**Breast Cancer Classification Dataset**

The topology selected was a 3-layer network, because it was not as simple as a shallow network but also not a complex as a very deep network. After trial and error, more hidden layers did not make any significant difference, only increasing computation time and making no more than ±1% difference. One more hidden layer increased accuracy by 0.5%, 5 hidden layers reduced it by 0.7%; more than 5 hidden layers, only led to a slow and progressive reduction of accuracy. The learning rule, given as 'optimizer=adam', is a stochastic gradient descent algorithm: this worked well. Initial results were excellent, changes to the learning rule did not yield better results. The learning rate we found to be best was the default for the 'adam' optimizer: which was 0.001. Higher rates either had no difference or, it was too high and significantly lowered accuracy. Using a value of 150 upwards for the epoch tended to result in negligible results for accuracy other than increasing the learning time drastically. Using lower values such as 50 lowered the accuracy of the neural network.

An 60/20/20 train/validation/test split was used. The size of the dataset resulted in higher train and lower validation/test splits resulting in overfitting. Reduced training size led to lower accuracy. Thus, this was the best compromise.

The activation function for the hidden layers was the RELU function because we had only positive values and it gave better accuracy, ~ 1%, whilst also being slightly faster than the sigmoid function. As this was a binary classification problem, the sigmoid function is the default decision. The number of hidden layer neurons was 16 and the input layer neuron count was 30. This dimension reduction led to better results, any less, down to 10, or anymore, up to 50, led to worse results.

The classification results per fold were as follows:

Accuracy for fold 0 is 96.49122807017544%

Accuracy for fold 1 is 96.49122807017544%

Accuracy for fold 2 is 94.73684210526315%

Accuracy for fold 3 is 94.73684210526315%

Accuracy for fold 4 is 98.24561403508771%

Accuracy for fold 5 is 98.24561403508771%

Accuracy for fold 6 is 98.24561403508771%

Accuracy for fold 7 is 98.24561403508771%

Accuracy for fold 8 is 100.0%

Accuracy for fold 9 is 96.42857142857143%

with average accuracy for 10 folds being 97.18671679197995%.

The task of using 10-fold cross-validation aids generalisation of the network but another feature implemented was the use of the 'Dropout' method available in Keras. Set to 0.1, so for each training set, 10% of values would be reduced to zero whilst scaling all other values by 1/(1-rate). The main advantage of this method is that it prevents all neurons in a layer from synchronously optimizing their weights. This adaptation, combined with cross-validation, prevents the neurons from converging to the same goal, thus decorrelating the weights.

**Student Performance Maths Regression Dataset**

The dataset has 32 features, 1 label(G3), and 395 data points. 4 of the categorical features in the dataset are multi class, so a OneHotEncoder had to be used on these 4 features, this resulted in the number of features increasing to 41. Therefore, there are slightly less data points than 10 times the number of attributes, so learning may have been affected by the Curse of Dimensionality. We did not remove any features from the dataset as we thought that they all had a high correlation with the label.

A 60/20/20 train/validation/test split was used to train the model to minimize the Root Mean Square Error (RMSE). We found that the optimal topology was a 4-layer network, with 3 hidden layers. After trial and error, we found that less hidden layers increased the RMSE and more than 3 hidden layers did not significantly reduce the RMSE, but increased computation time. The learning rule that was used was 'optimizer=sgd', as it produced a lower RMSE than 'optimizer=adam'. The learning rate that we used was 0.001. Smaller learning rates than 0.001 did not make a significant difference to the RMSE. We found that a value of 100 was the optimal number of epochs, as using a lower number increased the RMSE, whereas a higher number than 100 had little impact on the RMSE other than increasing the learning time drastically.

The activation function for the hidden layers that we used was the RELU function because we only had positive values. The RELU function also gave a slightly smaller RMSE than the sigmoid function and a significantly smaller RMSE then the softplus function. As this is a regression problem, no activation function is used in the output layer. We had 41 features and 1 label, so the number of inputs was 41 and the number of outputs was 1. The three hidden layers all had 41 neurons, as more and less neurons in the hidden layers (such as 22 and 50 neurons) led to worse results. The optimal model that we found produced a Root Mean Square Error of 1.7685020017857032.

The regression results per fold were as follows:

Root Mean Square Error for fold 1 is 1.3090511180570041

Root Mean Square Error for fold 2 is 2.729231257189508

Root Mean Square Error for fold 3 is 1.5986677980378814

Root Mean Square Error for fold 4 is 3.0376100132687

Root Mean Square Error for fold 5 is 1.7460510836223513

Root Mean Square Error for fold 6 is 1.9775940664456242

Root Mean Square Error for fold 7 is 2.6170762764841484

Root Mean Square Error for fold 8 is 1.9077135044502156

Root Mean Square Error for fold 9 is 2.6631769380638675

Root Mean Square Error for fold 10 is 1.8292895547304342

With the average Root Mean Square Error being 2.2095539635642205

As with the classification dataset, 10-fold cross-validation aids with generalization and the 'Dropout' method (also set to 0.1) was used to overcome the problem of overfitting.