# 2014_IPG-082-MTP-PPT-PLAG

*by* Shivam Sinha

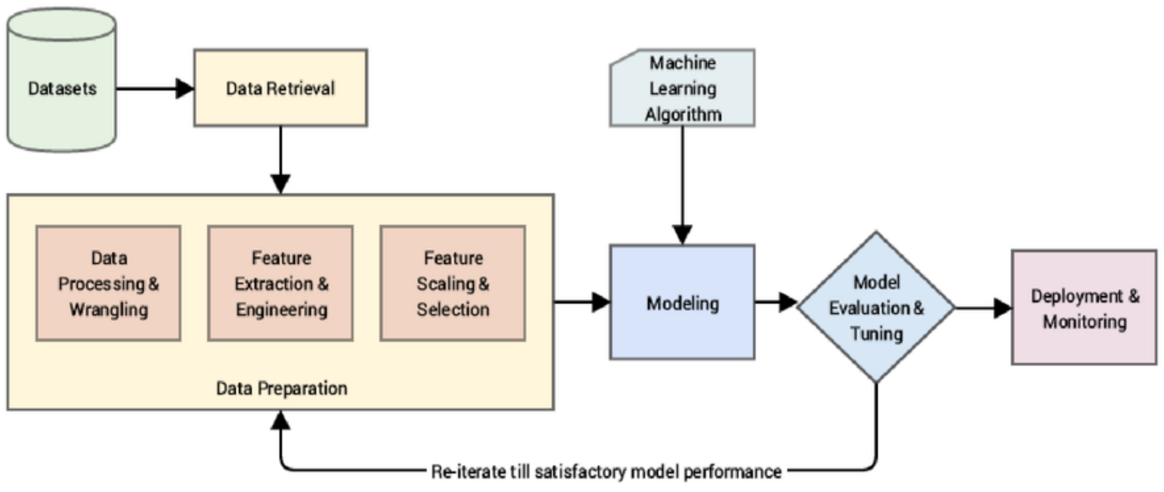
Datasets
Data Retrieval
Machine Learning Algorithm
Data Processing & Wrangling
Feature Extraction & Engineering
Feature Scaling & Selection
Data Preparation
Modeling
Model Evaluation & Tuning
Deployment & Monitoring
Re-iterate till satisfactory model performance

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 else goto step 2 and redo the procedure with different hyperparameters, algorithms, etc.

- These earlier research were unable 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 outclass the drawbacks of conventional machine learning algorithms.
- The thesis aims to use nature-inspired algorithms for feature selection.
- 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

Health Care Readmissions is extremely difficult task for both sufferer and health center. In our thesis, we design a robust algorithm to determine the amount of patients admitted again to a health care. We measured the performance of different deep learning algorithms 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 remaining ML algorithms in the prediction value. We also establish that the result of the combination of LSTM and Convolution layer was remarkable on this dataset. This architecture can be used in current's health system to aim at high possibility patients, decrease the degree of readmission, and provide excellent health care.

- 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.

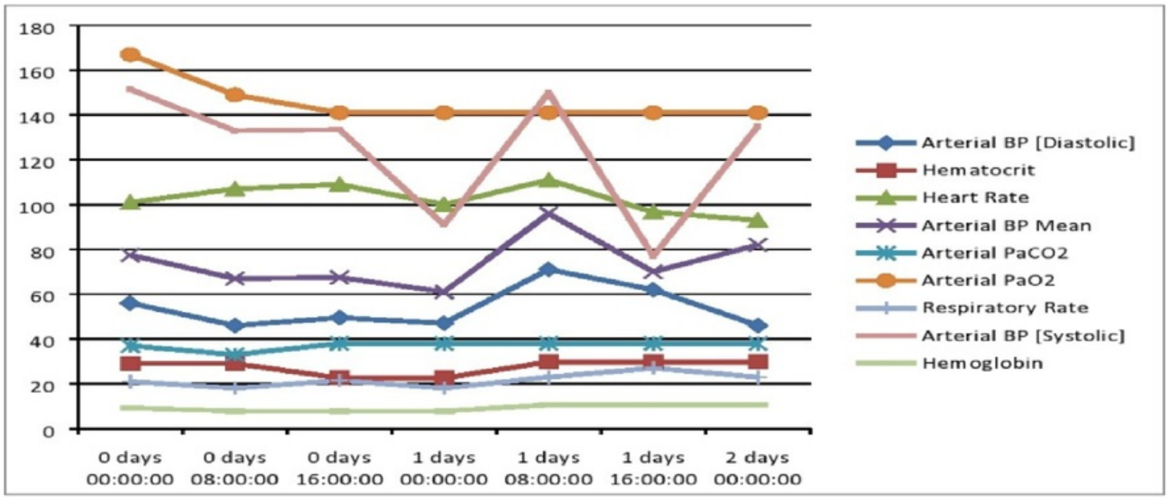

Figure 3.1: Some of the Clinical recordings associated with each patient[23]

## 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 confirmed as suffering from various kinds 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 admitted again in 30 days.
- Patients admitted again 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] scaling here.

- Min-Max Scaling

  Let a matrix "M":-

$$M = \begin{Vmatrix} m11 & m12 \\ m21 & m22 \end{Vmatrix} \tag{3.1}$$

  So for scaling we compute

$$min1 = min(m11, m21), \tag{3.2}$$

$$max1 = max(m11, m21) \tag{3.3}$$

$$min2 = min(m12, m22), \tag{3.4}$$

$$max2 = max(m12, m22) \tag{3.5}$$

  And "M" becomes:-

$$M = \begin{Vmatrix} (m11 - min1)/(max1 - min1) & (m12 - min2)/(max2 - min2) \\ (m21 - min1)/(max1 - min1) & (m22 - min2)/(max2 - min2) \end{Vmatrix} \tag{3.6}$$

### 3.2.2 Feature Selection

Features are selected through Nature Inspired Algorithms:

**Searching**

- Search according to aplha, beta and delta wolves. They separate to look for prey and come together to ambush on prey.
- Modeled by using $w$ random variable greater than 1 or less than -1.
- When $|w| > 1$ wolves are required to separate from prey to find better solution.

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

$$\vec{d} = |\vec{c}\,\vec{x}_p(t) - \vec{x}(t)| \tag{3.13}$$

$$\vec{x}(t+1) = \vec{x}_p(t) - \vec{w}\,\vec{d} \tag{3.14}$$

Where t is the current iteration, $\vec{w}$ , $\vec{c}$ are coefficient vectors and $\vec{x}_p$ is position of prey and $\vec{x}$ is the position of grey wolf.

**Updation of Coefficients**

$$\vec{w} = 2\vec{a}\,\vec{r1} - \vec{a} \tag{3.15}$$

$$\vec{c} = 2\vec{r2} \tag{3.16}$$

Where $\vec{a}$ linearly decrease from two to zero over the course of iteration. $\vec{r1}$ and $\vec{r2}$ are arbitarary vectors $\in$ [0,1]. **Attacking** The attack is mostly led by alpha. The beta and delta may also join in hunting. We first save three best solutions i.e. alpha, beta and delta which are the best answers and update the locations of other wolves based on these three. It is modeled as:

$$\vec{d}_\alpha = |\vec{c}_1\vec{x}_\alpha - \vec{x}| \tag{3.17}$$

$$\vec{d}_\beta = |\vec{c}_2\vec{x}_\beta - \vec{x}| \tag{3.18}$$

$$\vec{d}_\delta = |\vec{c}_3\vec{x}_\delta - \vec{x}| \tag{3.19}$$

$$\vec{x}_1 = \vec{x}_\alpha - \vec{w}_1\vec{d}_\alpha \tag{3.20}$$

$$\vec{x}_2 = \vec{x}_\beta - \vec{w}_2 \vec{d}_\beta \tag{3.21}$$

$$\vec{x}_3 = \vec{x}_\delta - \vec{w}_3 \vec{d}_\delta \tag{3.22}$$

$$\vec{x}(t+1) = (\vec{x}_1 + \vec{x}_2 + \vec{x}_3)/3 \tag{3.23}$$

Grey wolf attack it's target when it is immobile. In GWO vector $w$ is a arbitrary value within an range [-2a,2a] $a$ decrease from two to zero throughout the iteration. When $|w| < 1$ wolves ambush on victim.

### 3.2.3 Feature Extraction

#### 3.2.3.1 Convolutional Neural Network

A convolution neural network(CNN) is an efficient machine learning approach from deep learning and is close to standard multi layered networks. It consists of three layers:

- Convolution Layer
- Pooling Layer
- Fully Connected Layer

**Convolution Layer** This layer is the most important layer among all in CNN. It contains rectangular grids or cubic blocks called filters. The filter is moved at every block of input neuron to produce output. These filters are trainable. Output is determined by equation:

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

There are three hyperparametrs in convolutionlayer that decides the output:

- Depth
- Stride
- Zero Padding

The number of filters used in convolution layer with certain stride is the depth of the output neuron. For Example say, the RGB image, with a depth of three. Given an image of size (5x5x3) and given ten filters of size (3x3x3). Here, when the value of stride is given 1, the filters hovers a single pixel over an image then the output neuron is of size (3x3x10). Whilst when the value of stride is given 2, the filters will skip two pixels, and output neuron is of size (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 the input neuronś size is $W$, filterś (kernel) size is F,paddinǵ size is P, and stride is S. Linear algebra are also used in the convolutional neural network. Now consider that given a matrix of dimension $AxB$ (A is the number of rows and B is the number of columns). Now two dimensional convolution operation with two matrices $(A_M, B_M)$ is a dimension of matrix M, and $(A_N, B_N)$ is a dimension of matrix N. Convolution of these matrix is given by below equation:

$$P(i,j) = \sum_{a=0}^{A_M-1} \sum_{b=0}^{B_M-1} M(a,b) * N(i-a, j-b) \tag{3.26}$$

**Max-pooling layer:** Pooling layer executes the next operation after each convolution layer. These 13 layers are used to reduce the size of the neurons. These are small rectangular grids that acquires small portion of convolutional layer and filters it to give a result from that block. The most commonly used layer is max pooling that fetch that maximum pixel from the block. A expression for a max-pooling layer is expressed in a equation:

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

17 **Fully Connected layers:** The final layer of a 12 convolutional neural network(CNN) is a fully connected layer that is formed from the attachment of all preceding neurons. It reduces the spatial

information as it is fully connected like in artificial neural network. It contains neurons beginning at input neurons to the output neurons.

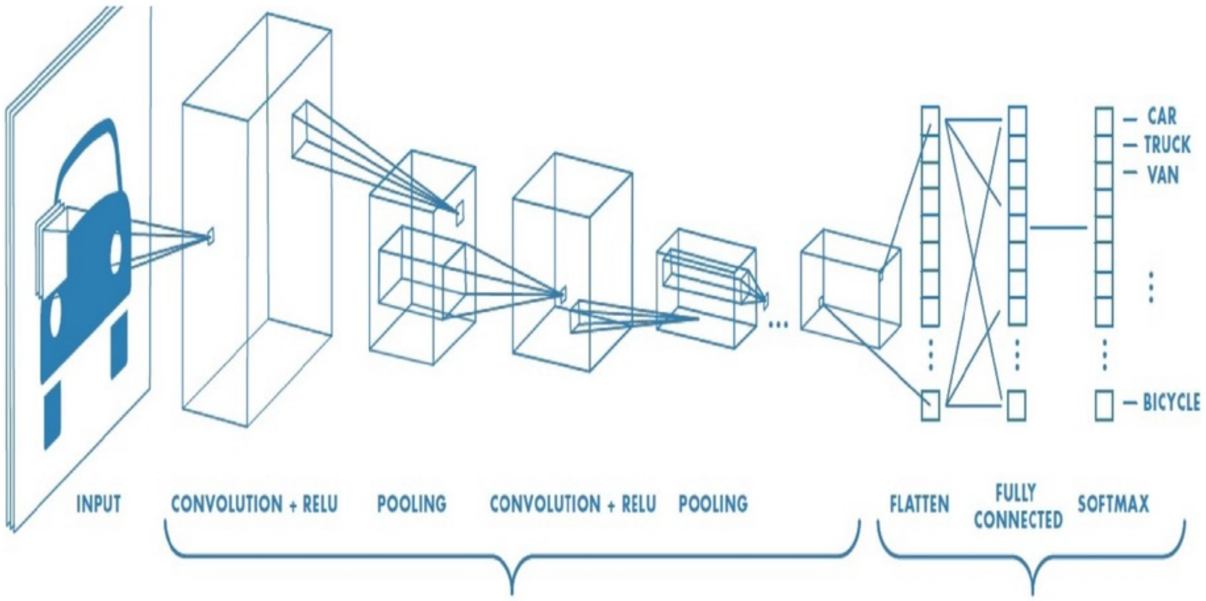

Figure 3.2: Convolution Neural Network Architecture

### 3.2.4 Algorithms

#### 3.2.4.1 Recurrent Neural Network

Recurrent neural networks (RNN) are utilized to represent the time 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 the weight matrix is represented as $W_i$, the bias represented as $b_i$ and $\sigma$ 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 models were designed to tackle the issue of vanishing gradients and were developed to model long-range dependencies efficiently. LSTMs can fix this by maintaining an internal state that represents the memory cell of the LSTM neuron. This internal state can only be written and read via gates which decides what information will through gates. The following diagram shows the recurrent neural network.

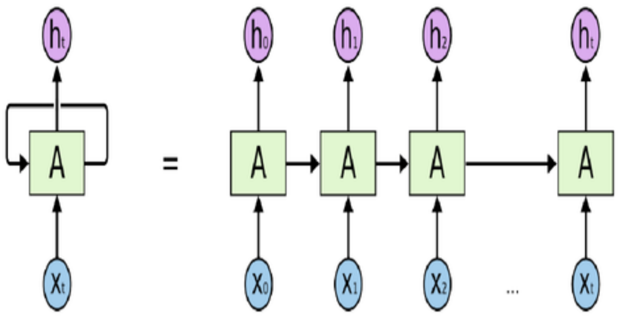

Figure 3.3: An unrolled RNN[17]

#### 3.2.4.2 Long Short Term Memory(LSTM) Networks

Long Short Term Memory networks are commonly called LSTMs. They are a particular kind of RNN, that are good in understanding long-term dependencies. They were introduced by [19] and were polished and familiarized by many people. They work very effectively on a large diversity 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 duration of time. All recurrent neural networks contains a series of replicating chunks of the neural network. In traditional RNNs, this replicating chunk have a straightforward architecture, such as a single tank layer. LSTMs also have this series-like architecture, but the recurrent chunk has a disimilar architecture. Rather than having a single layer, there are four layers which are connecting extraordinarily. The intermediate information is stored in a single hidden layer l and its state changes over time ($l_{t-1}$, $l_t$, $l_{t+1}$). On the final hidden state vector $l_T$, 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 $l_t$.

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

$$u_t = \sigma(W_u X_t + R_u l_{t-1} + b_u) \tag{3.31}$$

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

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

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

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

$u_t$, $o_t$ and $f_t$ are input gate, output gate and forget gate respectively. Forget gate decides which information to discard, input gates are used to update cell and output gates are used to decide the output of the cell state. Cell states are completely overridden in classical RNN, but LSTM has the potential to eliminate or append data 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, u, o and c. Here $\sigma$, $\Phi$ stands for an element wise operation of the sigmoid (logistic) and tanh function respectively. For matrix multiplication. The candidate values are computed in equation(3.33), and equation(3.34) old state is multiplied by $f_t$, and this helps in neglecting the things we decided to neglect. Then we add $u_t$*$\delta C_t$ in it. This is the new candidate values, increased by how much we planned to update each state value. The final output of an LSTM unit is given by equation(3.35). Here * represents the Hadamard (element-wise) multiplication process. Figure 3 shows the LSTM network.

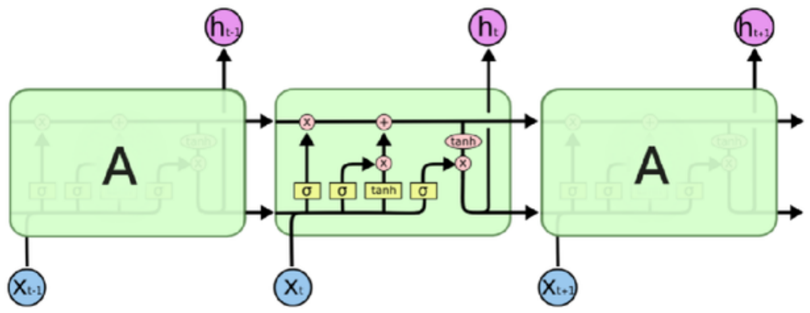

Figure 3.4: The replicating module in an LSTM[17]

#### 3.2.4.3 Bidirectional Recurrent Neural Networks (BRNN)

Bidirectional Recurrent Neural Networks also known as BRNN is just like RNN but it trains simultaneously on both directions of the time dependent data[31]. This algorithm gives the better result in both regression and classification problem. BRNN computes both forward ($\overrightarrow{1}$) and backward ($\overleftarrow{1}$) hidden sequence.

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

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

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

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

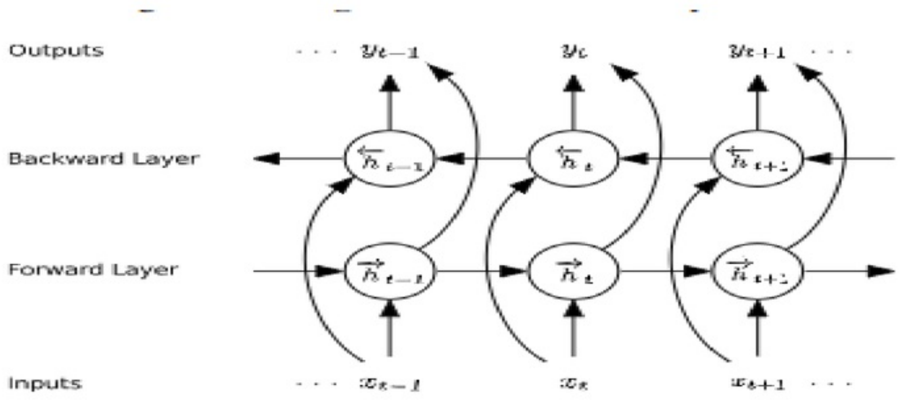

Figure 3.5: Bidirectional Recurrent Neural Networks[15]

#### 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

## 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 LSTM (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 bunch size of 1000. First, the convolution layer was used with 45 filters of size $5 \times 5$. 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

| **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 | 0.87 | 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

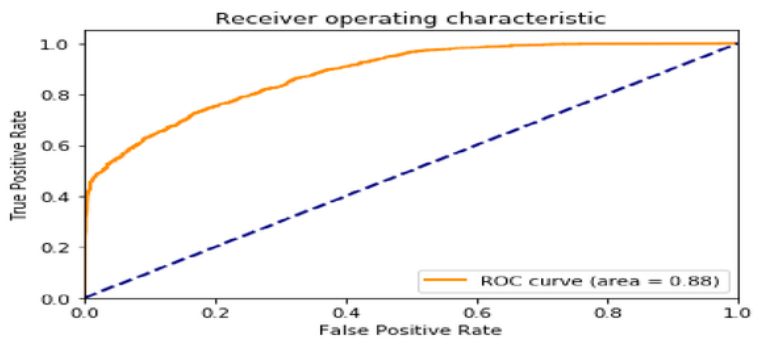

Figure 4.1: ROC Curve: LSTM.

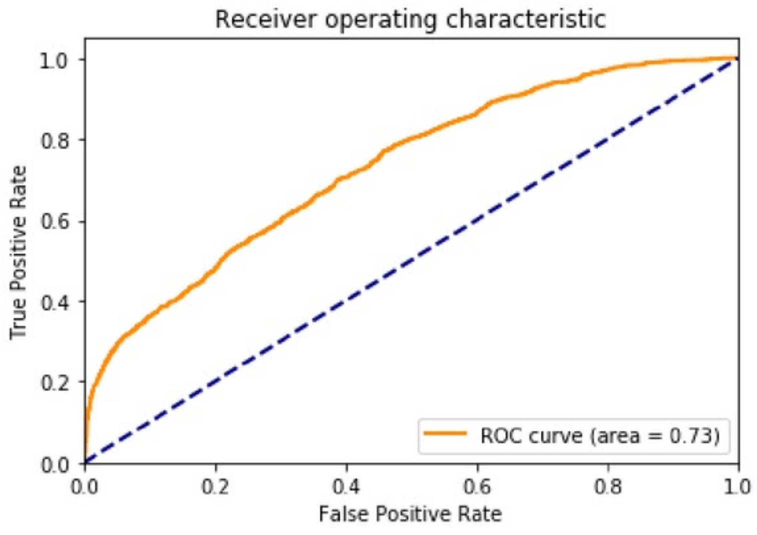

Figure 4.2: ROC Curve: LSTM Auto-Encoders.

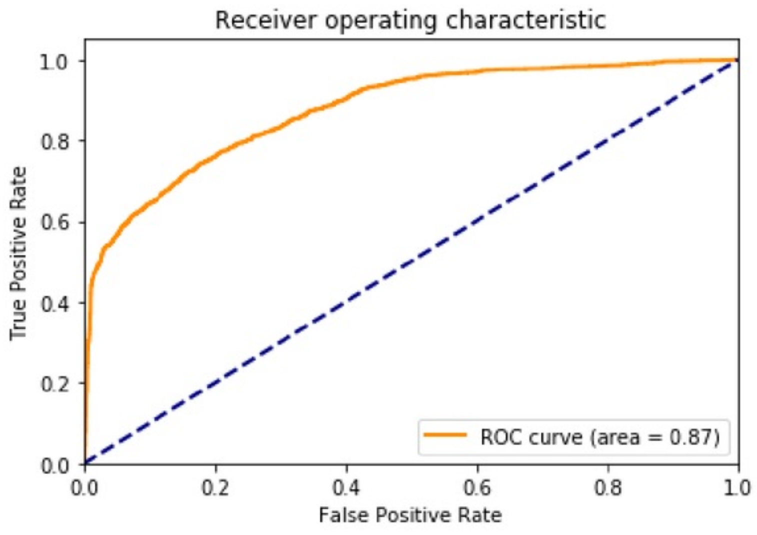

Figure 4.3: ROC Curve: CLSTM.

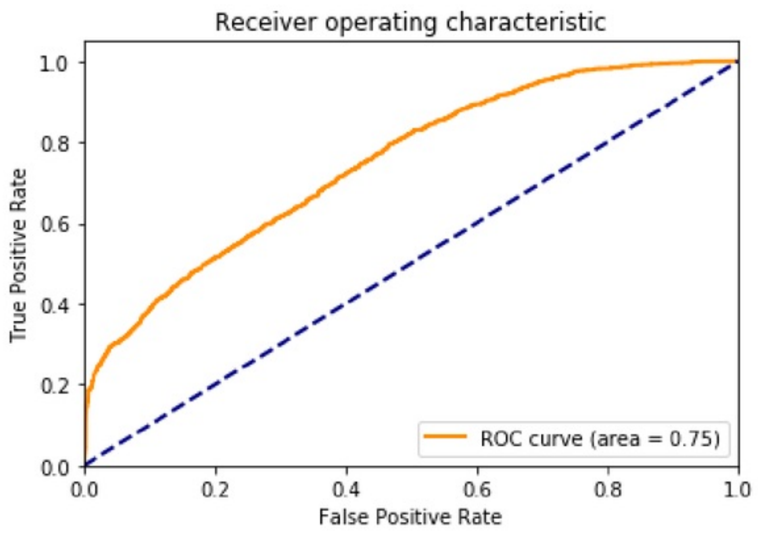

Figure 4.4: ROC Curve: LSTM with Bat Algorithm.

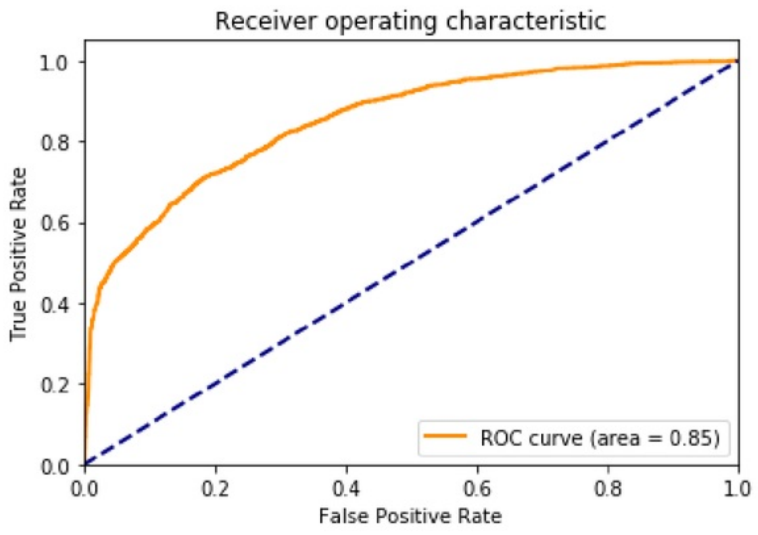

Figure 4.5: ROC Curve: LSTM with Grey Wolf Optimizer Algorithm.

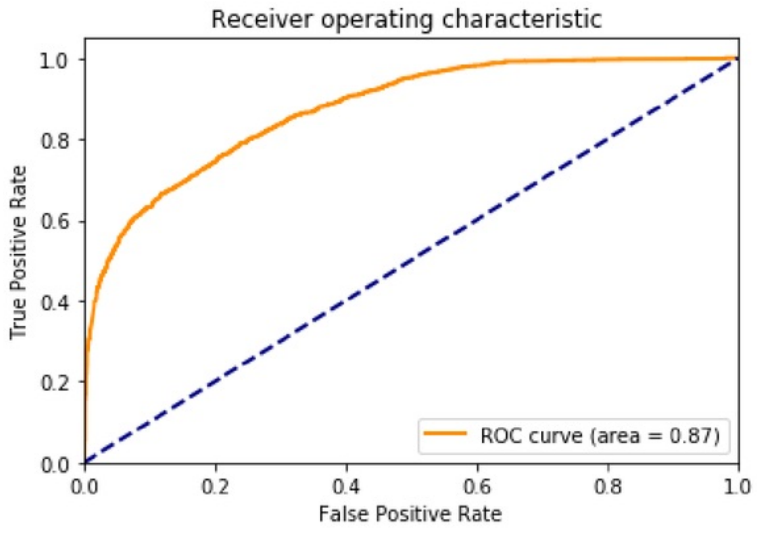

Figure 4.6: ROC Curve CLSTM Auto Encoders.

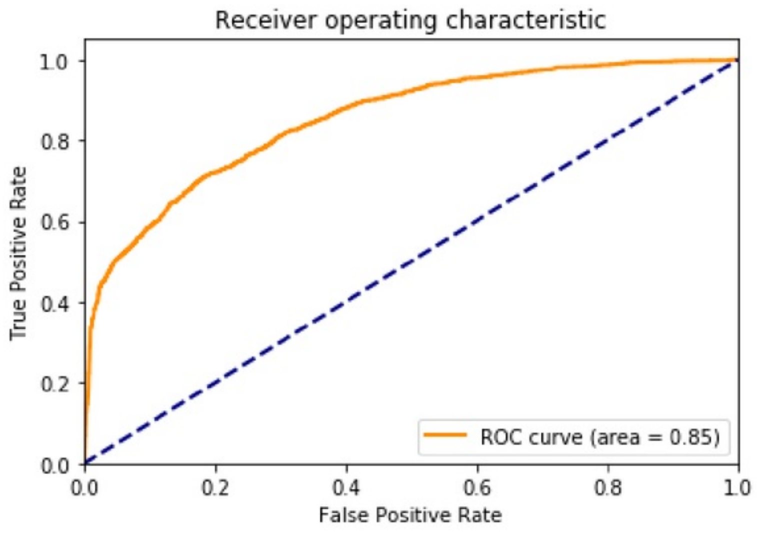

Figure 4.7: ROC Curve Bi-Directional LSTM.

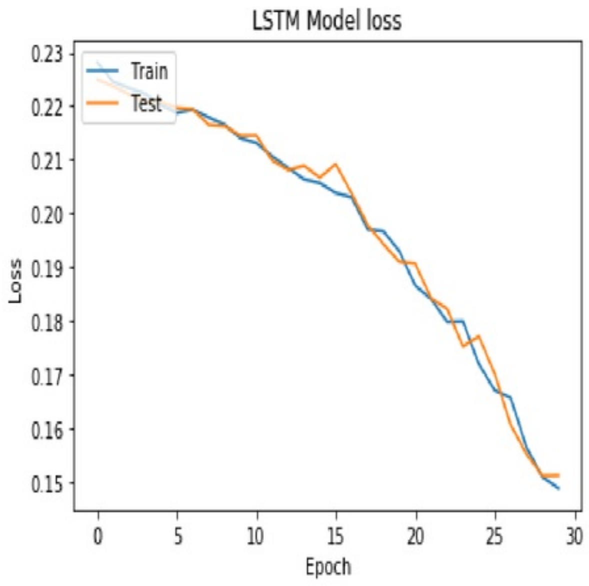

Figure 4.8: Loss Curve:LSTM.

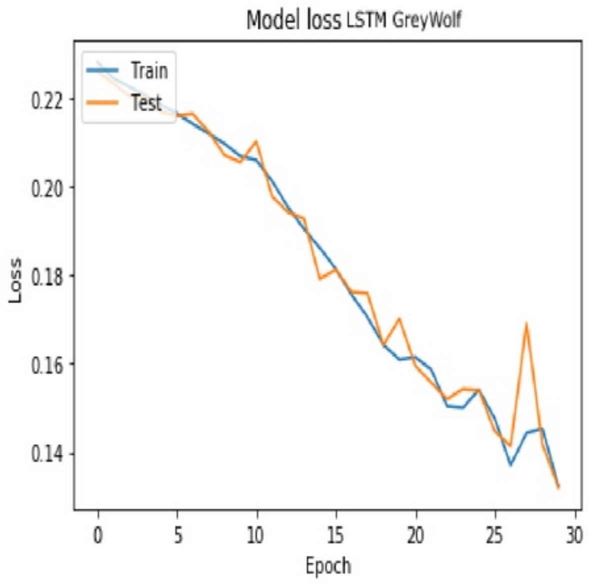

Figure 4.9: Loss Curve: LSTM Grey Wolf.

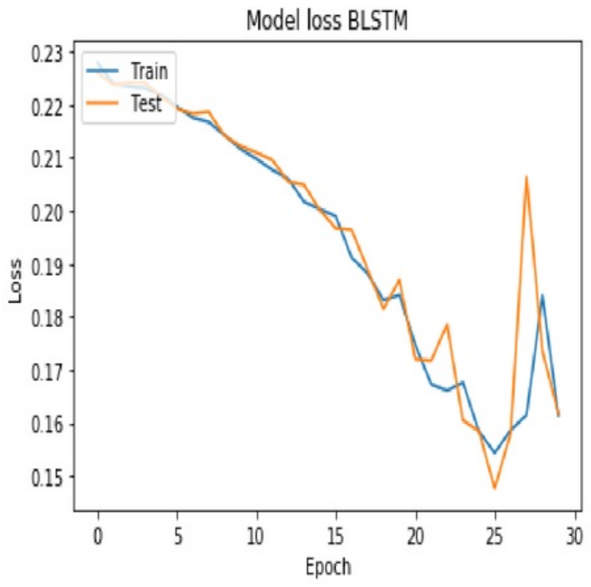

Figure 4.10: Loss Curve: Bidirectional LSTM.

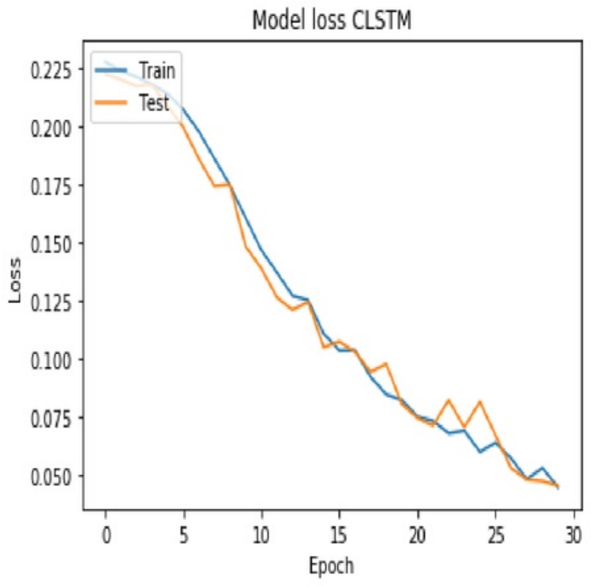

Figure 4.11: Loss Curve: Convolution LSTM.

# 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 algorithms on MIMIC-III dataset based on area under the roc curve, accuracy anf F1Scores.
- 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 admitted again 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 admitted again in 30 days.
- Patients admitted again after 30 days.
- Patients that were never readmitted.

2014_IPG-082-MTP-PPT-PLAG

ORIGINALITY REPORT

| 7% | 2% | 3% | 7% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | Submitted to Punjab Technical University<br>Student Paper | 2% |
| 2 | Phan Quoc Dzung, Nguyen The Tien, Nguyen Dinh Tuyen, Hong-Hee Lee. "Selective harmonic elimination for cascaded multilevel inverters using Grey Wolf Optimizer algorithm", 2015 9th International Conference on Power Electronics and ECCE Asia (ICPE-ECCE Asia), 2015<br>Publication | 1% |
| 3 | Submitted to University College London<br>Student Paper | 1% |
| 4 | www.nature.com<br>Internet Source | 1% |
| 5 | Submitted to Monash University<br>Student Paper | <1% |
| 6 | Submitted to The University of Manchester<br>Student Paper | <1% |
| 7 | Shankho Subhra Pal. "Grey Wolf Optimization | |

| | | |
|---|---|---|
| | Trained Feed Foreword Neural Network for Breast Cancer Classification", International Journal of Applied Industrial Engineering, 2018<br>Publication | <1% |
| 8 | Submitted to University of Warwick<br>Student Paper | <1% |
| 9 | Submitted to Rochester Institute of Technology<br>Student Paper | <1% |
| 10 | Submitted to King's College<br>Student Paper | <1% |
| 11 | Submitted to Universiti Teknologi MARA<br>Student Paper | <1% |
| 12 | Submitted to City University<br>Student Paper | <1% |
| 13 | www.analyticsvidhya.com<br>Internet Source | <1% |
| 14 | Submitted to Universiti Sains Malaysia<br>Student Paper | <1% |
| 15 | Submitted to Kuwait University<br>Student Paper | <1% |
| 16 | Submitted to University of Melbourne<br>Student Paper | <1% |
| 17 | Submitted to IIT Delhi<br>Student Paper | <1% |