Abstract

Deep learning healthcare applications have evolved over the last few years. This advancement leads to new applications and possibilities in the field of health care. Due to different variants of deep learning algorithms like convolutional and recurrent neural network these advancements in healthcare application made possible. Healthcare has shown promises, as there is a huge share of the budget in intensive care medicine. This has expanded the interests of healthcare researchers, and they are focusing on providing the best medication to critically-ill patients. DNN models are advanced machine learning models, which have been widely applied for clinical applications in recent times. Due to the lack of ability to capture time dependency in the data which is a major shortcoming, basic machine learning algorithms like Logistic Regression, SVM, Decision Tree, etc. are not very useful. There are several types of deep learning algorithm like Deep belief networks, Deep convolutional neural network, Recurrent neural network, etc. The LSTM variant of the Recurrent neural network have shown promises to work better on data involving time series prediction, and it can transfer the results of previous cell state onto the next cell state so that dependency on time is maintained throughout. In this thesis, the proposed methodology is to use LSTM, and its different variants like Gated Recurrent Unit (GRU), LSTM attention on healthcare applications like hospital readmission prediction using ICU data. The different architectures of LSTM will produce different results based on the data on which architecture is being trained. Then the comparison of performances can be done between LSTM, and it's variants on various healthcare applications and provide the best model suitable for each of the different healthcare data being tested. The comparison of time independent machine learning models like Logistic Regression, SVM, Naive Bayes can be done with the results of Deep learning models.

Keywords: Deep learning, LSTM, Healthcare, Recurrent Neural Network, GRU.

Chapter 1

Introduction

This chapter presents an overview of the context as a part of Hospital Readmission project and how machine learning used in health care applications in section 1.1 and 1.2, respectively. In section 1.3, 1.4, and 1.5, we discuss the algorithms used in this project. In section 1.6, the problems and motivations are presented. Next, in section 1.7, we discuss the objectives of this project.

1.1 Hospital Readmission

Medical clinic Readmissions are distinguished as a sign of the low quality of consideration, for example, inadequate release arranging and care coordination. Emergency clinic readmission[32] happens when a patient release from a medical clinic is readmitted inside a predetermined interim. Readmission can happen for any decided or undecided reason. These elements drove Hospital associations to consider controlling readmission rates; an expansion in readmission rates prompts an increment in punishments. There has been an expanded use of Readmission rates as a result rule in wellbeing administrations investigation and as a quality benchmark for wellbeing frameworks. For Medicare patients, hospitalizations are exceptionally distressing, and it is considerably more when they result in ensuing readmissions. A ton of research concentrates demonstrated that emergency clinics could take part in a few exercises like explaining patient release directions, planning with post-intense consideration suppliers, lively cleaning system to lessen the rate of readmissions of patients. Be that as it may, these individual subsequent meet-ups can be costly.

1.2 Machine Learning For Health Care

Machine learning leads to provide intelligence by providing new knowledge. Since the onset of machine learning, it has been widely used in achieving paramount goals as the healthcare sector is evolving. The healthcare sector is producing an enormous mass of data, and it is beyond human capability to manually analyze this vast data and produce inferences based on that. Machine learning has proven to be a benefit at this level. The machine learning algorithms automatically find patterns in the data, which enable to get inferences from the data easily. Machine learning can assist healthcare providers in a variety of tasks like hospital readmission of patients suffering from chronic illness. This can not only lower the lifetime risk of the patient but also can save millions of dollars if the algorithm can predict efficiently and correctly. Other applications of machine learning in the healthcare domain includes drugs combination which should be avoided taking together, classifying images for different diseases like skin cancer. Keeping this in the picture, it can be said that machine learning can revolutionize the healthcare sector.

1.3 Deep Learning

Deep learning is one of the extended branches of Machine learning, which works on a specific machine learning algorithm, that is the artificial neural network. It's leveraging the recent advancements in computing power and thus using specific purpose neural network to learn from a plethora of data and make predictions based on the patterns detected. The structure consists of several hidden layers which are meant to extract the information by exploiting the structures present in the data. Deep neural networks are specialized in solving problems, including data, which is greatly structured. It also promises to replace hand-engineered features with feature extraction techniques that are hierarchical, unsupervised, or semi-supervised. Multiple hidden units are being used between input and output in case of Deep Neural Network (DNN). Similar to the artificial neural network, the complex non-linear relationships can be modeled using DNN[2]. Similar to the artificial neural network, the complex non-linear relationships can be modeled using DNN.

1.4 Recurrent Neural Network

An artificial neural network has a particular type called recurrent neural network (RNN), and it has directed cycles between connection. The internal state of the network exhibits dynamic temporal behavior. Unlike ANN, arbitrary sequences can be processed by using the internal state of RNN. This is the fundamental architecture developed in the 1980s: neuron-like units in a network, a directed connection between every neuron. There is a time-varying real-valued activation in every unit. There is modifiable real-valued weight in each connection. There are input, output, and hidden nodes. One input vector is supplied at every time steps. At every time step, the non-linear function of the weighted sum of all the activations of connected units is calculated by each non-input unit[25].

1.5 Long Short Term Memory

The Long short-term memory (LSTM) network, a deep learning RNN, is used by numerous researchers, published by Schmidhuber & Hochreiter in 1997[19]. This deep learning system overcomes the vanishing gradient problem of traditional RNN. The recurrent gates called forget gates are augmented in LSTM. The vanishing or exploding of backpropagated error is prevented in LSTM RNNs. There are an unlimited number of virtual layers in LSTM RNNs, through which the error can flow backward. LSTM can learn from events that occurred thousands of time steps ago so it can be used for very deep learning tasks. LSTM has a unique architecture that consists of a memory cell, which can maintain the state which it is currently in, over time. Non- Linear gating units are used in LSTM to regulate the information flow in and out of the memory cell. Over time, many improvements have been suggested and made in the standard architecture of LSTM to make it more efficient. Today, LSTMs are being used in a variety of learning problems which differ in scale and nature significantly when compared to the problems which were initially solved by LSTM.

1.6 Motivation

- For Medicaid patients, hospitalizations can be distressing, even more so when they occur in frequent readmissions. A number of researches show that health cares can involve in several projects to lower their degree of readmissions, such as simplifying patient discharge procedure, coordinating with post acute care providers etc.
- The purpose of this thesis is to use deep neural networks which can work effectively to predict the result that can be utilized to avoid unnecessary hospital readmissions.

1.7 Objectives

The primary objective of this thesis is:

- To develop a Deep Neural Network(DNN) models for Healthcare application, i.e., Hospital Readmission.
- To predict hospital readmission of patients using trained deep neural network models.
- To compare results between Deep Learning Models and with existing algorithms.

1.8 Research work flow

This section focuses on how the research is being carried out, how the different algorithms, models and data prepossessing are implemented.

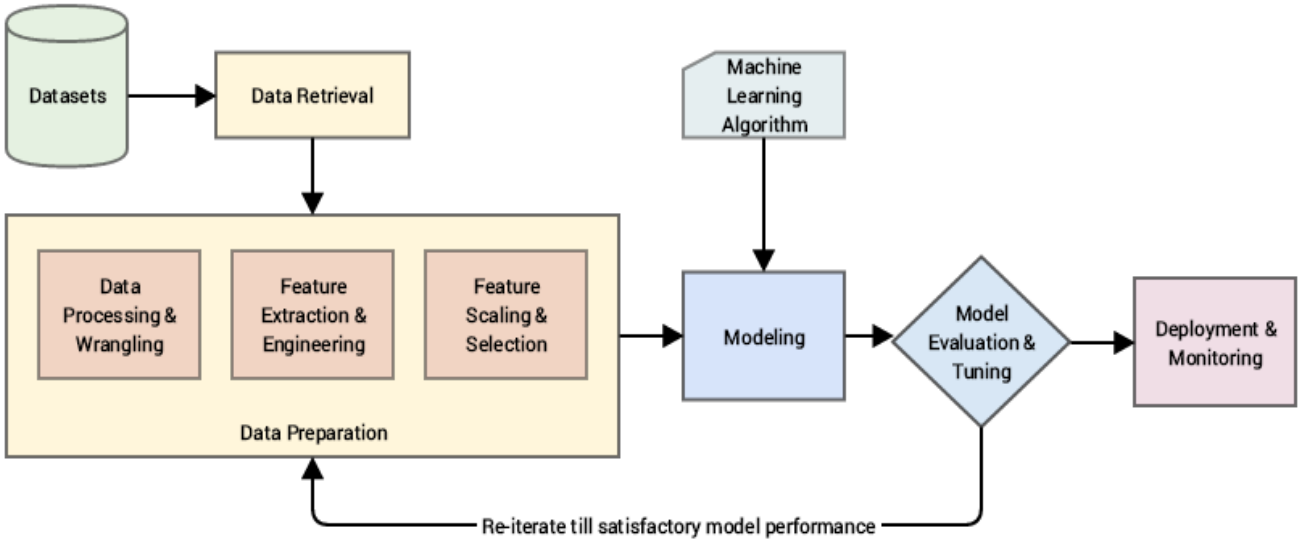

Figure 1.1: Flowchart of proposed methodology

According to the research objectives, the report will describe the work flow as below:

Step 1: Collect a vast readmission dataset with several features.

Step 2: Do data-preprocessing to deal with missing values.

Step 3: Convolution layers can be used for feature extraction, i.e., to retrieve the meaningful data out of the dataset.

Step 4: Do Feature scaling to normalize the dataset to reduce biases.

Step 5: Feature selection can also be done using nature-inspired algorithms to fetch important features from the dataset.

Step 6: Feeding the retrieved dataset into the model.

Step 7: Calculate the Accuracy, FScore, AUC, etc. based on the actual and predicted output.

Step 8: If the results are satisfied, then stop otherwise go to step 2 and repeat the process with different hyperparameters, algorithms, etc.

- These earlier works were not able to model high dimensional nonlinear relations as good as RNN.
- Descriptive statistics were being used in earlier methods. However, these statistics like mean, median & mode are always under the risk of losing some vital information.
- Deep learning algorithms are still under a shadow for a variety of healthcare applications.

2.4 Novelity

- With this thesis, the aim is to use deep learning algorithms to overcome the drawbacks of conventional machine learning algorithms.
- The thesis aims to use nature-inspired algorithms for feature extraction.
- The thesis also aims to use the hybrid model (Convolution Recurrent Neural Network), which can further increase the accuracy of the proposed model.

2.5 Conclusion

Hospital Readmissions is a very difficult thing for both sufferer and health center. In our research, we built a robust algorithm to determine the number of patients readmitted to a health care. We compared different deep learning models with and without Nature-inspired algorithms to predict readmission probability and concluded that Long Short Term Memory with grey wolf optimizer performed better than the rest of the machine learning algorithms in the prediction quality. We also establish that the result of the combination of LSTM and Convolution layer was remarkable on this dataset. This framework can be applied in today's health system to aim at high risks patients, reduce the rate of readmission, and deliver better health care.

Chapter 3

Methodology

This section introduces the dataset description and the analytical validation of the proposed solution.

3.1 Data description: MIMIC-III

Medical Information Mart for Intensive Care (MIMIC-III)[23] consists of data about patients admitted to various critical care units in a large hospital. MIMIC-III is generally viewed as a large and single-center database. A large number of different parameters are present in MIMIC III database. These parameters include information such as vital signs, medications, laboratory measurements, observations, fluid balance, procedure codes, diagnostic codes, imaging reports, hospital length of stay, survival data, and others. The database consists of information of around 58,576 distinct patients who were admitted to various critical care units of the hospital between 2001 and 2012. The data comprises of patients aged 16 years or above only. Descriptive statistics were performed on this dataset, and it was found that the median age is 65.8 years for adult patients(Q1-Q3: 52:877:8). Out of total patients, there were 55.9% male patients, and only 11.5% of the cases had in-hospital mortality. It was calculated from the data that on an average a patient stayed for 2.1 days in ICU and had an average of 6.9 Days of hospital stay. MIMIC-III database have several idiosyncratic properties about it, and these are mentioned below:

- The dataset is a vast database accumulated throughout more than a decade and consists of very detailed information of each patient under care.

- MIMIC-III database requires a user to oblige by a user data agreement and once itis accepted the analysis is boundless.
- It contains discharge summaries as well as reports of ECG, imaging studies and information about various codes like International Classification of Disease, 9thEdition (ICD-9) codes, Diagnosis Related Group (DRG) codes, and Current Procedural Terminology (CPT) codes.
- It is a time series of data. Clinical variables are recorded concerning time for each patient. Figure 1 shows the chart of timing clinical variables recorded in the dataset.

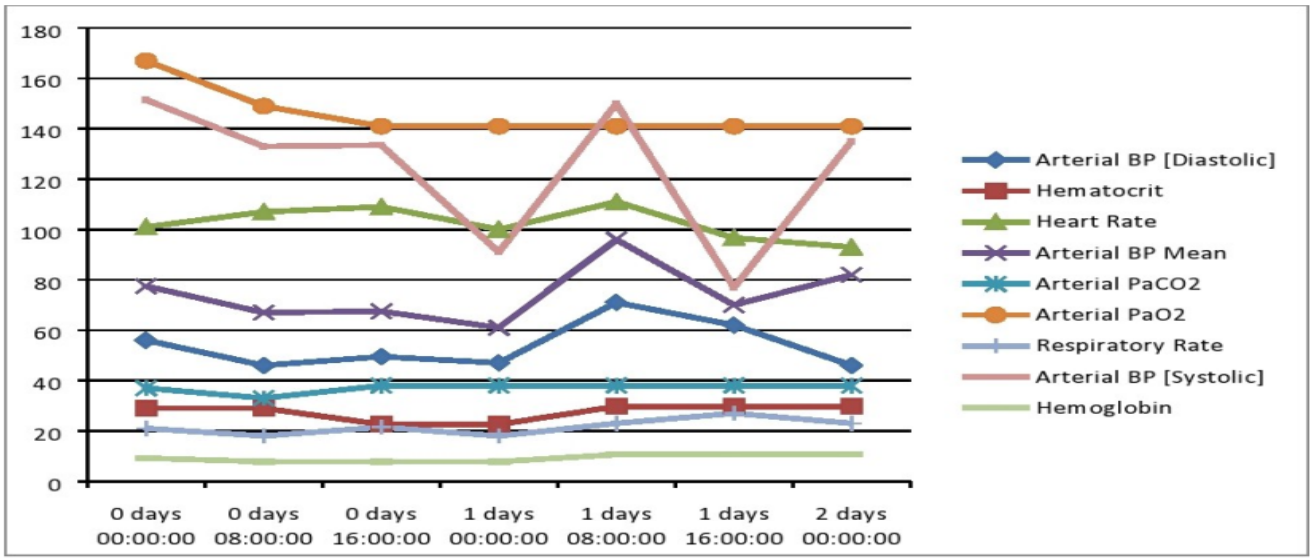

Figure 3.1: Some of the Clinical recordings associated with each patient[23]

3.2 Mechanism/Algorithm

3.2.1 Dataset Preprocessing

3.2.1.1 Data Extraction

MIMIC-III dataset consisted of around 58,576 patients who were diagnosed as suffering from different types of disease and hence mortality due to those diseases. These diseases include Pulmonary disease, Circulatory disease, Trauma, a disease of the digestive system, and many more.

Since the dataset is very large, we only consider data of those patients who were readmitted again, which gives the details of 7,534 patients. The data set is divided into two classes:-

- Patients readmitted in 30 days.
- Patients readmitted after 30 days.

3.2.1.2 Missing Values

MIMIC-III dataset contains missing values in some of the features. The feature will be removed if it contains more missing value otherwise mean will be used to fill the missing values.

3.2.1.3 Normalization

Normalization is used to reduce the biases among the attributes. It presents the data on a command scale. It standardizes the span of independent attributes or variables of data, called feature scaling. We use min-max[21] normalization here.

- Min-Max Normalization

  Let a matrix "A":-

$$A = \begin{Vmatrix} a11 & a12 \\ a21 & a22 \end{Vmatrix} \tag{3.1}$$

  So for normalization we calculate

$$a = min(a11, a21), \tag{3.2}$$

$$b = max(a11, a12) \tag{3.3}$$

$$c = min(a12, a22), \tag{3.4}$$

$$d = max(a21, a22) \tag{3.5}$$

  And "A" becomes:-

$$A = \begin{Vmatrix} (a11 - a)/(b - a) & (a12 - c)/(d - c) \\ (a21 - a)/(b - a) & (a22 - c)/(d - c) \end{Vmatrix} \tag{3.6}$$

3.2.2 Feature Selection

Features are selected through Nature Inspired Algorithms:

3.2.2.1 Bat Algorithm

Bat Algorithm is an optimization algorithm inspired by the echolocation behavior of microbats[34]. Bats uses echolocation to distinguish between food/prey and they also sense distance and other background barriers. Bats fly at random with velocity v_i at location x_i with a fixed frequency f_{min}, varying wavelength and loudness A_0 to search for the prey. They can automatically change the wavelength (or frequency) of their emitted pulses and regulate the rate of pulse emission r in the scale of [0, 1], depending on how close is their target. Although the loudness can vary in many ways, we assume that the loudness changes from a high (positive) A_0 to a minimum constant value A_{min}. Rules to update position x_i and velocity v_i of bats at time step t is given by:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{3.7}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_b)f_i \tag{3.8}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{3.9}$$

where $\beta \in$[0,1] is a random vector extract from a uniform distribution and x_b is the current global best solution which is found after comparing all the solutions among all the n bats. For the local search part once a solution is selected among the current best solutions, a new solution for each bat is generated locally using a local random walk:

$$x_{new} = x_{old} + \epsilon A^t \tag{3.10}$$

where $\epsilon \in$[-1,1] is a random number. A^t is the average loudness of all the bats at that time step. Variation of Loudness and Pulse Emission:

$$A_i^{t+1} = \alpha A_i^t \tag{3.11}$$

$$r_i^{t+1} = r_i^0[1 - exp(-\gamma t)] \tag{3.12}$$

where α & γ are constant.

3.2.2.2 Grey Wolf Optimizer

Grey Wolf Optimizer (GWO) Algorithm inspired by grey wolves[27]. The GWO algorithm mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. Four types of grey wolves are employed for simulating the leadership hierarchy such as alpha, beta, delta, and omega. Also, the three main steps of hunting, searching for prey, encircling prey, and attacking victim, are implemented.

Alpha wolf is a leader male or female. They make decisions like the sleeping place, hunting, etc. Other wolves acknowledge alpha wolf by their tail down. Beta wolf help alpha wolf in making decisions. They are an advisor for alpha and discipliner for the pack. They also ensure all subordinate obey the order of alpha and give feedback to alpha. Delta wolves also called subordinate. They dominate omega wolves.

Categories of Delta wolves:-

- Scouts - Watch boundaries.
- Sentinels - Protect pack.
- Elders - Aplha or beta wolf sometimes.
- Hunters - Help aplha and beta wolf in hunting.
- Care Taker - Care ill weak and wounded wolves.

Omega wolves are like the scapegoat in the pack. They are last allowed wolves to eat having weak fitness.

Search Process in GWO Algorithm:-

- Searching
- Encircling
- Attacking the prey

Searching

- Search according to aplha, beta and delta wolves. They diverge to search for prey and converge to attack prey.
- Modeled by utilizing A random variable greater than 1 or less than -1.
- When $|A| > 1$ wolves are forced to diverge from prey to find better solution.

Encircling Prey It is the process in which wolves encircle the prey for attack. It is modeled as:

$$\overrightarrow{\mathrm{D}} = |\overrightarrow{\mathrm{C}}\,\overrightarrow{\mathrm{X}}_p(t) - \overrightarrow{\mathrm{X}}(t)| \tag{3.13}$$

$$\overrightarrow{\mathrm{X}}(t+1) = \overrightarrow{\mathrm{X}}_p(t) - \overrightarrow{\mathrm{A}}\,\overrightarrow{\mathrm{D}} \tag{3.14}$$

Where t is the current iteration, $\overrightarrow{\mathrm{A}}$, $\overrightarrow{\mathrm{C}}$ are coefficient vectors and $\overrightarrow{\mathrm{X}}_p$ is position of prey and $\overrightarrow{\mathrm{X}}$ is the position of grey wolf.

Updation of Coefficients

$$\overrightarrow{\mathrm{A}} = 2\overrightarrow{\mathrm{a}}\,\overrightarrow{\mathrm{r1}} - \overrightarrow{\mathrm{a}} \tag{3.15}$$

$$\overrightarrow{\mathrm{C}} = 2\overrightarrow{\mathrm{r2}} \tag{3.16}$$

Where $\overrightarrow{\mathrm{a}}$ linearly decrease from 2 to 0 over the course of iteration. $\overrightarrow{\mathrm{r1}}$ and $\overrightarrow{\mathrm{r2}}$ are random vectors $\in$ [0,1]. **Attacking** The hunt is usually guided by alpha. The beta and delta might also participate in hunting. We first save three best solutions i.e. alpha beta and delta which are the best solutions and update positions of other agents based on these three. It is modeled as:

$$\overrightarrow{\mathrm{D}}_\alpha = |\overrightarrow{\mathrm{C}}_1\,\overrightarrow{\mathrm{X}}_\alpha - \overrightarrow{\mathrm{X}}| \tag{3.17}$$

$$\overrightarrow{\mathrm{D}}_\beta = |\overrightarrow{\mathrm{C}}_2\,\overrightarrow{\mathrm{X}}_\beta - \overrightarrow{\mathrm{X}}| \tag{3.18}$$

$$\overrightarrow{\mathrm{D}}_\delta = |\overrightarrow{\mathrm{C}}_3\,\overrightarrow{\mathrm{X}}_\delta - \overrightarrow{\mathrm{X}}| \tag{3.19}$$

$$\overrightarrow{\mathrm{X}}_1 = \overrightarrow{\mathrm{X}}_\alpha - \overrightarrow{\mathrm{A}}_1\,\overrightarrow{\mathrm{D}}_\alpha \tag{3.20}$$

$$\overrightarrow{\mathrm{X}}_2 = \overrightarrow{\mathrm{X}}_\beta - \overrightarrow{\mathrm{A}}_2 \overrightarrow{\mathrm{D}}_\beta \tag{3.21}$$

$$\overrightarrow{\mathrm{X}}_3 = \overrightarrow{\mathrm{X}}_\delta - \overrightarrow{\mathrm{A}}_3 \overrightarrow{\mathrm{D}}_\delta \tag{3.22}$$

$$\overrightarrow{\mathrm{X}}(t+1) = (\overrightarrow{\mathrm{X}}_1 + \overrightarrow{\mathrm{X}}_2 + \overrightarrow{\mathrm{X}}_3)/3 \tag{3.23}$$

Grey wolf attack prey when it stops moving. In GWO vector A is a random value within an interval [-2a,2a] a decrease from 2 to 0 throughout the iteration. When $|A| < 1$ wolves attack prey.

3.2.3 Feature Extraction

For feature extraction Convolution Neural Network(CNN) is used.

3.2.3.1 Convolutional Neural Network

A convolutional neural network is a productive machine learning approach from deep learning, and it is close to standard Neural Networks. A convolutional neural network is a network with convolutional layers. A convolutional neural network is composed of three steps of neural layers to build its architectures: Convolutional, Pooling, and Fully-Connected.

Convolutional layers: The convolutional layer is an important piece of the convolutional neural network. In the convolutional layer, the output is obtained from the input by processing in certain conditions. Convolutional layers consist of a rectangular mesh or cubic chunk of neurons. It means input or output layers with filters can be a rectangular mesh or cubic chunk of neurons.
The filter is given from uppermost left to downside right. In every location of the filter, the weighted volume of the pixels is determined by the equation

$$WI^x + b \tag{3.24}$$

and a new neuron is acquired. In the convolutional layer, three sorts of hyperparameters decide the volume of the resultant neurons: depth, stride, and zero-padding. A number of the filters with certain strides controls the depth. For the original RGB image, depth is equal to three. Given an

image of size (5x5x3) and given ten filters of size (3x3x3). Here, whereas the stride equal to 1, the filters runs a single pixel. In this case, an output neuron is 3x3x10. Whereas the stride equal to 2, the filters are skipped two pixels, and output neuron size is 2x2x10.

Zero-padding is the process of filling zeros around the edge of the input neuron. Zero-padding is commonly used to adjust the size of input neuron during the filter process when we need to main the output neuron based on input neuron size.

$$Output = (W - F + 2P)/S + 1 \tag{3.25}$$

where W is the input neurons size, F is the filter (kernel) size, P is the size of the padding, and S is the stride. Linear algebraic operations are also used in the convolutional neural network. Suppose that matrix dimensions are a and b (a rows, b columns). 2D convolutional cube calculates the two-dimensional convolution with two input matrices. (M_A, N_A) is dimensions of the matrix A, and (M_B, N_B) is dimensions for the matrix B. In case, the cube determines the complete output size, convolution equation is as below

$$C(i,j) = \sum_{a=0}^{M_A-1} \sum_{b=0}^{N_A-1} A(a,b) * B(i-a, j-b) \tag{3.26}$$

Max-pooling layer: Pooling layer executes the next operation after each convolutional layer. These layers are used to reduce the size of the neurons. These are small rectangular grids that acquires small portion of convolutional layer and samples it to provide a single output from that block. The most commonly used method is max pooling that fetch that maximum pixel from the block. A formulation for a single type of the pooling layer, max-pooling is presented in equation

$$h_l^j(a,b) = max \in \mathcal{N}(a), b \in \mathcal{N}(b) h_j^{l-1}(\bar{x}, \bar{b}) \tag{3.27}$$

Fully Connected layers: The last layer of a convolutional neural network is fully connected layer that is formed from the connection of all previous neurons. It reduces the spatial information as it is fully connected like in artificial neural network. It contains neurons from all input neurons until all output neurons.

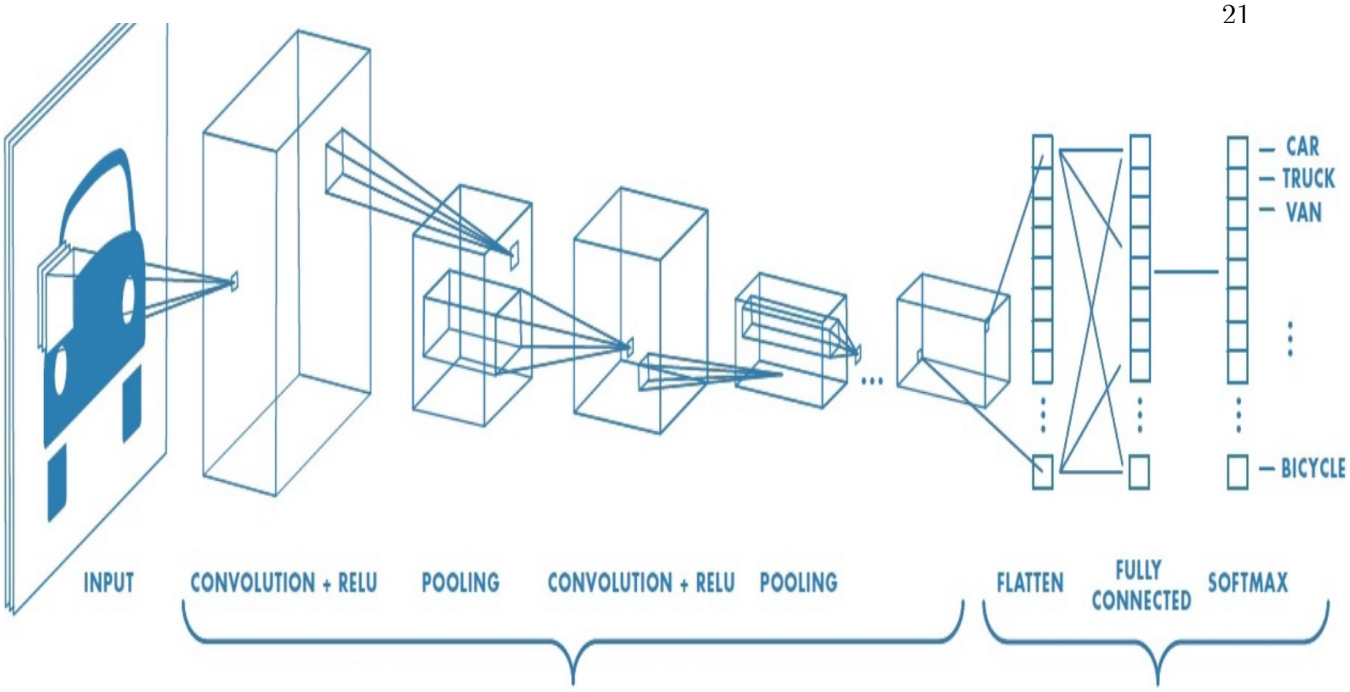

Figure 3.2: Convolution Neural Network Architecture

3.2.4 Algorithms

3.2.4.1 Recurrent Neural Network

Recurrent neural networks (RNN) are used to capture the temporal dependency in the time series data[5]. As most of the healthcare data is a series of temporal recordings, hence RNNs have been widely adopted in the healthcare domain. We are usually provided with a series of observations $x_1 ... x_T$ and we train a classifier to generate hypotheses $\hat{y}$. The recurrent connections are added in feed-forward neural networks to make it RNN. The output of a neuron in a typical NN is as follows:

$$y_i^t = \sigma(W_i x^t + b_i) \tag{3.28}$$

Where W_i is the weight matrix, b_i is the bias and represents the sigmoid function. While in the case of RNN, a neuron is fed with the output of the neuron at time t-1. The following equation shows the new activation function:

$$y_i^t = (W_i x^t + V_i x^{t-1} + b_i) \tag{3.29}$$

As RNN uses the previous outputs as recurrent connection, their current output depends upon the previous states. This property of RNN makes it very useful in sequence labeling tasks. The backpropagation through time can be used to train RNNs. It was demonstrated by that learning long-term dependencies is difficult using gradient descent. This is mainly because the backpropagating error can vanish which makes the network inefficient in learning long-range dependencies, or frequently explode which makes convergence impossible.

LSTM networks were designed to tackle the problem of vanishing gradients and were developed to model long-range dependencies efficiently. LSTMs can fix this by keeping an internal state that represents the memory cell of the LSTM neuron. This internal state can only be written and read through gates which decides the information flowing through the cell state. The following diagram shows the recurrent neural network.

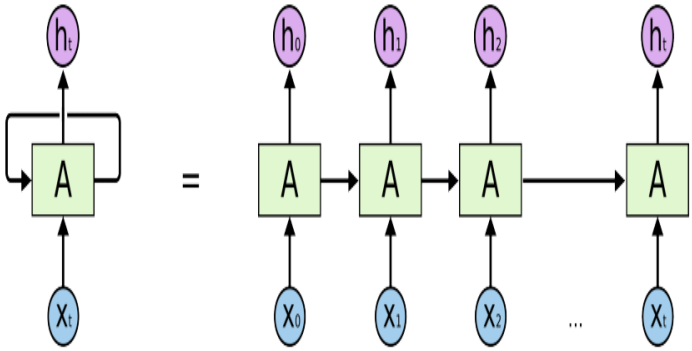

Figure 3.3: An unrolled recurrent neural network[17]

3.2.4.2 Long Short Term Memory (LSTM) Networks

Long Short Term Memory networks usually just known as LSTMs are a special type of RNN, that are good in learning long-term dependencies. They were introduced by [19] and were polished and popularized by many people. They work very well on a large variety of problems and are now used by many researchers. To solve long-term dependency problem, LSTMs are precisely formulated. Their default behaviour is to remembering something for a long period of time. All recurrent

neural networks contains a chain of repeating modules of the neural network. In traditional RNNs, this repeating module will have a pretty simple structure, such as a single tank layer. LSTMs also have this chain-like structure, but the recurrent module has a different structure. Rather than having a single neural network layer, there are four layers which are interacting extraordinarily. The intermediate information is stored in a single hidden layer h and its state changes over time (h_{t-1}, h_t, h_{t+1}). On the final hidden state vector h_T, we used a fully connected layer followed by sigmoid function. For loss function, we used log loss. The following equations can be used for calculation of the current hidden layer h_t.

$$f_t = \sigma(W_f X_t + R_f h_{t-1} + b_f) \tag{3.30}$$

$$i_t = \sigma(W_i X_t + R_i h_{t-1} + b_i) \tag{3.31}$$

$$o_t = \sigma(W_o X_t + R_o h_{t-1} + b_o) \tag{3.32}$$

$$\tilde{C}_t = \Phi(W_C X_t + R_C h_{t-1} + b_c) \tag{3.33}$$

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \tag{3.34}$$

$$h_t = o_t \phi(C_t) \tag{3.35}$$

f_t , i_t and o_t are forget gate, input gate and output gate respectively. To decide which historical information will be discarded from the cell state forget gates are used, the update of cell state is decided by the input gate, and the output gate decides the output of the cell state. Cell states are completely overridden in classical RNN, but LSTM has the potential to add or remove information to the cell state. The input weight of each gate, recurrent weight, and the bias are expressed as W* , R* , b* respectively where * can be f, i, o and c. Here σ, Φ stands for an element-wise application of the sigmoid (logistic) and tanh function respectively. For matrix multiplication. The

candidate values are computed in equation (6), and equation(7) old state is multiplied by f_t, and this helps in forgetting the things we decided to forget. Then i_t*δC_t is added in it. This is the new candidate values, scaled by how much we decided to update each state value. The final output of an LSTM unit is given by equation 8. Here * represents the Hadamard (element-wise) multiplication operation. Figure 3 shows the LSTM network.

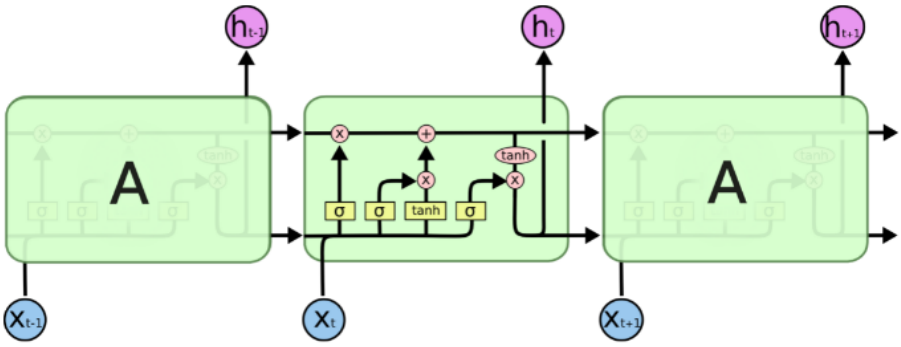

Figure 3.4: The repeating module in an LSTM contains four interacting layers[17]

3.2.4.3 Bidirectional Recurrent Neural Networks (BRNN)

Bidirectional Recurrent Neural Networks also called BRNN is just like RNN but it trains simultaneously on both sides of the time series data[31]. This model gives the better result in both regression and classification problem. BRNN computes both forward ($\overrightarrow{\mathrm{h}}$) and backward ($\overleftarrow{\mathrm{h}}$) hidden sequence.

$$\overrightarrow{\mathrm{h}}_t = \mathcal{H}(W_{x\overrightarrow{\mathrm{h}}}x_t + W_{\overrightarrow{\mathrm{h}}\overrightarrow{\mathrm{h}}}\overrightarrow{\mathrm{h}}_{t-1} + b_{\overrightarrow{\mathrm{h}}}) \tag{3.36}$$

$$\overleftarrow{\mathrm{h}}_t = \mathcal{H}(W_{x\overleftarrow{\mathrm{h}}}x_t + W_{\overleftarrow{\mathrm{h}}\overleftarrow{\mathrm{h}}}\overleftarrow{\mathrm{h}}_{t+1} + b_{\overleftarrow{\mathrm{h}}}) \tag{3.37}$$

$$y_t = W_{\overrightarrow{\mathrm{h}}y}\overrightarrow{\mathrm{h}}_t + W_{\overleftarrow{\mathrm{h}}y}\overleftarrow{\mathrm{h}}_t + b_o \tag{3.38}$$

The long range context can be accessed in both directions by combining BRNNs with LSTM which gives bidirectional LSTM[16].

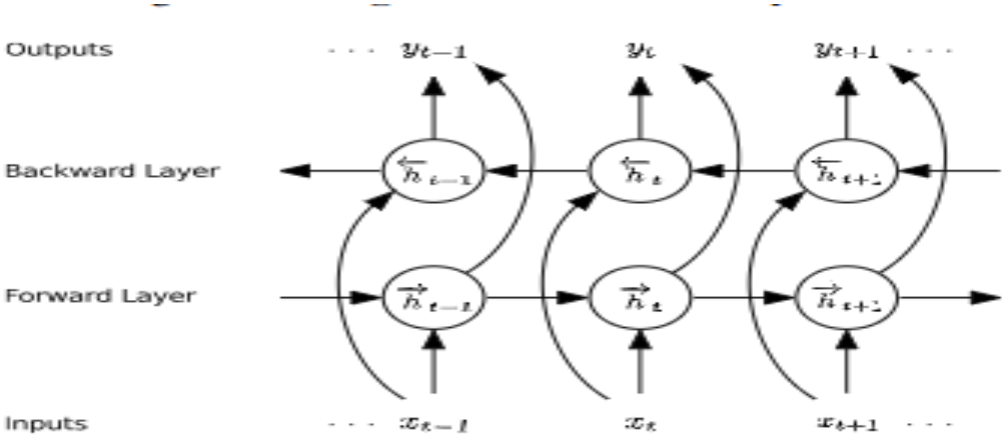

Figure 3.5: Bidirectional Recurrent Neural Networks[15]

3.2.4.4 Auto Encoders

An auto-encoder[26] is a neural network model that seeks to learn a flatten representation of the data. They are an unsupervised learning algorithm, although commonly, they are trained using supervised learning algorithms, mentioned as self-supervised. They are commonly trained as part of a deeper model that attempts to recreate the input. The design of the auto-encoder model purposefully makes this challenging by limiting the architecture to a bottleneck at the center of the model, from which the reconstruction of the input data is performed. In this case, once the algorithm is fit, the regeneration aspect of the model can be dropped, and the model up to the point of the bottleneck can be adopted. The result of the model at the congestion is a fixed length vector that produces a compressed representation of the input data. Input data from the region can then be supplied to the model, and the result of the model at the congestion can be used as a attribute vector in a supervised learning model, for visualization, or more generally for reducing dimensions.

3.3 Conclusion

The methodology and the algorithms used to compute, generate, test, and verify the process produces good results.

Chapter 4

Experiments and results

This section discusses the implementation details and the results obtained using varoius algorithms.

4.1 Implementation Details

The entire dataset was split using stratified sampling and 10% of the dataset was used as a validation set and random search was used for hyperparameter optimization. The rest 80% of the dataset was used for training purpose and was tested on 10% of the dataset. The LSTM model was trained with 30 epochs using Adam optimizer. LSTM layer uses 100 memory cell with no dropout and 25 memory cell with 20% dropout and these architectures are found after validation performance. Also train this model with Bat Algorithm.

Another model train on this dataset was convolution long short term memory(CLSTM). It is a hybrid model in which convolution layers are used for feature extraction. The CLSTM model was trained with 30 epochs and the batch size of 1000. First, the convolution layer was used with 45 filters of size 5×5. Then the LSTM layer that uses 100 memory cell with 20% dropout. Also train this model with Grey Wolf Optimizer Algorithm.

In the training phase, the model was saved and also compute these performance metrics (AUC, F1, precision, and recall).

Several built-in modules of python will be utilized, namely:

- Numpy

- Pandas
- Scikit-Learn
- Keras
- Matplotlib
- NiaPy

The computations were performed on Ubuntu 16.04 LTS with following hardware specifications:

- **Processor** Intel i7 6th Gen. broadwalle processor.
- **RAM** 24 GB DDR4
- **Secondary Memory** 250 GB SSD
- **Graphics Card** Nvidia 1080Ti 11GB

4.2 Results Obtained

This segment shows the readmission prediction results acquired from various predictive models. The prediction accuracy will be differentiate in terms of two variables AUC under ROC (Receiver Operating Characteristic) Curve[22], and final accuracy. Both area under ROC curve and final accuracy relies on the classifiers power to rank examples for positive class, however in the case of final accuracy, it also relies on the capability to calculate the threshold in the ranking to classify the positive class. Applying various modles on the dataset, with default parameters, we got results as referenced in Table 1.

Algorithm type	**Algorithm**	**AUC**	**Accuracy**	**F1 Score Micro**	**F1 Score Macro**
Deep Learning without Nature Inspired Algorithms	Long Short Term Memory	0.88	81.71%	0.80	0.76
	Long Short Term Memory Auto-Encoders	0.73	75.00%	0.70	0.62
	Convolution Long Short Term Memory	0.89	85.63%	0.71	0.66
	Convolution Long Short Term Memory Auto-Encoders	0.87	82.41%	0.80	0.78
	Bi-Directional Long Short Term Memory Auto-Encoders	0.85	83.42%	0.78	0.75
Deep Learning with Nature Inspired Algorithms	Long Short Term Memory with Bat Algorithm	0.75	77.58%	0.70	0.63
	Long Short Term Memory with Grey Wolf optimizer Algorithm	0.85	87.48%	0.79	0.76

Table 4.1: Comparison among different Models implemented so far

Methods	Area Under ROC Curve	Accuracy
Logistic Regression	0.54	58.09%
Support Vector Machines	0.54	59.37%
Decision Tree	0.52	55.68%
Random Forest	0.53	60.49%

Table 4.2: Comparison with Baseline Models

4.2.1 Curves showing Receiver Operating Curve (ROC) and Loss Curve of different models

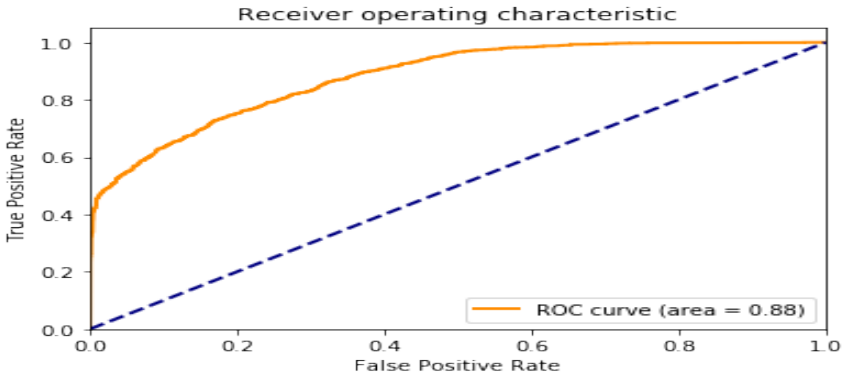

Figure 4.1: Receiver Operating Characteristic Curve: Long Short Term Memory.

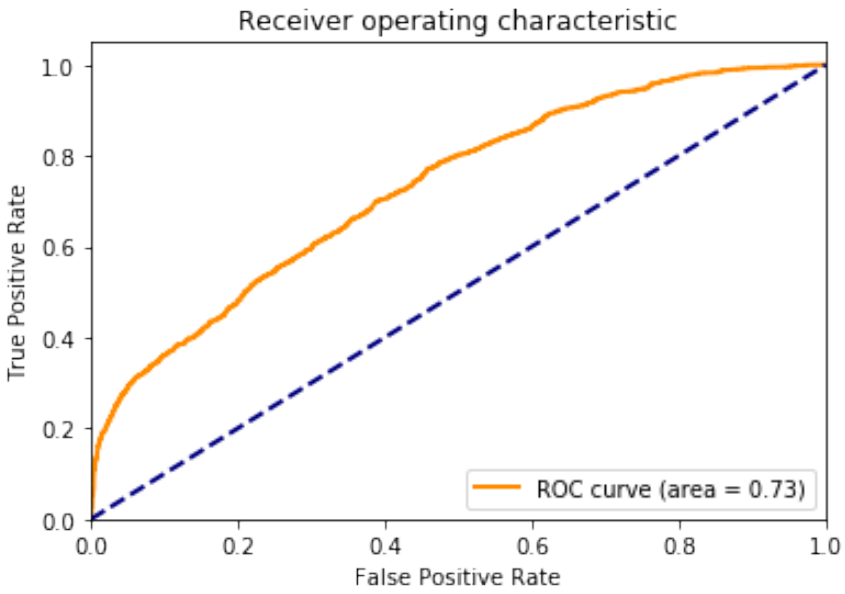

Figure 4.2: Receiver Operating Characteristic Curve: Long Short Term Memory Auto-Encoders.

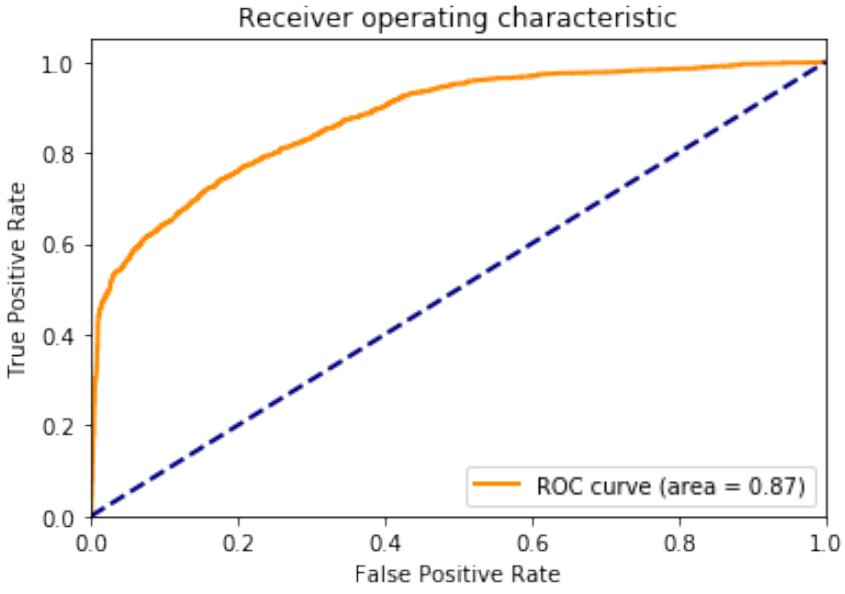

Figure 4.3: Receiver Operating Characteristic Curve: Convolution Long Short Term Memory.

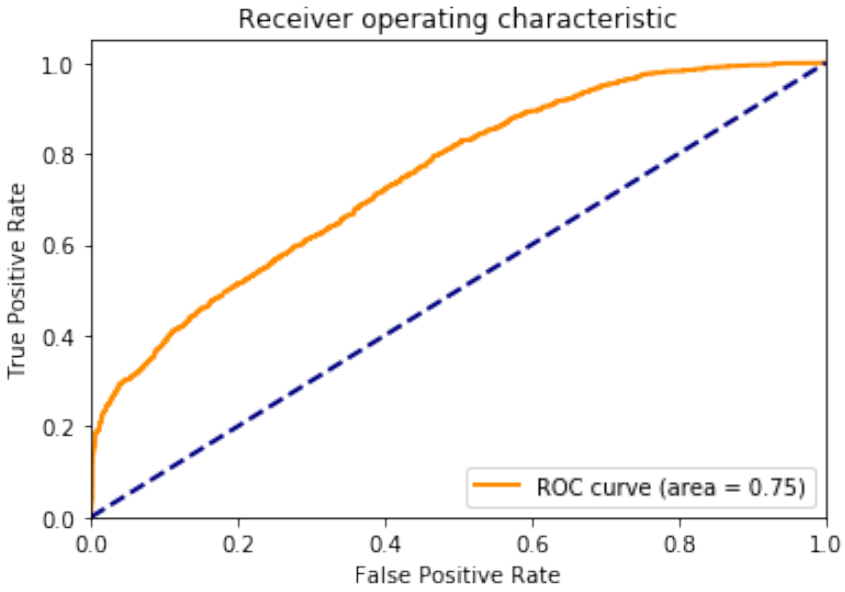

Figure 4.4: Receiver Operating Characteristic Curve: Long Short Term Memory with Bat Algorithm.

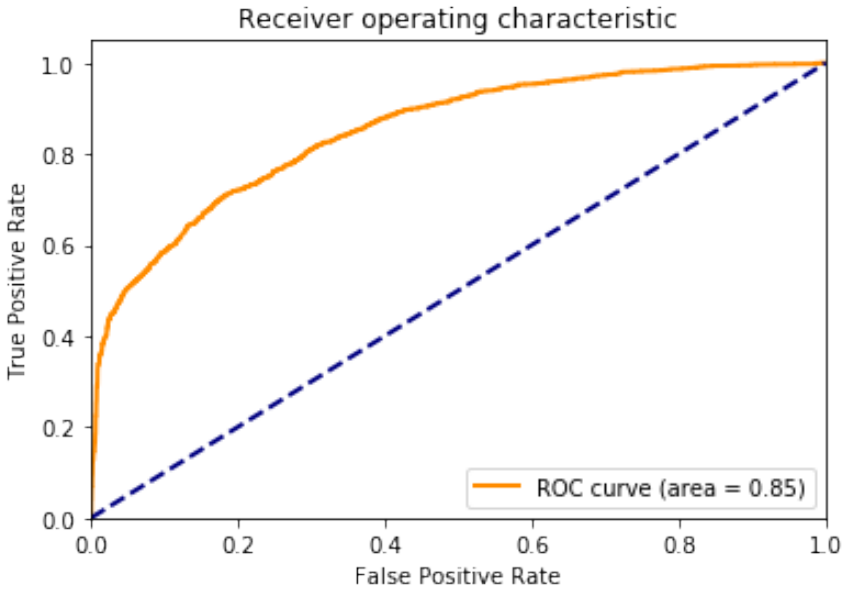

Figure 4.5: Receiver Operating Characteristic Curve: Long Short Term Memory with Grey Wolf Optimizer Algorithm.

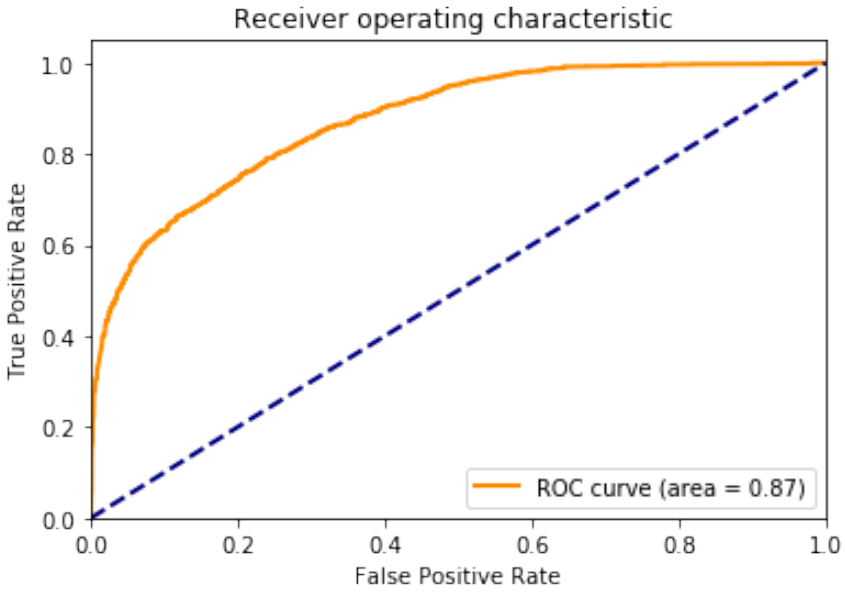

Figure 4.6: Receiver Operating Characteristic Curve Convolution Long Short Term Memory Auto Encoders.

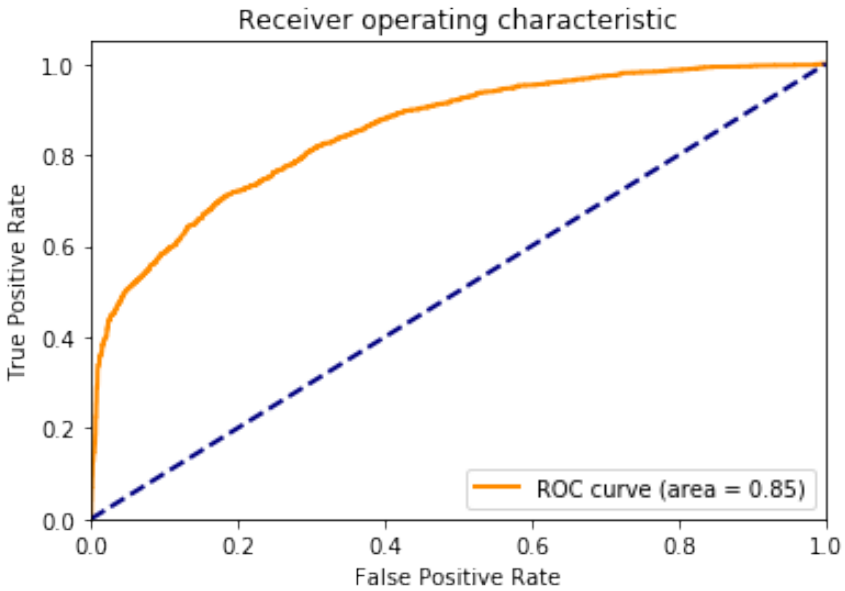

Figure 4.7: Receiver Operating Characteristic Curve Bi-Directional Long Short Term Memory.

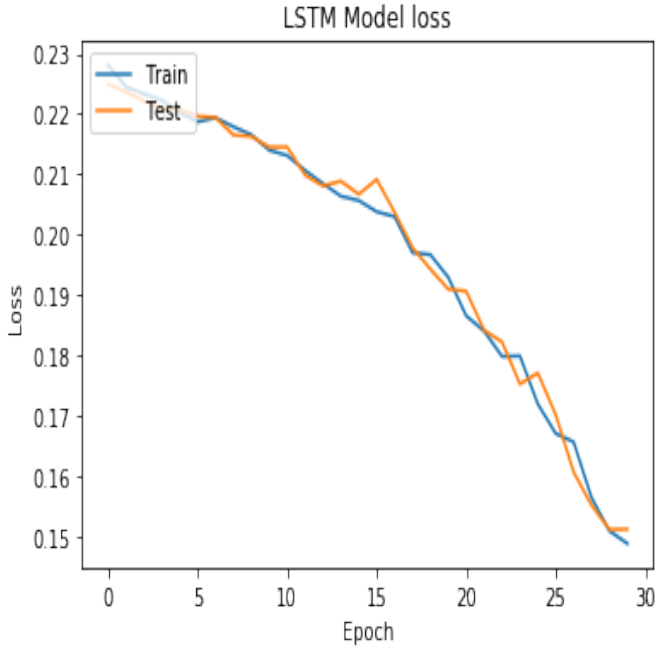

Figure 4.8: Loss Curve: Long Short Term Memory.

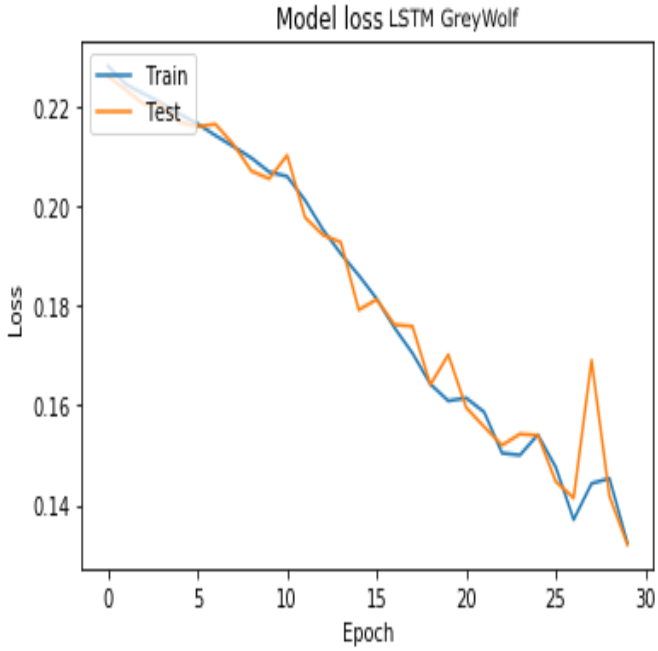

Figure 4.9: Loss Curve: Long Short Term Memory Grey Wolf.

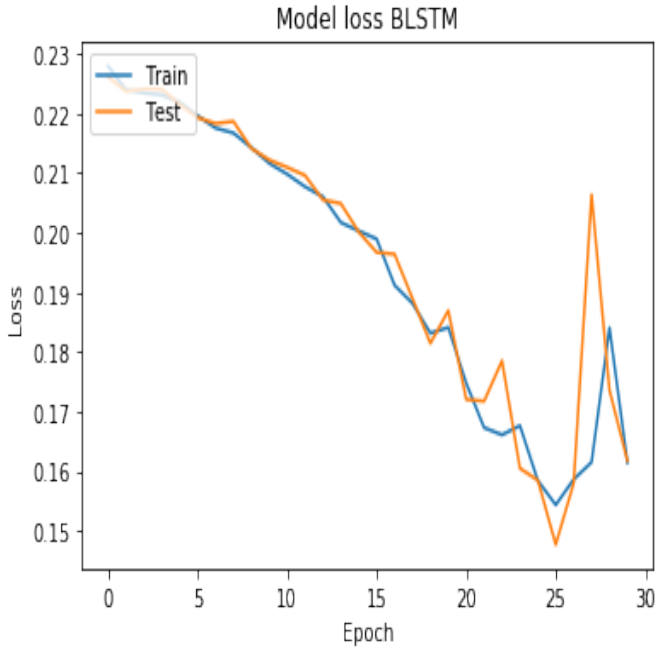

Figure 4.10: Loss Curve: Bidirectional Long Short Term Memory.

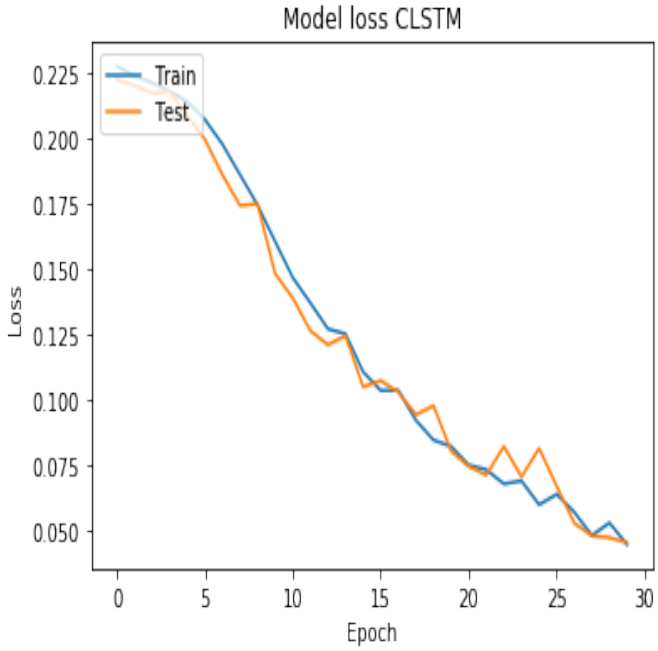

Figure 4.11: Loss Curve: Convolution Long Short Term Memory.

Chapter 5

Discussions and conclusion

5.1 Contributions

Following are the contributions made in the research work.

- To compare the performance of various deep learning models on MIMIC-III dataset based on accuracy and area under the roc curve.
- Compare the results of deep learning models with baseline models to point the advantages of deep learning in the field of health-care applications.
- Comparison of deep learning models with or without nature-inspired algorithms.

5.2 Limitations

The computation power and the memory required to process and compute such a big dataset, and the heavy amounts of output becomes one of the major limitations of the model. Since the data is already 35gb and as well as we use the LSTM model and hybrid model, i.e., CLSTM also with nature-inspired algorithms which store a lot of data while computation, therefore memory requirements becomes considerably large.

5.3 Future scope

Presently, this study only targets patients who were readmitted to the hospital for sure, whether within 30 days or after 30 days. This study can be extended, and one can classify the data

in three categories:

- Patients readmitted in 30 days.
- Patients readmitted after 30 days.
- Patients that were never readmitted.