Comparative Study of Classification Techniques (SVM, Logistic Regression and Neural Networks) to Predict the Prevalence of Heart Disease

Divyansh Khanna, Rohan Sahu, Veeky Baths, and Bharat Deshpande

***Abstract*—This paper does a comparative study of commonly used machine learning algorithms in predicting the prevalence of heart diseases. It uses the publicly available Cleveland Dataset and models the classification techniques on it. It brings up the differences between different models and evaluates their accuracies in predicting a heart disease. We have shown that lesser complex models such as logistic regression and support vector machines with linear kernel give more accurate results than their more complex counterparts. We have used F1 score and ROC curves as evaluative measures. Through this effort, we aim to provide a benchmark and improve earlier ones in the field of heart disease diagnostics using machine learning classification techniques.**

***Index Terms*—Cleveland heart disease dataset, classification, svm, neural networks.**

I. Introduction

An important aspect of medical research is the prediction of various diseases and the analysis of factors that cause them. In this work, we focus on Heart Disease, specifically the University of California (UCI) heart disease dataset. Various researches have investigated this dataset for better prediction measures. Through our effort, we bring out a comparative understanding of different algorithms in estimating the heart disease accurately. Plan of this paper is as follows: Section II provides an insight into the dataset used. We follow that up with past research in this field under Section III. Section IV has an overview of the classification models implemented. It tries to give an understanding of the working of the models and what makes them so successful. Section V has the performance evaluation mechanisms employed frequently in this field of analysis. Section VI has our results and Section VII concludes the paper with a summary of findings and future research directions.

II. Dataset details

The dataset used in our study is the publicly available Cleveland Heart Disease Dataset from the UCI repository [1].

The UCI heart disease dataset consists of a total 76 attributes. However, majority of the existing studies have used the processed version of the data consisting of 303 instances with only 14 attributes.

Different datasets have been based on the UCI heart disease data. Computational intelligence researchers, however, have mainly used the Cleveland dataset consisting of 14 attributes.

The 14 attributes of the Cleveland dataset along with the values and data types are as follows:

- Age, in years
- Sex: male, female
- Chest Pain type (a) typical angina (angina), (b) atypical angina (abnang), (c) non-anginal pain (notang), (d) asymptomatic (asympt). These are denoted by numbers 1 to 4
- Trestbps: Patient's resting blood pressure in mm Hg at the time of admission to the hospital
- Chol: serum cholesterol in mg/dl
- Fbs: Boolean measure indicating whether fasting blood sugar is greater than 120 mg/dl: (1 = True; 0 = false)
- Restecg: electrocardiographic results during rest
- Thalach: maximum heart rate achieved
- Exang: Booelan measure indicating whether exercise inducing angina has occurred
- Oldpeak: ST depression induced by exercise relative to rest
- Slope: the slope of the ST segment for peak exercise
- Ca: number of major vessels (0 - 3) colour by fluoroscopy
- Thal: the heart status (normal, fixed defect, reversible defect)
- The class attributes: value is either healthy or heart disease (sick type: 1, 2, 3, and 4). But for our purposes, we indicated a heart disease by 1 and healthy by 0.

For purpose of this research, the multi-class classification problem is converted to binary classification problem. This facilitates better application of the models and also gives a better outlook to the overall problem statement at hand.

For this study, the data was split into two equal parts i.e., *training data* and *testing data*. The models were trained on one half and after selection of parameters through cross-validation; it was tested for accuracy on the test data. This is done to keep a sufficient amount of data from biasing the models and thus giving a completely fresh perspective for testing.

Manuscrupt received December 28, 2014; revised April 22, 2015.

Divyansh Khanna, Rohan Sahu, and Bharat Deshpande are with the Department of Computer Science, Birla Institute of Technology and Sciences, Pilani, Goa Campus, 403726 India (e-mail: divyanshkhanna09@gmail.com, rohan9605@gmail.com, bmd@goa.bits-pliani.ac.in).

Veeky Baths is with the Department of Biological Science, Birla Institute of Technology and Sciences, Pilani, Goa Campus, 403726 India (e-mail: veeky@goa.bits-pilani.ac.in).

III. Past Research

Over the past years, a lot of work and research has gone into better and accurate models for the Heart Disease Dataset. The work by Nahar J. *et al*. (2013) [2] gives a knowledge driven approach. Initially Logistic Regression was used by Dr. Robert Detrano to obtain 77% accuracy (Detrano, 1989 [3]). Newton Cheung utilized C4.5, Naive Bayes, BNND and BNNF algorithms and reached the classification accuracies of 81.11%, 81.48%, 81.11% and 80.96%, respectively (Cheung, 2001 [4]). Polat *et al*. proposed a method that uses artificial immune system (AIS) and obtained 84.5\% classification accuracy (Polat *et al*., 2005 [5]). More results were reported by using ToolDiag and WEKA tools. Our study has utilized Python and machine learning supporting libraries. In the case of medical data diagnosis, many researchers have used a 10-fold cross validation on the total data and reported the result for disease detection, while other researchers have not used this method for heart disease prediction. For our work, we have used both test-train split idea along with cross-validation for optimal parameters selection.

IV. Brief Introduction to Models Implemented

In this section, we give an understanding of the techniques we have used in this study. We discuss *Logistic Regression*, *Support Vector Machines* and *Neural Networks*. Along with each of them, provided are the implementation details of the models, such as the cross-validation, number of hidden units used, etc used for prediction of results. Throughout this section we try and maintain a balance between the intuitive understanding and the mathematical formulation, though the former overshadows the other in certain cases for better expression of ideas.

A. Logistic Regression

Logistic Regression is a standard classification technique based on the probabilistic statistics of the data. It is used to predict a binary response from a binary predictor. Let us assume our hypothesis is given by $h_\theta(x)$. We will choose:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \tag{1}$$

where is called the logistic function or the sigmoid function. Assuming all the training examples are generated independently, it is easier to maximize the log likelihood. Similar to the derivation in case of standard linear regression, we can use any gradient descent algorithm to achieve the optimal points. The updates will be given by $\theta := \theta - \alpha \Delta_\theta l(\theta)$, where l is the log likelihood function.

$$g(z) = \frac{1}{1+e^{-z}} \tag{2}$$

In our use of the logistic regression, we have used L-2 regularization along with 5 fold and 10 fold cross validation on the training dataset. LR model gives a good enough test data accuracy of 86%-88% and an impressive F1-score.

B. Support Vector Machines

Support Vector Machines, *SVMs*, are clearly one of the most popular and effective machine learning algorithms widely used in classification and recognition tasks in supervised learning. They have a very strong theoretical background that makes them indispensable in this field. The basic idea behind SVMs is as follows: there is some unknown and non-linear dependency (mapping, function) $y = f(x)$ between some high-dimensional input vector x and the vector output y.

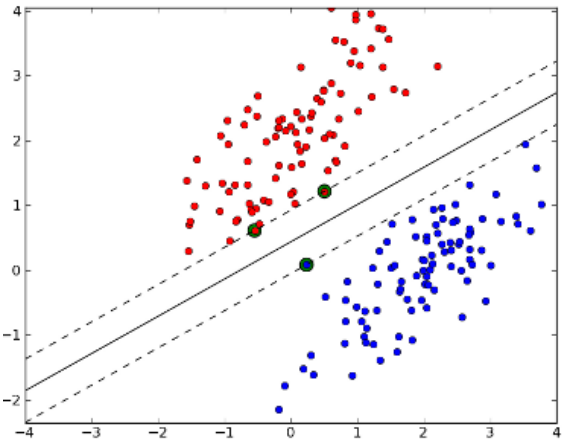

Fig. 1. Linear margin classifier.

There is no information about the under-lying joint distributions of the input data vectors. The only information available is the training data. Hence, making them a true member of the supervised learning algorithms class.

SVMs construct a hyperplane that separates two classes. Essentially, the algorithm tries to achieve maximum separation of the classes.

Fig. 1 shows a maximal classifier for a two dimensional data problems. The same can be achieved for any dimensional data.

The *support vectors* are those data points which fall on the boundary planes. As the name suggests, these vectors can be understood to be supporting the hyperplane in classifying the data according to the learned classifier. The following is the primal optimization problem for finding the optimal margin classifier:

$$\min_{\gamma,\omega,b} \frac{1}{2} \| \omega \|^2 \tag{3}$$

$$y^{(i)}(\omega^T x^{(i)} + b) \geq 1, i = 1...m$$

We can write the constraints as

$$g_i(\omega) = -y^{(i)}(\omega^T x^{(i)} + b) + 1 \leq 0 \tag{4}$$

We now have one such constraint for each training example. We construct the Langrangian for our optimization problem and take it up as a dual optimization problem and solve with KKT constraints. After solving

$$\begin{aligned} \omega^T x + b &= (\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)})^T x + b \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b \end{aligned} \tag{5}$$

If data is not linearly separable, as in Fig. 2, the function

$\phi(.)$ may be used to map each data point x_i in to a higher dimensional space, and then try to obtain a maximally separable hyperplane in that space as a classifier. Specifically, given a feature mapping ϕ, we define corresponding *kernel* to be

$$K(x,z) = \phi(x)^T \phi(z) \tag{6}$$

Now we could replace (x, z) everywhere in our algorithm by the kernel $K(x, z)$ in the algorithm. Now given a ϕ we can easily compute the kernel. But, because of the high dimensionality involved, it is computationally very expensive to calculate $\phi(x)$. The *kernel trick* issued for obtaining the dot products without explicitly mapping the data points into a higher dimensional space. This helps us evade the *curse of dimensionality* in a simple way.

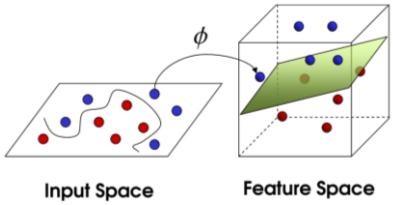

Fig. 2. Non-linear margin classifier.

In our implementation, we work with a grid search technique. Grid search trains an SVM with each pair (C, γ) in the Cartesian product (where C and γ is chosen from a manually specified dataset of hyper parameters) of these two sets and evaluates their performance on a held-out validation set. We search for optimal hyper-parameters of the model which give the least error and better approximation. We use three SVM kernels, *linear kernel*, *polynomial kernel* and a *radial basis function kernel*. The model for each kernel chooses a set of parameters from a given set and fits them using cross-validation. For different kernels, the results vary. Mostly, the accuracies on the test data vary in the range of 84%-87%.

C. Neural Networks

Neural Networks are an extremely popular set of algorithms used extensively in all sub-fields of Artificial Intelligence, concisely introduced in [6], and thoroughly explained in the [7]. The strength of a connection between neuron i and j is referred to as w_{ij}. Basically, a neural network consists of three sets, the visible set, V, the hidden set, H, and the output set of neurons O. The set V consists of neurons which receive the signals and pass onto the hidden neurons in set H.

In supervised learning, the training set consists of input patterns as well as their correct results in the form of precise activation of all output neurons. Each neuron accepts a weighted set of inputs and responds with an output, which is the weighted sum along with the bias processed by an activation function. The learning ability of a neural network depends on its architecture and applied algorithmic method during the training. Training procedure can be ceased if the difference between the network output and desired/actual output is less than a certain tolerance value. Thereafter, the network is ready to produce outputs based on the new input parameters that are not used during the learning procedure.

1) Single layer perceptron and back propagation

A single layer perceptron (*SLP*) is a feed-forward network having only one layer of variable weights and one layer of output neurons. Along with input layer is a bias neuron. For better performance, more than one trainable weight layers are used in the perceptron before the final output layers. A *SLP* is capable of representing only linearly separable data by straight lines. Whereas, the two-stage perceptron is capable of classifying convex polygons by further processing these straight lines.

An extremely important algorithm used to train multi-stage perceptron with semi-linear functions is the *Back-propagation of errors*. The idea behind the algorithm is as follows. Given a training example (x, y), we will first run forward pass to compute all the activations throughout the network. Then, for each node i and layer l, the error term is computed which measures how responsible that node is for any errors in the output. For an output node, the error is direct difference between the network's activation and the true target value which is given to us. But for the intermediate error terms $\delta_i^{(l)}$ of the hidden unit i in layer l, we use the weighted average of the error terms of the nodes that uses $a_i^{(l)}$ as an input.

We make use of a *momentum* value of 0.1. It specifies what fraction of the previous weight change is added to the new weight change. Momentum basically allows a change to the weights to persist for a number of adjustment cycles. The magnitude of the persistence is controlled by the momentum factor. In our implementation, we use 15 *sigmoid* hidden units for appropriate feature extraction. Also, the final result is calculated over 20 epochs (so that the weights get learned well enough for sensible prediction), with a *softmax* layer as the output layer of neurons. The model gives a test and train data accuracy in the range 83%-85%.

2) Radial basis function network

RBFN is an alternative to the more widely used MLP network and is less computer time consuming for network training. RBFN consists of three layers: an input layer, a hidden (kernel) layer, and an output layer. The nodes within each layer are fully connected to the previous layer as elaborated in [8] and Fig. 3.

The transfer functions of the hidden nodes are RBF. An RBF is symmetrical about a given mean or center point in a multi dimensional space. In the RBFN, a number of hidden nodes with RBF activation functions are connected in a feed forward parallel architecture. The parameters associated with the RBFs are optimized during the network training.

The RBF expansion for one hidden layer and a Gaussian RBF with centers u_i and width parameters σ_i is represented by

$$Y_k(K) = \sum_{i=1}^{H} W_{ki} \exp\left(-\frac{\| X - \mu_i \|^2}{\sigma_i^2}\right) \tag{7}$$

where H is the number of hidden neurons n the layer, W are the corresponding layer's weight and X is the input vector.

Estimating μ_i can be a challenge in using RBFNs. They can choose randomly or can be estimated using K-Means clustering. In our study, we use K-Means to find centroids, μ for the 15 RBF neurons by fitting in the training data. For σ, we take the standard deviations of the points in each cluster. This also goes by the intuition behind RBF activation. These

are then used in the RBF activations of the neural network.

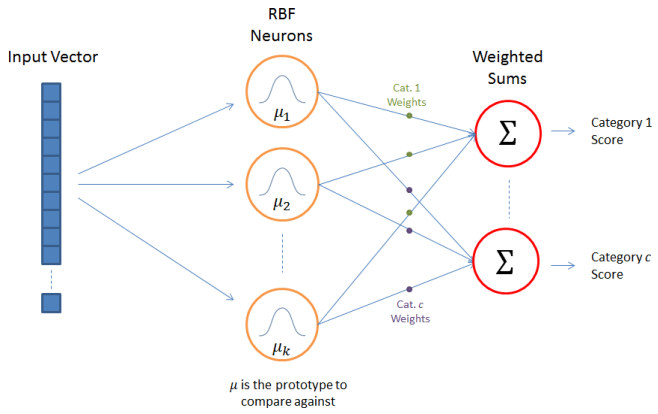

Fig. 3. Radial basis function network.

This model gives a test and train data accuracy in the range 78%-84%. The variation occurs because of the selection mechanism of the centers.

3) Generalized regression neural network

A GRNN (Specht 1991 [9]) is a variation of the radial basis neural networks, which is based on kernel regression networks. A GRNN does not require an iterative training procedure as back propagation networks. It approximates any arbitrary function between input and output vectors, drawing the function estimate directly from the training data. In addition, it is consistent that as the training set size becomes large, the estimation error approaches zero, with only mild restrictions on the function.

GRNN consists of four layers: input layer, pattern layer, summation layer and output layer as shown. The summation layer has two neurons, S and D summation neurons.

S summation neuron computes the sum of weighted responses of the pattern layer. On the other hand, D summation neuron is used to calculate un-weighted outputs of pattern neurons. The output layer merely divides the output of each S-summation neuron by that of each D-summation neuron, yielding the predicted value to an unknown input vector.

$$Y_i = \frac{\sum_{i=1}^{n} y_i . \exp - F(x, x_i)}{\sum_{i=1}^{n} \exp - F(x, x_i)} \quad (8)$$

$$F(x, x_i) = \exp(-D_i^2 / 2\sigma^2) \quad (9)$$

$$D_i^2 = (X - X_i)^T . (X - X_i) \quad (10)$$

F is the radial basis function between the x, the point of inquiry and x_i, the training samples which are used as the mean. The distance between the training sample and the point of prediction is used as a measure of how well each training sample can represent the point of prediction. As this distance becomes bigger, the $F(x, x_i)$ value becomes smaller and therefore the contribution of the other training samples to the prediction is relatively small.

The smoothness parameter is the only parameter of the procedure. For our study, this value was chosen to be 1.30.

The search for the smoothness parameter has to take several aspects into account depending on the application the predicted output is used for. We used the holdout method. In the holdout method, one sample of the entire set is removed and for a fixed σ GRNN is used again to predict this sample with the reduced set of training samples. The squared difference between the predicted value of the removed training sample and the training sample itself is then calculated and stored. The removing of samples and prediction of them again for this chosen σ is repeated for each sample-vector. After finishing this process the mean of the squared differences is calculated for each run. Then the process of reducing the set of training samples and predicting the value for these samples is repeated for many different values of σ. This way we get the most suitable σ with the least error. GRNN gives a test data accuracy of 89% of the dataset.

V. Performance Evaluation and Comparison

In this section, we go through the results and comparative measures implemented in the study. In all studies the comparison techniques play an important role. They define how different models are to be compared and thus whether the predicted results will be useful for further applications.

First, we start with the measures used followed by a discussion on our findings.

1) F1-Score: It is a very commonly used measure of a test's accuracy. It embodies both *precision* and *recall* of the test to compute the score. Precision is the number of true positives divided by the sum of true positives and false positives. Similarly, recall is the number of true positives divided by the sum of true positives and false negatives, which is the total number of elements that actually belong to the positive class. In binary classification, recall is also referred to as *sensitivity*. It isn't competent to measure just the recall (100% sensitivity can be achieved by predicting all the test cases to be positive), so it's usually combined together with precision in the form of F1 score. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. For our purpose, we use the balanced F1 score, which is the geometric mean of precision and recall.
2) ROC: ROC Curve, or *receiver operating characteristic* is a graphical plot that plots true positive rate (which is same as *sensitivity*) against the false positive rate (which is same as the complementary of *specificity*), at various threshold settings. Our concern is the Area under the ROC Curve (AUC). It tells how well the test can separate the group being tested into those with and without the disease in question. Recently, the correctness of ROC curves has been questioned because in some cases AUC can be quite noisy as a classification measure. Nevertheless, it gives a good enough result in our case. The more the area, the better it is.

In Table I and Table II, we can see the $F1$ scores of each of the models along with their respective accuracies. A similar intuition is reflected through the ROC Curves in Fig. 4 and Fig. 5. The RBF network doesn't fare as well as other networks.

TABLE I: LOGISTIC REGRESSION AND SVM RESULTS

MODEL	KERNEL	CV	ACCURACY (TEST DATA) %	ACCURACY (TRAIN DATA) %	F1 SCORE
LOGISTIC REGRESSION	NA	5 FOLD	86.8	84.8	0.87
LOGISTIC REGRESSION	NA	10 FOLD	88.2	82.8	0.88
SVM	LINEAR	5 FOLD	87.6	84.2	0.88
SVM	LINEAR	10 FOLD	87	82	0.87
SVM	RBF	5 FOLD	84.8	86	0.85
SVM	RBF	10 FOLD	84.8	86	0.85
SVM	POLY	5 FOLD	84.7	85.5	0.85
SVM	POLY	10 FOLD	84.7	85.5	0.85

TABLE II: NEURAL NETWORK RESULTS

MODEL	NO OF HIDDEN UNITS	PARAMETERS	ACCURACY (TEST DATA) %	ACCURACY (TRAIN DATA) %	F1-SCORE
BACK PROPAGATION	15	NA	83-85	83-85	0.86
RBF (K-MEANS)	15	NA	78-84	78-84	0.82
GRNN	NA	1.3	89	NA	0.88

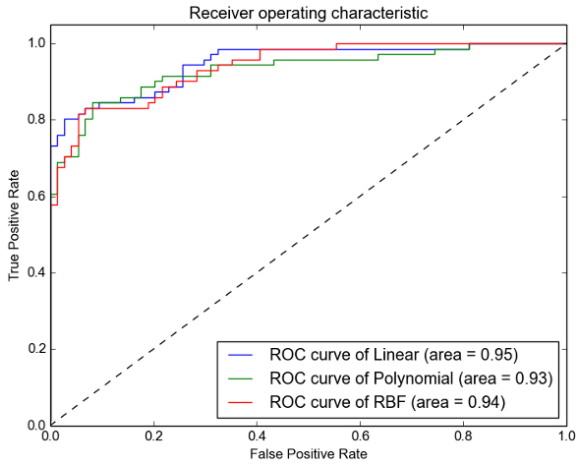

Fig. 4. ROC curve for gridsearch (SVM).

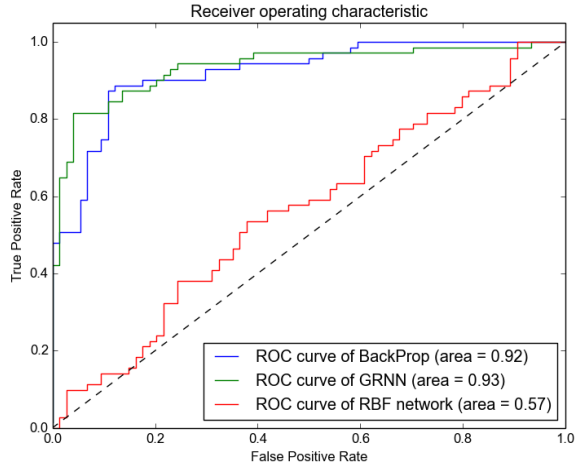

Fig. 5. ROC curve for neural networks.

VI. RESULTS AND DISCUSSION

A common approach to reporting performance of classifiers is by performing a 10-fold cross validation on a provided dataset and report performance results on the given dataset. However, this method is expected to be biased to the training data and may not reflect the expected performance when applied on real-life data. So, in addition to generally used 10-fold cross validation, we have also performed a train-test split on the dataset and then used a 10-fold cross validation to select the best parameter for training. Performance results were presented based on the prediction outcomes of the test set.

As evident from the results from the adjoining tables and plots, Logistic Regression and the SVM approach give better performance, particularly with linear kernel. Among neural networks, the GRNN method stands out while the RBF NN doesn't prove to be very useful.

Also, the number of hidden units plays a small role in defining the shape of the predictions. For network with a very large number of hidden neurons, for e.g. 150, the training accuracy increases by a significant margin but suffers a minor fall on the F1 Score. The ROC of RBF network also signifies that for a given set of parameters the area under the curve is comparatively low as compared to other network classification techniques.

VII. CONCLUSIONS

This study provides a benchmark to the present research in the field of heart disease prediction. The dataset used is the Cleveland Heart Disease Dataset, which is to an extent curated, but is a valid standard for research. This paper has provided details on the comparison of classifiers for the detection of heart disease. We have implemented *logistic regression*, *support vector machines* and *neural networks* for classification. The results suggest SVM methodologies as a very good technique for accurate prediction of heart disease, especially considering classification accuracy as a performance measure. Generalized Regression Neural Network gives remarkable results, considering its novelty and unorthodox approach as compared to classical models.

Overall for the heart disease dataset, simpler methods like logistic regression and SVM with linear kernel prove to be more impressive. This study can be further extended by utilizing these results in making technologies for accurate prediction of heart disease in hospitals. It can enhance the capabilities of traditional methods and reduce the human error, thereby making a contribution to the science of medical diagnosis and analysis.

ACKNOWLEDGEMENTS

We would like to sincerely thank the Cardiology

Department at Goa Medical College for helping us understand the medical details of this project. Their support also helped us validate the 14 attributes chosen for this project. We would also like to thank the Dean of Goa Medical College for allowing us to interact with doctors at GMC.

REFERENCES

[1] Uci. 2010. V. A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D. [Online]. Available: http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/heart-disease.names

[2] J. Nahar, "Computational intelligence for heart disease diagnosis: A medical knowledge driven approach," *Expert Systems with Applications*, pp. 96-104, 2013.

[3] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu *et al*., "International application of a new probability algorithm for the diagnosis of coronary artery disease," *American Journal of Cardiology*, vol. 6, pp. 304-310, 1989.

[4] N. Cheung, "Machine learning techniques for medical analysis," B.Sc. Thesis, School of Information Technology and Electrical Engineering, University of Queenland, 2001.

[5] K. Polat, S. Sahan, H. Kodaz, and S. Gunes, "A new classification method to diagnosis heart disease: Supervised artificial immune system (AIRS)," in *Proc. the Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN)*, 2005.

[6] D. Kriesel. A Brief Introduction to Neural Networks. [Online]. Available: http://www.dkriesel.com/en/science/neural_networks

[7] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of Statistical Learning*, Springer-Verlag, New York, 2001.

[9] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Network*s. vol. 2, no. 6, 1991.

[10] Y. S. Abu-Mostafa. Learning with data. [Online]. Available: http://amlbook.com/support.html

[11] V. N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, Inc., New York, 1998.

[12] S. A. Hannan, R. R. Manza, and R. J. Ramteke, "Generalized regression neural network and radial basis function for heart disease diagnosis," *International Journal of Computer Applications*, vol. 7, no. 13, 2010.

Divyansh Khanna is a student completing his B.E. (hons) in computer science and M.Sc. (hons) in mathematics from BITS Pilani KK Birla Goa Campus. He did his schooling from New Delhi. His research interests include machine learning, neural networks and parallel computing. He has worked as a summer technology intern in a tech firm in Hyderabad and will be working towards his thesis. His current projects include porting popular machine learning algorithms to a parallelized implementation to increase speedup. He was awarded the Inspire Scholarship for meritorious performance in national level board examination.

Rohan Sahu is a computer science graduate from BITS-Pilani, Goa. He has worked at Oracle Bangalore in the field of database development. He has spent the last year working on machine learning and data science projects that includes a research stint at BITS-Pilani, Goa, where he worked on applying machine learning to the field of healthcare. Rohan is currently working at a digital health firm in Gurgaon, India where he handles analytics and pre-sales roles.

Veeky Baths is an associate professor in BITS Pilani Goa. Veeky's research areas and core competencies are in the field of systems biology, machine learning, biological sequence analysis and metabolic network analysis. Veeky is applying graph theory and computational approach to understand the integrity and robustness of metabolic network, which will be a great help for knowledge based drug discovery. When complex biological networks are represented as a graph then it becomes amenable to various types of mathematical analysis and there is a reduction in the complexity. Graphs or networks are abstract representation of more complex biological process. Veeky joined the Department of Biological Sciences in 2005. He obtained his B.Sc degree from Ravenshaw University and M.Sc. in bioinformatics from the Orissa University of Agriculture and Technology. He completed his Ph.D. degree in science from Bits Pilani K.K.Birla Goa Campus in 2011. He then obtained an MBA from Goa Institute of Management.

Bharat Deshpande received his Ph.D. degree from IIT Bombay in 1998. Then he received postdoctoral fellowship from Department of Atomic Energy. Dr. Deshpande joined BITS Pilani in 2001 and moved to BITS Pilani, Goa Campus in 2005. Since 2006 he has been heading the Department of Computer Science & Information Systems at Goa. Apart from basic courses in computer science, he has taught specialized courses like algorithms, theory of computation, parallel computing, artificial intelligence & few more. His research interests are in areas of complexity theory, parallel algorithms, and data mining. Over some years he has supervised numerous masters and doctoral students. He has many national and international publications to his credit. He is also on the board of studies of University of Goa and College of Engineering, Pune. Dr. Deshpande was also the vice president of the Goa Chapter of Computer Society of India and is currently vice president of the ACM Professional Goa Chapter.