Heart Attack Prediction using Artificial Neural Networks

Jobis James
Student Id: 10902871
jamesj29@uni.coventry.ac.uk

***Abstract-* Heart disease cases are growing at an unprecedented pace, and it's important and concerning to be able to predict those diseases in advance. This is a difficult diagnosis to make, so it must be done correctly and quickly. The key emphasis of the research paper is on which patients are more likely to have heart disease based on different medical features. We developed a heart disease prediction system that uses the patient's medical history to determine whether or not the patient will be diagnosed with heart disease. To predict and identify patients with heart disease, we used various artificial neural networks (ANN) such as Single Layer Perceptron, Multilayer Perceptron, Radial Basis Function and Long Short-Term Memory (Recurrent neural network). The proposed model's strength was very satisfying, as it was able to predict evidence of developing heart disease in a given person using RBF and MLP, which demonstrated a high degree of precision as opposed to other used methods such as LSTM. The given heart disease prediction system improves medical care and reduces the cost.**

I. INTRODUCTION

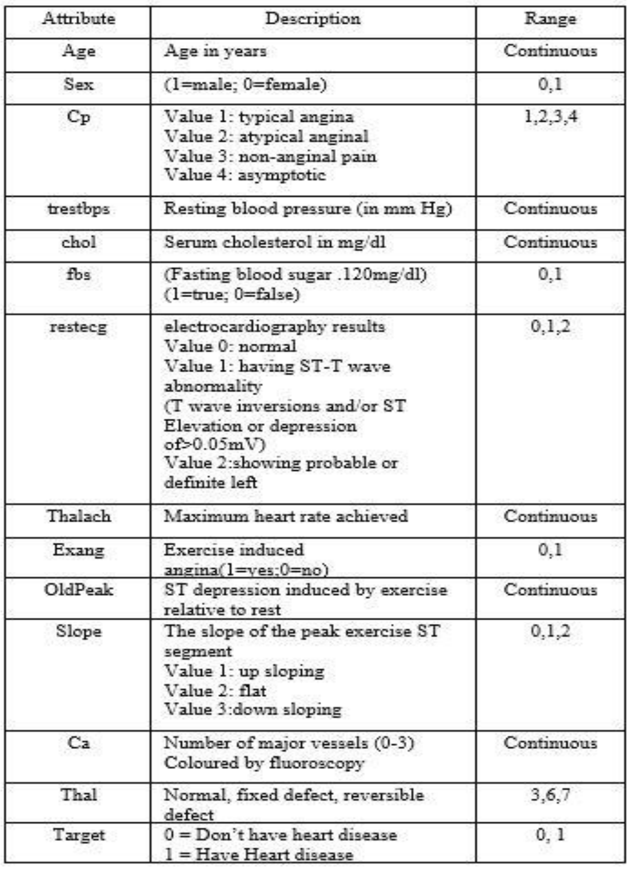
Cardiovascular diseases are a broad term that refers to a variety of conditions that can damage the heart. According to the World Health Organisation, 17.9 million people die each year from CVDs (cardiovascular diseases) [1]. It is the leading cause of death for adults. With the benefit of their medical records, our project demonstrates how Artificial Neural Networks (ANN) used to predict a patient's heart disease status. It identifies who has certain heart disease signs, such as chest pain or elevated blood pressure, which may aid in diagnosing disease with less diagnostic examinations and more successful therapies, enabling them to be handled properly.

The aim of this project is to check whether the person is likely to be diagnosed with any heart diseases based on their medical record such as gender, age, fasting sugar level, chest pain etc. A dataset with the patient's medical records and characteristics is taken from the Kaggle. Using this dataset, we will estimate the patient may have heart disease or not. To do so, we assess a patient based on 13 medical characteristics to see if he is at risk for heart failure. Four artificial neural network algorithms are used to train these medical attributes: Single Layer Perceptron (SLP), Radial Basis Function (RBF) , Multilayer Perceptron (MLP)) and Long Short Term Memory (LSTM). MLP is the best method for this problem, with an accuracy of 100 percent.

II. DATASET

The dataset used in this course work have taken from Kaggle [2], which is a simplified kind of the dataset found in UCI [3]. The dataset contains clinical records of 1025 patients from Long Beach, Switzerland, Cleveland, and Hungary from 1988 onwards. There are 14 columns in the dataset with various attributes such as sex, age, cholesterol level, and so on, as well as a target column that tells us whether that person has heart disease or not. Except for the target column, which will be our dependent variable, we'll consider all the columns as independent variables. Table 1 lists the properties of the dataset and their meanings. The target attribute indicates whether or not the patient has heart disease. It is integer-valued, with 0 indicating no heart disease and 1 indicating heart disease.

Table 1 Dataset Attributes



Attribute	Description	Range
Age	Age in years	Continuous
Sex	(1=male; 0=female)	0,1
Cp	Value 1: typical angina Value 2: atypical anginal Value 3: non-anginal pain Value 4: asymptotic	1,2,3,4
trestbps	Resting blood pressure (in mm Hg)	Continuous
chol	Serum cholesterol in mg/dl	Continuous
fbs	(Fasting blood sugar .120mg/dl) (1=true; 0=false)	0,1
restecg	electrocardiography results Value 0: normal Value 1: having ST-T wave abnormality (T wave inversions and/or ST Elevation or depression of>0.05mV) Value 2:showing probable or definite left	0,1,2
Thalach	Maximum heart rate achieved	Continuous
Exang	Exercise induced angina(1=yes;0=no)	0,1
OldPeak	ST depression induced by exercise relative to rest	Continuous
Slope	The slope of the peak exercise ST segment Value 1: up sloping Value 2: flat Value 3:down sloping	0,1,2
Ca	Number of major vessels (0-3) Coloured by fluoroscopy	Continuous
Thal	Normal, fixed defect, reversible defect	3,6,7
Target	0 = Don't have heart disease 1 = Have Heart disease	0, 1

Once we've properly pre-processed the results, we can break it into training (80 percentage of total data) and testing (20 percentage of total data) datasets.

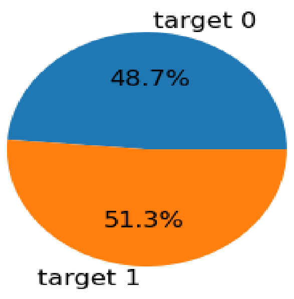


Fig. 1 Target Class Pie Chart

Fig 1 illustrates, about half of the people in the dataset has been diagnosed with heart disease.

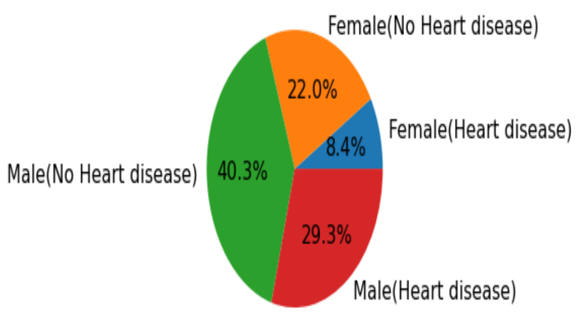


Fig. 2 Target class distribution with respect to Gender

From Fig.2 , we can understand that males have more chance of heart disease when compared to females.

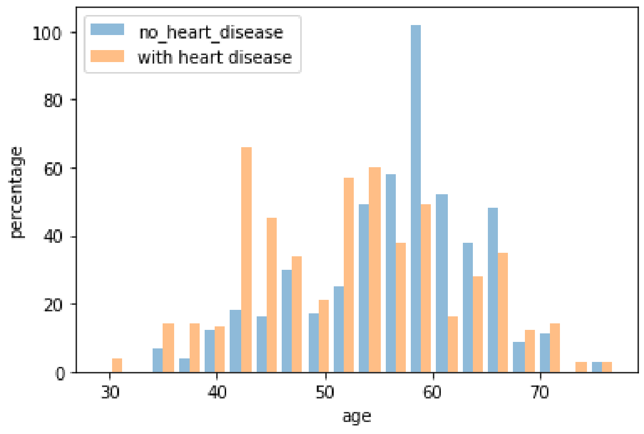


Fig. 3 Distribution of Target variable w.r.t. Age

In Fig 3., around the age of forty, the ratio of people having heart disease increases. That is, people above the age of forty are at a greater risk of heart disease.

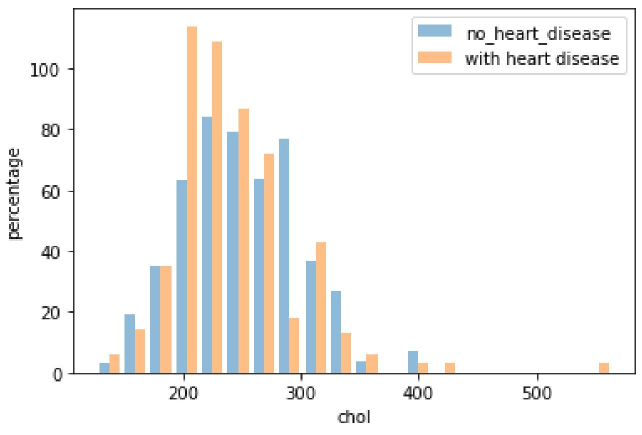


Fig. 4 Distribution of Target variable w.r.t. cholestrol

There is a high chance of heart disease for people have cholesterol of more than 200mg/dl. The standard cholesterol level should be less than 200 mg/dl according to the study.

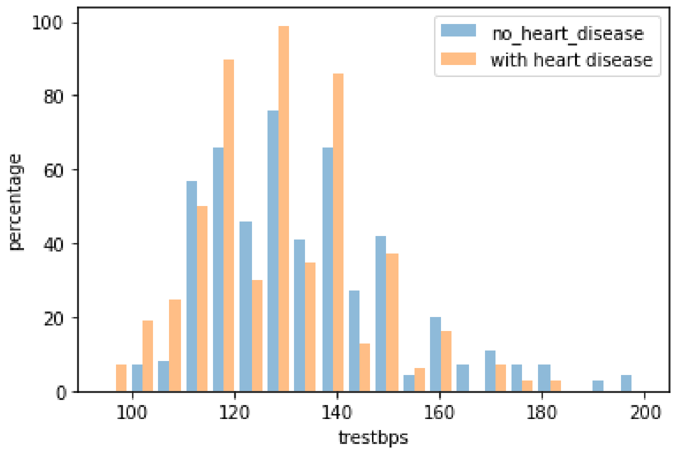


Fig. 5 Distribution of Target variable w.r.t. trestbps

Blood pressure should be less than 120 mmHg in the optimum situation. Whether or not the patients have heart disease, more than half of them have high blood pressure.

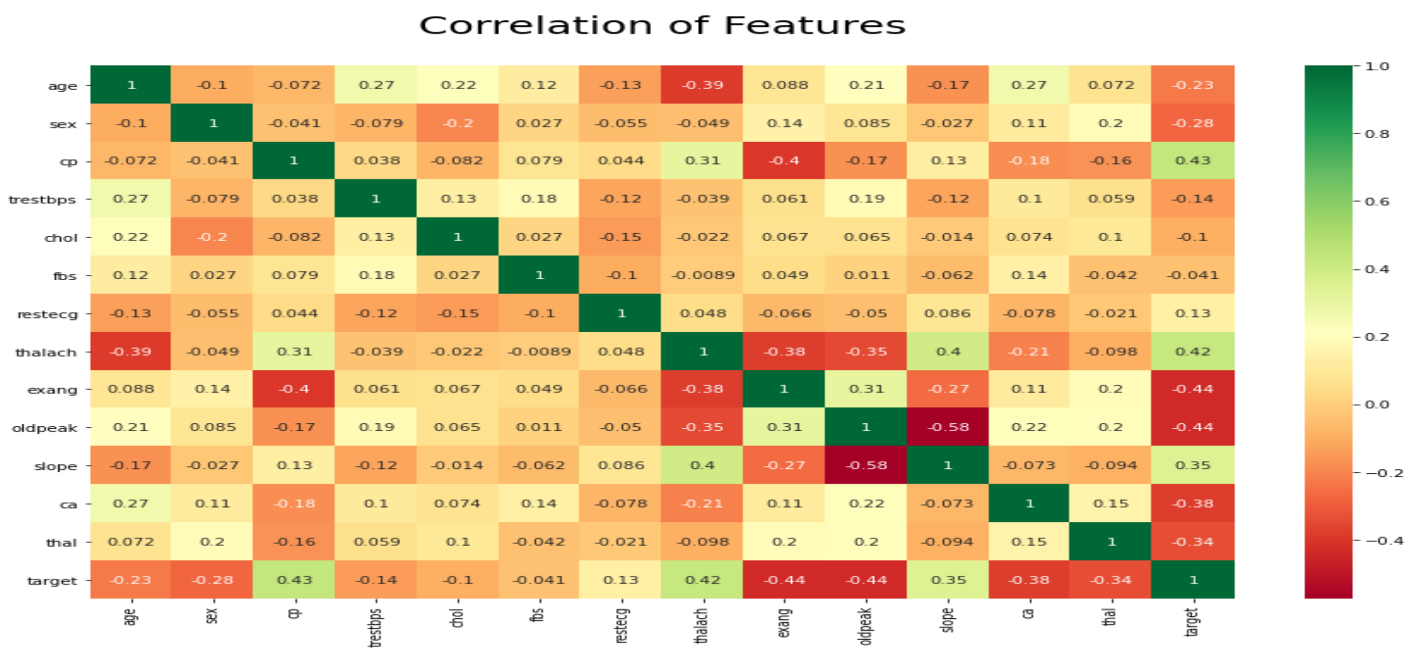


Fig. 6 Correlation of Features

The correlation between two unique variables is seen in each cell of Fig 6. The correlation between "target" and "cp” (chest pain type") is 0.43 in the cell of Fig 6, suggesting that they are highly positively correlated. That means the value of target has a high dependence on chest pain. Cp, thalach and slope attributes have a high positive correlation with the dependent variable. Exang, old peak and ca have a negative correlation with the target variable. Correlation values give an idea about the features to select for training the model.

III. METHODOLOGY

This paper focuses to find the best variety of artificial neural network algorithms, including Single Layer Perceptron (SLP), Multilayer Perceptron (MLP), Radial Basis Function (RBF) and Long Short Term Memory (LSTM), all of which can assist physicians or medical analysts in correctly diagnosing Heart Disease. The work includes gathering data sets, pre-processing them, and constructing binary classifier models with different ANN algorithms. Then we train the models with 80% of the data. After that, we use the trained models to predict the output of 20% of the data. After that, we will evaluate the results of each model and find the best model. We used the python language with Keras library to complete this project.

a) Data Preprocessing

The raw input data must be pre-processed after selecting the features; otherwise, the neural network would not provide reliable forecasts. There are no missing values in our dataset. The 14 attributes of the dataset came from a big dataset of 76 attributes. And these 14 attributes are used for all heart disease predictions. So, we are also selecting all these 14 attributes to train our model.

The decisions taken during this stage of growth are vital to a network's success. Two often-used pre-processing approaches are standardisation and normalisation. Standardization is a pre-processing method for transforming continuous data into a naturally distributed distribution. By subtracting the mean and dividing by the standard deviation for each input variable, itmodifies the distribution to have a mean of zero and a standard deviation of one. Therefore, we standardised our dataset.

b) ANN model life cycle in Keras

The steps in the ANN model life cycle in Keras are network definition, compile network, fit network, evaluate network and make predictions.

1. Define Network

In Keras, a model is depicted as a progression of layers. We start with a Sequential model and bit by bit add layers until we are happy with our model’s performance. We may try different things with the count of layers and their sorts, and the correct organization setup is constantly found by experimentation testing. The Dense class is used to describe completely connected layers. The count of neurons or nodes in the layer can be specified as the first parameter, and the activation function can be specified as the second argument. An activation function is a function that is applied to the output of a neural network layer before being moved on to the next layer as data. Nowadays, the ReLU activation feature is used to boost efficiency. We use a sigmoid on the output layer with a default threshold of 0.5 to guarantee that our network output is between 0 and 1, making it easy to map to either a probability of class 1 or a hard classification of either class.

2. Compile Network

We must define certain further properties required at the time of training the network when compiling. Note that when we train a network, we're searching for the right weights to map our dataset's inputs to outputs. We need to specify the loss function that will be used to measure a set of weights. During preparation, the optimizer is used to find various network weights as well as any extra measurements we choose to gather and record. The loss function in the proposed models is cross entropy. Each estimated class likelihood is compared to the real class desired outcome, which is either 0 or 1, and a score/loss is calculated that penalises the probability based on how much it differs from the intended result. Adam is the optimizer that we used in our proposed models. Adam is a hybrid of RMSprop and Stochastic Gradient Descent, but with the addition of momentum. It uses squared gradients to scale the learning rate, similar to RMSprop, and it uses a running average of the gradient rather than the gradient itself to take advantage of momentum, similar to stochastic gradient descent with momentum. Finally, since this is a binary classification problem, we gather and report classification accuracy, which is calculated by the metrics parameter.

3. Fit the Model

After compilation our model is ready for efficient computation. After that, the model must be checked or validated. By calling the fit() function on the model, we can train our model on our test. Training takes place in epochs, with each epoch separated into batches. An epoch is a pass-through to all the records of a training dataset. The batch size is a gradient descent hyperparameter that determines how many training samples must be processed before the model's internal parameters are modified. We want to train the model so that it learns how to map input data rows to classification outputs. The model will still have some error, but for a given model setup, the amount of error will eventually level out. This is known as Model convergence.

4. Evaluate the Model and Make predictions

We will determine the network's performance using the testing dataset (20% of total data), which was not used during preparation. This will require an assessment of the network's ability to make future predictions of unknown outcomes. The model assesses the loss of overall test trends as well as any other metrics defined at the time the model was created, such as classification accuracy. The result is a list of assessment metrics. We can use our fit model to make predictions on new data once we are pleased with its results. It is as simple as invoking the model's *predict*() function for an array of new input patterns.

Some many experiments were done to decide the optimum number of hidden layers and neurons, learning rate, epoch, batch size, dropout, and activation function of each ANN models.

c) *Artificial Neural Network*

Similar to biological neurons in the brain, an artificial neural network is made up of highly integrated processors known as neurons. The neurons are linked together by weighted connections that transfer signals from one neuron to the next. Weights in ANN are equivalent to synapses in biological neural networks, and they serve as a means of long-term memory in ANN, indicating the importance of a neuron input. ANN learns through a constant update of these weights; each neuron receives several inputs signals through its connection but only produces a single output.

1. *Single Layer Perceptron*

A single-layer neural network is the most basic type of neural network, with just one layer of input nodes sending weighted inputs to a matching layer of receiving nodes, or in some cases, to a single receiving node. This single-layer perceptron served as the base for a number of more complex systems. The perceptron will return a function dependent on inputs, which would be single neurons in the human brain's physiology. Perceptron models are similar to logic gates in that they perform specific functions: depending on the weighted inputs, a perceptron can either give a signal or not. SLP neural network does not contain any hidden layer. SLP can solve a linearly separable problem. XOR cannot be applied by a SLP. Since the output of XOR are not linearly separable. That means we can't separate the output classes with a straight line. Multi-layer networks were invented because of this.

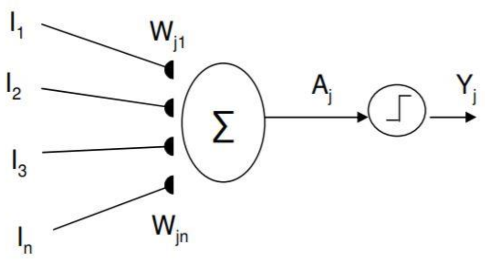
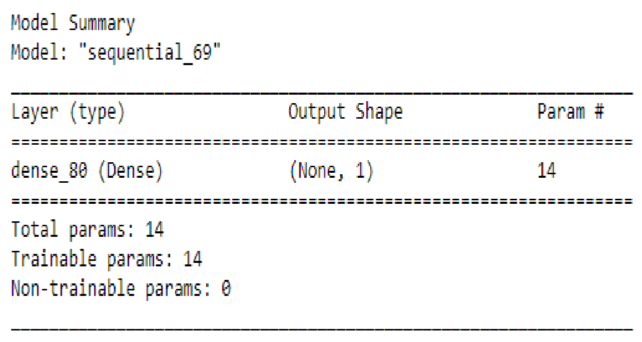


Fig. 7 Single Layer Perceptron

Fig 7 is the architecture of SLP.



```
Model Summary
Model: "sequential_69"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_80 (Dense)             (None, 1)                 14
=================================================================
Total params: 14
Trainable params: 14
Non-trainable params: 0

_________________________________________________________________
```

Fig. 8 Model Summary of SLP

Fig 8 is the model summary of SLP architecture. For our SLP model, we choose 60 epochs with a batch size of 10 to fit the model after trying different values.

2. *Multi-Layer Perceptron*

Multi-Layer Perceptron (MLP) is a feed-forward network. One or more hidden layers are present in an MLP (apart from one input and one output layer). As in Fig 9, the input layer receives the input signals or values. The output layer is responsible for tasks such as prediction and classification. Between the input and output layers, the MLP's real computational engine is an infinite number of unknown layers. Each node, with the exception of the input nodes, has a nonlinear activation function. In an MLP, data flows from input to output layer in the forward direction, similar to a feed-forward network. The backpropagation learning algorithm is used to train the neurons in the MLP. An MLP can learn non-linear functions as well as linear functions. In supervised learning problems, MLP's are often used. They practice on a series of input-output pairs to learn how to model the dependencies between inputs and outputs. During preparation, the model's parameters, or weights and biases, are modified to minimise error. Backpropagation is used to change the weight and bias in relation to the error, which can be calculated in some ways, including root mean squared error (RMSE).

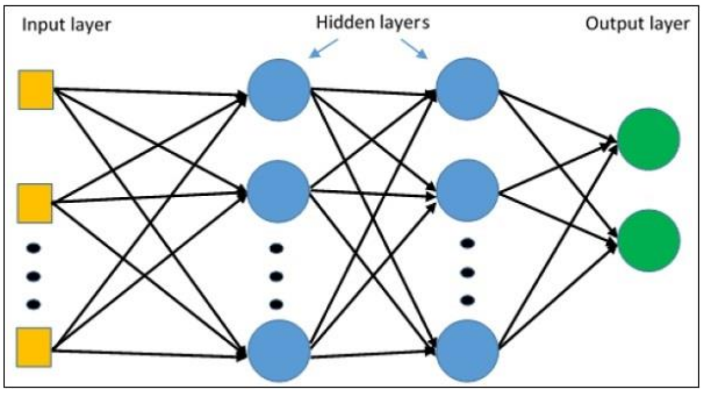
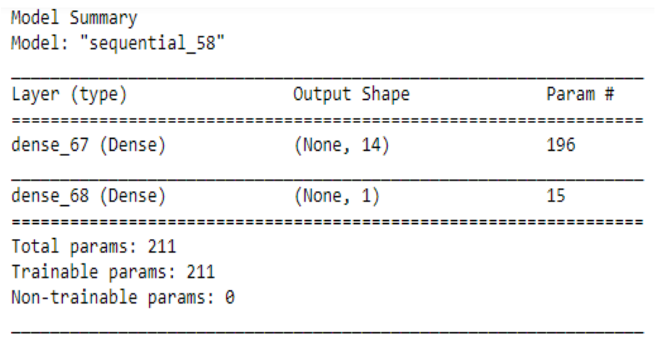


Fig. 9 MLP Architecture

In the forward transition, the pulse passes from the input layer to the output layer, going through the hidden layers, and the output layer's decision is linked to the ground truth points. Backpropagation and the chain rule of calculus are used to backpropagate partially derivatives of the error function with respect to various weights and biases using the MLP in the backward pass. Differentiation produces an error environment, or gradient, along which the parameters can be tweaked to get the MLP closer to the error minimum. This can be achieved with any gradient-based optimisation algorithm, such as stochastic gradient descent. This state is referred to as convergence.

We created MLP with a single hidden layer and two hidden layers. The activation function used in hidden layers is rectified linear (ReLU). The loss function and Optimizer used in the MLP model are cross-entropy and *adam* optimizer.

Fig 10 shows the model summary of MLP model



```
Model Summary
Model: "sequential_58"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_67 (Dense)             (None, 14)                196

_________________________________________________________________
dense_68 (Dense)             (None, 1)                 15
=================================================================
Total params: 211
Trainable params: 211
Non-trainable params: 0

_________________________________________________________________
```

Fig. 10 Model Summary of MLP with Single Hidden Layer

3. *Radial Basis Function*

A feed-forward network, the RBF neural network can be equipped using supervised learning algorithms. The RBF network's key benefit is that it only has one hidden layer and employs the radial base function as an activation function [7]. These functions are particularly useful for approximation. Many researchers were attracted to successfully implement in a variety of problem areas. RBF networks learn even more quickly than backpropagation networks. This type of network is less vulnerable to intermittent input issues compared to the behaviour of wireless networks. Basic functions are hidden blocks.

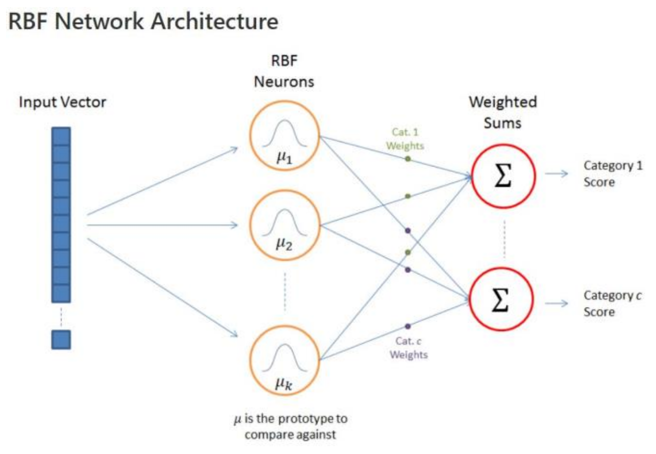
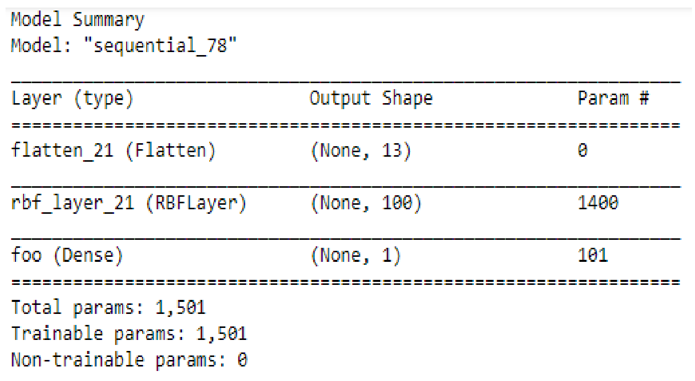


Fig. 11 RBF Architecture

Although RBF has only one hidden layer, it can solve any classification problem. If changing the weight does not solve the error we found, sometimes we may need to add some hidden layers. If the centre and deviation are already known from the program area and can also be hard-coded, the best results can be achieved. In these cases, it is generally recommended to use RBF instead of a perceptron. As long as there are multiple output neurons, RBF will be better than Perceptron, despite the hidden single-layer problem.

The input layer converts the data into hidden neurons containing radial basis functions as shown in Fig 11 . his function is used to evaluate the distance between the network inputs and the hidden layer's centre. Returns the sum of the hidden layer's contribution with a fixed weight. The number of neurons in the hidden layer is one of the most interesting and crucial tasks since the number of neurons in the hidden layer will cause mismatch and overfitting problems. If there are not enough neurons, it may cause a mismatch, which means that the network cannot identify patterns properly. Similarly, overfitting can lead to poor comprehensiveness**.**



```
Model Summary
Model: "sequential_78"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_21 (Flatten)         (None, 13)                0

_________________________________________________________________
rbf_layer_21 (RBFLayer)      (None, 100)               1400

_________________________________________________________________
foo (Dense)                  (None, 1)                 101
=================================================================
Total params: 1,501
Trainable params: 1,501
Non-trainable params: 0
```

Fig. 12 Model Summary of RBF

We created RBF model with 100 neurons in the hidden layer. There are 1501 trainable parameters for our model.

4. *Long Short-Term Memory (LSTM)*

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) that can learn order dependency in sequence prediction problems. LSTM was developed to address the issue of error back-flow. Memory blocks are recurrently connected blocks that make up an LSTM layer. These blocks are a distinguishable version of the memory chips in a digital computer. There are one or more recurrently attached memory cells in each block, as well as three multiplicative units (input, output, and forget gates) [4]. For the cells, this offers continuous analogues of write, read, and reset operations. The neural net can only interact with the cells via the gates. An LSTM can be used to conserve gradients. As long as the forget gate is open and the input gates are closed, the memory cell remembers the first input. The most important tuneable LSTM hyperparameters are learning rate and network size. Fig 13 shows the LSTM Architecture

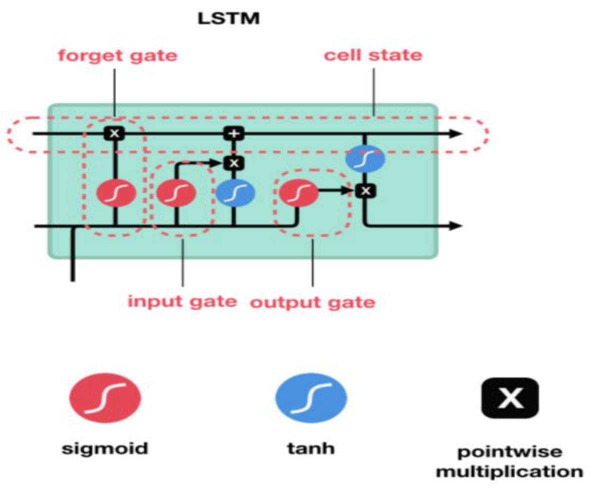
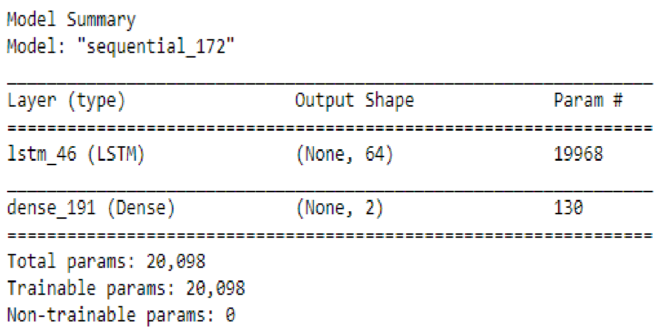


Fig. 13 LSTM Architecture

In the case of training non-sequential data with LSTM, if everything works properly, the recurrent paths and state variables will just be ignored. So, the recurring stuff is a waste of hardware that doesn't support or harm network capacity — may just add a little noise.



```
Model Summary
Model: "sequential_172"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_46 (LSTM)               (None, 64)                19968
_________________________________________________________________
dense_191 (Dense)            (None, 2)                 130
=================================================================
Total params: 20,098
Trainable params: 20,098
Non-trainable params: 0
```

Fig. 14 Model Summary of LSTM

LSTM has more parameters when compared to other proposed neural networks. We used the activation function as softmax.

IV. EXPERIMENTAL SETUP AND RESULT EVALUATION.

In Python, we created a binary classification model using single-layer perceptron, multilayer perceptron, radial basis function and long short term memory for forecasting heart disease.

A. *Performance Evaluation*

For this binary classification problem, we are using accuracy, precision, recall and F1 score to evaluate the performance of each model. The counts of test records correctly and incorrectly estimated by a classification model are used to evaluate the model's performance. The confusion matrix gives a more detailed image of not only a predictive model's results but also which classes are being forecast correctly and incorrectly, as well as the types of errors made.[5]

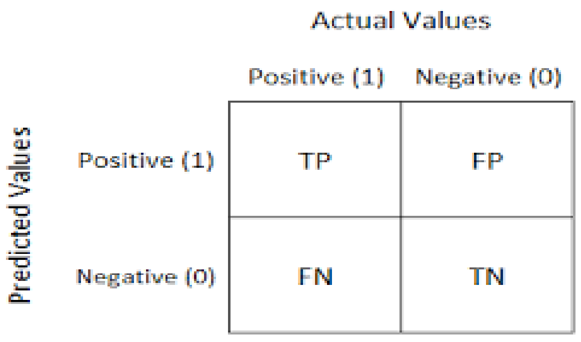


Fig. 15 Confusion Matrix

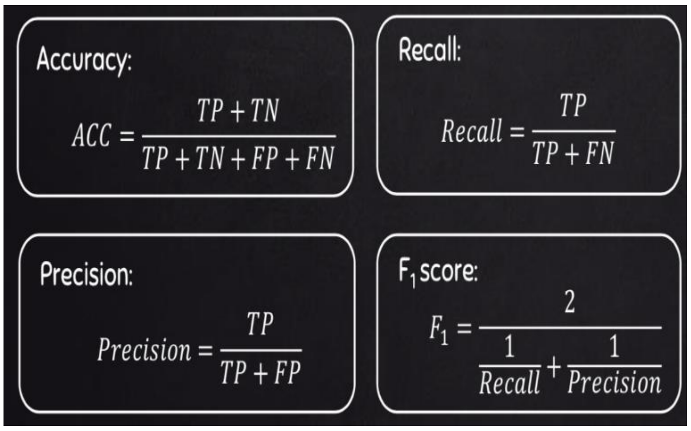
Accuracy - The number of classifications a model correctly forecast divided by the total number of predictions is known as model accuracy.

Precision - The ratio of True Positives to all the positives predicted by the system is known as precision.

Recall - The percentage of True Positives to all positives in your Dataset is known as Recall (Sensitivity).

F Score - The F-score, which is known as the harmonic mean of the model's precision and recall, is a way of combining the model's precision and recall.

Fig 16 contain the equation of performance evaluation metrics.



Accuracy:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall:

$$Recall = \frac{TP}{TP + FN}$$

Precision:

$$Precision = \frac{TP}{TP + FP}$$

F_1 score:

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

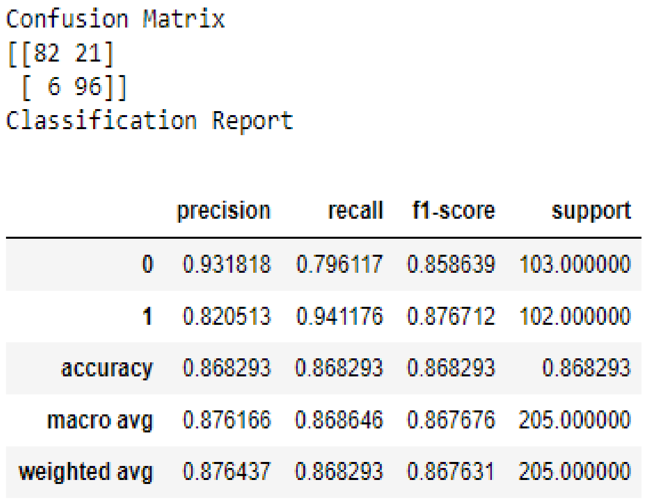
Fig. 16 Performance Evaluation Metrics

All the models have better performance on standardised data rather than raw data. LSTM has almost similar performance on the non-standardised dataset. We use a validation dataset as a test dataset (20 percentage of total data). The validation dataset is a technique to avoid overfitting of data.

Let's take a look at the results of each model we developed.

1) *Single Layer Perceptron (SLP)*

From the confusion matrix in Fig 17, we can understand that the SLP model correctly predicts 96 patients who have heart disease. But it fails to predict the heart disease of 6 patients. The classification report displays the accuracy of the model for the test dataset is 86.8 percentage. Precision is larger than recall for this model.



```
Confusion Matrix
[[82 21]
 [ 6 96]]
Classification Report
```

	precision	recall	f1-score	support
0	0.931818	0.796117	0.858639	103.000000
1	0.820513	0.941176	0.876712	102.000000
accuracy	0.868293	0.868293	0.868293	0.868293
macro avg	0.876166	0.868646	0.867676	205.000000
weighted avg	0.876437	0.868293	0.867631	205.000000

Fig. 17 Confusion Matrix and Classification Report of SLP

From the Fig. 18 of model accuracy, we can see that the model attains maximum accuracy near 50 epochs and became stable after 50 epochs. If we increase the number of epochs, the model will overlearn the data. That means the model will learn the noise as part of data and that will lead to overfitting [6].

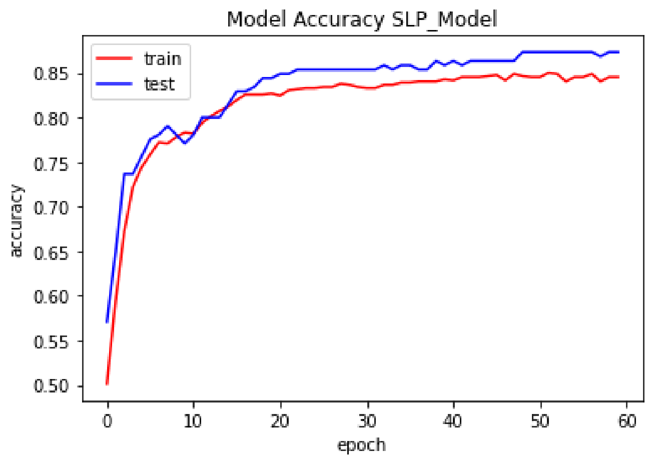


Fig. 18 Model Accuracy Plot

We can see from the loss plot Fig 19 that the model performs similarly on both the train and validation datasets (test dataset). If these parallel plots begin to diverge consistently, it might be time to avoid training at a previous epoch.

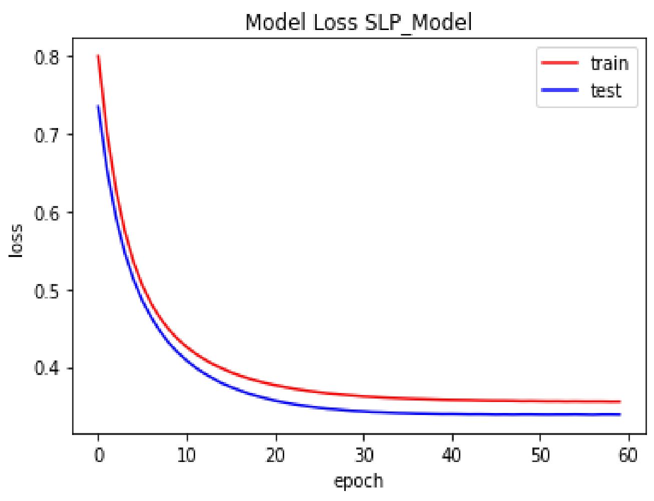
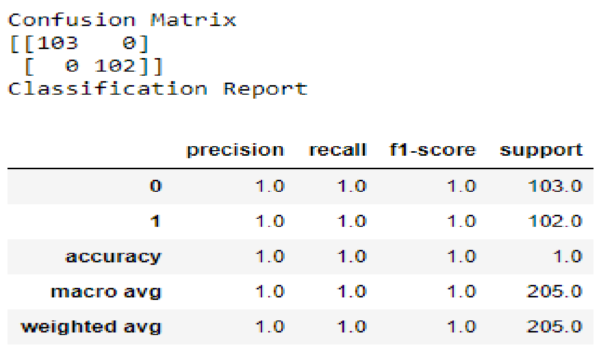


Fig. 19 Model Loss Plot

2) Multi-Layer Perceptron (MLP)

MLP with a single hidden layer with 14 neurons achieved 100 % accuracy in 300 epochs of batch size 10. At the same time, MLP with 2 hidden layers achieved 100 % accuracy in 150 epochs of batch size 10.



```
Confusion Matrix
[[103   0]
 [  0 102]]
Classification Report
```

	precision	recall	f1-score	support
0	1.0	1.0	1.0	103.0
1	1.0	1.0	1.0	102.0
accuracy	1.0	1.0	1.0	1.0
macro avg	1.0	1.0	1.0	205.0
weighted avg	1.0	1.0	1.0	205.0

Fig. 20 Confusion Matrix and Classification Report of MLP

From the Fig 21 of model accuracy, we can see that the model attains maximum accuracy after 250 epochs.

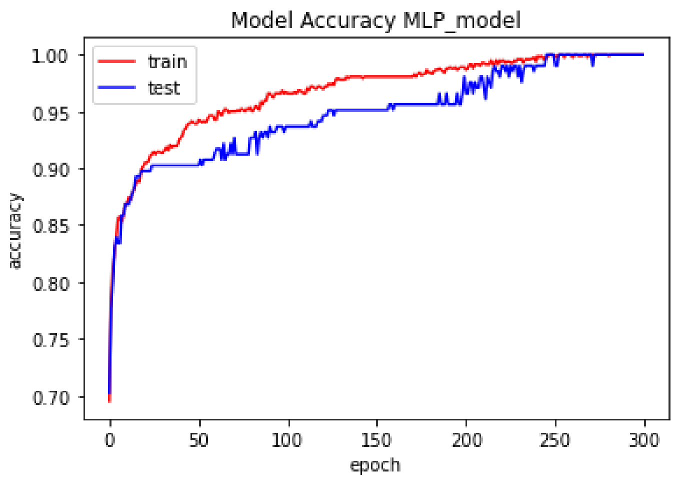


Fig. 21 Model Accuracy Plot

We can see from the loss plot Fig 22 that the model performs similarly on both the train and validation datasets (test dataset). The plot shows a small divergence between train and test data after 20 epochs and returns after 150 epochs. After 250 epochs, the graph became parallel.

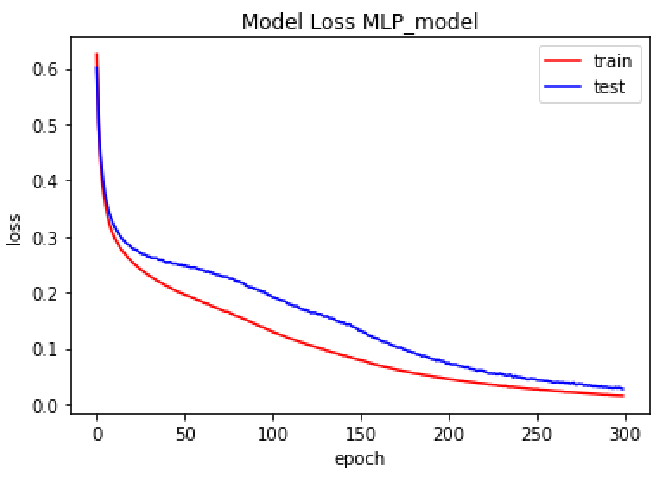
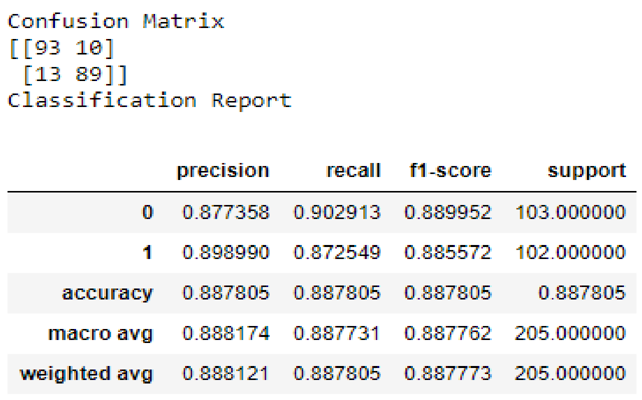


Fig. 22 Model Loss Plot

3) Radial Basis Function (RBF)

From the confusion matrix in Fig 23, it is clear that the RBF model correctly predicts 89 patients who have heart disease. But it fails to predict the heart disease of 13 patients. The classification report displays the accuracy of the model for the test dataset is 88.7 %.



```
Confusion Matrix
[[93 10]
 [13 89]]
Classification Report
```

	precision	recall	f1-score	support
0	0.877358	0.902913	0.889952	103.000000
1	0.898990	0.872549	0.885572	102.000000
accuracy	0.887805	0.887805	0.887805	0.887805
macro avg	0.888174	0.887731	0.887762	205.000000
weighted avg	0.888121	0.887805	0.887773	205.000000

Fig. 23 Confusion Matrix and Classification Report of RBF

Model Accuracy graph in Fig 24 of RBF model shows consistent accuracy of 88 % after 50 epochs.

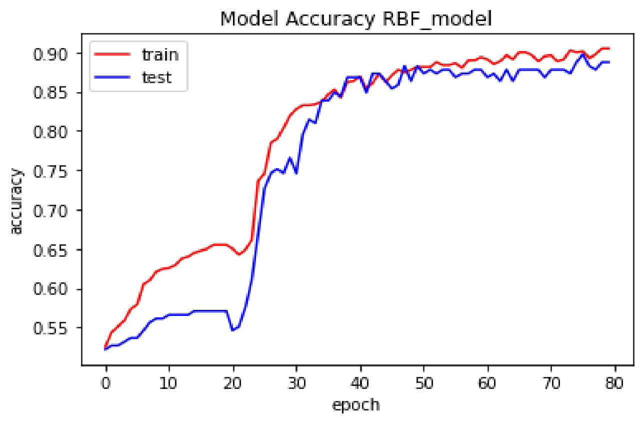


Fig. 24 Model Accuracy Plot

Model Loss plot of RBF model in Fig 25 illustrates a divergence on the loss values of train and test data after 50 epochs. This is a warning to reduce the epoch.

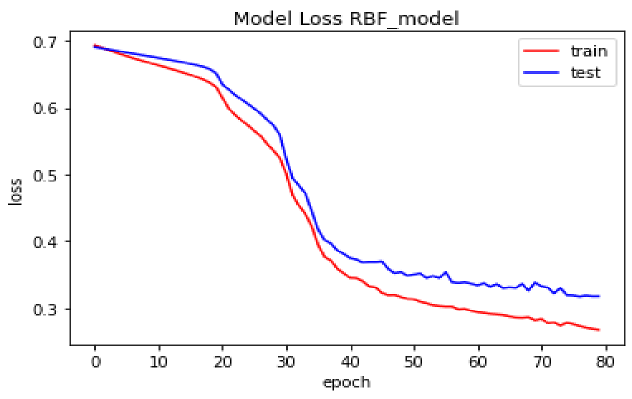
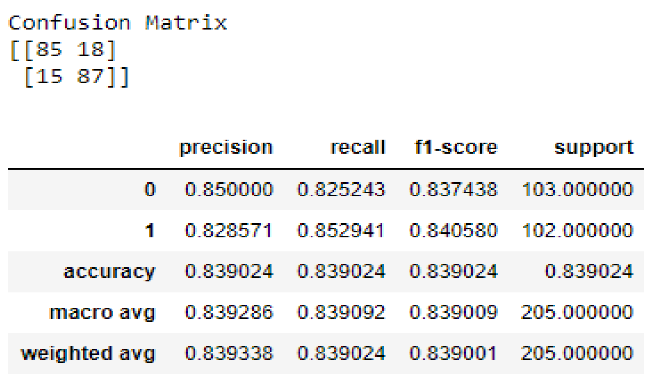


Fig. 25 Model Loss Plot

4) <u>Long Short-Term Memory</u>

From the confusion matrix in Fig 26, it is clear that the RBF model correctly predicts 87 patients who have heart disease. But it fails to predict the heart disease of 15 patients. It also correctly predicts 85 people does not have the chance of heart disease. The classification report displays the accuracy of the model for the test dataset is 83.9 %. The precision, recall, f1 score are around 83.9 %.



```
Confusion Matrix
[[85 18]
 [15 87]]
```

	precision	recall	f1-score	support
0	0.850000	0.825243	0.837438	103.000000
1	0.828571	0.852941	0.840580	102.000000
accuracy	0.839024	0.839024	0.839024	0.839024
macro avg	0.839286	0.839092	0.839009	205.000000
weighted avg	0.839338	0.839024	0.839001	205.000000

Fig. 26 Confusion Matrix and Classification Report of LSTM

From the Fig. 27 of model accuracy, we can see that the model attains the accuracy of 83 % after 40 epochs and became stable after that. If we increase the number of epochs, the model will overlearn the data. There is a layer with dropout to avoid overfitting of data.

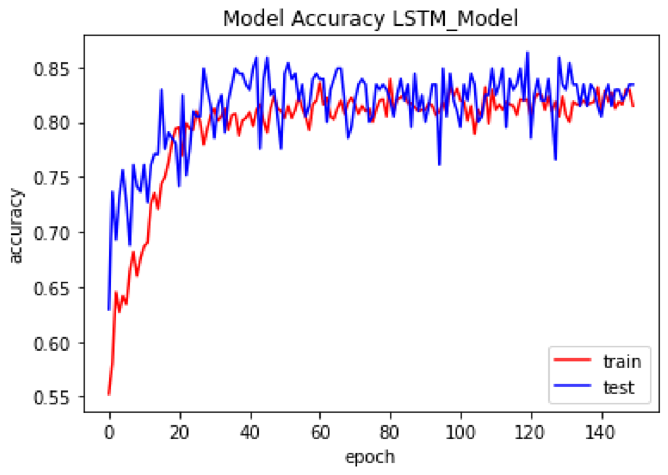


Fig. 27 Model Accuracy Plot

The model loss Fig. 28 also became stable after 50 epochs. That means we can reduce the number of epochs. LSTM works well with sequential data. Our data is not sequential. So, the hardware to memorise the inputs is not useful for our problem.

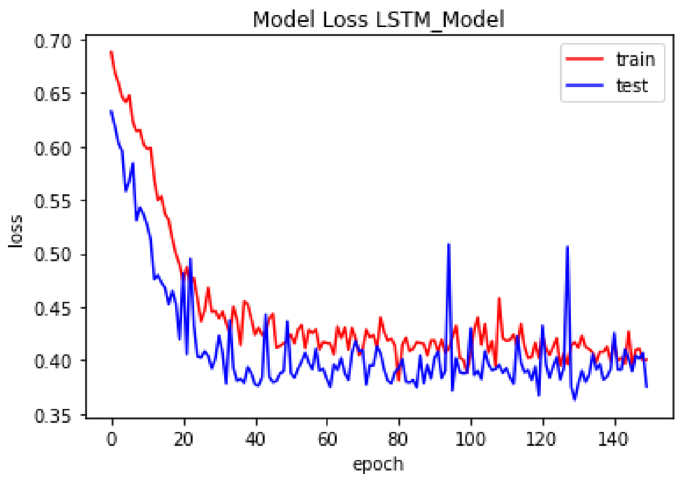
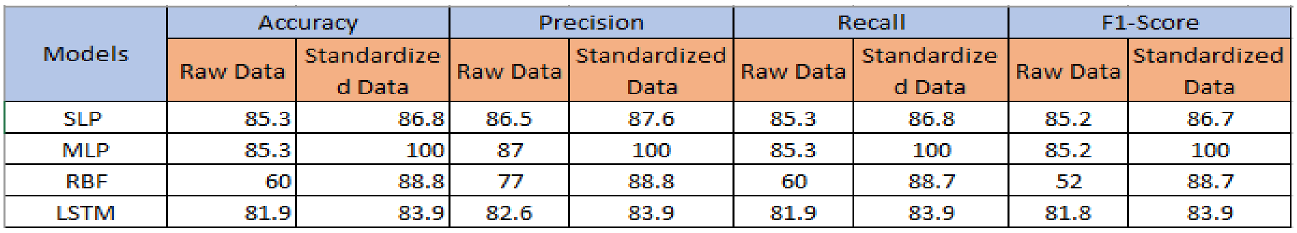


Fig. 28 Model Loss Plot

B. RESULT EVALUATION

The Table 2 shows the values of performance evaluation of four ANN models SLP, MLP, RBF and LSTM with raw and standardized data. It is obvious that MLP is the best model with 100% accuracy, precision, recall and f1 score for the standardised data. MLP with a single hidden layer requires 300 epochs to become the best model. But MLP with two hidden layers only requires 150 epochs. RBF is better than SLP and LSTM with an accuracy of 88.8%t on standardised data. LSTM has the lowest accuracy of 83.9 among other ANN models. LSTM works well with sequential data.

Table 2 Model Evaluation Summary



Models	Accuracy		Precision		Recall		F1-Score	
	Raw Data	Standardized Data	Raw Data	Standardized Data	Raw Data	Standardized Data	Raw Data	Standardized Data
SLP	85.3	86.8	86.5	87.6	85.3	86.8	85.2	86.7
MLP	85.3	100	87	100	85.3	100	85.2	100
RBF	60	88.8	77	88.8	60	88.7	52	88.7
LSTM	81.9	83.9	82.6	83.9	81.9	83.9	81.8	83.9

V. CONCLUSION

In this report, various artificial neural network approaches were used to compare the classification of the Heart Disease dataset for positive and negative diagnosed participants. Single Layer Perceptron (SLP), Multilayer Perceptron (MLP), Radial Base Function (RBF), and Long Short Term Memory are the algorithms used to solve the problem .To determine the network's optimum parameters, multiple experiments were performed. Overall, the improved ANN system had high diagnosing accuracy. Overall, the improved ANN method had around 100 % diagnosing accuracy, indicating its effectiveness in the diagnosis of heart disease. The ANN methods like MLP, RBF and SLP provided very promising results in terms of classification accuracy as compared to the LSTM neural network, with a classification accuracy of 100, 88, and 86%, respectively.

VI. ACKNOWLEDGEMENT

I would like to express my gratitude to Mr Abhijith Shaji, with whom I collaborated on this coursework. We have preferred five neural network models, three of which have shared by both, and the other two were validated and evaluated separately. We had collaboratively worked on ANN algorithms such as Single Layer Perceptron (SLP), Multilayer Perceptron (MLP) and Radial Basis Function (RBF).

VII. APPENDIX

a). Source Code

https://livecoventryac-my.sharepoint.com/:f:/g/personal/jamesj29_uni_coventry_ac_uk/Eqsh0TaI0XVLpHRoDM7bJjgB93E544AW3_CKkTsT9gnv9g?e=eeMfCQ

VIII. REFERENCES

[1] WHO [Online]: Cardiovascular diseases (who.int)

[2]Kaggle Dataset [Online]: https://www.kaggle.com/johnsmith88/heart-disease-dataset

[3] UCI Dataset [Online]: https://archive.ics.uci.edu/ml/datasets/Heart+Disease.

[4] Machine Learning mastery [Online] https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

[5]Kdnuggets {Online] https://www.kdnuggets.com/2020/04/performance-evaluation-metrics-classification.html

[6] Elite Data Science[Online] https://elitedatascience.com/overfitting-in-machine-learning

[7] Science Direct

https://www.sciencedirect.com/topics/engineering/radial-basis-function-network