% (2.s.f) and 5.5% (2.s.f). The mean average ranges approx. 1.5%, from 93.5% (3.s.f) to 95.0% (3.s.f). Training sets 1 and 2 have outliers at approx. 90% accuracy while the lower whisker for dataset 3 extends to approx. 88% accuracy.

Figures 4, 5 & 6 were gathered using a neural network with 8 layers of 1080 nodes each. They were also trained for 10 epochs.

Figures 5 & 6 show the distribution of accuracy and loss for the three datasets. The median accuracy (figure 5) for dataset 1 is consistent with those in figure 2. However, the median accuracy for the datasets 2 and 3 are much lower. This is likely a sign of overfitting as the lower quartile for dataset 3 is approx. 86.0% (3.s.f), over 6.5% lower than the lowest lower quartile in figure 2.

The locations of the upper and lower quartiles and whiskers for datasets 1 & 2 compared to those for dataset 3 are in a similar relationship to their counterparts in figure 2.

# 4. Discussion

The model used can never reliably provide perfect results. The data generated in place of the NaN values is the main cause of this. If a NaN mask was able to be used, the accuracy percentage would likely be higher.

For all iterations of the neural network, the loss was consistently at 0.673 (3.s.f). It is unknown why this value is both that high and so consistent. This may be due to the loss function used, binary cross-entropy.

The initial thought on the level of fluctuation (figures 1 & 4) was that it was likely due to underfitting. However, there is a greater level of fluctuation in the larger network (figure 4). It is believed that the drop in minimum accuracy is due to the model overfitting to the datasets provided. Because of this, the original model was used to make the final predictions.

# 5. Graphs and Illustrastions

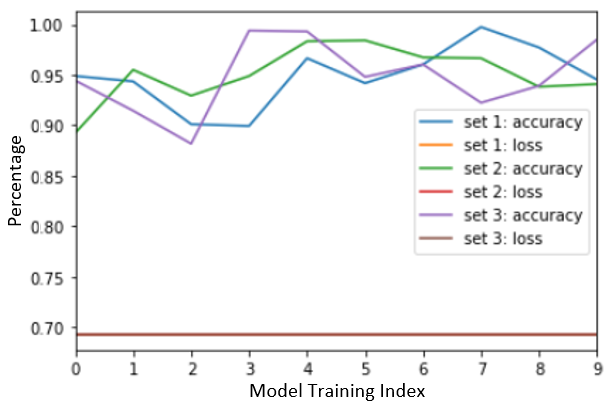


Figure 1: A plot of the loss and accuracy for 10 instances of the model training for each of the training sets of data with a model of 4 layers of 128 nodes

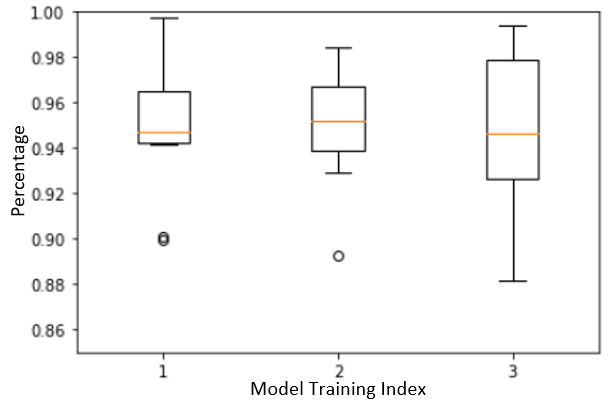


Figure 2: A boxplot of the accuracy for 10 instances of the model training for each of the training sets of data with a model of 4 layers of 128 nodes

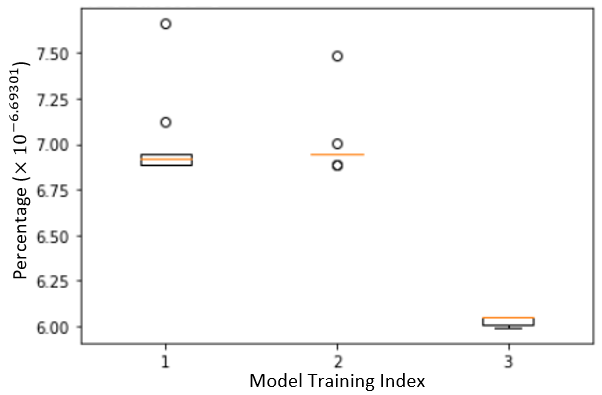


Figure 3: A boxplot of the loss for 10 instances of the model training for each of the training sets of data with a model of 4 layers of 128 nodes

Chart, line chart

Description automatically generated

Figure 4: A plot of the loss and accuracy for 10 instances of the model training for each of the training sets of data with a model of 8 layers of 1080 nodes

Chart, box and whisker chart

Description automatically generated

Figure 5: A plot of the loss and accuracy for 10 instances of the model training for each of the training sets of data with a model of 8 layers of 1080 nodes

Chart, box and whisker chart

Description automatically generated

Figure 6: A plot of the loss and accuracy for 10 instances of the model training for each of the training sets of data with a model of 8 layers of 1080 nodes