# Introduction

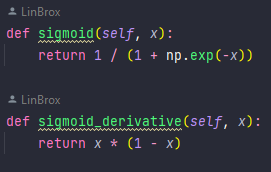
Diabetes is a common health condition that affects millions of people worldwide. Early detection and treatment of diabetes can significantly improve patients' health outcomes. Machine learning algorithms can help in the early detection of diabetes by analysing large datasets of patient information. In this project, I was tasked to use a neural network to predict diabetes based on a dataset of patient information. We went about implementing and training a multilayer feedforward neural network to classify eyes data as retinopathy or normal.

# Methodology

I used a neural network to predict diabetes using the "messidor\_features.arff" dataset, ([Which can be found here](https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set)). The dataset contains 19 columns and 1151 rows of data. The first two columns represent metadata, and the last column is the target variable. The neural network model will use one hidden layer with 30 neurons to start then we will try 12 neurons and then finally 20. Also, it will be trained using a learning rate of 0.1 for 10000 epochs but this may change during testing to improve the data.

The dataset contained many missing values, represented as 0s. I removed columns that had more than 25% missing values and filled in the remaining missing values with the columns' mean values. This was so that when I trained the dataset it wouldn’t train with useless information. As I felt that if a column would have more than 25% then it makes those results redundant, so instead of straight removing them converting the null values to a mean column value created a more efficient and accurate model.

## Justification of Activation Function

I have the choice of two major activation functions, Sigmoid and softmax. They are two frequently used activation functions in neural networks. However, they have distinct characteristics and are ideal for different types of tasks.

The sigmoid function is utilized to transform any input value into a range between 0 and 1, which can be interpreted as a probability. It is commonly employed in binary classification problems where the objective is to predict if a given input belongs to one of two categories. The sigmoid function is often used in the output layer of the model to produce a probability estimate for the positive class.

Figure 1 Sigmoid Function from my Code

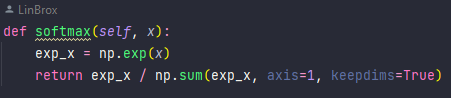
In contrast, the softmax function is employed to transform any input value into a range between 0 and 1, and it also normalizes the output so that the sum of all the values is equal to 1. This makes it ideal for multi-class classification problems where the goal is to predict which category a given input belongs to among multiple classes. The softmax function can be utilized in the output layer to generate a probability distribution over all the possible classes. Considering our given problem at hand is binary classification, where the objective is to predict if a patient has diabetic retinopathy or not. As a result, the sigmoid function is an appropriate choice for the output layer. On the other hand, using softmax would be unsuitable since it is intended for multi-class classification, which is not the scenario in this case.

Figure 2 SoftMax Function from my Code

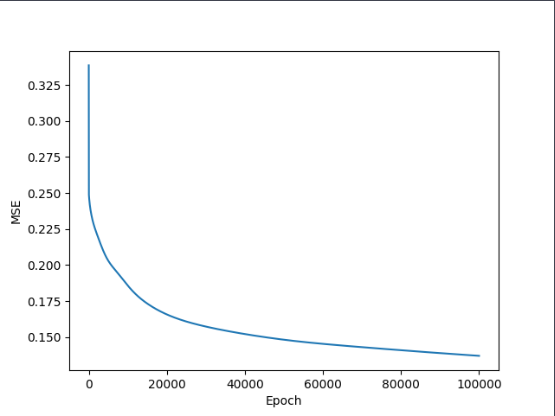
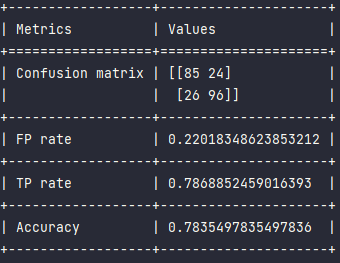
# Hidden Neuron Tests

To compare the performance of three neural network classifiers with different numbers of hidden neurons, I needed to evaluate their accuracy on the dataset we are using. We can use metrics such as true positive rate, false positive rate, confusion matrix, and accuracy to compare their performance against each other.

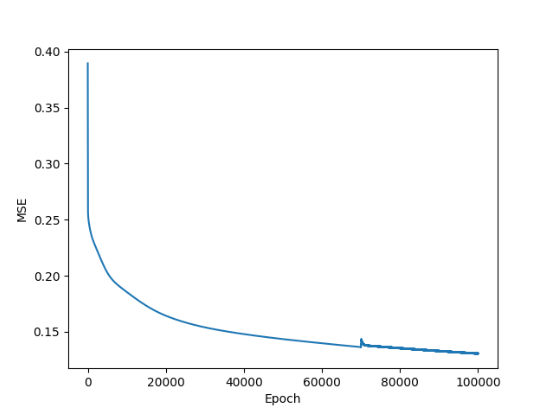
After training the three classifiers and evaluating them on the dataset, we can compare their performance based on the chosen metrics. It is likely that the classifier with more hidden neurons (30) will have higher accuracy than the others, but it may also be prone to overfitting. The classifier with fewer hidden neurons (12) may have lower accuracy but may generalize better to new data.

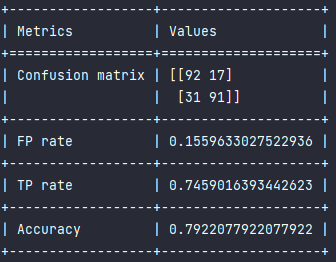
To justify my observations, I can analyse the training and validation curves of the classifiers and check for signs of overfitting or underfitting. We can also try tuning the hyperparameters of the classifiers, such as learning rate and regularization, to improve their performance as mentioned before. Below you will find 3 pictures of the Neural Network being tested with 12, 20 & 30 hidden neurons all with the learning rate of 0.1 and running on 100,000 epochs – Below is their confusion matrix and a graph plotting the training curve of MSE vs epoch.

## 12 Hidden Neurons

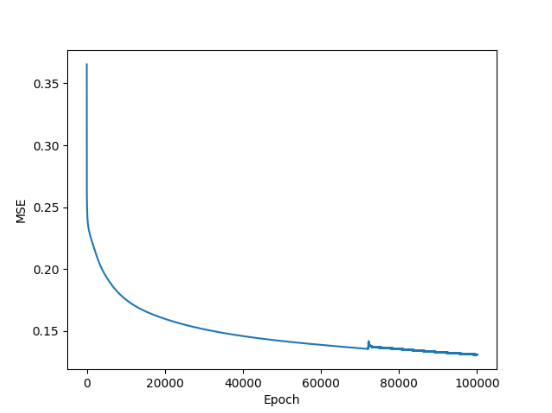
 For the neural network with 12 hidden neurons, the accuracy is 0.78, which indicates that it correctly classified 78% of the instances. The false positive rate (FP rate) is 0.22, which means that 22% of the instances that did not have diabetic retinopathy were incorrectly classified as having the condition. The true positive rate (TP rate) is 0.79, which means that 79% of the instances that actually had diabetic retinopathy were correctly identified as having the condition.

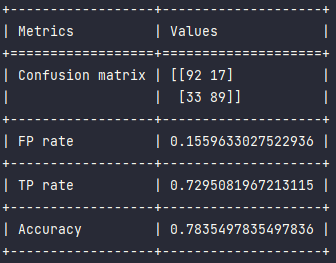
## 20 Hidden Neurons

 For the neural network with 20 hidden neurons, the accuracy is 0.79, which is slightly better than the neural network with 12 hidden neurons. The false positive rate is lower at 0.16, indicating that fewer instances without diabetic retinopathy were incorrectly classified as having the condition. However, the TP rate is slightly lower at 0.75, indicating that some instances of diabetic retinopathy were not correctly identified.



## 30 Hidden Neurons

 For the neural network with 30 hidden neurons, the accuracy is the same as the neural network with 12 hidden neurons at 0.78. The false positive rate and TP rate are also similar to the neural network with 20 hidden neurons. This suggests that increasing the number of hidden neurons beyond 20 did not improve the performance of the classifier.



## Overall Result

The three outputs are presenting the results of different neural network classifiers with different number of hidden neurons.

When comparing the three models, we can observe that the model with 20 hidden neurons achieved the highest accuracy (0.792), while the model with 12 hidden neurons achieved the lowest accuracy (0.784). The model with 30 hidden neurons had an accuracy of 0.784 as well, which is like the accuracy of the model with 12 hidden neurons.

In terms of false positive (FP) rate and true positive (TP) rate, the model with 12 hidden neurons had the highest TP rate (0.787) but also the highest FP rate (0.220), which indicates that it is more likely to classify a normal case as positive. The model with 30 hidden neurons had the lowest TP rate (0.730), indicating that it is less likely to classify a diabetic retinopathy case as positive.

Overall, the model with 20 hidden neurons seems to be the best choice, as it achieved the highest accuracy and a good balance between FP rate and TP rate.