**REPORT**

**Question 1**

We split the dataset into 80% for training and 20% for testing, the main purpose behind training the model with 80% is that we tried to avoid over fitting the model with much data and training with more data can help algorithms detect the signal better and then we assigned each feature with a random weight that will be updated as the model run 1000 times. This means that the random weights will be updated 1000 times together with the bias. The initial value of bias is 0.03.

The model is executed 1000 because if we only run it few times, there is a probability that we will end up having higher rate of errors, therefore we updated the weight values 1000 times to reach optimal value. And then we used sigmoid function because we wanted the results to be values between 0 and 1.

The ipywidgets are used to enable or create input boxes where end user will input the number (the first n records) of values he or she wishes to view from the dataset, and also the ipywidgets are used to create the buttons that will display requested output or perform the predictions.

**Question 2**

To complete this task a model was created using sequential algorithm.

To train the model we used, sequential algorithm which is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. The reason behind using this model is that, it does not require layer sharing, again it is appropriate when classifying elements. To create this model, we used the add( ) method.The model consists of 4 layers (one input layer, two hidden layers and one output layer).

Within the hidden layers ReLU activation function was used as it does not activate all neurons simultaneously. Models that use ReLU is easier to train and often achieves better performance in relation to those using other activation functions.

Softmax activation layer was used as the activation for the last layer (output layer) of classification network because the results could be interpreted as a probability distribution.

To compile the model we used optimizer too reduce model loss and increase model accuracy. To determine probabilistic model loss, we used sparseCategoricalCrossentropy which computes the Crossentropy loss between the labels and predictions. We used Crossentropy loss function because there are more than two label classes in the provided dataset, and also because the labels for the provided dataset are integers.

To speed up the learning process of a model we normalized images and reduced their image range.

To test our model’s model performance we used accuracy matrix and loss matrix.