Neural Networks Hyper Parameter Experiment

Sailesh Dwivedy

METHODS:

In this deep learning experiment I used the heart attack analysis and prediction data set from Kaggle[1], which is a binary classification problem. Here we predict the target column where 0 (138 records) represents less chance of heart attack and 1 (165 records) represents more chance of heart attack. There are 303 records in total with 14 columns in including the output/target column. There are is no missing values in any of the columns.

To train our model we created a 75/25 train test split of the dataset. Train comprises of 227 records and test comprises of 76 records. The input features were not on the same scale so we standardised the train(X_train) and test data (X_test) using a standard scaler because standardization helps to bring input features on same scale helping models learn better. I have used to random state as 42 for the experiment to ensure reproducible results.

I used MLPclassifier[2] as the model for the experiments. In this experiment I observed the effects of Hidden Layer sizes, Neurons per hidden layer and activation functions on the model performance keeping the number of iterations for training and the batch size constant.

Each of the models is built with either 1 or 2 hidden layers, 10 or 20 neurons per hidden layer and logistic or tanh as activation function. The number of iteration is 100 and batch size is 20 which has been kept constant across all models. The total model that I got out of this setting is 8.

After training, each model was evaluated on the test set using confusion matrix, precision, recall and accuracy.

RESULTS:

Here, I have taken class 0 as negative which is less chance of having heart attack and class 1 as positive which is more chance of having heart attack. Below I have recorded the performance metrics on the test set for different combinations. For every model shown below the number of iterations is 100 and and batch size is 20.

Table 1: Hidden Layers=1, Neurons Per Layer=10, Activation=logistic

	Precision	Recall	F1-score	Support
Negative (0)	0.85	0.83	0.84	35
Positive (1)	0.86	0.88	0.87	41

Accuracy = 0.8552

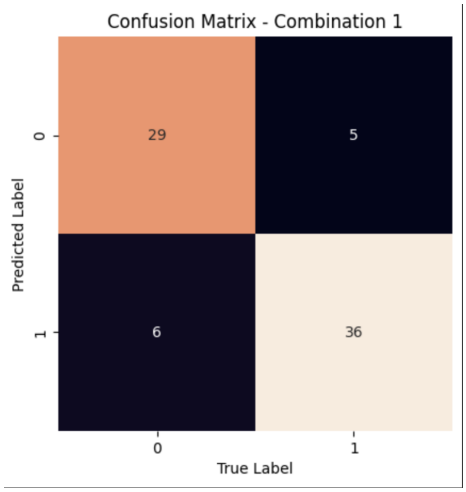


Table 2: Hidden Layers=1, Neurons Per Layer=10, Activation=tanh

	Precision	Recall	F1-score	Support
Negative (0)	0.85	0.80	0.82	35
Positive (1)	0.84	0.88	0.86	41

Accuracy = 0.8421

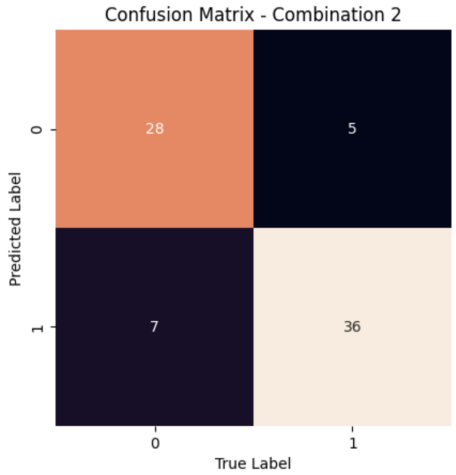


Table 3: Hidden Layers=1, Neurons Per Layer=20, Activation=logistic

	Precision	Recall	F1-score	Support
Negative (0)	0.85	0.83	0.84	35
Positive (1)	0.86	0.88	0.87	41

Accuracy = 0.8552

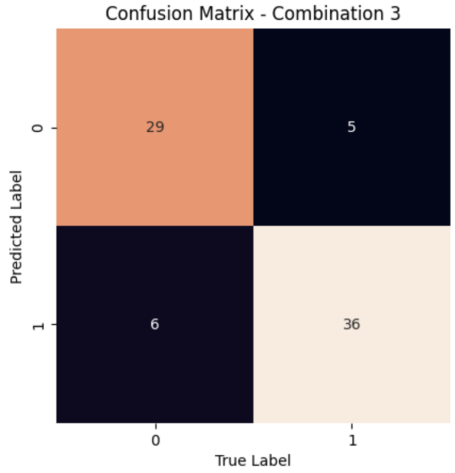


Table 4: Hidden Layers=1, Neurons Per Layer=20, Activation=tanh

	Precision	Recall	F1-score	Support
Negative (0)	0.88	0.86	0.87	35
Positive (1)	0.88	0.90	0.89	41

Accuracy = 0.8815

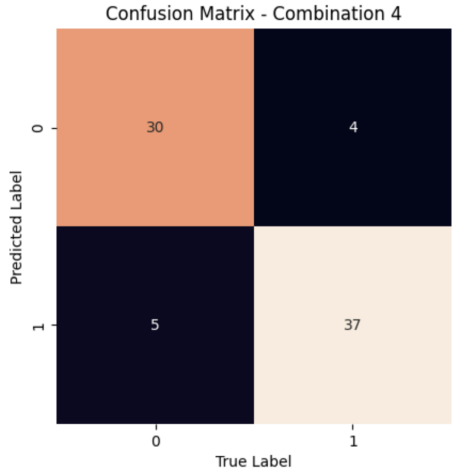


Table 5: Hidden Layers=2, Neurons Per Layer=10, Activation=logistic

	Precision	Recall	F1-score	Support
Negative (0)	0.85	0.83	0.84	35
Positive (1)	0.86	0.88	0.87	41

Accuracy = 0.8552

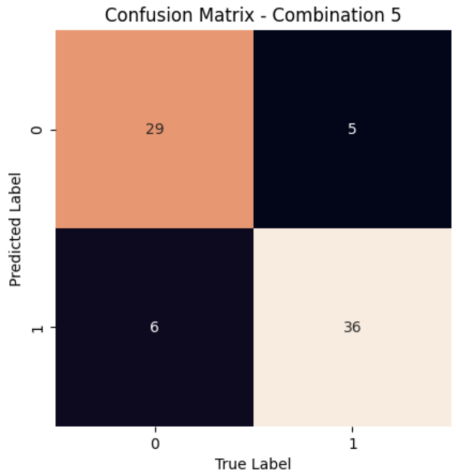


Table 6: Hidden Layers=2, Neurons Per Layer=10, Activation=tanh

	Precision	Recall	F1-score	Support
Negative (0)	0.83	0.83	0.83	35
Positive (1)	0.85	0.85	0.85	41

Accuracy = 0.8421

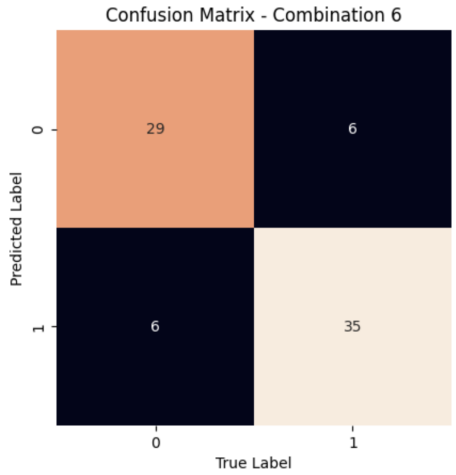


Table 7: Hidden Layers=2, Neurons Per Layer=20, Activation=logistic

	Precision	Recall	F1-score	Support
Negative (0)	0.85	0.80	0.82	35
Positive (1)	0.84	0.88	0.86	41

Accuracy = 0.8421

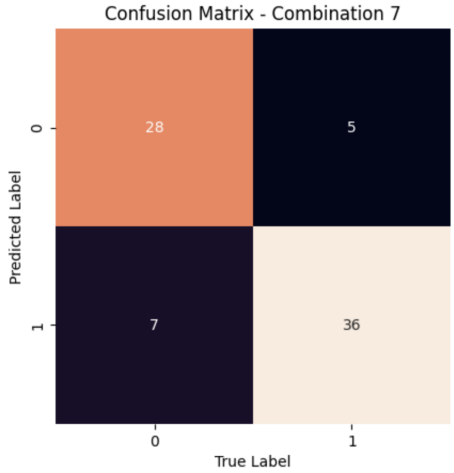
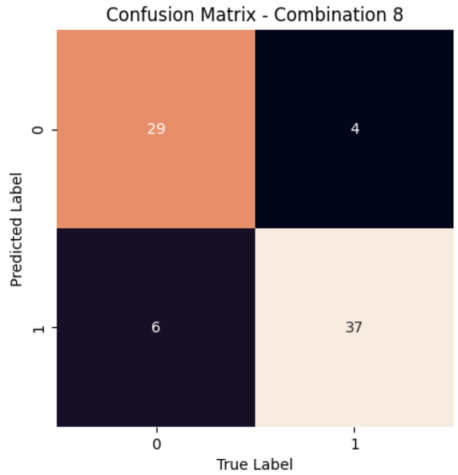


Table 8: Hidden Layers=2, Neurons Per Layer=20, Activation=tanh

	Precision	Recall	F1-score	Support
Negative (0)	0.88	0.83	0.85	35
Positive (1)	0.86	0.90	0.88	41

Accuracy = 0.8684



ANALYSIS:

Based on the results seen above for 8 combination of neural network architecture two general trends emerge:

Trend 1 - Activation Function relative to Neuron Count

The best performance is given by use of Tanh activation function when the neuron per layer is increased to 20 irrespective of the network having 1 or 2 hidden layers.

When models are less complex that is having 10 neurons per layer rather than 20, logistic/sigmoid activatio gives comparable or slightly better performance.

So, tanh is more benefited by increasing the model complexity than logistic/sigmoid.

Reason:

The best performance is given by tanh with a complex model, which might be because of the fact that the output for tanh is wider in range of values (-1 to 1) compared to that of sigmoid (0 to 1) because of which the gradients will be better during back propagation leading to more effective learning in large networks.

The reason for good performance of sigmoid for simpler architecture could be that more simpler models its property of giving values between 0 to 1 is sifficient but as model becomes more complex tanh with a larger range of output value performs better.

Trend 2 - Impact of Network size on Performance

The best performance was got by increasing the number of neurons irrespective of number of layers with the tanh activation function.

Reason:

A larger model capacity that is more neurons allows the model to capture better patterns irrespective of the network having 1 or 2 hidden layers, showing that model's capacity increase through more neurons can be more beneficial than only adding more layers.

Insights from Evaluation Metrics:

Precision measures out of all predicted positive instances how many are correctly predicted as positive and recall measures, out of all actual positive instances how many were correctly predicted as positive. Based on the above results we can see that some models are better at reducing false negatives(high recall) and some better at reducing false positives(high precision).

The tanh activation function with 20 neurons per layer and 1 hidden layer (Combination 4) does a better job at balancing both precision and recall. This striking of balance is very necessary for real world applications