# TITLE Optimizing Neural Networks for medical data sets: A case study on Neonatal Apnea prediction.

# ABSTRACT

**Objective:** The neonatal period of a child is considered most crucial phase of its physical development and future health. As per the WHO, India has the greatest number of pre-term births [1], with over 3.5 million babies born prematurely, and up to 40% of them are babies with low birth weights, highly prone to a multitude of diseases such as Jaundice, sepsis, apnea etc. Apnea is the primary concern for caretakers of neonates in intensive care units. The real-time medical data is known to be noisy and nonlinear in nature and to address the resultant complexity in classification and prediction of diseases; there is a need for optimizing learning models to maximize predictive performance. The study attempts to build optimized neural network architectures to predict the persistence of apneic episodes after the first week of admission at Neonatal Intensive Care Units (NICU).

**Method:** The data being inherently complex and noisy, Kernel PCA is used to reduce dataset dimensionality for the analysis such as interpretations and visualization of the dataset. Hyper-parametric and parametric optimization, in different categories are considered, including learning rate updater algorithms, regularization methods, and activation functions, etc. based on their performance on the validation set, to obtain a holistically optimized neural network, that best model the given complex medical dataset. Deep Neural Network Architectures such as Deep Multilayer Perceptron's, Stacked Auto-encoders and Deep Belief Networks are employed to model the dataset, and their performance is then compared to the optimized neural network obtained from the parametric exploration. ~~The models were evaluated with K -Fold Cross Validation using (AUROC) Area Under the Receiver Operating Characteristic curve as the performance evaluating criteria.~~ Further, the results are also compared with Support Vector Machine (SVM), Decision Tree (DT) and Random Forest algorithms.

**Result:** The results indicates that optimized Multilayer perceptron with six hidden layers (AUC 82 %) and Deep Auto-encoders (AUC 83 %) are the best predictors of neonatal apnea over the shallow models.

**Conclusion:** Hyperparametric optimized Multilayer perceptron is proved to be as efficient as Deep neural network models such as Deep Belief Networks and Deep Auto-encoders for noisy and nonlinear data set. The investigated models can help neonatologists as a diagnostic tool.

# INTRODUCTION

Machine Learning over the last decade, has come leaps and bounds, with various advancements in solving numerous problems once considered intuitive, and hence inherently difficult for computers to solve such as Face Recognition [2], Disease prediction [3], Speech Recognition [4], Anomaly detection [5], along with a myriad of others. As this field, has matured, new opportunities have emerged for various applications in the Medical domain. Neural networks have proved to be highly powerful models in this domain, being used to solve various classification and regression problems, especially medical diagnosis. Artificial Neural Network (ANN) a black box method and a versatile learner, can be applied to nearly any learning tasks such as classification and numeric prediction. Neural networks are a form of supervised learners trained by back-propagating the error between the required output and that of the network output, to the layers between the input and the output layers, called the hidden layers. Neural network with multiple layer is called a universal function approximator, as it can learn a diverse set of functions without explicit structure. It also handles complex interactions as well as highly nonlinear relationships between input and output. The last decade has seen exponential growth in the field of Machine Learning and Artificial Intelligence, primarily due to the introduction of Deep Neural Networks [6] that have drastically improved the ability of machines to solve problems which rely on intuitive decisions learnt from experience rather than pre-determined rules. The shallow models such as Support vector machines (SVM), Decision Trees (DT) are incapable of capturing underlying concepts and corresponding relationships [7]. The basic requirement of shallow models is that they require well designed feature representation. Therefore, a model with automatic feature learning and classification or regression will be an effective solution. Deep architectures can simulate and express complex problems at a different concept levels that are difficult to be expressed in a problem domain. Multilayer neural network architecture is capable of learning the true underlying features and feature logic and therefore generalize very well, but weight learning algorithms did not work on multilayer architecture leading to a way Deep learning.

Deep Learning is a subfield of machine learning which attempts to learn high level abstractions in data by utilizing hierarchical architecture [8] [9]. The concept used for deep learning is to train the layers in sequence where each of the non-output layer is trained to be an auto encoder with weight adjustment algorithm. Basically, it is forced to learn good features that describes what comes from the previous layers. Lastly, final layers are trained to predict class based on output from previous layers.

 The basic neural networks are unable to handle selectivity – invariance dilemma and also suffers from vanishing gradients which are addressed by deep learning [10]. Deep learning techniques by reinforcing the discriminative power of simple neural networks helps in solving selectivity – invariance dilemma [11]. Further they work by automatically extracting essential features from raw data and makes them robust with respect to variations in input. Higher classification accuracy even with lesser data, better accuracy with many hidden layers by using nonlinearity, optimally weighted and lower dimensional production mechanism are the merits of deep learning over shallow networks [12]. For Deep networks, back propagation algorithm has found to be unreliable due to over fitting and random network parameter initialization. Therefore, an appropriate learning strategy is required to handle the above-mentioned restrictions. Learning methods in Deep networks uses unsupervised as well as supervised approach. In medical science where we do not have sufficiently labeled data sets, unsupervised approach is more applicable. The real time medical data is known to be noisy and nonlinear in nature and therefore there is a need to optimize learning models to maximize predictive performance based on hypermetric optimization.

Deep Neural Network Architectures such as Deep Multilayer Perceptron’s, Stacked Auto-encoders and Deep Belief Networks were employed to model the dataset and their performance was compared to the optimized Multi-Layer Perceptron (MLP) obtained from our parametric exploration. The models were evaluated with K (10) - Fold Cross Validation using (AUROC) Area Under the Receiver Operating Characteristic curve as the performance evaluating criteria. Further, the results were also compared with Support Vector machine (SVM), Decision Tree (DT) and Random Forest algorithms.

Among the neonatal diseases, Apnea is predominant in premature neonates. Apnea is a breathing disorder for more than 10 to 15 seconds often associated with bradycardia, cyanosis or both [1]. Apnea of prematurity (AOP) is the major concern for caretakers of neonates in intensive care units. In premature babies, apnea can result in failure of cerebral blood flow, causing ischemia and eventually even leukomalacia.  Further, as per the National Institutes of Child Health and Human Development (NICHD), crucial conditions that are responsible for the occurrence of AOP are poorly understood and aren’t integrated into the care facilities [13].

Prediction of apnea episodes, at the earliest, is a challenging task in clinical practice. Proposed work deals with use of optimized neural network algorithms to predict the presence of persistence of apneic episodes after the first week of admission at NICU.

The remainder of the paper is organized as follows. Section 2 describes the background study dealing with Deep learning approaches and Neural network architectures and applications used in medical domain. Section 3 focuses on methodology adopted to build deep networks. Section 4 presents thorough experimental results and discussion and section 5 concludes the research work.

# BACKGROUND

## Deep Learning approaches

A neural network is composed of the following elements such as learning process, set of neurons or weights and connectivity functions. This set of elements enables us to build a broad range of neural networks, ranging from directed acyclic graph such as Multi – Layer Perceptron (MLP) with creative alternatives of Deep networks to specialized Deep learning methods such as Restricted Boltzmann Machine, Stacked Denoising etc. Learning methods in Deep networks uses unsupervised as well as supervised approach. Supervised approach requires class labels to modify weights and also a mechanism to communicate error at lower layers. Multilayer perceptron does not get trained properly due to diffusion of gradient and slow training. Unsupervised training between layers can decompose the problem into distributed sub problems with higher level of abstraction to be further decomposed at subsequent layers. In medical diagnosis, where we do not have sufficiently labeled data sets, unsupervised approach is more applicable. Table 1 describes the different deep learning models with learning approach and architecture. Most of the deep learning models uses greedy layer wise training approach with unsupervised learning followed by supervised learning approach.

Table 1 Overview of Deep learning approaches [10] [11] [12]

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Characteristics** | **Advantages / Disadvantages** | **Architecture / Learning** |
| Restricted Boltzmann Machines (RBMs) | * No lateral connections between hidden (h) and visible (x) nodes. * Symmetric weights * Not seeking global minima, but rather an incremental transformation of feature space. * Uses probabilistic logistic node. * Works with small training sets * Theoretical justification possible | * Hidden and visible layers are conditionally independent allowing sample of one layer using activation of other. * Formation of bipartite graph allows training using gradient based contrastive divergence algorithm | * Deep Belief Networks * Deep Boltzmann machines * Deep energy models |
| Deep Belief Networks (DBN) | * Constructed using multiple stacked RBMs. * Greedy layer wise training with each layer being RBM. * Directed connections at lower layers and undirected connections at top two layers. | * Type: generative * Initialization of networks weights preventing poor local optima. * Unsupervised training which removes necessity of labelled data. * Computationally expensive due to initialization process. | * Fully connected architecture with multiple hidden layers and partial bidirectional connections. * Unsupervised training (pre / fine tuning) |
| Deep Boltzmann machines (DBM) | * Multiple hidden layers, fully connected architecture with bidirectional connections | * Type: generative * Ambiguous inputs makes it robust by incorporating top down feedback. | * Pretraining: unsupervised with large amount of unlabeled data * Fine tuning: supervised |
| Deep Auto-encoder | * Learning efficient encodings with ANN. * Trained to reconstruct its own input, therefore having output vectors same dimensionality as input. * Unsupervised learning which tries to discover generic features of the data. | * Type: generative * Easier to train than RBMs with contrastive divergence and are thus preferred in contexts where RBMs train less effectively. * Effectivity decreases if errors are present in first few layers. * Pretraining the network with initial weight can solve this problem. | * Architecture:  multiple hidden layers, fully connected architecture with bidirectional connections * Pretraining: unsupervised, fine tuning: supervised * Conjugate gradient method a variant of back propagation is used to train the network. |
| Sparse Auto- encoder | * Extract sparse features from raw data. | * Making the complex data more meaningful by making categories more separable. * Simple interpretation of complex input data * More robust to noise. | * Achieved by penalizing the hidden unit biases or by directly penalizing the output of hidden unit activations |
| Stacked Auto- encoders | * Stack many (sparse) encoders in succession and then train them using greedy layer wise training. * Supervised training on last layer using final feature and then do supervised training on the entire network to fine tune all weights. * In place of fine tuning, a concatenation approach with both hidden features and original features in final or other layers is adopted. | * Type: Generative * Not as accurate as Deep Belief networks * With denoising encoders they are as competitive as Deep belief networks | * Architecture:  multiple hidden layers, fully connected architecture with bidirectional connections * Learning: * pretraining: unsupervised, fine tunning: supervised |
| Denoising Auto encoder | * Stochastically corrupt training instance each time, but still train auto encoder to decode the uncorrupted instances, forcing it to learn conditional dependencies within the instance. | * Robust to noise. * Not as generative as DBNs. | **----** |

## Neural network architectures and applications in medical domain

Neural network has found use in various medical applications. Shanthi et al. [11] proposed a Thrombo-embolic stroke disease predictor system that used a feed-forward multilayer perceptron architecture with 3 layers: 20 inputs, 1 hidden layer with 10 nodes and the output layer with 10 nodes. From the 25 physiological parameters considered as features for the prediction of heart stroke, 20 features were selected using a backward stepwise method based on their correlation importance with the provided class label. The model was then trained using the Back-propagation algorithm and the right model was selected by its performance over the validation set. The trained, optimized artificial neural network model provided an accuracy of 89% overall for heart stroke prediction. Further, a decision-reference system was proposed by Vanisree et al. [14] for diagnosis of Congenital Heart Diseases. The system used a Multilayer feed forward Neural network that was trained on a benchmarked dataset. Various physiological features of a patient were considered in this work, including signs, symptoms and medical test parameters. This system managed to achieve an accuracy of 90% in providing an accurate measure of the patient’s condition. In another work by Dangare et al. [15], a Heart Disease Prediction System was developed based on neural networks. The model predicts the probability of a patient developing a heart disease using 13 continuous medical parameters like blood pressure and cholesterol, discrete features such as obesity and smoking habits, as features. The study manages to diagnose heart disease with 99.25% accuracy by training a comparatively simple neural network with 1 hidden layer and 570 training examples. This study is a good example of how effectively neural networks even in their nascent form, assisted in diagnosis of medical diseases.

Daphne et al. [16] proposed maximum likelihood and single layer Gradient Descent Multi-Layer for neonatal risk prediction which proved to be inconsistent. Further, Kolmogorov’s superposition theorem is proposed to justify the number of neurons in network. Choudhury et al. [1] proposed a single hidden multi-layer perceptron for neonatal disease diagnosis with Genetic method for feature selection. The accuracy found to be 75 percent which could be increased with different hyperparametric variations and the use of deep networks. Noguchi et al. [17] proposed Neural Network Fetal heart rate diagnosis model which is build based on single hidden layer MLP with backpropagation model but lacks hyperparametric variations.

Neural networks, since 2006, under the aegis of Deep Learning, have been applied to various other medical domains such as Biomedical Imaging, Medical transcription, Medical history analysis among numerous others, with great success. Recently, a Natural Language processing technique was proposed to automate medical transcription and translation i.e. Automated Speech Recognition (ASR) system developed by Wołk et al. [18]. This work utilized a sophisticated recurrent neural network with 750 hidden layers trained on sentences of Polish-to-English and vice-versa. Each sentence composed of 50 words or fewer and was preprocessed through various stages: tokenization, factorization, and cleaning. The recurrent neural network was trained on the resulting vocabulary with 500,000 iterations and the model managed to achieve a working accuracy of up to 64.36% on the METEOR metric, under strict hardware constraints. Some of the major work in Deep learning are summarized in Table 2.

Table 2 Literature review based on Deep learning models

|  |  |  |  |
| --- | --- | --- | --- |
| **Authors** | **Objectives** | **Methodology** | **Result / Discussion** |
| Zhaohui et al. [7] | * Conventional Deep Belief Network as an effective training method | * Unsupervised feature extraction with DBN followed by supervised learning by standard SVM. | * Variants of deep learning methods can be implemented for medical data analysis. * Knowledge extraction based on extracted features. |
| Maryam et al. [9] | * Utility of Deep learning concept in Big data analytics | * Deep learning methods for complex pattern extraction, semantic indexing, data tagging, information retrieval and for simplifying descriptive tasks. | * Study deals with applications of deep learning algorithms and architectures for Big data analytics and challenges of adaptation of deep learning algorithms. |
| Premaladha et al. [12] | * Efficient algorithms to predict Melanoma based on Computer Aided Diagnosis (CAD) system. | * Hybrid approach is adopted with Deep learning based neural network and Adaboost SVM. | * High classification accuracy with of 93% better than ANN. |
| Nicholas et al. [17] | * Compare the performance of logistic regression and ordinary regression with ANN based predictive methods for disease risk prediction. | * ANN: varying network architecture structure. * Back propagation: training the network. | * Results suggest a potential for applying deep learning methods to improve disease risk prediction. |
| Nhathai et al. [19] | * To study human behavior prediction for overweight and obese people through online social networks. | * Social Restricted Boltzmann machine (SRBM) is proposed. | * Model predicts future activity levels of users more accurately and more stably then conventional models. |
| Hinton et al. [20] | * Describes a nonlinear generalization of PCA that uses an adaptive, multilayer “encoder” network to transform the high dimensional data into a low dimensional code and a similar “decoder” network to recover the data from the code. | * To optimize weights in nonlinear encoders with multiple hidden layers a “pretraining” procedure using RBM is proposed. | * Layer by layer pretraining can also be used for classification and regression. * Pretraining helps in generalization as very little information in the labels are used to slightly adjust the weights found by pretraining. |
| Kang et al  [21] | * Proposed a deep learning based real time video analyzing module inside the CCTV device to detect and analyze disasters. | * Deep neural network architectures. | * Evaluation showed that the average detection time and accuracy for situations involving “fire” demonstrated a higher detection rate than those involving “car accidents”. |

       Thus, in most of the work on medical diagnosis, it is found that neural networks outperform other machine learning models. Most study have used simple ANNs with a single hidden layer architecture, back propagation algorithm using gradient descent for optimization, without optimized networks to prevent over fitting of their models on possibly skewed medical datasets. Deep network architecture can perform better than the shallow architectures on nonlinear datasets. For complex, noisy and nonlinear data sets there is a need to optimize multilayer layer perceptron based on hyperparametric and parametric optimization so as to better “fit” the latent distributions and perform better predictions. ~~Deep Learning architectures can attain higher classification accuracy even on lesser ldata set [11]~~ .Deep Learning architectures can attain higher classification performance on highly complex distributions that simple shallow models cannot “fit”, as their bias is too high, and the variance in noisy distributions are inherently higher.

# METHOD

The main objective of the study is to train deep Multi - Layer Perceptron (MLP) networks by altering hyperparameters to build an optimized neural network to predict neonatal apnea, implemented using Python Packages such as Sklearn, Keras, Matplotlib etc. with Java library such as DeepLearning4j to build neural networks models. The overall methodology is described in Figure 1, consists of following steps: a) Data exploration and Preprocessing, b) Training Deep MLP networks, c) Altering hyperparameters, d) Training Deep Belief networks and Deep autoencoder networks. Further, based on evaluation criteria's the optimized MLP are compared with Deep Belief Networks (DBN) and Deep Auto-Encoders.

## 

Figure 1 Overall Methodology

## Data exploration and preprocessing

To perform study and validate the performance of the proposed models, a data set containing 367 cases of neonates from NICU, Kasturba Hospital Manipal is considered. The outcome variable signifies the total number of apnea episodes from the time of admission till the first week. At the end of the first week if the apneic episodes were present it is coded as "1" otherwise as "0". ~~Out of 50 input features, only 20 are selected as medically relevant by a medical expert.~~ A medical expert was consulted on the relevance of the 50 features in the dataset. Of these 50 features, 20 were deemed as medically relevant after careful critiquing.

A combination of summary statistics such as mean, median, variance, and visualization techniques such as t-SNE and PCA are employed to explore data. On exploration, the data is found to noisy, and several examples with missing features are discovered. Techniques to approximate these missing features, such as nearest neighbor model approximations, functions of mean and mode, proved to be unreliable and inconsistent due to the inherent noise in the data. As a result, the examples with missing features are discarded. ~~The~~ variables which are entirely textual, descriptive or deemed medically irrelevant by the expert are removed~~.~~ This was done to ensure that only features that showed correlation to the class labels in consideration were retrained, and noise was minimized. These features were the 20 numeric features considered belongs to (a) Demographic such as birth weight, birth cry, gestation age, Apgar score, etc. (b) Maternal covariates includes mode of delivery, steroids, and surfactant, etc. (c) Physiological parameters such as heart rate, desaturation, etc. The initial preprocessing of the dataset, resulted in a reduction of 14.1% of the dataset, from 367 examples to 315, and from 50 features to 20 medically relevant features selected initially.

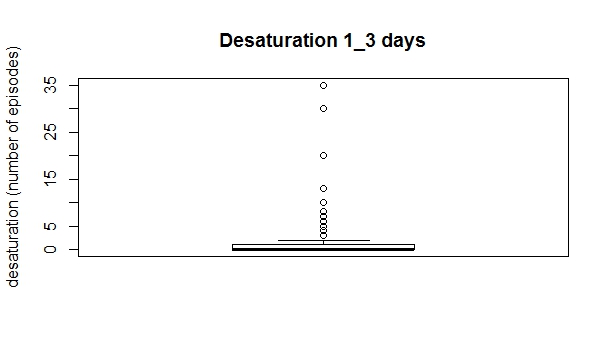
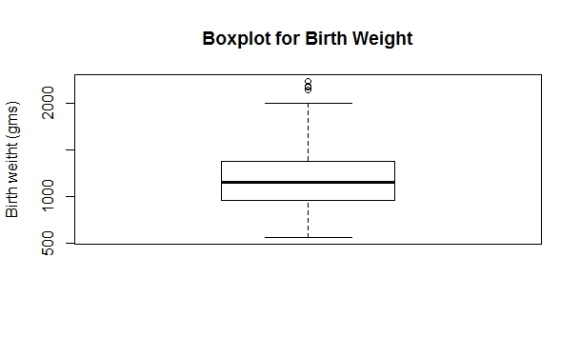
Forest of trees, a feature selection method [22], has been used to select statistically relevant features from the set of 20 features. Random Forest model was considered most appropriate for dataset modelling due to it’s ability to generalize well in distributions with class-imbalance. It is proven to be a highly accurate model on noisy datasets, as it generates its own unbiased estimate of the generalization error.

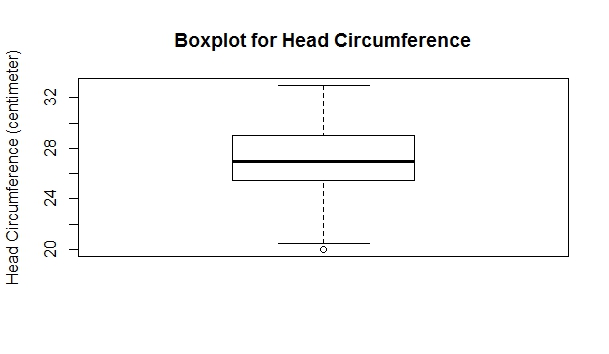
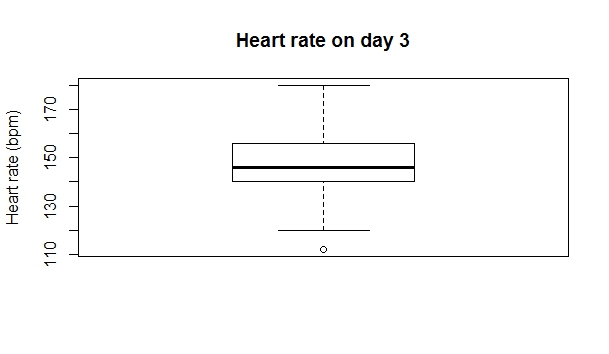
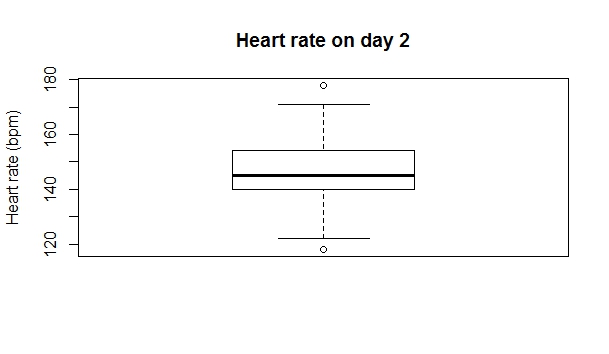
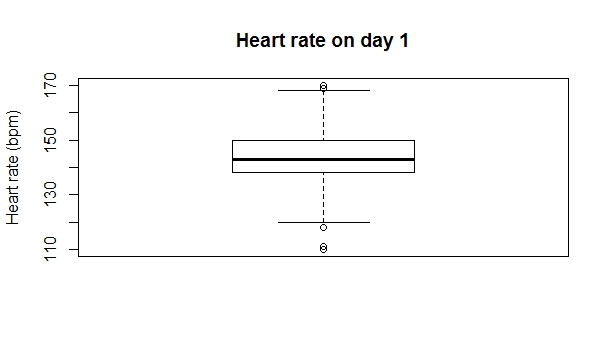
Table 3 represents features with their decreasing importance. To analyze the importance of the various features in the dataset, a Random Forest model is trained on the whole dataset. The resultant trained model is used to provide a measure of the importance of the various features by building a classification model. The feature importance of the different features having a cumulative sum of 100% as expected. The cumulative sum of the importance of the feature selected provides a measure of the confidence of the model in its classification process. With the aim to remove noise and select the right features, only the features with importance greater than 1.5% are retained for the final dataset. Therefore, four features, namely Birth Cry, Surfactant, Apparent Life Threatening Event (ALTE) and Bradycardia are removed from the original dataset, bringing the cumulative sum of the feature-importance of the remaining selected features to 95.8%.

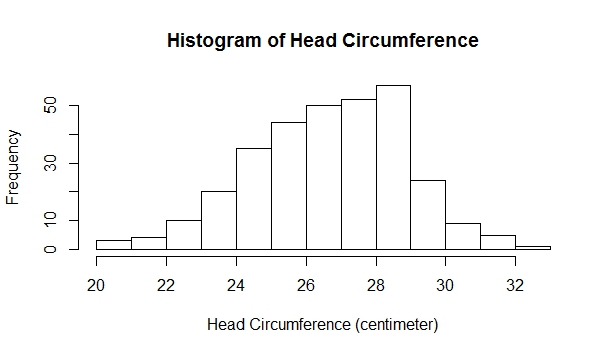
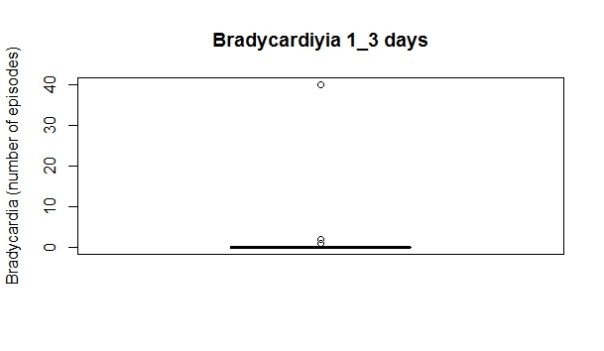
Table 3 Selection of features based on importance in decreasing order

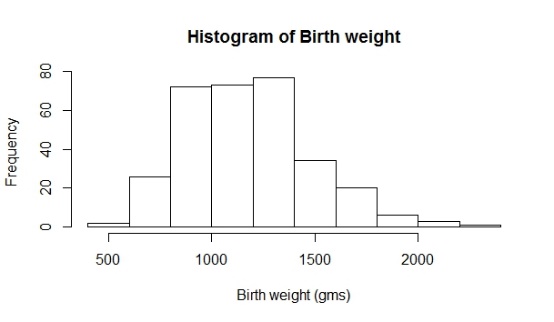
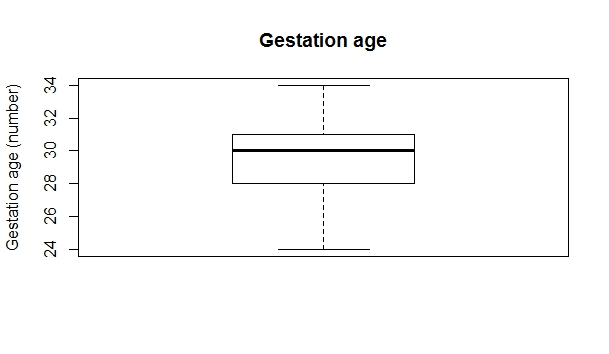
|  |  |  |
| --- | --- | --- |
| Sr.no | Features | Feature importance in % |
| 1 | Birth weight | 14.1 |
| 2 | Heart rate Day 3 | 12.06 |
| 3 | Heart rate Day 1 | 11.3 |
| 4 | Heart rate Day 2 | 10.0 |
| 5 | Desaturation | 7.18 |
| 6 | Gestation age | 7.14 |
| 7 | Head circumference Birth | 6.4 |
| 8 | Dexa / Betha | 4.8 |
| 9 | Mode of Resuscitation | 4.8 |
| 10 | Resuscitation | 3.5 |
| 11 | Steroids | 3.5 |
| 12 | Delivery mode | 3.8 |
| 13 | Apgar 1 minute | 2.5 |
| 14 | Apgar 5 minute | 2.3 |
| 15 | Sex | 2.2 |
| 16 | AGA / SGA | 1.9 |
| 17 | Surfactant | 1.4 |
| 18 | Birth cry | 1.2 |
| 19 | Bradycardia | 0.7 |
| 20 | ALTE | 0.7 |

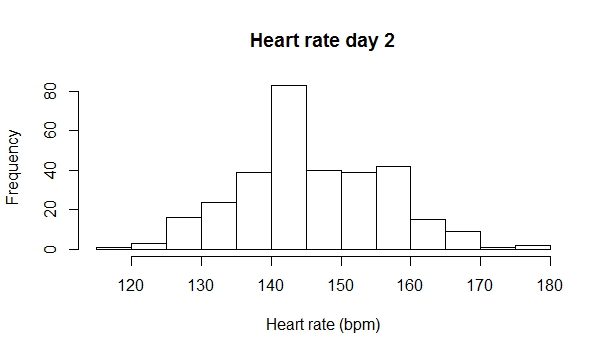
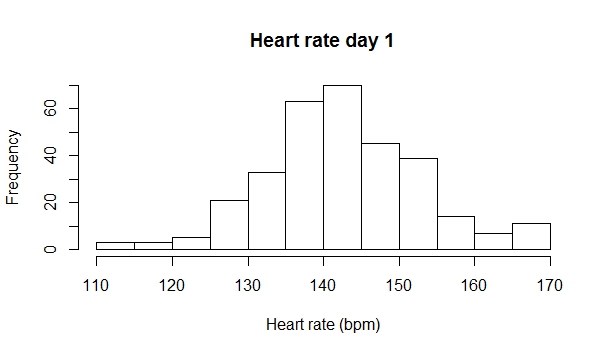
Data Exploration:

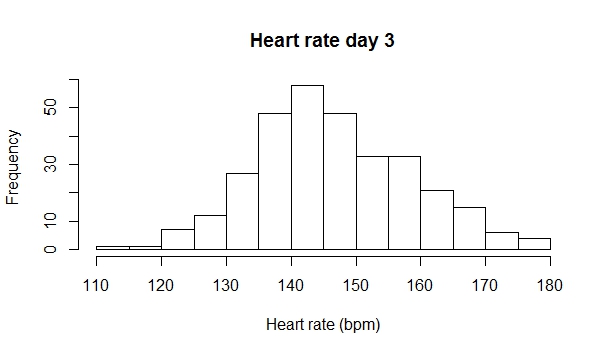
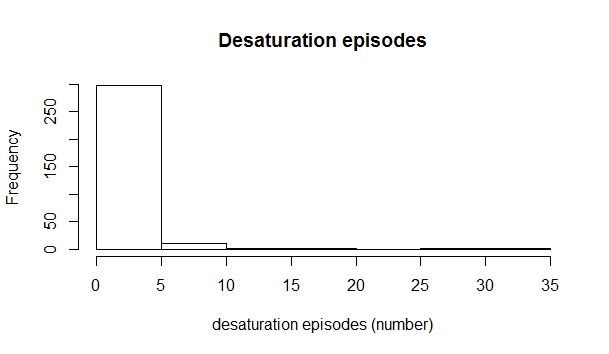


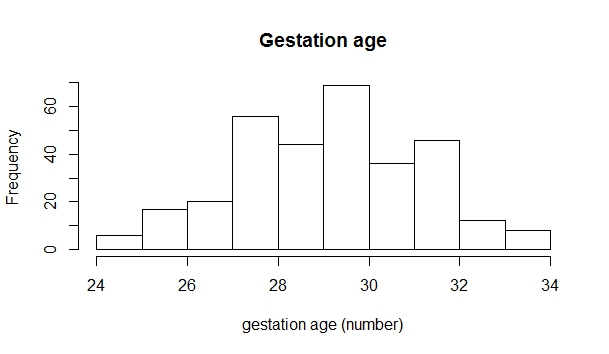
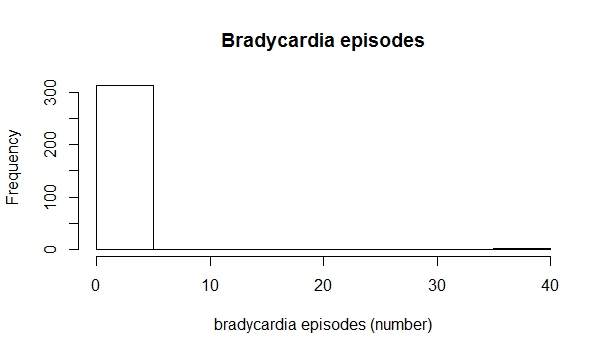


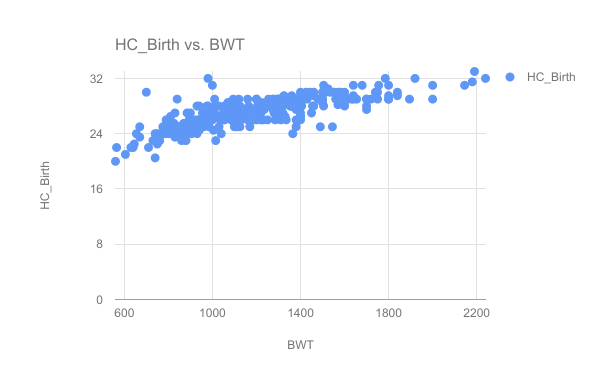


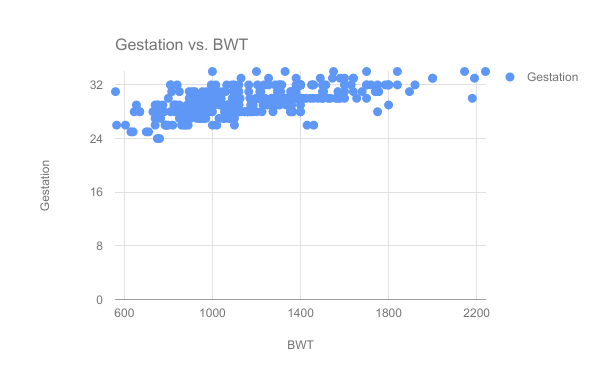
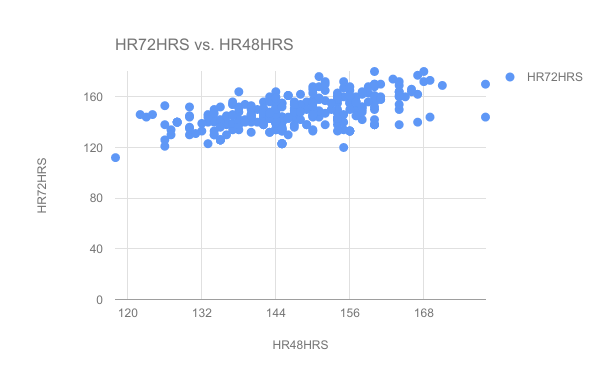












Gestation & Birth Weight have high linear correlation, as expected, as children born prematurely generally have lower Birth Weights, and children born after longer gestation periods, tend to be healthier with greater Birth Weights.

HC\_Birth vs BWT requires a comment.

To explore, visualize and understand the complexity, quality and nature of the data, the study uses Principal Component Analysis (PCA). ~~PCA is an algorithm that performs dimensionality reduction by reducing the feature components along the axis of the maximal variance in the data. It does this by converting a data set with highly correlated features to a set of linearly uncorrelated variables called principal components~~. PCA is an algorithm that discovers the eigen vectors of a distribution along the axis of maximal variances and uses these eigen vectors to identify multiple uncorrelated independent features called principal components having the highest corresponding eigen values. These principal components enables PCA to provide a more intuitive representation of high-dimensional data, into lower dimensional feature spaces, that can be visualized and interpreted. Hence, PCA is a great tool to analyze multi-dimensional distributions in lower dimensions.

But, for a complex multivariate K-Dimensional dataset, it isn’t possible to linearly separate, and hence classify the data points in a feature-space of dimensions lower than K [23]. On the other hand, it is almost always possible to separate K-dimensional information on a feature-space of greater than K dimensions. Therefore, using nonlinear functions that are decided heuristically, it is feasible to obtain an incrementally better linear separation of classes in a nonlinear higher dimensional feature space. Kernel Principal Component Analysis [24] is an extension of Linear PCA. Different Kernels allow for different representations of the same data in higher dimensional feature space, using a unique function that is selected heuristically. While Linear PCA or PCA, attempts to reduce to a low-dimensional linear feature sub -space, it fails if the dataset is restricted to a lower-dimensional albeit non-linear feature subspace. Kernel PCA is a succinct algorithm for analyzing such complex, especially noisy medical datasets as it addresses the sparsity problems that medical datasets are generally ailed by [24]. Kernel PCA is capable of inferring a representation of non-linear manifolds in noisy, complex distribution datasets, using kernels.

From Figure 2 it can be observed that the initially noisy uncorrelated Linear PCA representation of the data now finds a linear correlation, with reduced noise in the Kernel PCA representation of the data Figure 3. It is thus inferred that the data is correlated albeit non-linearly, and hence though the dataset is observed as noisy, as is with most medical datasets, it may be classifiable reliably on some hyper-plane using a high-dimensional non-linear classifier.

Deep Neural Models are especially successful on modelling such complex non-linear distributions, due to their ability to develop higher abstracted representations of the data, warping the plane of high-dimensional space to facilitate better classification. Deep Neural Networks are known to outperform conventional shallow models such as Support Vector Machines (SVM), Random Forests, Decision Trees, Naïve Bayes on such complex noisy distributions.

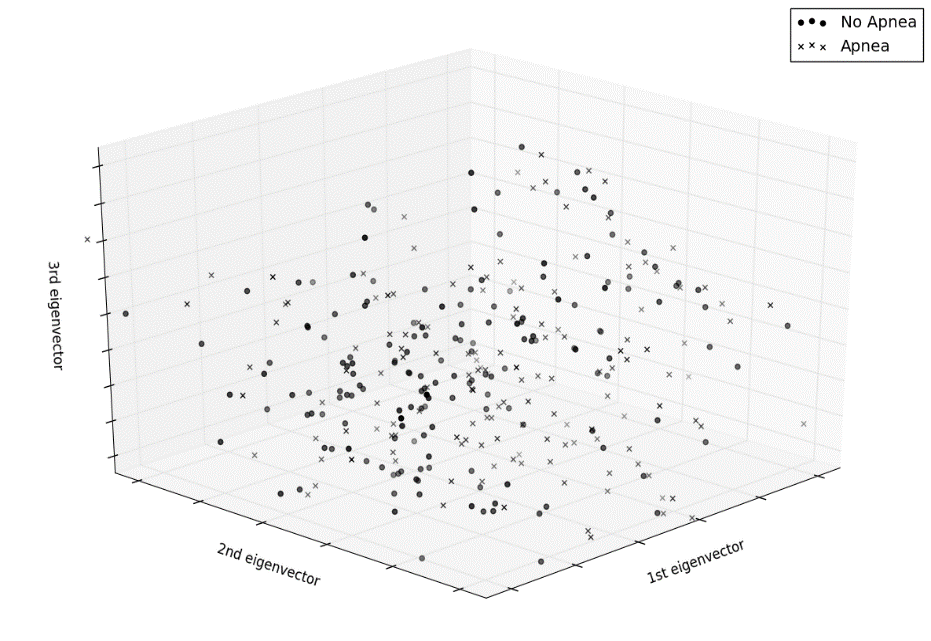


Figure 2 Data visualization using Linear PCA

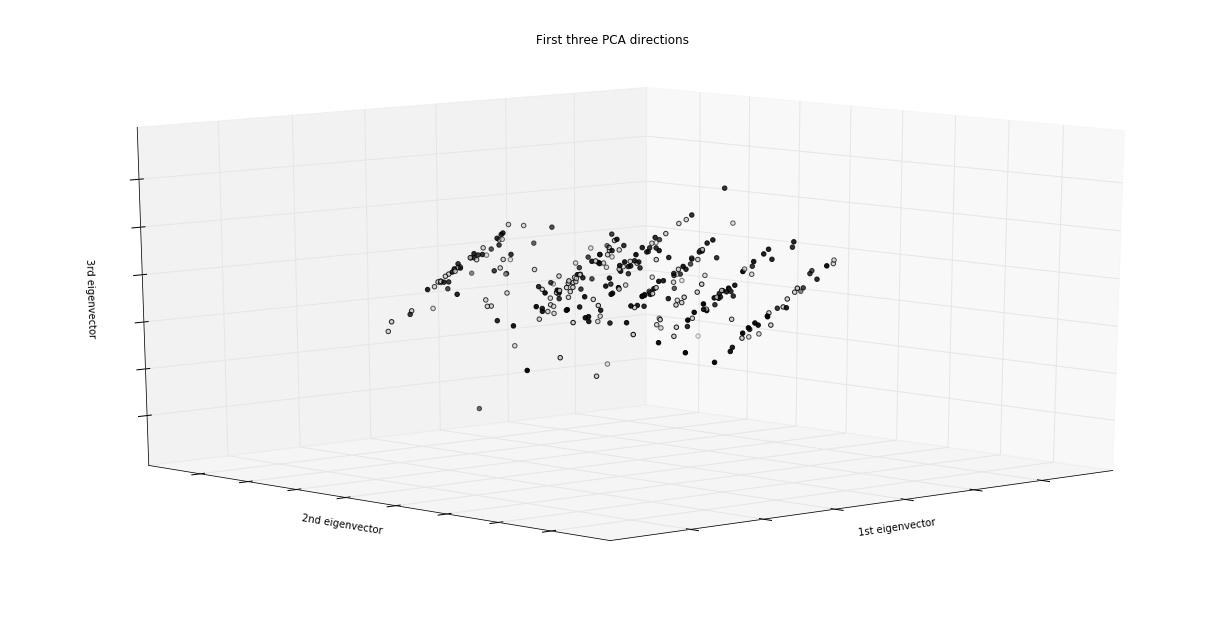
**

Figure 3 Data visualization using Kernel PCA

## Training Deep Multi – Layer Perceptron Network

Multilayer perceptron network (MLP) is a feed forward neural network, with a linear or non-linear activated neuron as its core unit, having one or more hidden layers. The network consists of an input layer of source neurons, at least one middle or hidden layer of computational neurons, and an output layer. The input signal is propagated in a forward direction on a layer by layer basis. The network is then trained using backpropagation algorithm, altering the weights according to the gradient of the error propagated, with the aim to achieve the optimal parametric settings. In the study, the Multi-Layer Perceptron (MLP) network is optimized, i.e. the performance of the network is empirically maximized by selecting, fine-tuning and gradually optimizing various parameters and hyperparameters of the network, as shown in Figure 4. Initial empirical results showed 4 hidden layer model was performing better than simple MLP with 1,2 or layers. Therefore, an MLP with four hidden layers is designed with Stochastic Gradient Descent as the gradient optimization algorithm. The various learning algorithms, parameters, and hyperparameters of the resultant network are altered gradually by selecting the highest performing model at each step.

For training, the dataset is normalized to a Standard Normal Distribution, with zero mean and unit variance with the aim to ensure that no feature is weighted arbitrarily more than the other. The weights in the network are sampled from a Gaussian Distribution with Zero Mean and Variance of 1/N, with N being the average of the number of inputs and outputs in the network [26]. The normalized dataset is randomly split 70:30 into train and test sets respectively. The models are trained for 5000 epochs, with an initial learning rate of 0.01, the momentum of 0.9 and the various other hyper parameters as mentioned in Table 4. The model is cross-validated at every 25 epochs, and the subsequent evaluations were logged and plotted as graphs for comparison.

Selecting best performing **Updater algorithm**.

Persisting with optimal *Activation function,*

*Gradient Descent Algorithm, Depth of network*

& *Regularization algorithm*

.

Selecting best performing **Regularization algorithm**.

Persisting with optimal *Activation function,*

*Gradient Descent Algorithm* & *Depth of network*.

Selecting best performing **Gradient Descent Algorithm.**

Persisting with optimal *Activation function*.

Selecting best performing **Activation function**.

Selecting best performing **Depth of network**.

Persisting with optimal *Activation function* &

*Gradient Descent Algorithm.*

Figure 4: Steps involved in selecting the optimized neural network

### Activation Functions

Activation functions are responsible for deciding the operation performed by each neuron i.e. the output of every neuron in the neural network for a group of inputs, and therefore greatly influences the behaviour of the network. Activation functions directly influence the capacity, and therefore the hypothesis space accessible by the neural network model. Non-linear activation functions like Tanh, Sigmoid allow propagation of nonlinear representations of the inputs through the network over a wider range of values. At the same time, non-linear activation functions also contribute to the inherent variance of the model with an increase in capacity and algorithmic complexity. Linear activations functions are comparatively simpler, maintain an inherently higher bias and lower variance, while allowing easier training, and therefore facilitating training at a lower computational cost. Some of the activation functions [27] which are used are defined below:

*Rectified Linear Unit (ReLU):*  A linear activation function that produces an activation linearly proportional to all positive inputs. It is preferred for larger networks due to its reduced computational complexity and for its ability to produce unsaturated activations even for large inputs. Mathematically, ReLU activation function can be represented as:

*Tanh***:** The output of a neuron with the Tanh activation function is a non-linear function of the input, represented by the Tanh function as:

*Sigmoid*: Non-linear activation function which is an alternative to the Tanh function that results in an output that is a non-linear function of the inputs to the neuron. Mathematically, the Sigmoid function is represented as:

It has been empirically found that Tanh function converges faster than the Sigmoid function.

*HardTanh*: Linear activation function that is a linear approximation of the tanh function with the following mathematical representation as:

HardTanh may be preferred over the Tanh function due to its reduced computational cost.

*LeakyReLU*: Rectified Linear Unit (ReLU) activated neurons don’t propagate any output when the input summand is negative. LeakyReLU allow a small constant negative error to be propagated for negative valued inputs, while still activate linearly for positive inputs. Mathematically expressed as:

For training, the initial design of the MLP model with four hidden layers, with hyperparameters initialized as per Table 4, is used. The above activation functions were individually applied to train the model, with all other hyper-parameters kept constant, and the validation scores for the corresponding models were logged for every 25 epochs during training. The Negative Log Likelihood error, along with the AUC scores for the validation sets, were compared in every instance, and corresponding inferences were made. The activation function corresponding to the highest performing model is retained for all future steps.

Table 4 Hyperparameters of MLP

|  |  |  |  |
| --- | --- | --- | --- |
| Learning Rate | Number of Epochs | Iterations (For each data point) | Momentum  (Nesterov) |
| 0.01 | 5000 | 5 | 0.9 |

### Gradient Descent Algorithms

The MLP network was trained on three popular gradient descent algorithms [28] namely: Stochastic Gradient Descent (SGD), Line Gradient Descent (LGD), and Conjugate Gradient Descent (CGD).

The Standard Gradient Descent is a first-order optimization algorithm also known as the method of steepest descent, which guarantees convergence to a local minimum for any convex function. The algorithm works by updating negatively the parameters which are to be optimized proportional to the differential or "gradient" of the value function G(w), where *wn* is the parameter to be optimized, *G(w)* is the loss/value function, and *α* is the proportionality constant or learning rate.

*Stochastic Gradient Descent:* SGD is a stochastic approximation of the gradient descent optimization algorithm that iteratively minimizes the error or objective function by considering the error for each data point. Defined as G (w), where G(w) is the estimated loss function obtained as a summand of Gi (w), associated with each loss/value function at each iteration i.

SGD is not as direct as the standard gradient descent but is shown to converge significantly faster than the Standard Gradient descent algorithm [29]. SGD can further be optimized by using various learning rate updater algorithms like Momentum, AdaGrad, Adam, to ensure smoother weight updates and a faster, more steady descent to the minima.

*Conjugate Gradient Descent:* CGD algorithm is another iterative variation of the standard gradient descent optimization algorithm that calculates a sequence of mutually orthogonal vectors, to compute the representation of the parameters using these resultant basis vectors. CGD allows approximation of the value function for large and sparse datasets at reduced computational complexity, albeit restricted to symmetric matrices. CGD algorithm guarantees convergence to minima in at most n iterative steps, equal to the number of mutually orthogonal vectors in the basis [28].

*Line Gradient Descent:* Stochastic Gradient Descent with Probabilistic Line Search or Line Gradient Descent (LGD). Probabilistic Line Search is constructed by combining the structure of deterministic methods with notions from Bayesian optimization [30].

The 4 Hidden Layer MLP model with the activation function lending the highest performance, obtained from the previous step, is selected and initialized with the hyper parametric values as per Table 4, retained from the previous step. Each of the above optimization algorithms is individually used to train the model, with the same hyper-parameters. The validation scores are logged for every 25 epochs during training. The Negative Log Likelihood error, along with the AUC scores for the validation sets, are compared at every instance, and corresponding inferences are made. Finally, the Optimization algorithm, lending the maximal performance is retained, as the updated model.

### Depth of the Network

The depth of a Neural Network is defined in terms of the number of hidden layers in the architecture and as it increases, so does its complexity and ability to learn complex data and abstractions. The number of layers in a network, critically determine the degree of bias and variance inherent in the model. With deeper networks, it is expected that the complexity and variance of the model will increase significantly, along with its tendency to overfit. Inferring from the Multi-Layer Perceptron model, with its selected optimization algorithm and activation, obtained from the previous step, MLP models varying from 1 to 8 hidden layers as shown in Table 5, are created. The initial hyper parametric settings from the updated model were retained and the selected optimization algorithm and activation function were used to train the model.

L1 regularization is introduced into the network to take the emphasis off the number of neurons in the network as L1 severely down-weights inactive neurons. Further, allowing the variations in the layers, and the subsequent learning of higher abstractions to influence the experiment. Again, the number of layers corresponding to the highest performing model is noted, and the particular model is selected as the updated MLP model, for further optimization.

Table 5 Defining structure of network

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Number of**  **Hidden Layers** | **Number of Neurons** | | | | | | | | | |
| **Layer 1** | **Layer 2** | **Layer 3** | **Layer 4** | **Layer 5** | **Layer 6** | **Layer 7** | **Layer 8** | **Layer 9** | **Layer 10** |
| 1 | Input  (16) | 16\*4 | Output  (2) |  |  |  |  |  |  |  |
| 2 | Input  (16) | 16\*4 | 16\*2 | Output  (2) |  |  |  |  |  |  |
| 3 | Input  (16) | 16\*4 | 16\*2 | 16 | Output  (2) |  |  |  |  |  |
| 4 | Input  (16) | 16\*4 | 16\*2 | 16 | 16/2 | Output  (2) |  |  |  |  |
| 5 | Input  (16) | 16\*4 | 16\*2 | 16 | 16/2 | 16/4 | Output  (2) |  |  |  |
| 6 | Input  (16) | 16\*8 | 16\*4 | 16\*2 | 16 | 16/2 | 16/4 | Output  (2) |  |  |
| 7 | Input  (16) | 16\*16 | 16\*8 | 16\*4 | 16\*2 | 16 | 16/2 | 16/4 | Output  (2) |  |
| 8 | Input  (16) | 16\*32 | 16\*16 | 16\*8 | 16\*4 | 16\*2 | 16 | 16/2 | 16/4 | Output  (2) |

*[Input: No. of neurons are 16, one for each input; Output (2) Neurons are the output neurons]*

### Regularization

Regularization algorithms are used to optimize the bias-variance trade off in machine learning models and to facilitate a better ‘fit' of the models on a given dataset. They achieve this by modifying the objective function to penalize large weights in the network. Regularization is widely used in practice to limit and manage the inherent bias in neural networks with the aim to prevent overfitting during the training of the model. It allows the network to emphasize a more uniform distribution of weights across the architecture as the training proceeds.

The literature [31] mentions several regularization techniques such as L1 and L2 regularization as well as combinations of the two. L1 and L2 norm regularization are derived from the Lasso and Ridge Regression methods used to optimize the bias-variance trade off. L2 regularization can be implemented by augmenting the error function with the squared magnitude of all the weights in the neural network. In the case of L1, λ a regularization-strength hyperparameter value is added to reduce overfitting.

To facilitate the comparison between the performance benefit provided by regularization techniques, the updated MLP model, obtained from the comparison of the MLP models with varying layers, is selected, modified with the corresponding regularization settings and trained with the retained initial hyper-parameters fixed as per Table 4. The models are trained using varying L1 and L2 regularization methods and combining of the two and with drop-off, with each as a separate model.

### Learning Rate Optimization Algorithms

The Stochastic Gradient Optimization algorithm has several variations to its learning rate decay scheduling method, which are shown to improve its performance and efficiency significantly [31]. The Learning Rate in a Neural Network model is crucial to its performance as it directly influences the rate of weight change in the model and the rate at which it learns the features in the high-dimensional functional space. By the use of optimized updater algorithms in the learning rate, the rate at which the network learns can be dynamically changed to facilitate optimized learning of the network and increase network performance on noisy and non-linear data. The several updater algorithms used to allow dynamic change of the learning rate in a neural network model are as follows: Momentum, Nesterov Momentum, Adam, RMS Prop, AdaDelta and SGD.

*Momentum:* Algorithm uses a hyperparameter as part of first order equation to change the value of the learning rate, depending on the previous gradient of the loss function. Further allowing a smoothed descent with steps of varying sizes, encourages the gradient descent algorithm to reach the local minima faster, and more reliably.

*Nesterov's Momentum:* Algorithm operates similarly to the standard Momentum algorithm, albeit adding a correction factor to allow anticipatory gradient updates that provide both stronger convergence and consistency, preventing the descent from straying into a non-optimal path [32].

*Adagrad:* Algorithm provides another first-order equation for scaling the learning rate according to the history of the gradient. It ensures that for larger gradients, the learning rate is reduced, and similarly, for smaller gradients the learning rate is enhanced.

*RMSProp:* It provides a second-order moment equation to change the learning rate by exponentially decaying the average and not the sum of gradients.

*AdaDelta:* AdaDelta provides two second-order moment equation to change the learning rate.

*Adam:* Algorithm uses both second-order and first-order moment equations to update the learning rate, but the moments decrease with time.

The updated model obtained from the comparison of various regularization techniques in the previous step is trained using Stochastic Gradient Descent algorithm with each specific updater algorithm individually, with all the other remaining hyperparameters initialized as per Table 4.

## Training Deep Belief Networks and Deep Auto encoders

Deep Learning, a machine learning subfield which attempts to learn a high level of abstractions in data by utilizing hierarchical architectures. It can be categorized as Convolutional Neural Networks (CNN), Restricted Boltzmann Machines (RBM), Autoencoder and Sparse coding. The work deals with implementation of Deep Belief Networks (DBN), an RBM based method and Stacked auto encoder based method.

### Deep Belief Networks

Probabilistic generative models with a joint probability distribution and an efficient layer by layer greedy learning strategy. It is fully connected architecture with partial bidirectional connections with unsupervised training which removes the necessity of labelled data for training. The Deep Belief Network (DBN), initialized with an architecture parameters as shown in Table 6, and has 6 Hidden Layers with three encoding and three decoding layers. It uses Linear Gradient Descent Algorithm for Optimization, and trained for 5000 epochs with 5 iterations for each training example. The model is initially pretrained for 1000 epochs, before using supervised training with backpropagation algorithm. Further, model is cross-validated at every 25 epochs and the resulting evaluations are logged for observations and analysis.

### Deep Stacked Autoencoder

A Generative multiple hidden layer, fully connected architecture with bidirectional connections which uses unsupervised learning for pretraining that applies backpropagation by setting the target values equal to inputs, is proposed. Unsupervised pretraining guides the learning towards basins of attraction of minima that supports better generalization from the training dataset [33]. Stacked Autoencoders can have several modifications that allow it to have better generalization on training, such as adding some random noise into the learned encoding, with the aim to force the model to learn the essential feature components only, these are known as Denoising Autoencoders [34]. With the advent of more computing power, the Stacked Autoencoders mode has been modified to include more encoding and decoding layers, to make Deep Stacked Autoencoders, which are now possible to be trained effectively with the help of pre-training for better performance optimization.

The architecture has 6-hidden layer, with three encoding and three decoding layers, as shown in Table 6. The Model is initially layer-wise pre-trained for 1000 epochs, after which it is fine-tuned by supervised and trained by backpropagation for 1000 epochs. The model is trained with an initial learning rate of 0.01, Nesterov's momentum of 0.5, and Stochastic Gradient Descent Algorithm, AdaGrad optimization, and ReLU activation function for all neurons in the hidden layer, with 5 iterations for every training batch/example.

Table 6 Deep network model configuration

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Deep Network Model Specification** | **Layer 1** | **Layer 2 Neurons** | **Layer 3 Neurons** | **Layer 4 Neurons** | **Layer 5 Neurons** | **Layer 6 Neurons** | **Layer 7 Neurons** | **Layer 8 Neurons** |
| Stacked Autoencoder | 16 Input | 16/2 Encoding | 16/4  Encoding | 16/8 Encoding | 16/4 Decoding | 16/2 Decoding | 16 Decoding | 16/8  Softmax Classifier Output |
| Deep Belief Network | 16 Input | 16/4 Encoding | 16/8 Encoding | 16/4 Decoding | 16/2 Decoding | 16 Decoding | 32 Decoding | 16/8  Softmax Output |

# RESULT AND DISCUSSION

The Multi-layer perceptron (MLP) with different hyperparametric optimization are evaluated based on evaluation criteria such as (AUROC) Area Under the Receiver Operating Characteristic curve and F1 score. Further, the optimized MLP models with varying hyperparameters are compared with Deep Belief Networks and Stacked Denoising Autoencoders.

## Hyperparametric optimization for MLP

### Activation Functions

AUC Score for different activation functions graph comparing training models at different iterations is shown if Figure 5. It can be seen that the ReLU and its variant Leaky-ReLU clearly outperform all the other activation function models, with the much higher stable performance on the validation set. Further, the rate of performance growth in the models with ReLU and Leaky ReLU is strikingly higher in the initial iterations of the training procedure. The higher performance of the ReLU activation functions can be attributed to the reduced complexity of linear activations allowing easier training of deeper models. Also, HardTanh, a linear approximation of the non-linear Tanh function, actually results in a model with higher performance, serving as another instance of the benefits of linear activation functions in training deep networks.

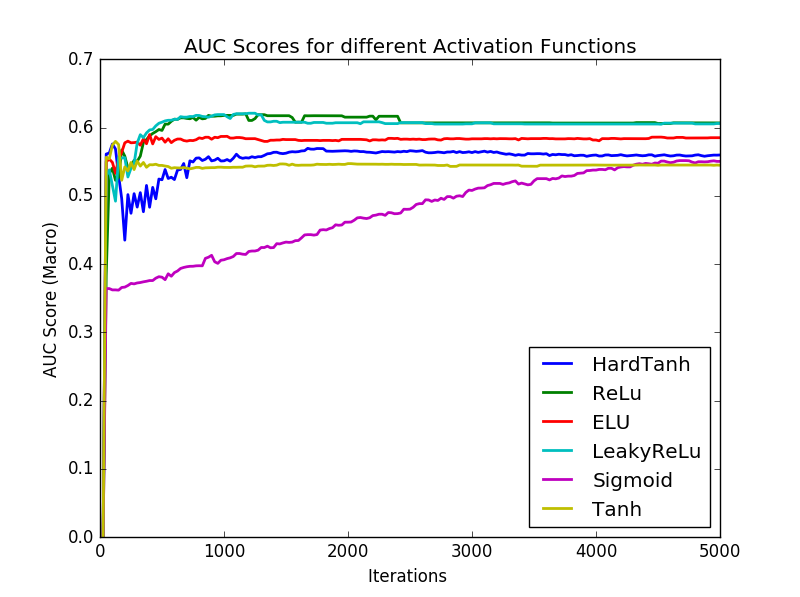
****

Figure 5 AUC scores versus Iteration graph for different activation functions

### The Optimization Algorithm

From Figure 6 depicting Cross Entropy vs Iteration graph of the models, it is observed that the Line Gradient Descent (LGD) algorithm converges to lower Cross Entropy values faster than both the Conjugate Gradient Descent (CGD) and Stochastic Gradient Descent (SGD) algorithms. It can be attributed to the iterative averaging of the gradient in batches from both the SGD and CGD algorithms, that causes the error rate to reduce gradually as compared to the LGD algorithm. The LGD algorithm alters the weights of the network based on its efficient Bayesian optimization algorithm, automating both the selection of a good learning rate and its decay scheduling [30]. Further, Figure 7, compares the graphs with AUC score on the validation sets for the three algorithms and it can be seen that the CGD algorithm consistently provides the highest performance model. The LGD algorithm settles to non-optimal local minima, after the 700th iteration as can be witnessed by the corresponding flat gradient, while the other two iterative batch algorithms vary drastically as the training proceeds with the performance of the SGD algorithm dropping eventually. It can be inferred that the CGD proves the most effective for the existing complex noisy dataset that can be attributed to its iterative solution. Further, optimizing the gradient by finding the conjugate directions ensures a more direct convergence to the local minima. Conjugate Gradient Descent, is guaranteed to approach a local minimum, in at most the number of mutually conjugate directions in the basis.

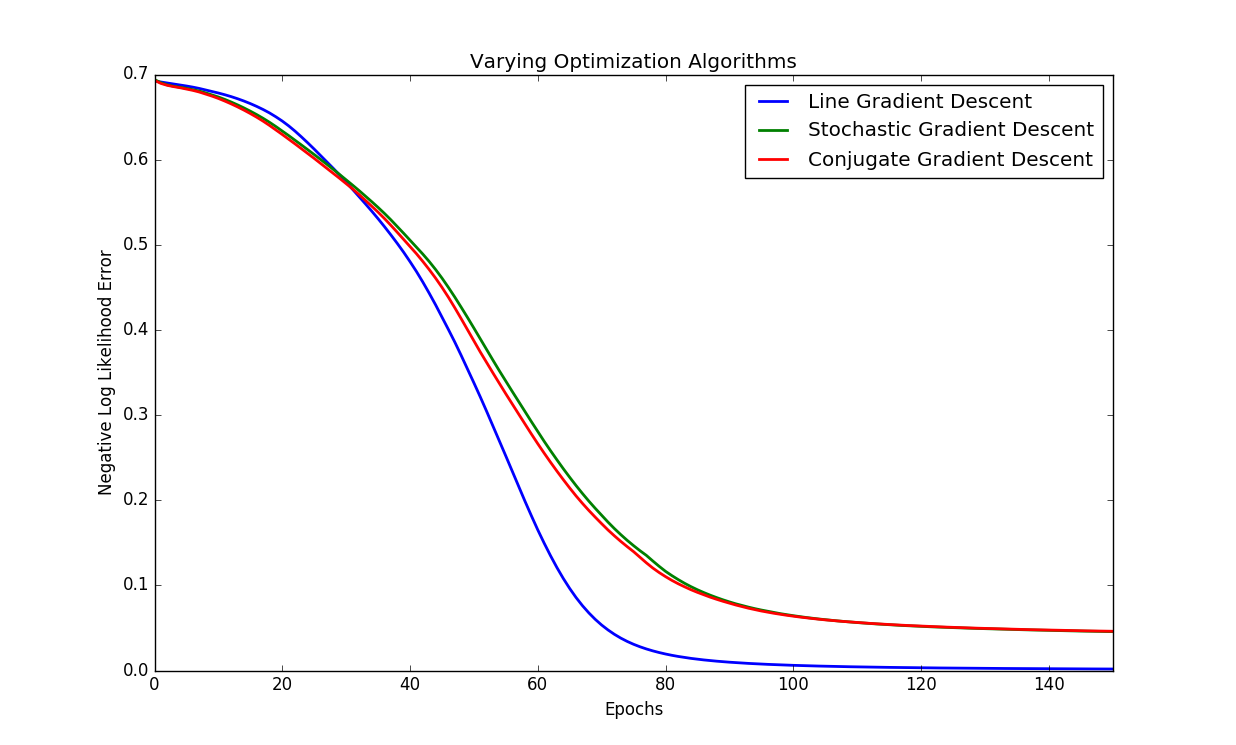
****

Figure 6 Cross Entropy versus Epochs graph for different optimization algorithms

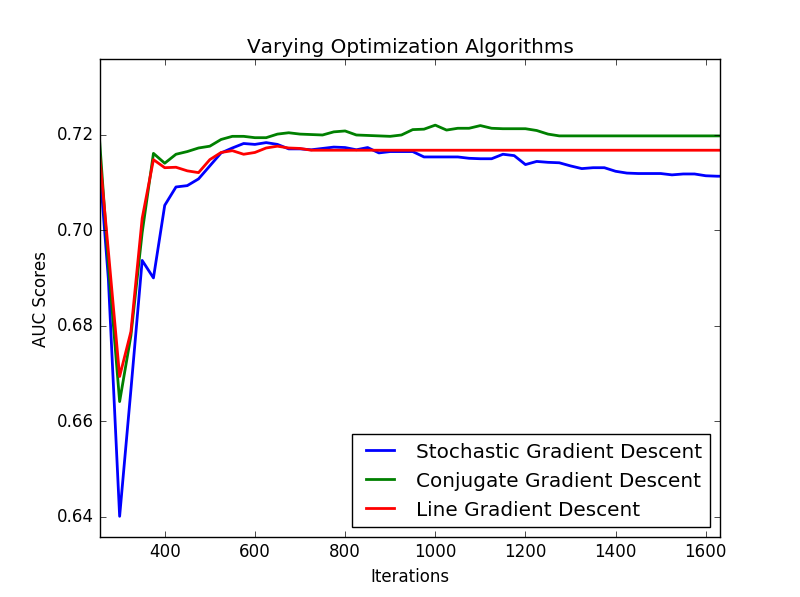
****

Figure 7 AUC versus Iteration graph for different optimization algorithms

### Depth of the Network

From the RMS Error vs. Iterations graph shown in Figure 8, for the models with varying network depth, it is observed that deeper networks i.e. with more hidden layers, converge to lower error rates exponentially faster than their shallower counterparts. Correspondingly from the validation set, AUC Score vs. Iteration graph for the models, shown in Figure 9, it can be inferred that the model with the greatest number of hidden layers, provides the highest performance model, clearly indicating the ability to "fit" on complex datasets, increases with the depth of the model. Although the models with higher hidden layers follow the higher performance, their tendency to overfit on the dataset also increases as expected. For the models with hidden layers from 2 to 4, in the AUC score vs. Iteration graph, it can be seen that performance, in fact, drops as the training proceeds that can be attributed to a possible local-minima or a resultant over-fit as the models gradually settle to a zero-gradient slope. It is also noted that 7 layer MLP overfits due to high variance and does not generalize very well with drop in performance.

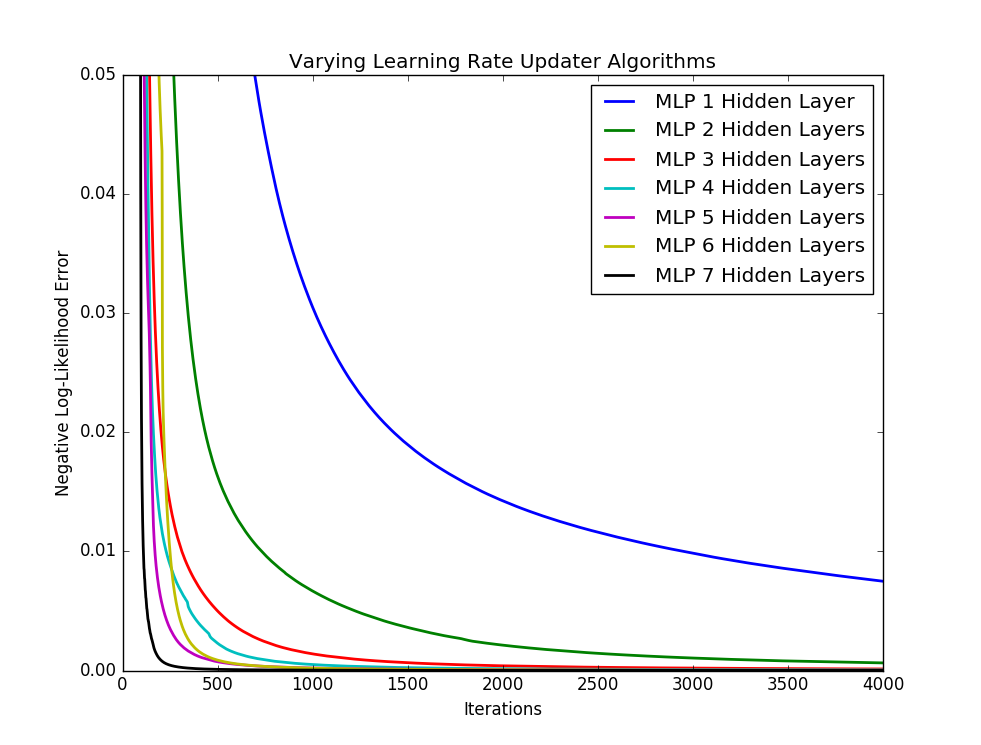
****

Figure 8 RMS Error versus Iteration graph for varying hidden layers

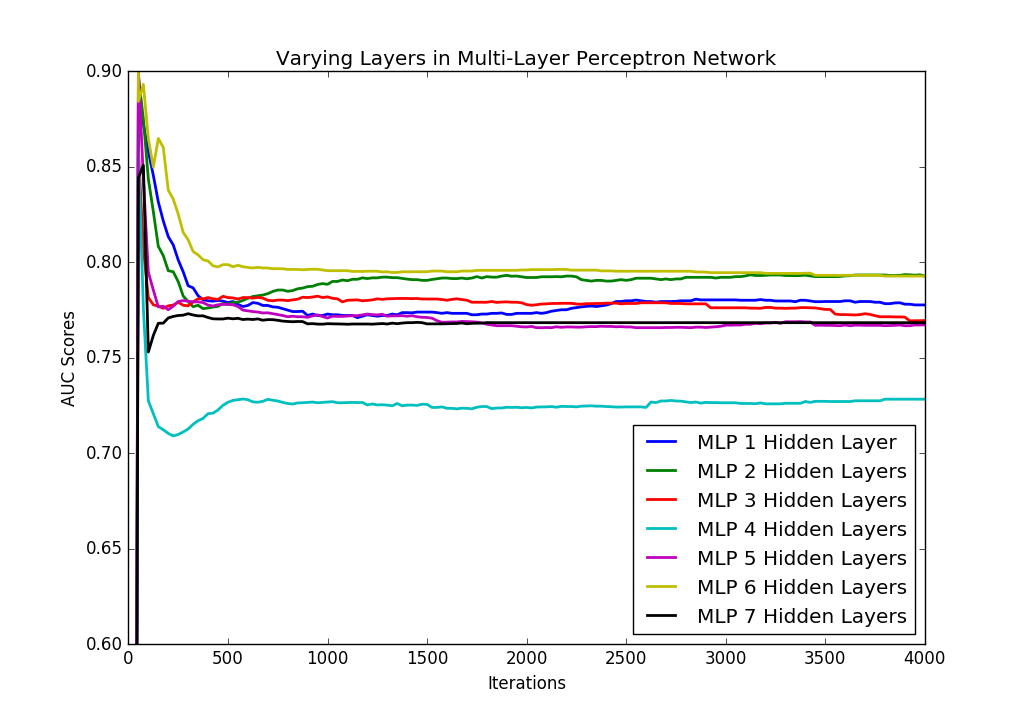
****

Figure 9 AUC scores versus Iteration graph for varying hidden layers

#### Regularization

From the obtained results in the comparison of models with varying regularization settings given in Figure 10, it is observed that regularization, as both L1 and L2, individually or as a combination, do not provide any significant advantage over the model with no regularization. The models with L1 regularization consistently result in lower performance. It can be inferred that penalizing large weights in the network, does not always result in better represented feature space, and for complex datasets, L2 regularization does not impact the model performance, with little impact on the sparsity of model weights.

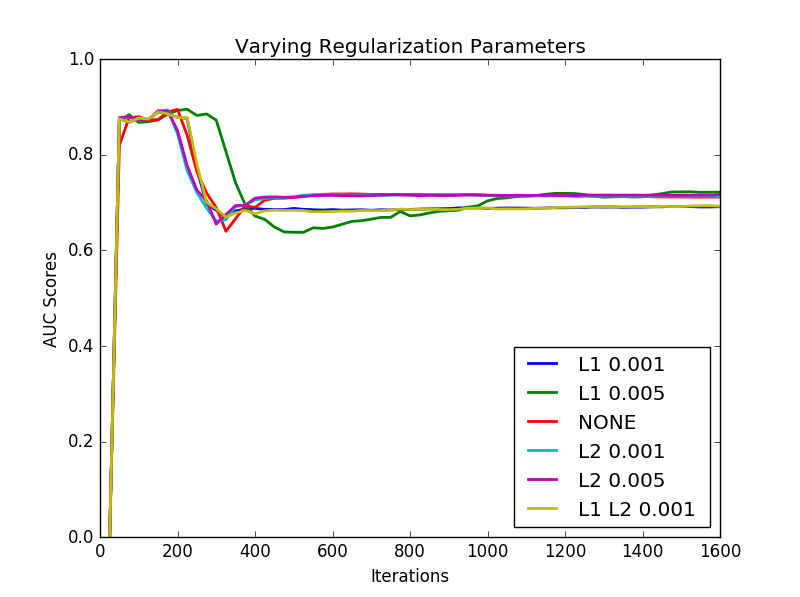
****

Figure 10 AUC versus Iterations graph for varying regularization parameters

### Learning Rate Updater Optimization Algorithms

From the Training RMS Error vs. Iteration graph based on different updater algorithms for Stochastic Gradient Descent algorithm, shown in Figure 11, it can be clearly observed that updating the learning rate through either a second-order or first-order algorithm makes a drastic difference in the model's ability to converge to a local-minima during training. Further, from the Figure 12, it can be observed that algorithms that utilize the second order moment like Adagrad and Adam, consistently performed higher than the other learning rate updater algorithms. Although SGD algorithm (without momentum) during the early iterations gave a higher performance due to the zig-zag nature of descent of the algorithm, the performance was not stable, and as the model slowly converged to the minima, it gradually shifted towards a non-optimal local optimum. On the other hand, Adam, AdaDelta consistently provide higher performances as the training proceeds, and gradually descent to a local-minima with a much higher performance than that of the SGD algorithm.

It can, therefore, be inferred, that a combination of second and first order momenta in updating the learning rate is beneficial in complex feature spaces with multiple local-minima’s, as can be expected from a noisy complex medical data, to obtain a more stable consistently higher performing model. From Figure 13, it can be seen that the optimized 6 Hidden Layer MLP network, trained using SGD with Adam Updater, in fact outperforms the 6 Hidden Layer MLP network trained using CGD.

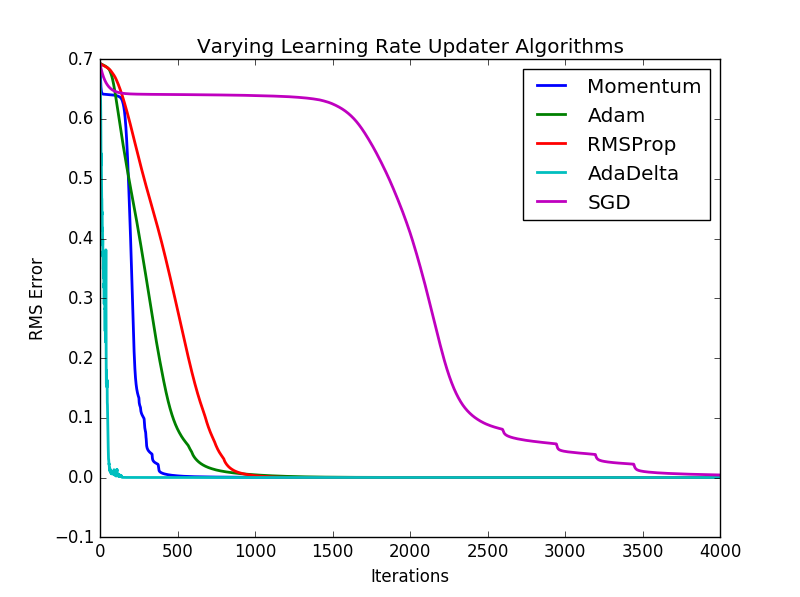
****

Figure 11 RMS error versus iteration graph for different updater algorithms

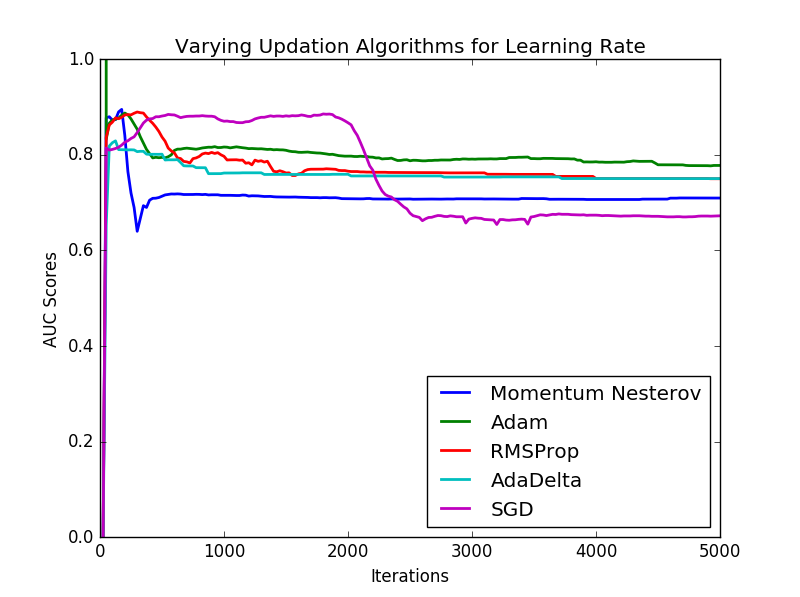
****

Figure 12 AUC scores versus iterations graph for different updater algorithm

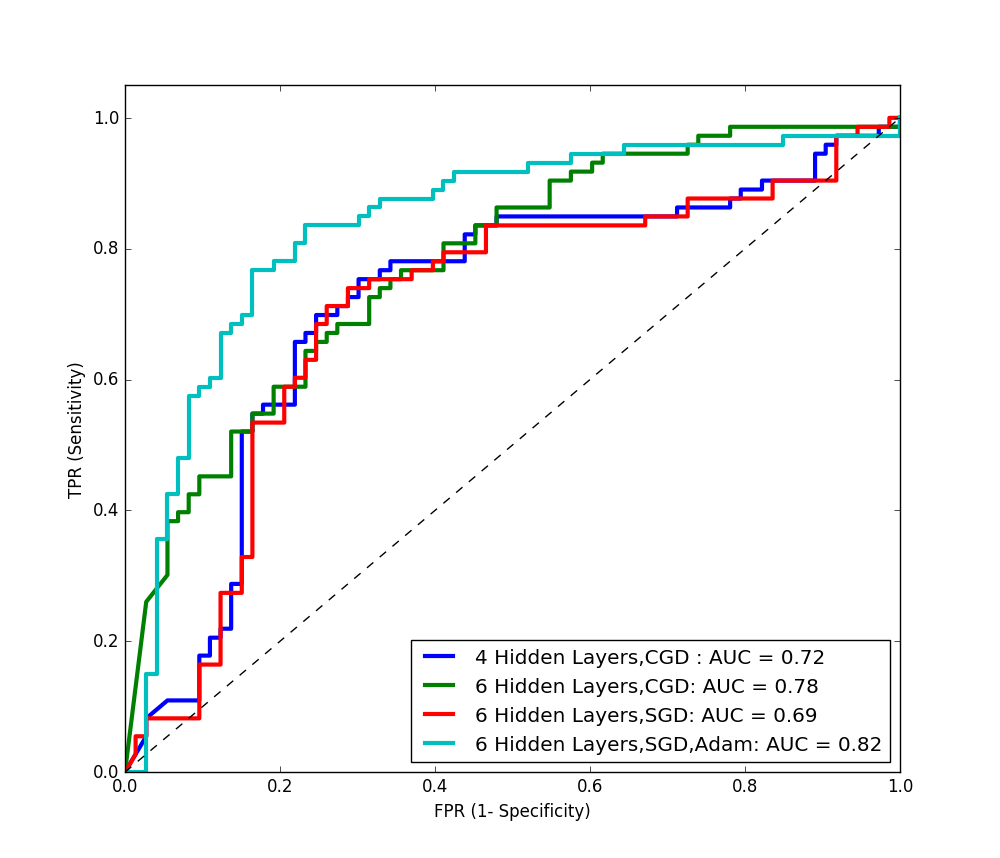


Figure 13 Comparison between various stages of optimization in MLP models

## Comparing the different models with evaluating criteria’s

Table 7 shows comparison of Optimized MLP models with its AUC score. The 6-hidden layer MLP proved to be most effective model with Adam Decay and Stochastic Gradient Descent with an AUC Score of 82%. Further, the shallow models such as Support Vector machine (SVM) AUC 0.66, Decision trees (DT) AUC 0.61 and Random forest AUC 0.72 were found to be less efficient compared to deep neural network models.

Figure: Comparison of performance of Shallow Models

|  |  |
| --- | --- |
| **Shallow Models** | **AUC Score** |
| Decision Tree Model (Optimized Classification and Regression Tree (CART)) | 0.61 |
| Random Forest Model  (Average Probabilistic prediction from Ensemble of forests) | 0.7808 |
| Support Vector Machine with RBF Kernel | 0.66 |

Table 7 Comparing different stages of optimization of the MLP model

|  |  |
| --- | --- |
| **Neural Network Model Specification** | **AUC Score** |
| MLP with 4 Hidden Layers, Conjugate Gradient Descent (CGD), ReLU | 0.7175 |
| MLP with 6 Hidden Layers,  Conjugate Gradient Descent (CGD), ReLU | 0.7808 |
| MLP with 6 Hidden Layers,  Stochastic Gradient Descent (SGD), ReLU | 0.6899 |
| MLP with 6 Hidden Layers,  ReLU, SGD with Adam Updater | 0.8247 |

The observed performance of the final optimized Multilayer Perceptron model with Deep Belief Network and Stacked Denoising Autoencoder have been logged in Table 8. AUC score cure for the same is shown in Figure 14.

Table 8 Comparing final Optimized MLP Model with other Deep Neural Network Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Neural Network Model Specification** | **AUC Score** | **Recall** | **F1 Score** | **Precision** |
| MLP with 6 Hidden Layers, SGD, ReLU activation function, with Adam Updater | 0.8247 | 0.7160 | 0.6798 | 0.6642 |
| Deep Autoencoder | 0.8354 | 0.7308 | 0.7202 | 0.710 |
| Deep Belief Network | 0.781 | 0.6961 | 0.6212 | 0.6231 |

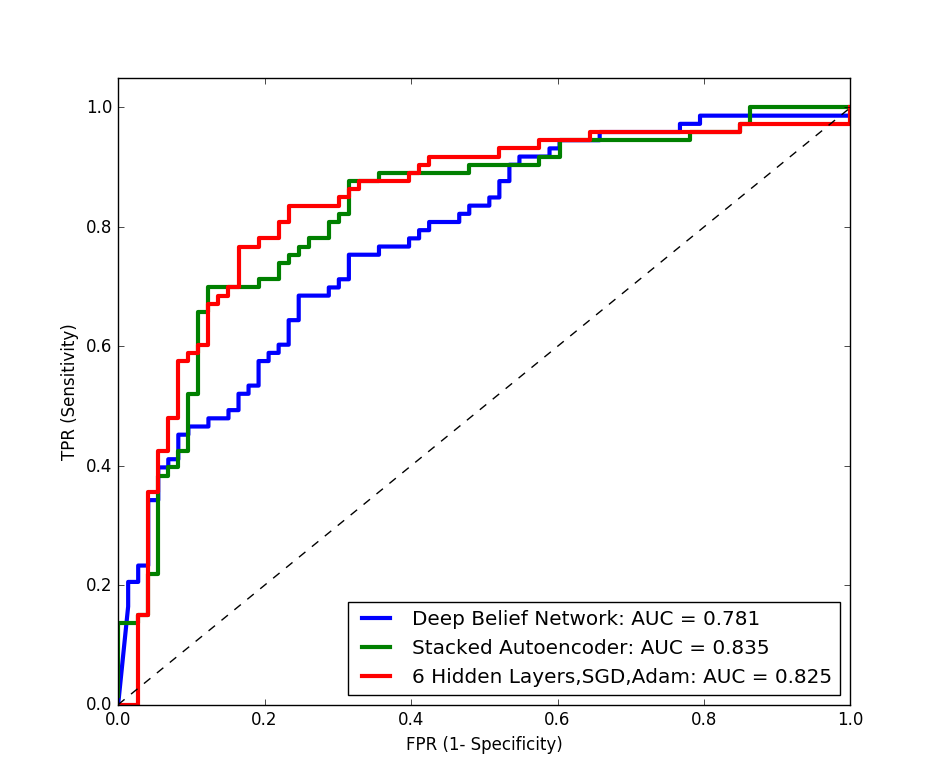


Figure 14 Sensitivity versus 1- Specificity graph (AUC score) for Deep network architectures

# CONCLUSION

The study presents different ways of optimizing neural network architectures for complex, nonlinear and smaller data set. The 8 Layer Multilayer Perceptron model, with Adam Decay and Stochastic Gradient Descent, with an AUC Score of 82% while the Deep Autoencoders AUC score of 83% proved to be the most useful model, and the DBN network had an AUC of 78%. Therefore, it is inferred that carefully Optimized Multi-Layer Perceptron models has proved to be nearly as effective at abstracting correlations and inferences from noisy, highly non-linear datasets as Deep neural network models such as DBN and Autoencoders. With the tuning and use of suitable hyperparameters in Deep MLP models, it is possible to extract significant improvements in performance on complex medical datasets. The study also revealed that, compared to shallow models such as Support Vector Machines, Decision Trees and Random forest algorithm optimized MLP, Deep autoencoders outperforms classification accuracy in predicting neonatal apnea. The steps involved in designing optimized MLP are generic and can handle any predictive problem with improved accuracy. Further the study can be extended in optimizing Deep Belief Network, Deep auto-encoders and use of ensemble approach to improve the predictive accuracy.

# ACKNOWLEDGEMENT

The authors are deeply indebted to Kasturba Hospital, Manipal University for providing an opportunity to work on the research problem.

# REFERENCES

[1] Chowdhury, R. Dilip, C. Mridula and R. K. Samanta. "An artificial neural network model for neonatal disease diagnosis." *International Journal of Artificial Intelligence and Expert Systems (IJAE),* vol. 2(3), pp. 96-106, 2011.

[2] Parkhi, M. Omkar, V. Andrea and Z. Andrew, “Deep face recognition”, *In British Machine Vision Conference,* vol. 1(3), 2015.

[3] D. Ferreira, A. Oliveira, and A. Freitas, “Applying data mining techniques to improve diagnosis in neonatal jaundice.” *Springer, BMC Medical Informatics and Decision Making*, vol. 12(143) December 2012.

[4] H. Sak, A. Senior, K. Rao, K. and F. Beaufays, 2015. “Fast and accurate recurrent neural network acoustic models for speech recognition.”  *arXiv preprint arXiv:* vol.1507.0694*7*, 2015.

[5] S. Guha and O. Schrijvers, “Robust Random Cut Forest Based Anomaly Detection on Streams”, *In Proceedings of the 33rd International Conference on Machine Learning* pp. 2712-2721, 2016.

[6] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks Elsevier*, vol. 61, pp.85-117, 2015.

[7] L. Zhaohui, Z. Gang, H. X. Jimmy and H. V. Qinminh,, “Deep learning for healthcare decision making with EMRs”, *IEEE In Bioinformatics and Biomedicine (BIBM),* pp. 556-559, November 2014.

[8] G. Yanming, L. Yu, O. Ard, L. Songyang, W. Song and L. S. Michael Liu. “Deep learning for visual understanding: A review,” *Elsevier, Neurocomputing,* vol. 187, pp.27-48, September 2016.

[9] N. M. Maryan, V. Flavio, K. M. Taghi, S. Naeem, W. Randall and M. Edin, “Deep learning applications and challenges in big data analytics”, *Springer, Journal of Big Data,*vol. 2 (1), p.1, 2015.

[10] Sonia, P. Sandeep and L. Singh “A review on advances in deep learning”, IEEE*, In Computational Intelligence: Theories, Applications and Future Directions (WCI),* pp. 1-6, December 2015.

[11] D. Shanthi, G. Sahoo, and N. Saravanan. "Designing an artificial neural network model for the prediction of thrombo-embolic stroke." *International Journals of Biometric and Bioinformatics (IJBB), vol.* 3(1), pp. 10-18, 2009

[12] J. Premaladha and K. S. Ravichandran, “Novel approaches for diagnosing melanoma skin lesions through supervised and deep learning algorithms”.  *Springer, Journal of medical systems,* vol. 40 (4), pp.1-12, 12 February 2016.

[13] N. Yasuaki, M. Fujihika, M. Kazuo and N. Takashi, "Neural network analysis and evaluation of the fetal heart rate." *Algorithms,*vol. 2 (1), pp. 19-30, November 2016.

[14] K. Vanisree, S. Jyothi, “Decision Support System for Congenital Heart Disease Diagnosis based on Signs and Symptoms using Neural Networks”, *International Journal of Computer Applications*, vol. 19(6), April 2011.

[15] S. C. Dangare and S. S Apte, S.S. “A data mining approach for prediction of heart disease using neural networks”. *International Journal of Computer Engineering & Technology (IJCET)*, vol. 3(3), pp.30-40, 2012.

[16] D. Townsend and E. Monique "Complimentary artificial neural network approaches for prediction of events in the neonatal intensive care unit." *IEEE, 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2008.

[17] A. Nicholas, A. Babak and H. A. David “Deep learning and the prediction of human disease risk,” *X 23andMe,* https://github.com/Lasagne/Lasagne,2015*.*

[18] K. Wołk and K. Marasek. “Neural-based machine translation for medical text domain. Based on European Medicines Agency leaflet texts”. *In Procedia Computer Science,* *Elsevier,*vol. 64, pp. 2-9, 2015.

[19] P. Nhathai, D. Dejing, B. Piniewski and K. David, “A deep learning approach for human behavior prediction with explanations in health social networks: social restricted Boltzmann machine (SRBM),” *Springer*, vol. 6 (79), 13 September 2016.

[20] G. E. Hinton and R. R. Salakhutdinov. “Reducing the dimensionality of data with neural networks”. *Science*, vol. 313 (5786), pp.504-507, 2006.

[21] K. Byungseok and C. Hyunseung 2016. “A deep-learning-based emergency alert system”. Elsevier, *ICT Express,* vol. 2, pp. 67-70, 21 May- June 2016.

[22] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection”, Journal *of machine learning research*, pp.1157-1182, 3 March 2003.

[23] M. Tipping and C. Bishop, “Probabilistic Principal Component Analysis”, *Journal of the Royal Statistical Society, Series B*, *(Statistical Research)*, vol. 61(3), pp. 611-622, 1999.

[24] B. Schölkopf, A. Smola and K. R. Müller, “Kernel principal component analysis”. *In International Conference on Artificial Neural Networks, Springer,* pp. 583-588, 1997.

[25] S. Mika, B. Schölkopf, A. J. Smola, Müller, M. Scholz and G. Rätsch, “Kernel PCA and De-Noising in Feature Spaces”. *In NIPS* vol. 11, pp. 536-542, 1998.

[26] G. Xavier, and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks." *Aistats*. vol. 9, 2010

[27] E. Shenouda and A. M. Andrews. "A quantitative comparison of different MLP activation functions in classification." *In International Symposium on Neural Networks. Springer Berlin Heidelberg,* 2006.

[28] R. Fletcher and C. M. Reeves. “Function minimization by conjugate gradients. *The computer journal*,” vol. 7(2), pp.149-154, 1964.

[29] Sebastian Ruder. “An optimization of Gradient descent optimization algorithm. *Cornell University Library,* <https://arxiv.org/abs/1609.04747>

[30] Mahsereci, Maren, and Philipp Hennig. "Probabilistic line searches for stochastic optimization." Advances In Neural Information Processing Systems. 2015.

[31] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov 2014.” Dropout: a simple way to prevent neural networks from overfitting”, Journal *of Machine Learning Research*, vol. *15*(1), pp.1929-1958, 2014.

[32] Sutskever, Ilya. Training recurrent neural networks. Diss. University of Toronto, 2013.

[33] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, P. Vincent and S. Bengio, “Why does unsupervised pre-training help deep learning?”, Journal *of Machine Learning Research*, pp. 625-660. 2010.

[34] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion”, Journal *of Machine Learning Research*, pp. 3371-3408, 2011.