2IIG0 - Homework 2 - Question 6

M.F.J. Moonen (1234115), Jin Ouyang (1608541), Bas Witters (1625187)

1 Subquestion A

We explored the data, and found that the values of both independent variables range from around 0 to almost 10,000. While the mean and standard deviation for both variables are similar at around 5000 and 3000 respectively, their distributions are not exactly the same as can be seen in figures 1 and 2. A scatter plot of the different points as in figure 3 reveals the clear pattern in the data: the plot can be split in quarters, where each one is dominated by one of the two outcomes.

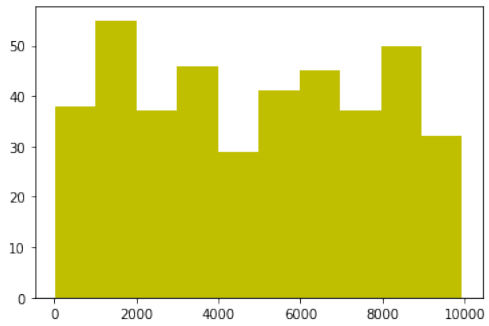

Fig. 1. Histogram visualising the value of the X_1 variable in the train set against the number of occurrences.

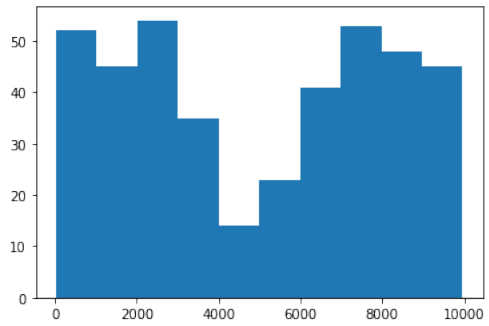

Fig. 2. Histogram visualising the value of the X_2 variable in the train set against the number of occurrences.

2 Subquestion B

We choose to use **min-max normalization** since the data distribution is not close to the normal distribution, which means it is not suitable for the Zero-mean-unit-variance normalization. Besides, the range of feature X1 and X2 of training data is huge; thus, we would like to scale them to the range of $[0, 1]$. The formula we used for updating each column j for every data point x_i is as follows: $x_{i,j} = \frac{x_{i,j} - min(x_j)}{max(x_j) - min(x_j)}$. After normalization, our first four samples look as shown in table 2.

3 Subquestion C

The size of the input layer is 1×2, and the size output layer of the MLP is 1×2 as well since this is a binary classification problem.

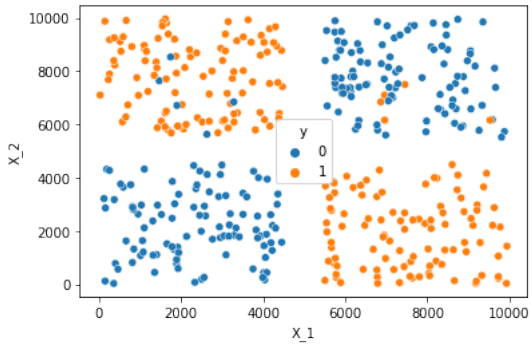

Fig. 3. Scatterplot visualising all values in the train set.

	X_1	X_2	y
0	0.291484	0.328300	0
1	0.187760	0.075766	0
2	0.419723	0.153595	0
3	0.818876	0.984258	0

Table 1. Table showing the first four samples of the dataset after normalization.

4 Subquestion D

We choose **a simple loss function** and **ReLU** as our loss function and activation function, respectively. We only use a simple loss function, which is calculated by the subtraction of the predicted labels and the true labels. The **ReLU** activation function is chosen as it can also avoid the gradient vanishing.

In order to get output from the MLP that actually works for this and is in the desired form, we use a **Softmax** function as activation between the final MLP layer and the output.

5 Subquestion E

The weights for all elements in our MLP are initially generated using random samples from the standard normal distribution. This was done so that we have an amount of variance in our initial weights that can be used to already from that start have different weights pulling subsequent values in different directions, so that if different relations of this kind turn out to be necessary while learning the network the basis is already present (to an extent).

This was chosen instead of for example all ones as initial state, because starting out with all weights as equally impactful on their connected nodes made less intuitive sense to us when we know that this is almost never the case in a trained network.

6 Subquestion F

As this is an implementation question, we refer to our notebook with the actual Python code. Early stopping was implemented in a way so that training is stopped before the configured maximum number of epochs is reached if no improvement is seen on the validation accuracy for several epochs in a row.

7 Subquestion G

After testing a lot of different values, we came to the conclusion that the following set of hyperparameters gives us the optimal validation accuracy:

- Learning rate: 0.01
- Batch size: 25
- Max number of epochs: 200
- Early stopping after: 8 epochs of no improvement on validation accuracy

8 Subquestion H

The development of training and test loss and validation can be seen together in Figure 4. The accuracy on both datasets over time can be seen in Figure 5. Based on these graphs we chose to trigger early stopping after 8 epochs of no improvement, so that the model does not start overfitting to the training data. It hits a maximum very quickly and stays there but can sometimes still improve slightly, so with 8 epochs we give it some time for that to occur (if it does) before we stop it so that it does not start overfitting.

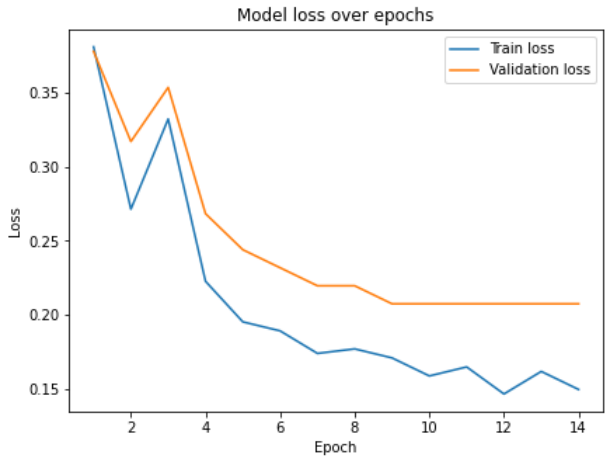

Fig. 4. Development of model loss over epochs of training. Training was cut off by early stopping after epoch 14.

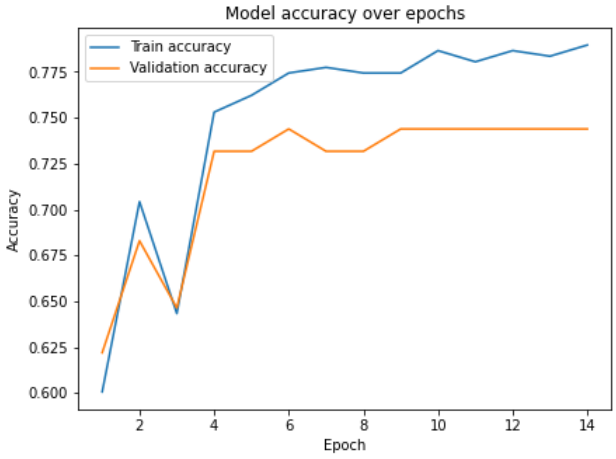

Fig. 5. Development of model accuracy over epochs of training. Training was cut off by early stopping after epoch 14.

9 Subquestion I

The final accuracy of our trained model on the train data is 0.79, and on the validation data is 0.74. The confusion matrices on both of these sets can be seen in figures 6 and 7.

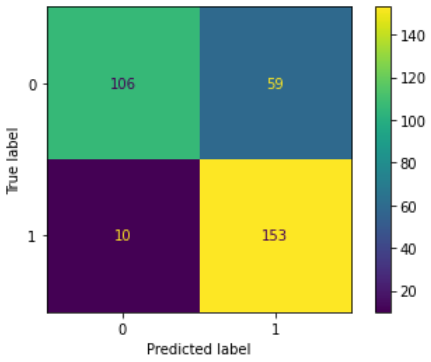

Fig. 6. Confusion matrix for trained model on full training dataset.

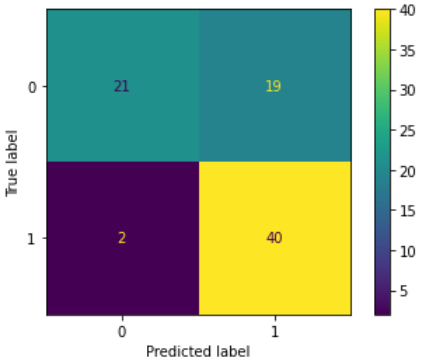

Fig. 7. Confusion matrix for trained model on validation dataset.