Implementing binary classifiers based on Neural Networks and Nearest Neighbor methods.

CS 260A - Machine Learning Algorithms
Final Project Report[1]

Amogh Param (704434779)
University of California, Los Angeles
aparam@cs.ucla.edu

1. Introduction

Machine Learning is a discipline that explores algorithms and models that learn from data. Primary tasks in Machine Learning involve classification and clustering of data. In this project, we explore two classes of well-known Machine Learning algorithms/methods for binary classification of patient data based on diastolic and systolic blood pressure measurements. We explore classification using Neural Networks and Nearest Neighbor methods.

1.1 Classification

Classification is the problem of identifying which class (among a set of classes or categories) a new observation of some type of data belongs to. This process of classification is done on the basis of training a model/algorithm on a set of data containing observations whose classification labels are known. An example would be labeling a patient as "healthy" or "unhealthy" as described by observed characteristics of the patient (diastolic blood pressure, systolic blood pressure etc.). In contrast to regression, which involves predicting a continuous valued output, classification problems predict a discrete valued output label for the data being classified.

1.2 Neural Networks

Neural Networks are a class of algorithms/models that can be used for classification problems. These kinds of networks are found to work well for classification problems. The primary idea behind these class of algorithms is that since humans are able to perform classification tasks well, building a model based on the way human minds work, specifically how the neurons in the brain work, provides a good way to solve classification problems. Similar to human brains, neural networks are composed of layers of neurons that receive input from either input sensors or from the output of layers of other neurons. Neurons either get activated or not, based on the combination of inputs exceeding some threshold. The fundamental way neural networks “learn” to classify is by strengthening connections between neurons of different layers.

The neural network first tries to predict the output for some representation of an instance/observation of the data. It then computes the error between the predicted output and the target output and then propagates this error back through the network to update the strength/weight between connections. As the network sees more observations to learn from, the strength/weight updates between connections converges and reaches a state where updates are either too small or there are no updates at all. Once the network reaches this state, it is said to have learned the classification. Essentially, that means that the network has learned to approximate a function that separates the data.

1.3 K - Nearest Neighbors

The nearest neighbor algorithm is another such classification algorithm. Unlike neural networks, the nearest neighbor algorithms are a class of lazy learning algorithms, as the function required for the separation of data is only approximated locally and all computation is deferred until classification. These algorithms work under the principle of finding a finite number of neighbors close to the point being considered for classification, checking the majority class label of these neighbors and classifying the point being considered to that of the majority. This process is done for all the points in the testing set.

1. **Github source**: https://github.com/jazzTheJackRabbit/machineLearningAlgorithms

2. Binary Classification of Patients based on Time Series Signals

In this project, we deal with the problem of classifying patients as 'Healthy' or 'Unhealthy' based on two time series signals for each patient. The signals that we deal with are that of diastolic and systolic blood pressure measurements for each patient over a course of 26 days. The primary task here is to find a representation for patients such that it helps us classify them. This task involves finding a unique set of features that provides a representation for each patient in a vector space and helps group similar patients together. The features selected for this project are discussed below.

2.1 Feature Selection

Since the project deals with time series data, where each patient is represented by his/her blood pressure measurements over a course of 26 days, we need to compress this data into an n-dimensional vector representation that adequately represents the patient's measurements. Therefore, we explore a number of statistical measurements as feature representations for each patient. The primary features considered in this project are as follows:

- `2D vector: [Mean of Diastolic Measurements, Mean of Systolic Measurements]`
- `2D vector: [Variance of Diastolic Measurements, Variance of Systolic Measurements]`
- `2D vector: [Difference of Maximum and Minimum values of Diastolic Measurements, Difference of Maximum and Minimum values of Systolic Measurements]`
- `2D vector: [Skewness of Diastolic Measurements, Skewness of Systolic Measurements]`
- `2D vector: [Kurtosis of Diastolic Measurements, Kurtosis of Systolic Measurements]`
- `2D vector: [Pearson's Correlation Coefficient between Diastolic and Systolic Measurements, Product of Standard deviations of Diastolic and Systolic Measurements]` **(Unique Contribution)**

2.2 Unique Contribution

In this project, a unique feature that I thought of using was to use the correlation between each patient's diastolic and systolic blood pressure levels. An assumption I made here was that different types of patients would have different types of correlation between the two blood pressure measurements. It turns out that using this measurement along with the product of standard deviation of the diastolic measurement and that of the systolic measurement, as a feature provided good results.

For the K-NN algorithm, I was able to achieve an accuracy rating of about 79.5% and a precision rating of 87% with the old dataset with k=3. With the new dataset, I was able to achieve an accuracy rating of about 74% and a precision rating of 92%, with k=9.

For the neural network, the best accuracy rating achieved was about 72% and a precision rating of 70% for the old dataset. For the new data, the best accuracy rating using this feature was about 64% with a precision rating of about 65%.

3. Source Code Description

This section provides a description of all the project source files.

`Data preparation.py`

`'DataPreparation.py'` is a class that extracts and structures the given datasets in a format that can be easily handled by the binary classifiers. Additionally, this class is responsible for creating `'Patient'` objects (described below), assigning them their classification labels and for also calling the feature computation methods of the `'Patient'` instances.

`Patient.py`

`'Patient.py'` defines the class for the patient data provided in the dataset. Instantiating this class is equivalent to creating an object record for a patient and assigning each of the instance variables to measurements, features and labels that uniquely identify that patient. This class also has methods to compute features like mean, variance, skewness, kurtosis, etc. for each patient.

`Classifier.py`

`'Classifier.py'` is the base class for the 2 binary classifiers (Neural Network and Nearest Neighbor). It provides methods for the required performance evaluation like, computing the classification confusion matrix (TP, FP, TN, FN), accuracy, sensitivity, specificity, precision, recall and f-measure.

`NeuralNetwork.py`

`'NeuralNetwork.py'` is a class that inherits from `'Classifier.py'` and therefore has all the methods to evaluate its own performance for the given dataset. The Neural Network class is designed to have one input layer, one hidden layer and one output layer, but the number of neurons/nodes in each layer can be changed as required. The input layer includes a bias node. A sigmoid activation function is used for the activations in each layer. Specifically, the hyperbolic tangent function is used. I previously tried experimenting with the exponential sigmoid function but I ran into many 'overflow' problems. The hyperbolic tangent function provides an output in the range of -1 to 1; therefore the outputs are scaled to be in the range 0 to 1 (since the problem we're interested in is binary classification). The weights for the neural networks are initialized to random values before every training phase. One of the primary methods in this class is the `'backPropagation'` method, which calculates the error between the actual output and the predicted output and updates the weights between nodes of different layers (back propagates the error). A training method predicts the output for every input set and updates weights based on the back-propagated error. Finally a classify method computes the class label that the input set should be classified into. By changing the parameters in the `'neuralNetworkConfigs'` dictionary, we can test the neural network under many different configurations and find the best performing configuration.

`NearestNeighbor.py`

Similar to the neural network described above, `'NearestNeighbor.py'` inherits from `'Classifier.py'`. In addition to the evaluation methods, this class provides methods to compute the distance between a reference data point (feature vector) and all the other training data points. The `'findKNearestNeighbors'` method looks at all the training data points and finds the top k nearest data points with respect to the reference point from the testing set that needs to be classified. The `'computeClass'` method then looks at the majority class label of the neighbors and assigns that as the class label of the reference data point. This process of classification is done for all the reference data points in the testing set and the final classification is achieved and evaluated.

4. Conclusion

Having implemented the two machine learning algorithms, we can conclude that in addition to the efficiency of implementation, the most important aspect of classifying data with good performance would be to choose the right set of features. The following sections consist of results obtained for various runs and configurations of the two algorithms. It is clearly evident that some features, like the one that includes the Pearson's Correlation Coefficient, do much better in comparison to features like the mean and variance. Therefore choosing the right feature is quintessential to classification problems as it can play a major role between a good classification and a bad classification.

5. Results

1. Algorithm: K-Nearest Neighbor
Dataset: Old

K - Nearest Neighbor (Old Dataset)				
Features			Actual Classification	
			1	0
Mean of Diastolic, Mean of Systolic	Predicted Classification	1	TP = 9.0	FP = 2.0
		0	FN = 10.0	TN = 18.0
Variance of Diastolic, Variance of Systolic	Predicted Classification	1	TP = 8.0	FP = 3.0
		0	FN = 11.0	TN = 17.0
Max - Min of Diastolic, Max - Min of Systolic	Predicted Classification	1	TP = 17.0	FP = 12.0
		0	FN = 2.0	TN = 8.0
Skewness of Diastolic, Skewness of Systolic	Predicted Classification	1	TP = 10.0	FP = 2.0
		0	FN = 9.0	TN = 18.0
Kurtosis of Diastolic, Kurtosis of Systolic	Predicted Classification	1	TP = 7.0	FP = 2.0
		0	FN = 12.0	TN = 18.0
Pearson Correlation Coefficient, SDD*SDS	Predicted Classification	1	TP = 13.0	FP = 2.0
		0	FN = 6.0	TN = 18.0

Figure 1.1 Confusion matrix for K-Nearest Neighbors, for the old dataset.

K - Nearest Neighbors (Old Dataset)						
Features	Accuracy	Sensitivity	Specificity	Precision	Recall	F-measure
Mean of Diastolic, Mean of Systolic	69.23%	47.36%	90%	81.81%	47.37%	60%
Number of Neighbors = 4						
Variance of Diastolic, Variance of Systolic	64.10%	42.10%	85.00%	72.72%	42.10%	53.30%
Number of Neighbors = 6						
Max - Min of Diastolic, Max - Min of Systolic	64.10%	89.47%	40.00%	58.62%	89.47%	70.83%
Number of Neighbors = 2						
Skewness of Diastolic, Skewness of Systolic	71.79%	52.63%	90.00%	83.33%	52.63%	64.51%
Number of Neighbors = 9						
Kurtosis of Diastolic, Kurtosis of Systolic	64.10%	36.80%	90.00%	77.77%	36.84%	50.00%
Number of Neighbors = 2						
Pearson Correlation Coefficient, SDD*SDS	79.48%	68.42%	90.00%	86.66%	68.42%	76.47%
Number of Neighbors = 3						

Figure 1.2 Performance measures for K-Nearest Neighbors, for the old dataset.

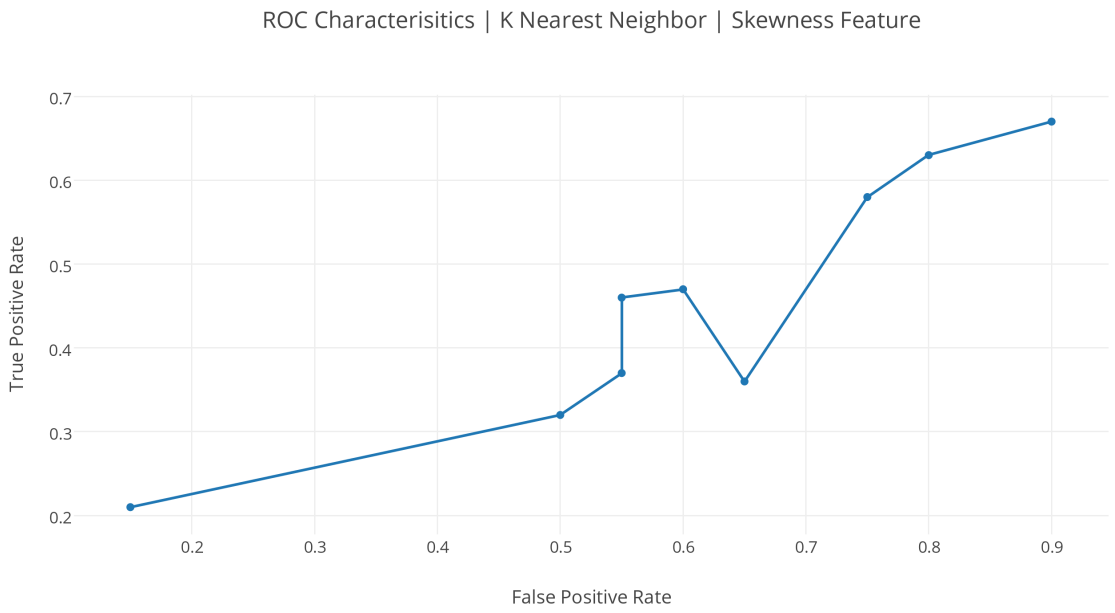

Figure 1.3 ROC Characteristics for K-Nearest Neighbors, for the old dataset. Runs using the Skewness feature and increasing values of ***k***.

0.21	0.15
0.32	0.50
0.37	0.55
0.46	0.55
0.47	0.60
0.36	0.65
0.58	0.75
0.63	0.80
0.67	0.90

Figure 1.4 Tabulation of runs using the Skewness feature in increasing values of ***k***, sorted by False Positive Rate.

2. Algorithm: K-Nearest Neighbor
Dataset: New

K - Nearest Neighbor (New Dataset)				
Features			Actual Classification	
			1	0
Mean of Diastolic, Mean of Systolic	Predicted Classification	1	TP = 17.0	FP = 9.0
		0	FN = 3.0	TN =10.0
Variance of Diastolic, Variance of Systolic	Predicted Classification	1	TP = 13.0	FP =9.0
		0	FN = 7.0	TN = 10.0
Max - Min of Diastolic, Max - Min of Systolic	Predicted Classification	1	TP = 18.0	FP =12.0
		0	FN = 2.0	TN = 7.0
Skewness of Diastolic, Skewness of Systolic	Predicted Classification	1	TP = 17.0	FP = 12.0
		0	FN = 3.0	TN = 7.0
Kurtosis of Diastolic, Kurtosis of Systolic	Predicted Classification	1	TP = 12.0	FP = 7.0
		0	FN = 8.0	TN = 12.0
Pearson Correlation Coefficient, SDD*SDS	Predicted Classification	1	TP = 11.0	FP = 1.0
		0	FN = 9.0	TN = 18.0

Figure 2.1 Confusion matrix for K-Nearest Neighbors, for the new dataset.

K - Nearest Neighbors (New Dataset)						
Features	Accuracy	Sensitivity	Specificity	Precision	Recall	F-measure
Mean of Diastolic, Mean of Systolic	69.23%	85.00%	53%	65.38%	85.00%	74%
Number of Neighbors = 4						
Variance of Diastolic, Variance of Systolic	58.97%	65.00%	52.63%	59.09%	65.00%	61.90%
Number of Neighbors = 8						
Max - Min of Diastolic, Max - Min of Systolic	64.10%	90.00%	36.84%	60.00%	90.00%	72.00%
Number of Neighbors = 7						
Skewness of Diastolic, Skewness of Systolic	61.53%	85.00%	36.84%	58.62%	85.00%	69.39%
Number of Neighbors = 8						
Kurtosis of Diastolic, Kurtosis of Systolic	61.53%	60.00%	63.15%	63.15%	60.00%	61.53%
Number of Neighbors = 5						
Pearson Correlation Coefficient, SDD*SDS	74.35%	55.00%	94.73%	91.66%	55.00%	68.75%
Number of Neighbors = 9						

Figure 2.2 Performance measures for K-Nearest Neighbors, for the new dataset.

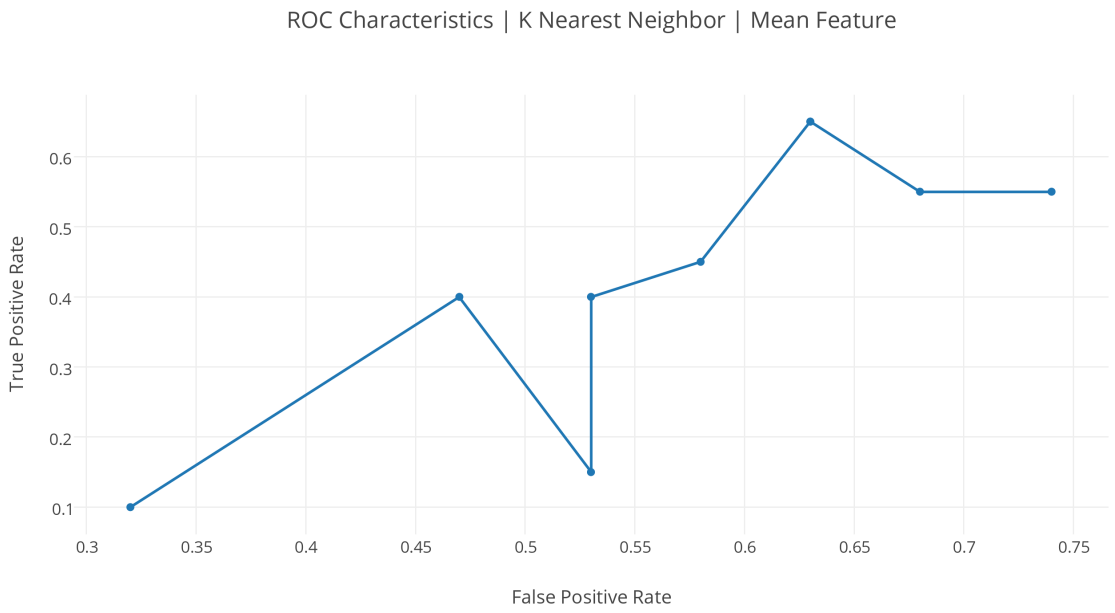

Figure 2.3 ROC Characteristics for K-Nearest Neighbors, for the new dataset. Runs using the Mean feature and increasing values of ***k***.

0.10	0.32
0.40	0.47
0.15	0.53
0.40	0.53
0.45	0.58
0.65	0.63
0.65	0.63
0.55	0.68
0.55	0.74

Figure 2.4 Tabulation of runs using the Mean feature in increasing values of ***k***, sorted by False Positive Rate.

3. Algorithm: Neural Network
Dataset: Old

Neural Network (Old Dataset)				
Features			Actual Classification	
			1	0
Mean of Diastolic, Mean of Systolic	Predicted Classification	1	TP = 13.0	FP = 8.0
		0	FN = 6.0	TN = 12.0
Variance of Diastolic, Variance of Systolic	Predicted Classification	1	TP = 12.0	FP = 9.0
		0	FN = 7.0	TN = 11.0
Max - Min of Diastolic, Max - Min of Systolic	Predicted Classification	1	TP = 10.0	FP = 6.0
		0	FN = 9.0	TN = 14.0
Skewness of Diastolic, Skewness of Systolic	Predicted Classification	1	TP = 9.0	FP =7.0
		0	FN = 10.0	TN = 13.0
Kurtosis of Diastolic, Kurtosis of Systolic	Predicted Classification	1	TP = 13.0	FP = 5.0
		0	FN = 6.0	TN = 15.0
Pearson Correlation Coefficient, SDD*SDS	Predicted Classification	1	TP = 14.0	FP = 6.0
		0	FN = 5.0	TN = 14.0

Figure 3.1 Confusion matrix for Neural Network, for the old dataset.

Neural Network (Old Dataset)						
Features	Accuracy	Sensitivity	Specificity	Precision	Recall	F-measure
Mean of Diastolic, Mean of Systolic	64.10%	68.42%	60%	61.90%	68.42%	65%
Neural Network configuration: iterations=400 | LearningRate=0.3 | Momentum:0.5						
Variance of Diastolic, Variance of Systolic	58.97%	63.16%	55.00%	57.14%	63.16%	60.00%
Neural Network configuration: iterations=300 | LearningRate=0.1 | Momentum:0.3						
Max - Min of Diastolic, Max - Min of Systolic	61.53%	52.63%	70.00%	62.50%	52.63%	57.14%
Neural Network configuration: iterations=300 | LearningRate=0.05 | Momentum:0.3						
Skewness of Diastolic, Skewness of Systolic	56.41%	47.36%	65.00%	56.25%	47.36%	51.42%
Neural Network configuration: iterations=300 | LearningRate=0.05 | Momentum:0.3						
Kurtosis of Diastolic, Kurtosis of Systolic	71.79%	68.42%	75.00%	72.22%	68.42%	70.27%
Neural Network configuration: iterations=500 | LearningRate=0.3 | Momentum:0.5						
Pearson Correlation Coefficient, SDD*SDS	71.79%	73.68%	70.00%	70.00%	73.68%	71.79%
Neural Network configuration: iterations=300 | LearningRate=0.3 | Momentum:0.5						

Figure 3.2 Performance measures for Neural Network, for the old dataset.

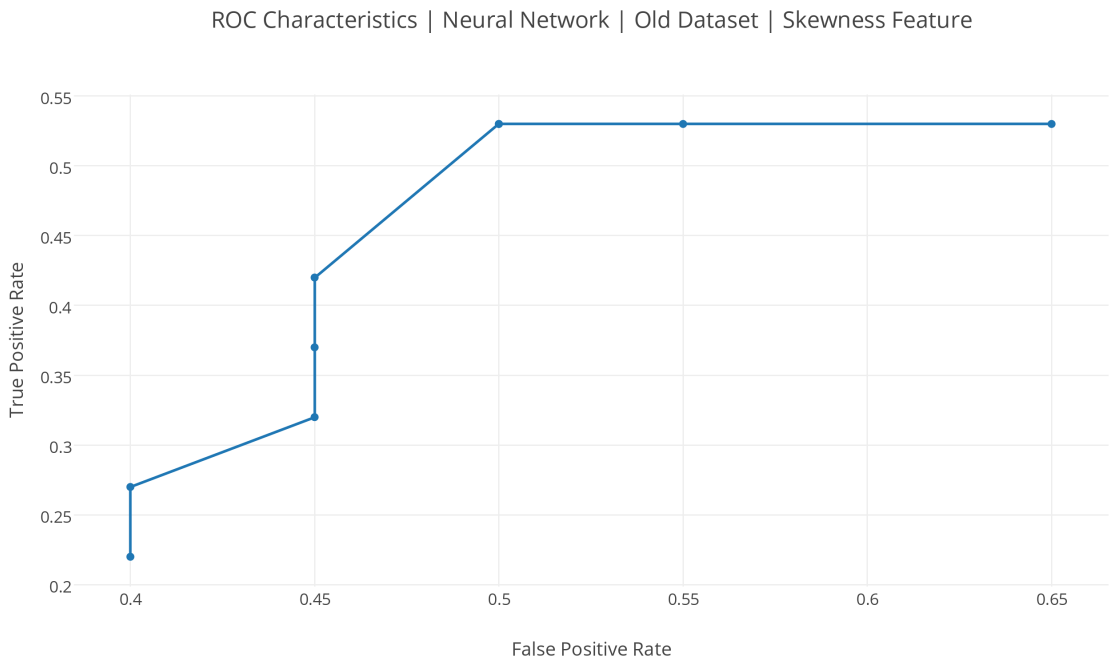

Figure 3.3 ROC Characteristics for Neural Networks, for the old dataset. Runs using the Skewness feature and different configurations of iterations, learning rate and momentum.

0.22	0.40
0.27	0.40
0.32	0.45
0.37	0.45
0.42	0.45
0.53	0.50
0.53	0.50
0.53	0.55
0.53	0.65

Figure 3.4 Tabulation of runs using the Skewness feature for different configurations of iterations, learning rate and momentum, sorted by False Positive Rate.

4. Algorithm: Neural Network
Dataset: New

Neural Network (New Dataset)				
Features			Actual Classification	
			1	0
Mean of Diastolic, Mean of Systolic	Predicted Classification	1	TP = 9.0	FP =5.0
		0	FN = 11.0	TN = 14.0
Variance of Diastolic, Variance of Systolic	Predicted Classification	1	TP = 14.0	FP = 6.0
		0	FN = 6.0	TN = 13.0
Max - Min of Diastolic, Max - Min of Systolic	Predicted Classification	1	TP = 13.0	FP = 8.0
		0	FN = 7.0	TN = 11.0
Skewness of Diastolic, Skewness of Systolic	Predicted Classification	1	TP = 12.0	FP =8.0
		0	FN = 8.0	TN = 11.0
Kurtosis of Diastolic, Kurtosis of Systolic	Predicted Classification	1	TP = 13.0	FP = 8.0
		0	FN = 7.0	TN = 11.0
Pearson Correlation Coefficient, SDD*SDS	Predicted Classification	1	TP = 13.0	FP = 7.0
		0	FN = 7.0	TN = 12.0

Figure 4.1 Confusion matrix for Neural Network, for the new dataset.

Neural Network (New Dataset)						
Features	Accuracy	Sensitivity	Specificity	Precision	Recall	F-measure
Mean of Diastolic, Mean of Systolic	58.97%	45.00%	74%	64.28%	45.00%	53%
Neural Network configuration: iterations=300 | LearningRate=0.1 | Momentum:0.3						
Variance of Diastolic, Variance of Systolic	69.23%	70.00%	68.42%	70.00%	70.00%	70.00%
Neural Network configuration: iterations=300 | LearningRate=0.05 | Momentum:0.3						
Max - Min of Diastolic, Max - Min of Systolic	61.53%	65.00%	57.89%	61.90%	65.00%	53.41%
Neural Network configuration: iterations=400 | LearningRate=0.3 | Momentum:0.5						
Skewness of Diastolic, Skewness of Systolic	58.97%	60.00%	57.89%	60.00%	60.00%	60.00%
Neural Network configuration: iterations=500 | LearningRate=0.1 | Momentum:0.3						
Kurtosis of Diastolic, Kurtosis of Systolic	61.53%	65.00%	57.89%	61.90%	65.00%	63.41%
Neural Network configuration: iterations=400 | LearningRate=0.3 | Momentum:0.5						
Pearson Correlation Coefficient, SDD*SDS	64.10%	65.00%	63.15%	65.00%	65.00%	65.00%
Neural Network configuration: iterations=300 | LearningRate=0.05 | Momentum:0.3						

Figure 4.2 Performance measures for Neural Network, for the new dataset.

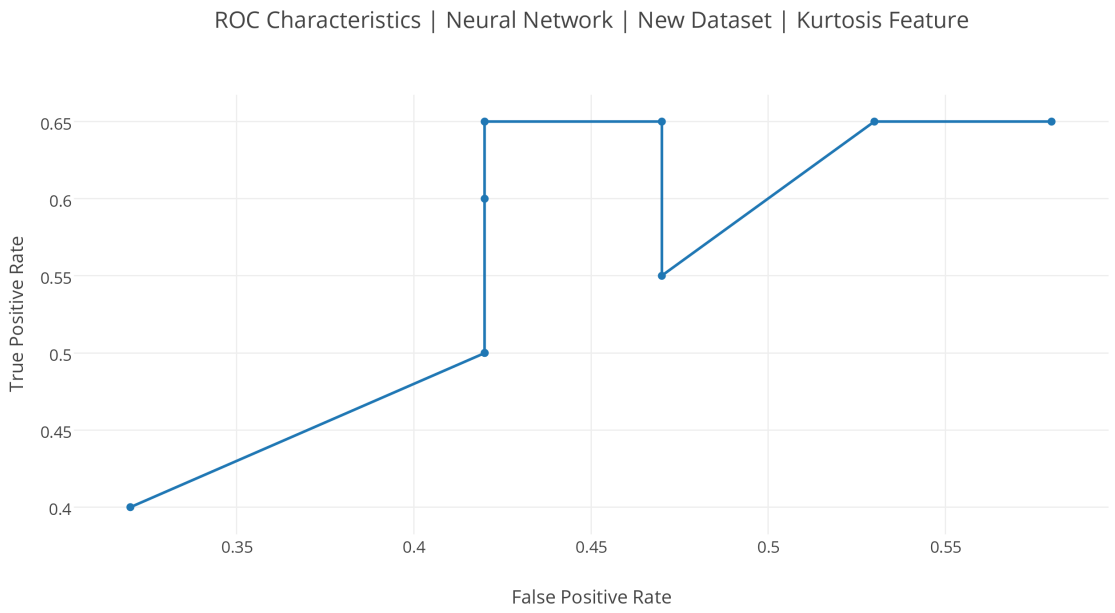

Figure 4.3 ROC Characteristics for Neural Networks, for the new dataset. Runs using the Kurtosis feature and different configurations of iterations, learning rate and momentum.

0.40	0.32
0.50	0.42
0.50	0.42
0.60	0.42
0.65	0.42
0.65	0.47
0.55	0.47
0.65	0.53
0.65	0.58

Figure 4.4 Tabulation of runs using the Kurtosis feature for different configurations of iterations, learning rate and momentum, sorted by False Positive Rate.