Dissertation Report

On

Analysis of Some Computational Intelligence Approaches for Software Reliability Prediction

Submitted by

Dola Spandana

Enrollment No: 16535010

Under the guidance of

Dr. Sandeep Kumar

Assistant professor

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

Roorkee – 247667

May, 2018

ABSTRACT:

Software engineering is partial without Software reliability prediction. For characterizing any software product quality quantitatively during phase of testing, the most important factor is software reliability assessment. Traditional models are mainly based on assumptions and approximations. But it is needed for developing such a single model which can be applicable for a relatively better prediction in all conditions and situations. For this the Neural Network (NN) model approach is introduced. In this thesis report the applicability of the models based on NN for better reliability prediction in a real environment is described and a method of assessment of growth of software reliability using NN model is presented. Mainly three types of NNs are used here. One is feed forward neural network, second is generalized regression neural network and third is radial basis function network. For modeling FFNN, back propagation learning algorithm is implemented and the related network architecture issues, data representation methods and some unreal assumptions associated with software reliability models are discussed. Different datasets containing software failures are applied to the proposed models. These datasets are obtained from several software projects. Then it is observed that the results obtained indicate a significant improvement in performance by using neural network models over conventional statistical models based on non homogeneous Poisson process.

AUTHOR’S DECLARATION

I, hereby declare that the work presented in this dissertation ***Analysis of Some Computational Intelligence Approaches For Software Reliability Prediction*** towards the fulfillment of the requirements for the award of the degree of ***Master of Technology in Computer Science and Engineering***, submitted to the Department of Computer Science and Engineering, ***Indian Institute of Technology Roorkee,*** *India*, is an authentic record of my own work carried out during May 2017 to May 2018, under the guidance of ***Dr. Sandeep Kumar, Assistant Professor,*** Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, India.

The content presented in this dissertation has not been submitted by me for the award of any other degree of this or any other institute.

Date:
Place:

Dola Spandana
M.Tech, C.S.E.,
Indian Institute of Technology, Roorkee

CERTIFICATE

This is to certify that the statement made by the author in the above declaration is correct to the best of my knowledge and belief.

Date:
Place:

Dr. Sandeep Kumar
Assistant Professor, C.S.E.,
Indian Institute of Technology, Roorkee

Acknowledgements

I would first like to thank my supervisor Dr. Sandeep Kumar, Assistant Professor, Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, for his thoughtful encouragement and careful supervision during my research work. His enthusiasm for Software Engineering kept me constantly engaged with my research work and his personal generosity helped make my time at I.I.T. Roorkee enjoyable.

I sincerely thank my family and friends for directly or indirectly helping me and giving me moral support that motivated me during the course of the work. I am also thankful to all the staff members of the Department of Computer Science and Engineering for their support.

Table of Contents:

List of Figures:

List of Tables

CHAPTER 1

INTRODUCTION

Many of our daily activities are done with the help of computers and embedded systems. These systems are administered by its own built in software. Thus we need our software to be reliable. The more reliable software provides more robust computer systems. Thus we need to predict the reliability of various software before it is ready to ship. [1]

The productiveness of a system is recognized to convey acceptability of that system for achievement of many deliberated tasks and productivity of using it. The suitability of executing intended tasks is firstly resolved by *reliability* and *quality* of many systems.

1.1 Reliability

The probability of a unit in a provided time period with the absence of failures is termed as reliability. Most accurate definition is: "Reliability of a unit or section is the probability that the unit perform its deliberate function appropriately for a particular period of time under stated operating conditions or environment." Through a single section it can signify an element, a system or a part of that system. Reliability definition majorly focuses on four elements:

1. Probability
2. Definite functions
3. Period, and
4. Manageable conditions

If T defines time until the specified part has occurrence of failure, then probability that failure will not happen in a specified period before time t is

$$R(t) = P(T > t)$$

This shows that reliability is termed as function of time. It mostly relay on environmental set-ups that vary irrespective of time. Since it is probability, mathematically its numerical number ranges from 0 to 1, and *R(t)* is a non increasing function in this range of limits.

$$R(0) = 1, R(\infty) = 0$$

1.2 Software Reliability

"As defined by IEEE, an error is a human action that results in a fault. Encountering a fault during system operation can cause a failure. A bug is synonymous with fault, and a defect is very similar. The words defect and bug are used to mean code that does not satisfy the user requirements, either because a requirement is incorrectly designed or implemented (the vast majority) or was not implemented. A failure is what a customer or tester encountered that caused them to report the defect." [2]

Reliability approximations of any software are managed in various different points, among them:

- To predict reliability of entire system
- To assign all the available resources at time of development and maintenance phase
- To analyze all maintenance costs.

The option of reliability-measures is helpful in consideration of whether the important penalty of the system failures depends on

1. The integral time span of system failures, or
2. The frequency of system failures.

If complete time span of failures is superior, then we can have the suitable measure associated to the availability of system. Otherwise, if frequency of system failures is crucial, then the suitable measure will be coupled to the system's mean up-time or down-time.

Consider an example; A system with a defined period of 1000 days in which it may fail twice and also inactive for five days every time, providing system availability of value 0.99. In other case the system mostly fail five times in similar time span and may be inactive for two days every time, which also generates availability score as 0.99. Thus the meantime to the first system failure can be a significant metric which specify the quantity of time the system can occupy with maximum capacity before initial total system failure. [3]

1.3 Commonly Used Techniques:

Software and its quality are served as most important factors for evaluating the global competitive status of any type of software product. To insist the nature, and to evaluate the performance of these software products, we have lots of software reliability prediction models proposed a few decades ago. Although we have nearly 200 traditional or standard software reliability prediction models among these most of them are mainly based on approximation of probabilities. Mostly these approximations may not supply the targeted accuracy for the evaluation of defects and also for ready to release time. Thus this can be obtained through artificial neural networks (ANN) where they give parametric estimation values without any assumptions. This also supports the parallel distributed approach of a given system. Still a vast range of research is progressing on the prediction of system reliability in order to magnify the productivity and quality of software developed.

1.4 Growth Models

Software Reliability and its Measures

- Failure Rate: failure occurrence rate. Also constitute total failures count in definite time span.
- Mean Time Between Failures (MTBF): Time between failures and their average. The total number of hours required to process prior to failure occurrence. MTBF is like inverse of failure rate.
- Reliability: The probabilities that object accomplish an essential function lacking its failure existence under defined context for a defined duration of time is called reliability. It considers mission time.
- Availability: The probability that an object in working condition at any provided time is called availability. Repairs and down time are taken into consideration.

The Software Reliability Growth Models (SRGM) comprise of two types. They are:

- The Parametric models
- The Nonparametric models

Parametric models are built on non homogeneous Poisson process. Non parametric model as neural network and is constructed on statistical failure data. Nonparametric models are more flexible. Various Reliability Measures:

- Next time to failure
- Time between failures
- Detected Cumulative failures

1.5 Objective

The prominent purpose of proposed thesis work is to execute distinct connection oriented models with discrete activation functions. A collection of disparate datasets having software failures are implemented to the models used. Different datasets are gathered from various software related projects. The variety of issues related to the method of data representation, some type of unrealistic premises integrated with the software reliability models, and architecture of network is examined.

We have implemented the feed forward neural network (FFNN) architecture initially with learning method called back propagation for reliability prediction. We can extend FFNN like optimizing the functions to see the variation with different types of datasets. General regression neural network and radial basis function are used to enhance and compare the error among the models. Following are prominent points of our execution.

- The Neural Network without back propagation
- Feed Forward Neural Network required hidden layers along with back propagation learning method
- Radial basis function Neural Network
- General Regression Neural Network
- Analysis of efficiency of above mentioned proposed models by applying distinct fulfillment criterion.

1.6 Organization of Thesis

Chapter 1 gives the brief view of all the basic terms of thesis. Remaining chapters are going to explain the following. Chapter 2 gives the overview of earlier works done on software reliability prediction techniques and observed gaps. Chapter 3 explains existing methods used and their approaches. It also describes some theoretical concepts of neural networks, back propagation algorithm and function optimization methods. Chapter 4 provides implementation details of project work their experimental results and implementation results. In Chapter 5 we have conclusions of thesis work with achievable future works.

CHAPTER 2
LITERATURE REVIEW

Almost all the papers main intention is to have efficient and accurate reliability prediction of software at its design phase itself. Early defect prediction is very effective in order to improve the availability of any software product. This can enhance the estimation cost and time at which software is ready for shipment. In this work the approach of neural network methods for fine reliability prediction in majority of the real environments are traversed practically and an evaluation method of extension for software reliability using artificial neural network (ANN) mode is introduced. [1]

Artificial Neural Network (ANN) is a strong approach for Software Reliability Prediction (SRP).

Werbos [19] explained that back propagation learning algorithm as an alternative for regression method where it is effective to recognize the origin of prediction in uncertainty applied at latest gas market model. This paper finally infers that NN models are extremely effective in forecasting the uncertainty of any data.

Shadmehr et al. [11] model parameters of pharmacokinetics system are approximated using FFNN multi layered. Noise present in the provided data is also predicted. The result with this method was compared to be better than Bayesian estimator.

ANN approaches and FFNN with back propagation learning are implemented both software reliability and quality estimation. [7,12,13]. Authors in this paper developed connectionist models and processed with a software failure dataset as input which generates reliability as output. The datasets and their depiction, architecture of model networks are discussed.

Karunanithi et al. [14] designed FFNN and recurrent networks for the prediction of software reliability. They considered 14 varied datasets of literature related software and differentiated among them. Through observations they suggested NN gives efficient predictive accuracy compared to existing analytical models.

Sitte [8] they analyzed two approaches for better reliability prediction. They are: neural networks and parametric recalibration methods. In terms of software reliability estimation these models are compared and deduced that NN are easy and superior predictors.

Tian et al. [15] recurrent neural network is used for SRP. The network is trained with Bayesian regularization approach. They concluded that their proposed model generates very minimum average relative error compared to most of the proposed prediction procedures.

RajKiran et al. [16] introduced wavelet neural networks for efficient prediction of software reliability. We can see implementation of two types of wavelets. They are: activation functions as Morlet wavelet and Gaussian wavelet. They have a comparison analysis among different methods like MLR, MARS, BPNN, TANN, PSN, and GRNN hence concluded that proposed model is more effective than these models.

Lo [17] proposed a model for SRP with the help of artificial neural networks (ANN). This proposed method inspects most of conventional reliability of software projects and their extension methods without supposing some practical things.

In this paper fuzzy wavelet neural network model (FWNN) is implemented [18]. The proposed method architecture is fabricated simply with the help of various failure datasets of software as input.

CHAPTER 3
PROPOSED WORK

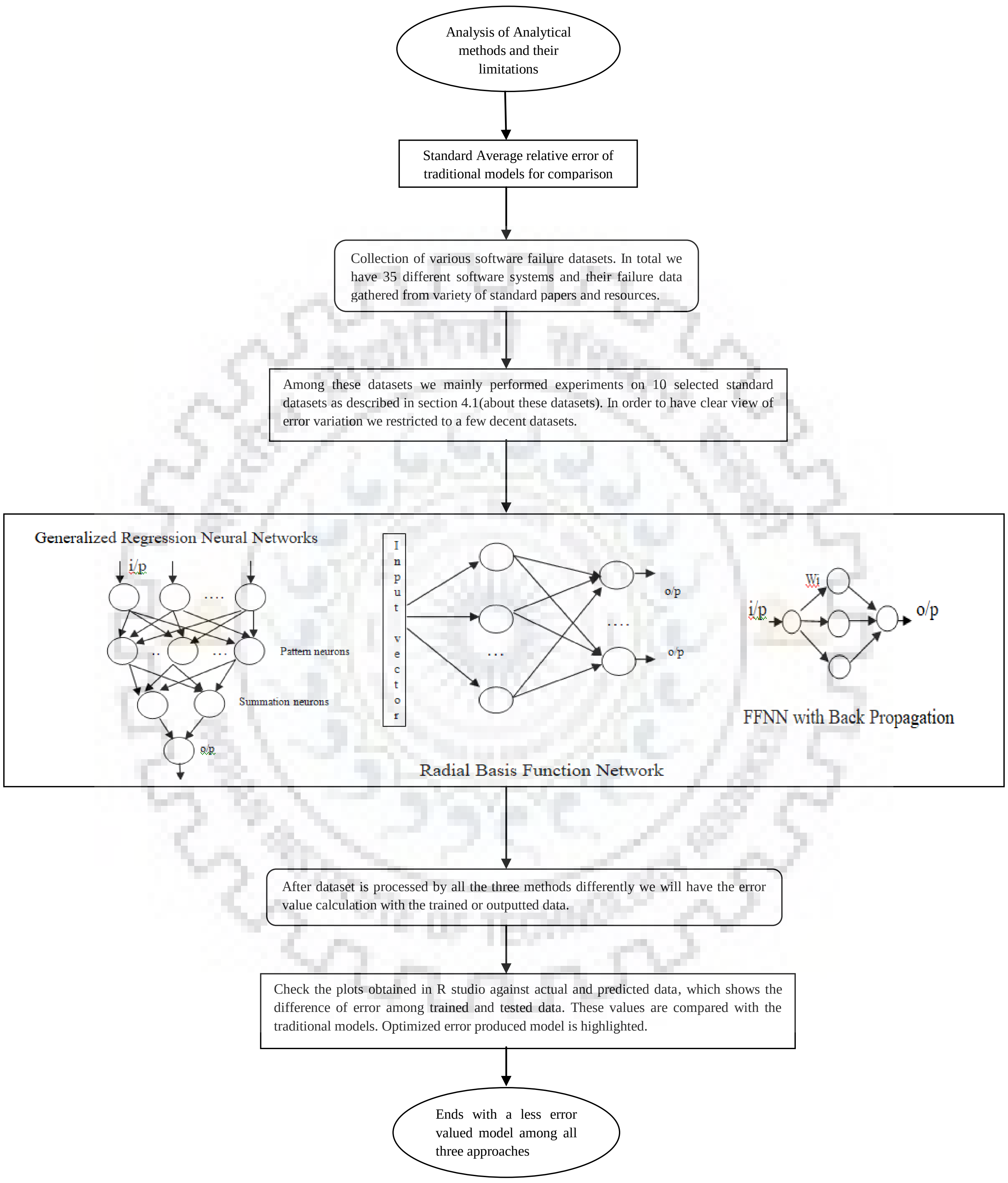
3.1 Motivation

In this fast running computerized world, software products play a key role in every aspect of human life. Thus we can see a tough competition among all software related industries in order to acquire top position on delivering more reliable software products. Due to this rapid growth of software technology, producers are very keen to quickly plan, execute, check and sustain the compound systems in order to satisfy the existing customer's requests. It became very challenging job to those software delivering companies to serve a promising quality and flawless software at perfect tenure. It's very crucial to verify the effects due to system failures which may lead to critical situations like financial loss, unpleasant situations and risk to human life and so on. Hence it became mandatory to check for the reliability of every software product before its delivery. Based on its reliability we can assume the sustainability of delivered products in real world.

Several different researches have been taken place for accurate prediction of reliability. Many scientists worked through variety of models among which neural network is most suggested one. They observed that these neural networks give superior performance when compared to earlier traditional analytical models. These models come under computational intelligence [21] where they have learning capability through data observation unlike analytical models which rely on approximations and assumptions. Computational Intelligence (CI) includes some of famous techniques namely: fuzzy logics, artificial neural networks and genetic algorithms. Neural networks play a superior role in software reliability prediction.

3.2 Proposed Approach

We have implemented a basic level feed forward neural network (FFNN) with and without back propagation learning algorithm. Later we replaced back propagation algorithm with other techniques. General Regression Neural network (GRNN) is implemented to check the performance variation and speed of calculation. Radial Basis Function Network (RBFN) is also introduced for having noise control in the input and enhances the optimization. The following is description about methods used and implemented:



Analysis of Analytical methods and their limitations

Standard Average relative error of traditional models for comparison

Collection of various software failure datasets. In total we have 35 different software systems and their failure data gathered from variety of standard papers and resources.

Among these datasets we mainly performed experiments on 10 selected standard datasets as described in section 4.1(about these datasets). In order to have clear view of error variation we restricted to a few decent datasets.

Generalized Regression Neural Networks
i/p
Pattern neurons
Summation neurons
o/p
Input vector
o/p
o/p
Radial Basis Function Network
Wi
i/p
o/p
FFNN with Back Propagation

After dataset is processed by all the three methods differently we will have the error value calculation with the trained or outputted data.

Check the plots obtained in R studio against actual and predicted data, which shows the difference of error among trained and tested data. These values are compared with the traditional models. Optimized error produced model is highlighted.

Ends with a less error valued model among all three approaches

3.2.1 Neural Networks

A network in which irregular neurons are interrelated among themselves is named as neural network. Neural network is a motivation from biological neuron system. The working of NN is to generate an output sequence when put up with an input sequence. Neural network basically defined as Artificial Neural Network (ANN) consists of collateral-diffusion architecture with huge number of neurons and their interconnection. ANN basically built with three primary components as shown bellow: [7]

- Nodes or neurons
- Network architecture
- Learning algorithm

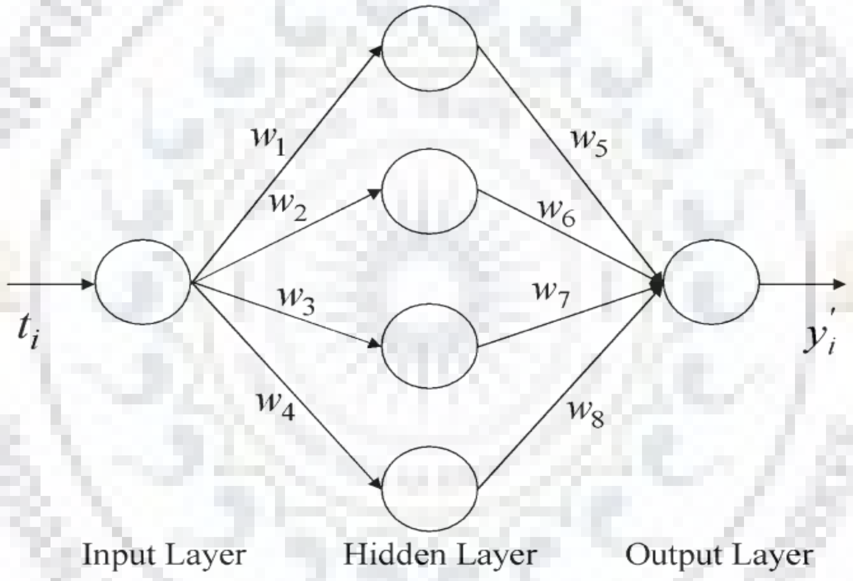


Figure 3.1: Feed Forward Neural Network (FFNN) Architecture [21]

The output of neural network using above three components is:

$$Y = f(A) \quad \text{and} \quad A = \sum_{j=1}^{P} wj\ xj$$

Where, P is number of input elements

F () is an activation function

Σ is summation function addition of inputs and weights

Wj is input weights

Y is neural network output [5]

1) Transfer Function with Hyperbolic Tangent:

$$Y = F(A) = \frac{e^{A} - e^{-A}}{e^{A} + e^{-A}}$$

Y differs from -1 to +1.

2) Transfer function with Log Sigmoid:

$$Y = F(A) = \frac{1}{1 + e^{-A}}$$

Y differs from 0 to +1.

The nature of above mentioned transfer functions is continuous for both hyperbolic and log sigmoid. For simplicity we have used log sigmoid as transfer function during experiments.

- Neural network resembles the working of human brain. It has in built learning mechanisms that are designed within it for designing the dependability.

- This neural network consists of elementary processing elements called as neurons. Numerous nodes (neurons) constitute neural network (ANN). Neurons present in this network are fastened among each other straightly through transmission links correlated with few random weights.

- Sequence of selected input is trained in NN. For a pre defined time span, the all possible outcomes are compared with the predictable sequence of output. This whole procedure is carried out by supervised learning.

- Until we are provided with the expected and satisfying outcomes by our network the training procedure will be carry forwarded. The neurons are organized level by level. The interconnection designs and their structure within neurons and among levels will assemble the architecture of network.

3.2.2 Back Propagation Learning Algorithm

Steps for Algorithm:

1. Weights are initialized
2. Repetition
3. Every training scheme
4. Train that scheme
5. Every training scheme error is noted and mean square error for all schemes
6. Calculate error level by level backward and reform the connecting weights everytime.
7. End
8. until error is admissibly low.

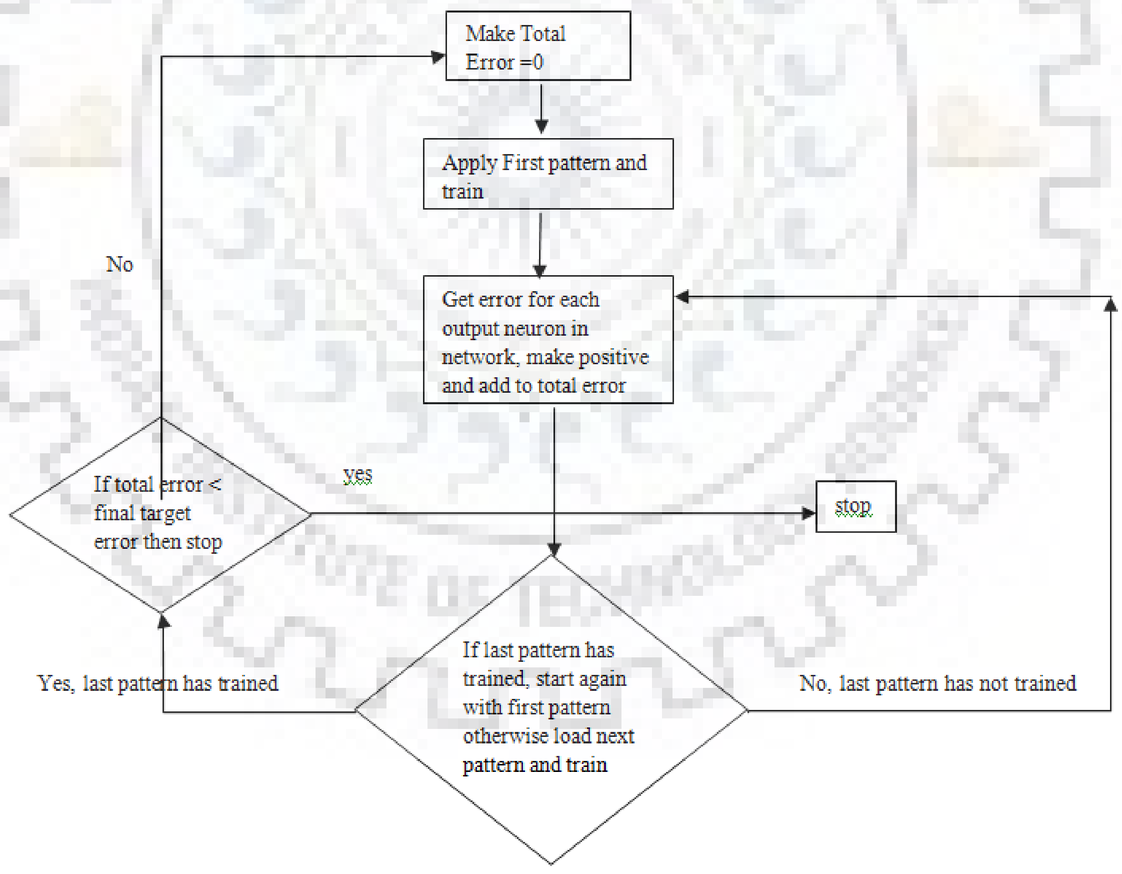

Make Total Error =0
Apply First pattern and train
Get error for each output neuron in network, make positive and add to total error
If total error < final target error then stop
yes
stop
No
If last pattern has trained, start again with first pattern otherwise load next pattern and train
Yes, last pattern has trained
No, last pattern has not trained

Figure 3.2: Flow chart description for Back propagation Algorithm [19]

3.2.3 Radial Basis Function Neural Network

In the context of mathematical modeling the radial basis function neural network is similar to artificial neural network with radial basis function as an activation function. The output pattern is termed as a definite integration of input radial basis functions and neuron parameters. The advantages on using radial basis function network are many. Few among them are function approximation, classification of data, time series related predictions and system control etc.

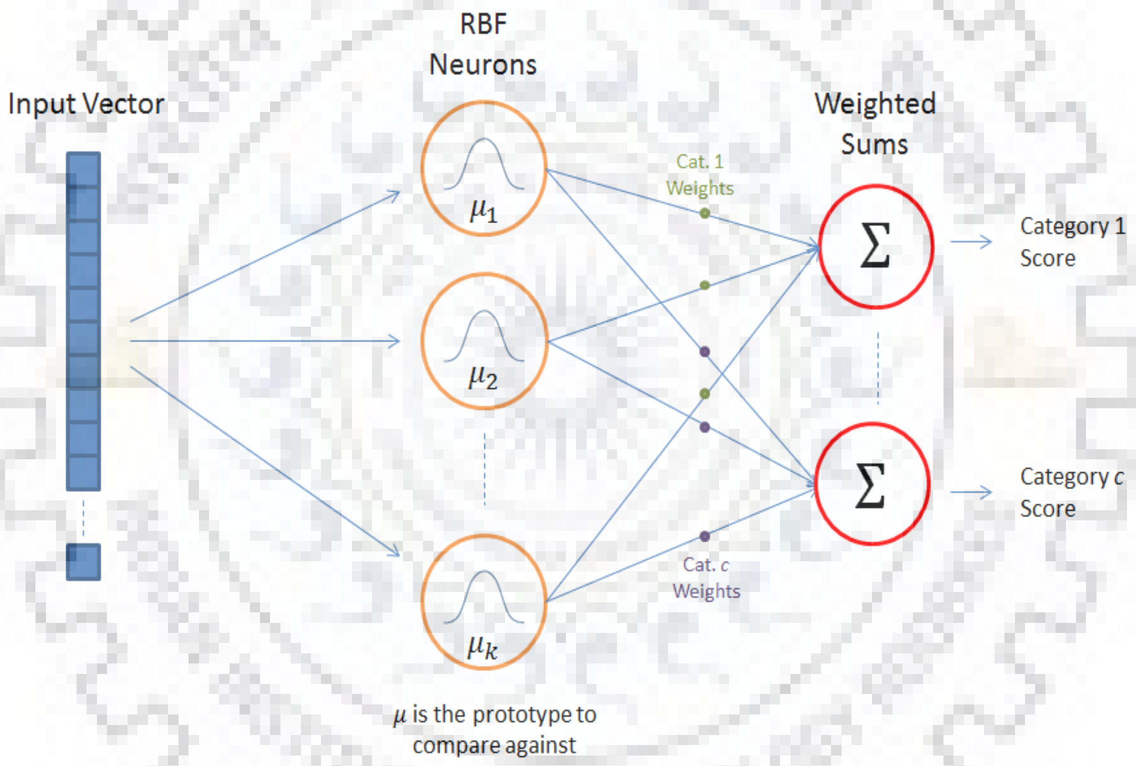


Figure 3.3: Radial Basis Function Architecture [24]

Input Vector:

It is an n-dimensional vector which is used for classification. The complete input vector is provided to every RBF neuron present in network. [9]

RBF Neurons

A prototype vector is reserved in each RBF neuron which is sample vectors from the training set. There will be an analogy among input vector and its prototype which provides an output value from 0 to 1 that resembles the similarity measure. For example, if input is equal to the stored prototype, then output value of that RBF neuron will be 1. If distance measure increases between input and its prototype, the output response decreases exponentially towards 0. Mathematically we can term the response of RBF neuron's as bell curve based on some illustrations. The response of neurons is coined as activation value. The stored prototype vector is named as neuron's center, based on its value at the center of bell curve. [24]

Output Nodes

The network output include a set of nodes that are categorized one per each which are attempting to classify. Every output node calculates the classify score for the related category. The highest scored category is assigned with input based on which the classification conclusion is decided.

An activation value from every RBF neurons is collected and their weighted sum is used to compute the score. Weighted sum signify that a multiplication between weight value of output node that bounds with each RBF neuron and neuron's activation. Output weight value is first used in weighted sum calculation before adding it to total response.

Activation Function

The main motive of this activation function is to compute the similarity measure between the input sample and its prototype vector that is selected from the training set. The most similar input vectors will return value near to 1. We are provided with many popular similarity functions but Gaussian is very well favoured method among all. Here we can see Gaussian equation with one-dimensional input.

$$Y = F(S) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where x represents input, mu is for mean value, and sigma gives standard deviation.

3.2.4 Generalized Regression Neural Networks

GRNN is also neural networks that build with function estimation and function approximation algorithm. We finally have the prediction of output with the provided input data. This follows the basic principle of NN thus there is a necessity of training data to train itself. The presence of input-output mapping is mandatory in training data. [9]

Thus the network is now trained with the training dataset and provided with latest testing dataset; it will appropriately deliver you the expected output or predicts the response for provided input data. Weights are deliberated by applying Euclidean distance between earlier training data sample and testing sample. Hence the output is predicted by weighted average of training dataset and their outputs. If the weight or distance is large then the weight will be very less and if the distance is small it will put more weight to the output. [21]

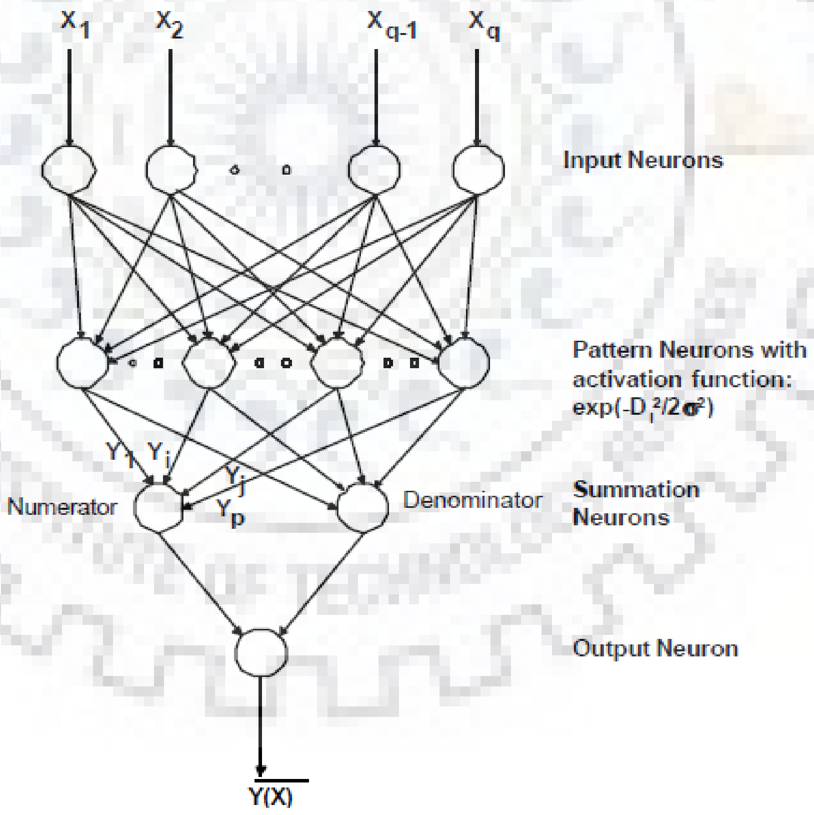


Figure 3.4: Generalized Neural Network [23]

Generalized neural network design contains four primary layers. They are described as follows:

Input layer

Input layer provides input sample to the following layer.

Pattern layer

Its main functionality is to calculate Euclidean distance and produces the activation function needed.

Summation layer

This layer has two main parts namely numerator and denominator. The summation of generated multiplication of activation function and training output data is handled by numerator part. The final summation of all activation functions is placed in denominator part. Both the parts values are provided by summation layer to next output layer. [23]

Output layer

In this layer consists of single neuron. This calculates the required output by dividing summation layer sub parts.

$$Y = F(x) = \frac{\sum Y_i e^{-\frac{d_i^2}{2\sigma^2}}}{\sum e^{-\frac{d_i^2}{2\sigma^2}}}$$

Where

$$d_i^2 = (x - x_i)^T (x - x_i)$$

The terms are: x as input, x_i as training pattern. Output Y_i is generated with input x_i . Euclidean distance d_i^2 from the x and x_i. Activation function used is $e^{-\frac{d_i^2}{2\sigma^2}}$. [22]

Through observations the value of d_i^2 clarifies the amount of training sample contribution in the test sample. Based on this d_i^2 value we can conclude that bigger the value smaller the contribution to output and vise versa. This parameter $e^{-\frac{d_i^2}{2\sigma^2}}$ shows the amount of weight the training sample will finally contribute overall.

CHAPTER 4
EXPERIMENTAL SETUP

4.1 Datasets

Project Code	**Project Name**	**Number of Failures**	**Development Phases**
Database1	Control System	136	Testing Operations
Database2	Control System	54	Testing Operations
Database3	Control System	38	Testing Operations
Database4	Control System	53	Testing Operations
Database5	Control System	73	Subsystem Test
Main	Cumulative Failures	136	Testing
Musa Dataset1	Control System	136	Testing Operations
Musa Dataset2	Iyer and Lee (1996)	191	System Test
CSR1	Time Between failures data	391	System Test
DATA7	A real-time control application consisting of 870,000 lines of code	109	Cumulative test time

Table 1: Datasets and their details

4.2 Feed Forward Neural Network and its process

a. Back propagation learning algorithm is used to this FFNN.
b. The entire basic FFNN architecture used here consists of two steps.
   1) Develop the basic feed forward neural network

2) Apply back propagation learning algorithm to this FFNN

c. From the weighted layer the input vector in generated. FFNN is composed of two layered plot network.

$$Y(n) = A(B(x(n))$$

d. Here we use the back propagation and its learning techniques to revise the weights of the defined network (A and B). This how training is processed for FFNN. [17]

e. ‘x’ as an input sample is generated with a layer corresponding weight W illustrated in the below equations. [10]

$$Y_j(n) = A\left(net_j\right)(n)$$
$$net_j(n) = \sum_i^p \left(W_{ji}\right)(X_i(n)) + \theta_i$$

Where p as input nodes count,

θ_i termed as bias lastly A as activation function.

d. Finally the desired output is mathematically calculated using hidden states and output related weight v.

$$Y_k(n) = B(net_k(n))$$
$$B\left(net_k(n)\right) = \sum_j^m Y_j(n) v_{kj} + \theta_k$$

Where, m as count of state or hidden nodes in network.

θ_k used as bias function

B is another activation function. For this function sigmoid function is used for further calculations. [9]

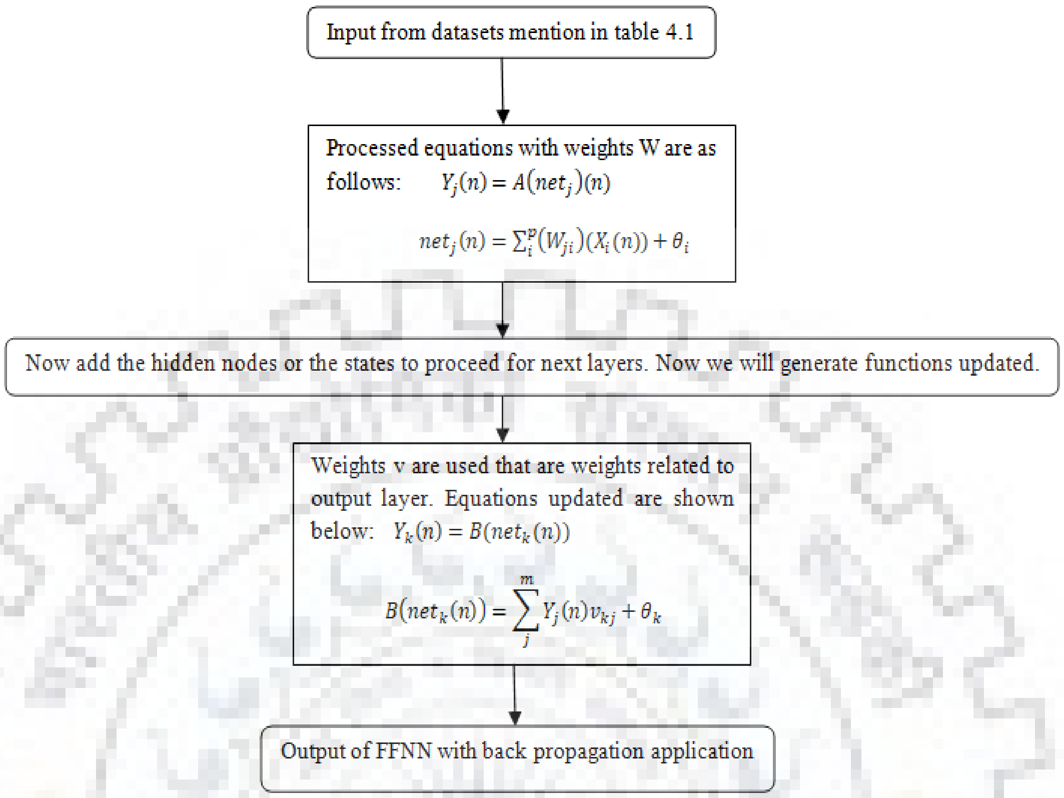

Input from datasets mention in table 4.1
Processed equations with weights W are as follows: $Y_j(n) = A(net_j)(n)$
$net_j(n) = \sum_i^p (W_{ji})(X_i(n)) + \theta_i$
Now add the hidden nodes or the states to proceed for next layers. Now we will generate functions updated.
Weights v are used that are weights related to output layer. Equations updated are shown below: $Y_k(n) = B(net_k(n))$
$B(net_k(n)) = \sum_j^m Y_j(n)v_{kj} + \theta_k$
Output of FFNN with back propagation application

Figure 4.1: Approach of FFNN with back propagation

4.3 Various Performance meters

Below mentioned are different performance calculations used for verifying the models used in this work:

- Relative Error percent (%): RE$_i$= $(|(E_i - O_i)/O_i|)$* 100
- Average Relative error (%): 1/n $\sum^n RE_i$
- Root Mean Squared Error (RMSE) = $\sqrt{[(\sum^n (E_i - O_i))^2/n]}$
- Mean Absolute Error (MAR): $[\sum^n |E_i - O_i|]$/n
- Mean Error : $[\sum^n (E_i - O_i)]$/n

Where,

E_i= expected or predicted value

O_i= absolute or actual value

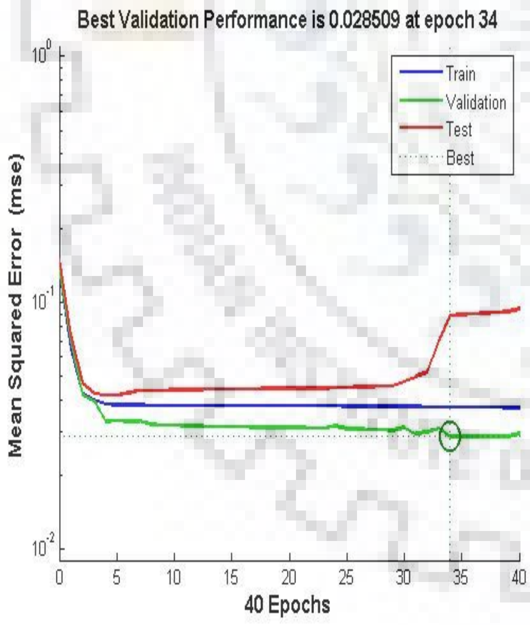
N = entire count of observations or patterns [4]

Basic overall view of FFNN model and is working, [7]

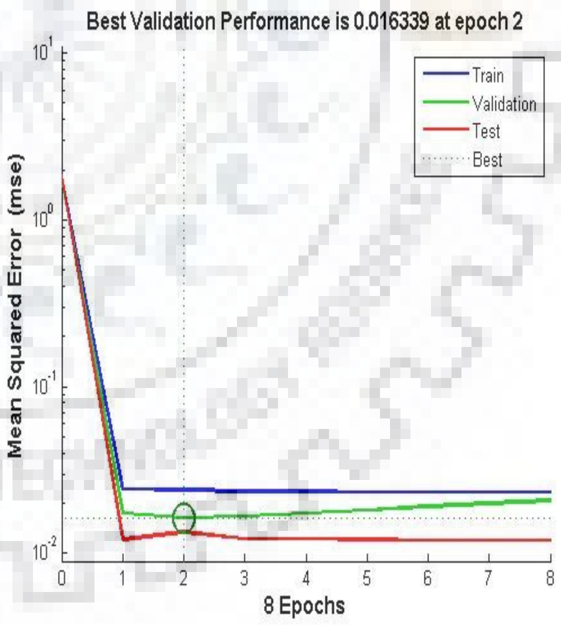
- Input is provided by datasets in the form of cumulative execution time.
- Output is defined by number of cumulative failures given by datasets
- Later these input and output terms are normalized in the range of 0 to 1.
- Then it is trained and tested which is represented by a plot against cumulative execution time on abscissas and number of cumulative failures on ordinates.

4.4 Graphs and Screenshots

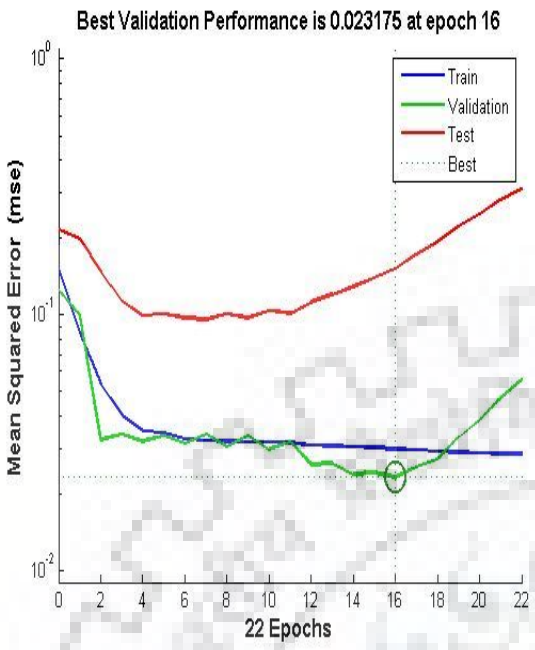
4.4.1 Initial Stage:

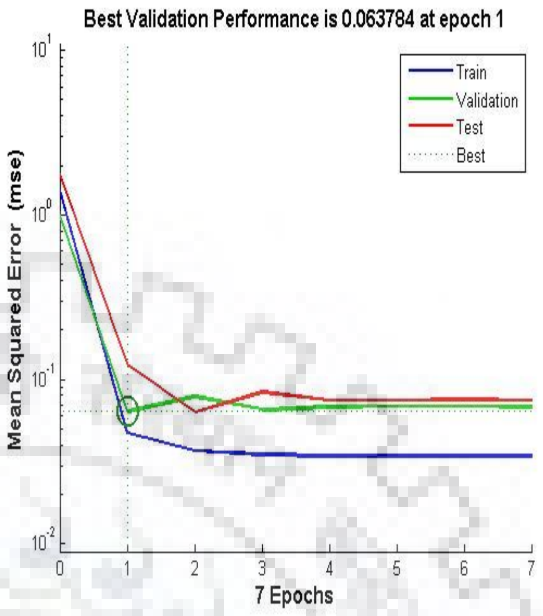


Dataset1



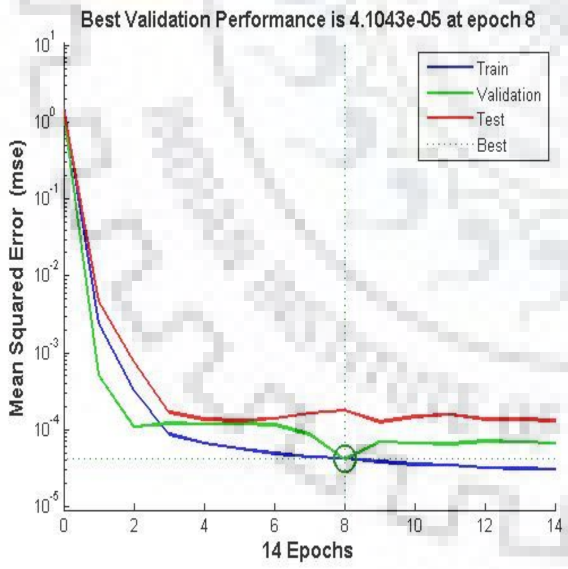
Dataset2

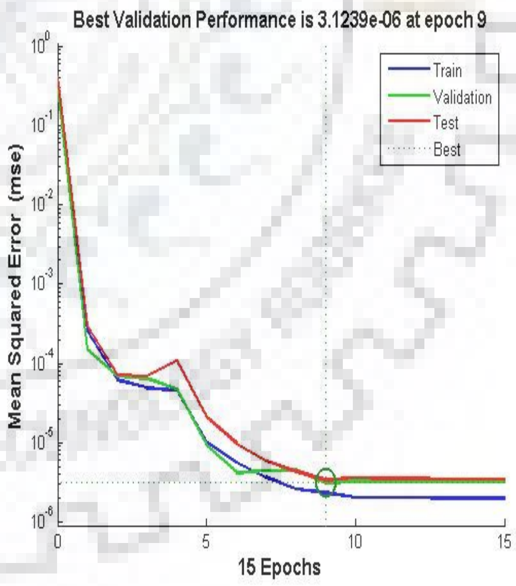




Dataset3

Dataset4





Musa dataset1

Musa dataset2

The graphs in section 4.4.1 are plots between number of epochs and error rate during training. For FFNN the mean square error rate gradually decreases with number of epochs.

FFNN Model is tested with different failure datasets. After running with all different types of datasets we can have the observation of variation of error without back propagation algorithm. These results are obtained from MATLAB environment. Datasets 1 and 3 have shown best performances at large epoch values where as datasets 2 and 4 showed best performance at smaller epoch values 2 and 1 from where we have constant decrease in error. Musa dataset 1 and 2 also have best validation performance at nearest epochs like 8 and 9 before which the error is having drastic changes.

From the plot figure 6, we have observed the epochs variation along with RMS value. This is FFNN without back propagation learning algorithm. Though we have a decrease in epochs with RMS during training we can see more error rate among training and testing data. At epoch 2 this shows best performance after this point of epoch the error value tend to decrease continuously.

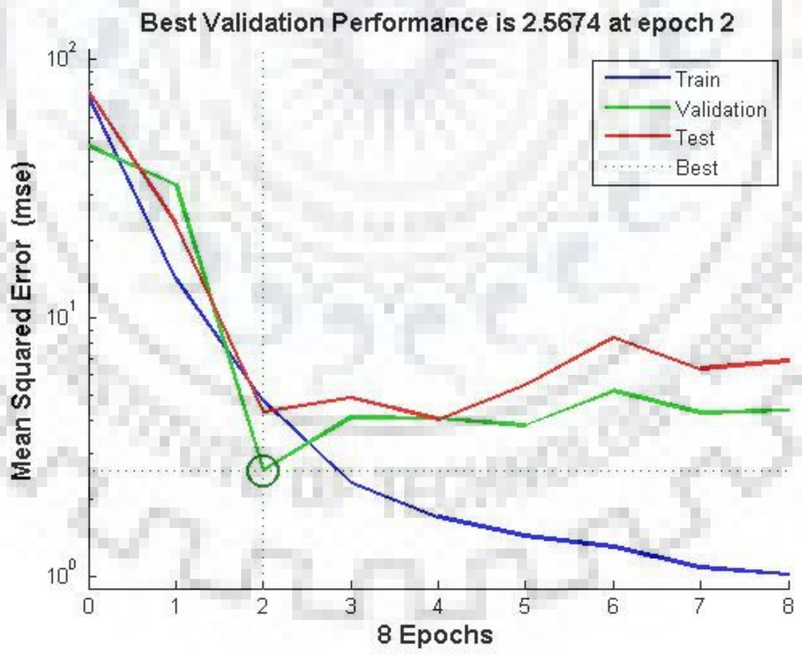


Figure 4.2: Error without Back Propagation Algorithm

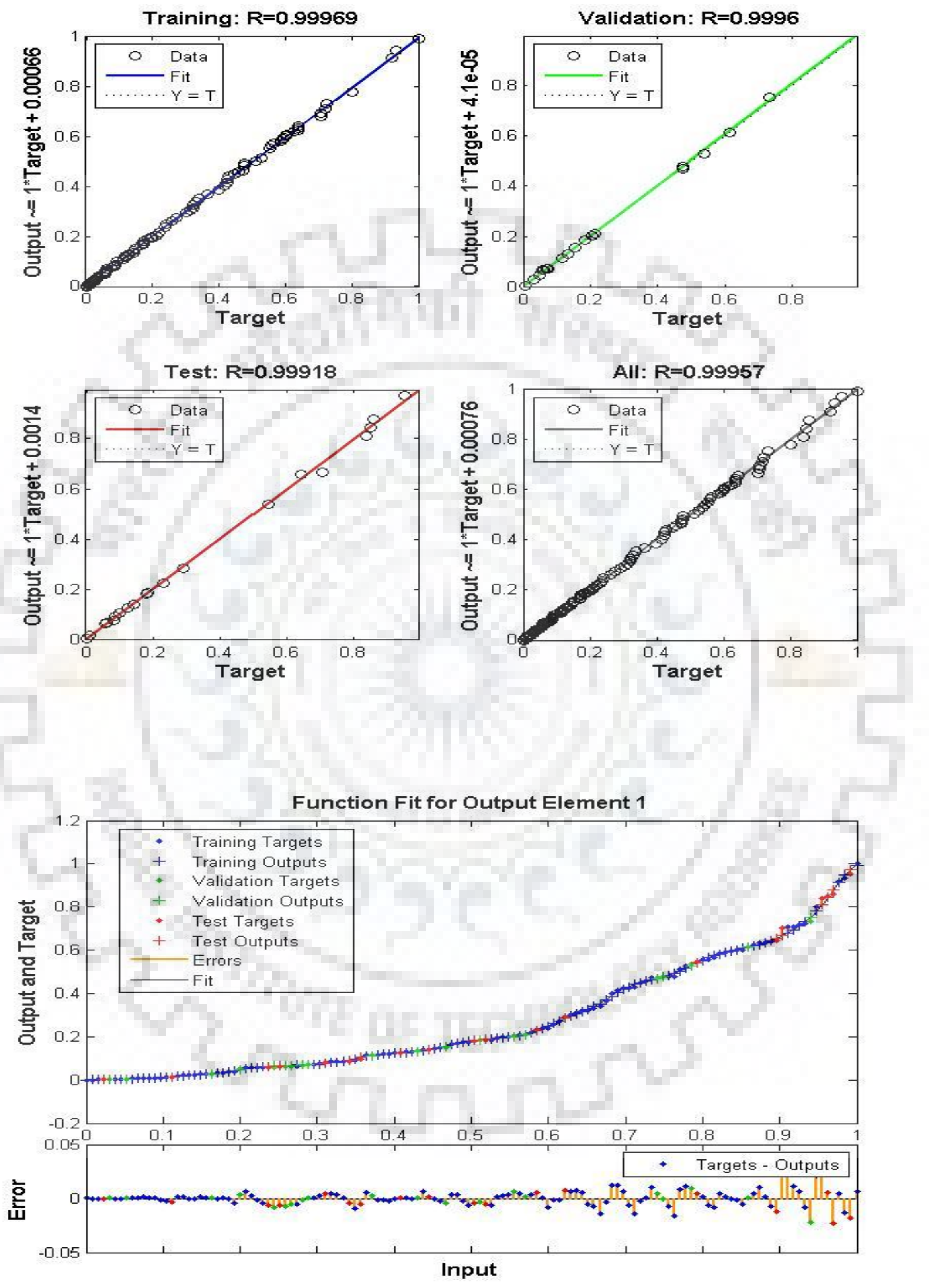


Figure 4.3: Actual and predicted data accuracy during Training and validation for Musa DB1

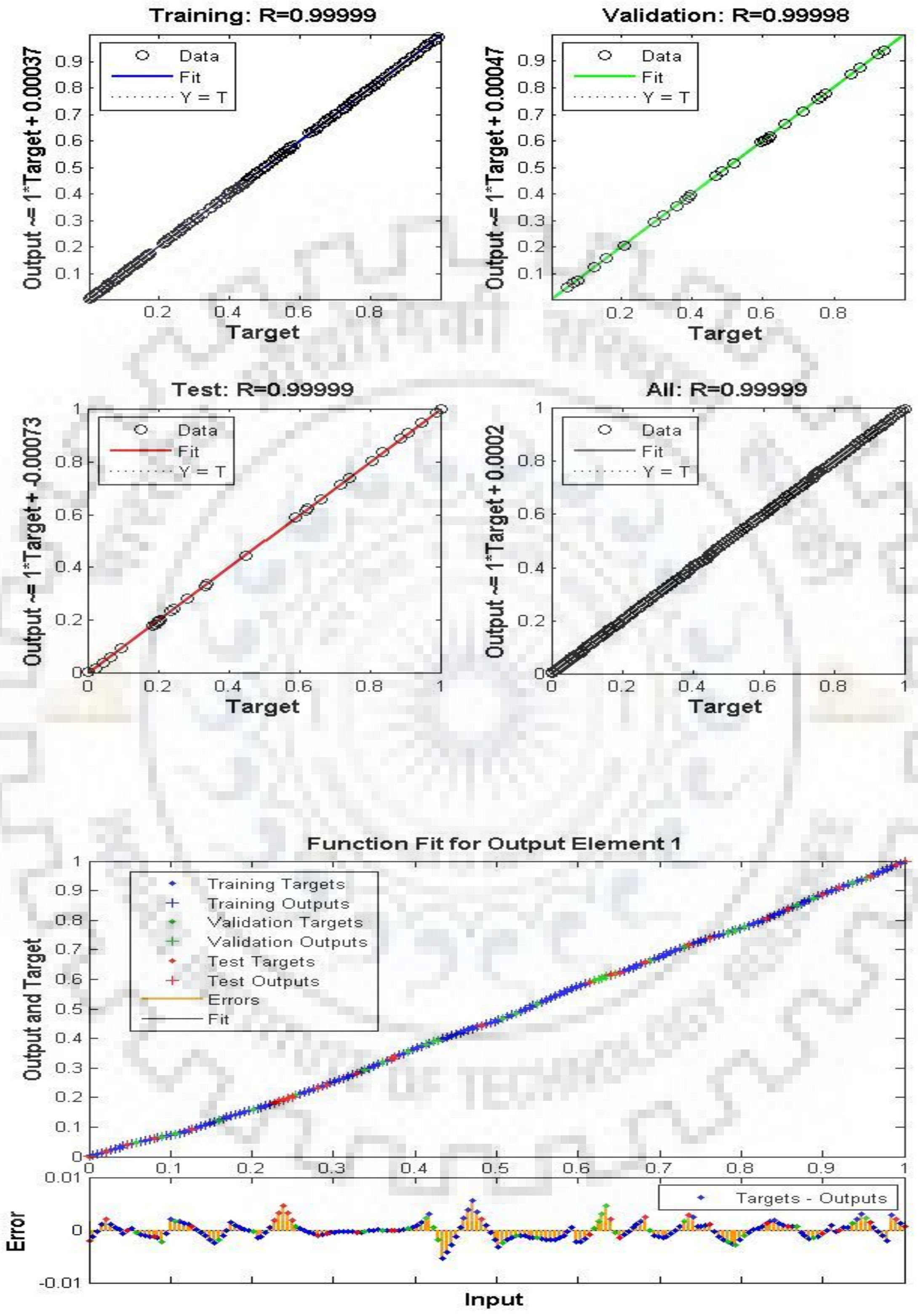


Figure 4.4: Actual and predicted data accuracy during Training and validation for Musa DB2

From figures 7 and 8 we can observe the regression defined value at various stages like training, testing and validation phases. We can see the variation of data among target and expected output. Also we can have the function curve fit for the provided output elements. The standard datasets like Musa dataset 1 and 2 are observed for proper function fit and error variation. Musa dataset 2 is having linear plot fitting and less error rate compared to musa dataset 2 function fitting plot which is little non linear.

4.4.1 Feed Forward Neural Network with Back propagation:

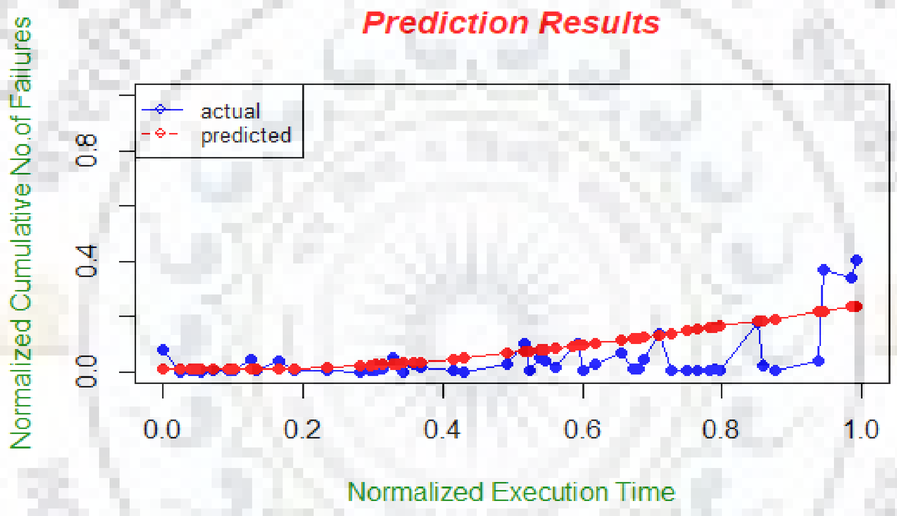


Figure 4.5: Interpretation of FFNN for dataset CSR1

Figure 9 shows the prediction results of FFNN with back propagation. It is a plot against normalized execution time and cumulative no. of failures. This CSR1 dataset shows the error between actual and predicted data. Red color represents predicted data and blue color represents actual data. Figure 10 shows the network of model FFNN with neurons and their weights.

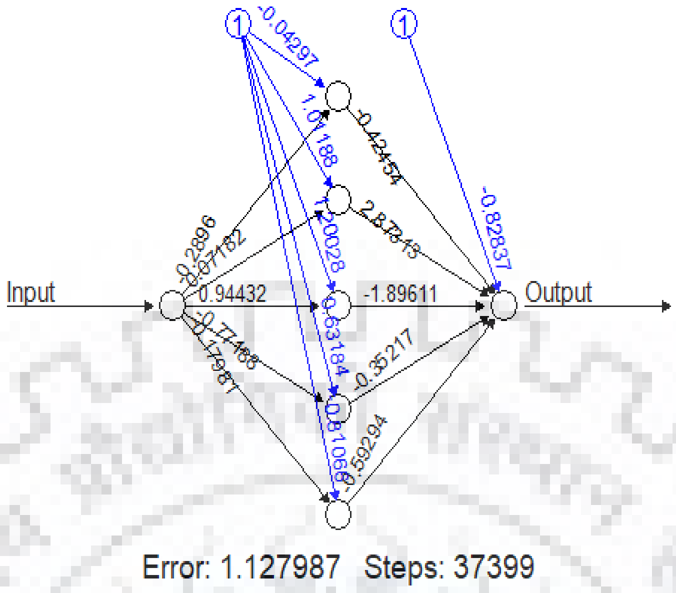


Figure 4.6: Network plot for CSR1

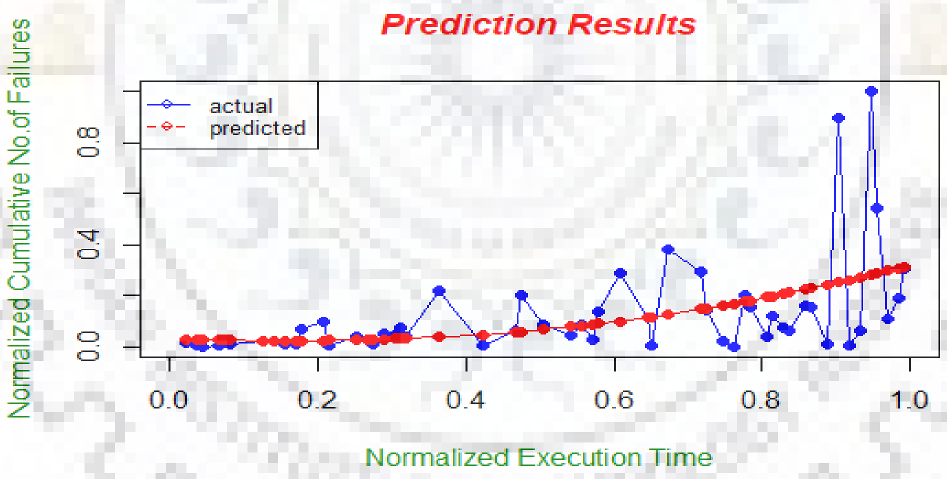


Figure 4.7: Interpretation of FFNN for dataset1

Dataset1 shows more error rate among actual and predicted data due to more noise in the dataset. Dataset 2 also shows more error rate with trained and tested data. Among these datasets we observed more relative error due to prediction of varied noisy actual data.

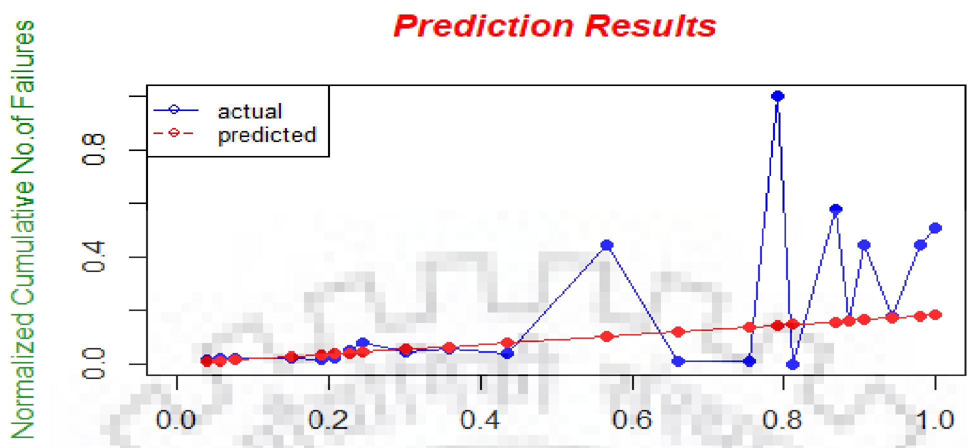


Figure 4.8: Interpretation of FFNN for dataset2

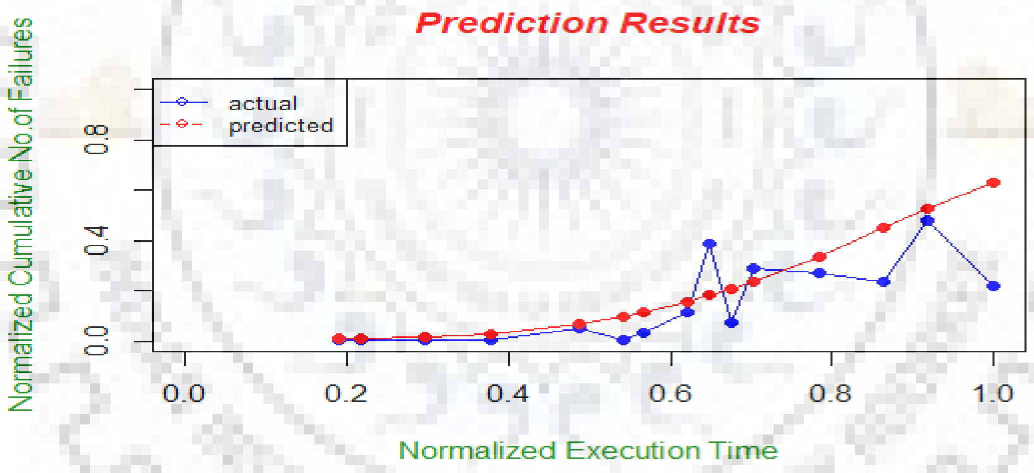


Figure 4.9: Interpretation of FFNN for dataset3

Dataset 3 and 4 shows its performance with FFNN back propagation model. The blue colored line depicts actual data which is having little more error rate. Error difference is observed based on the plot of red colored line of predicted data.

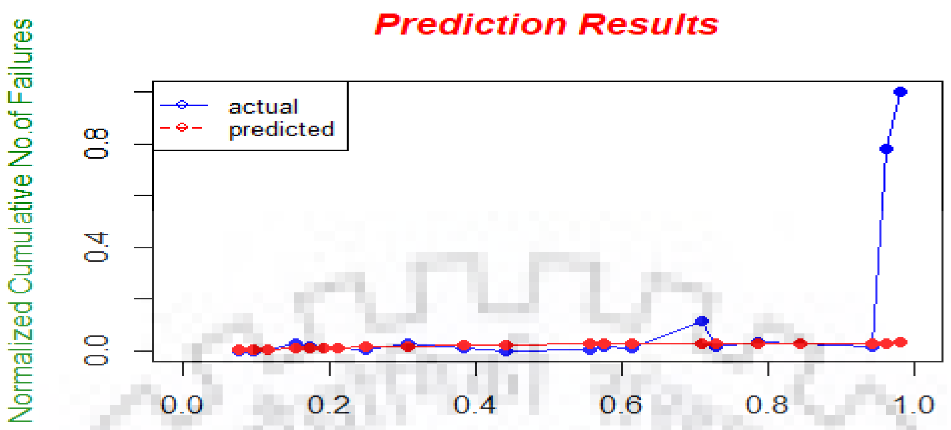


Figure 4.10: Interpretation of FFNN for dataset4

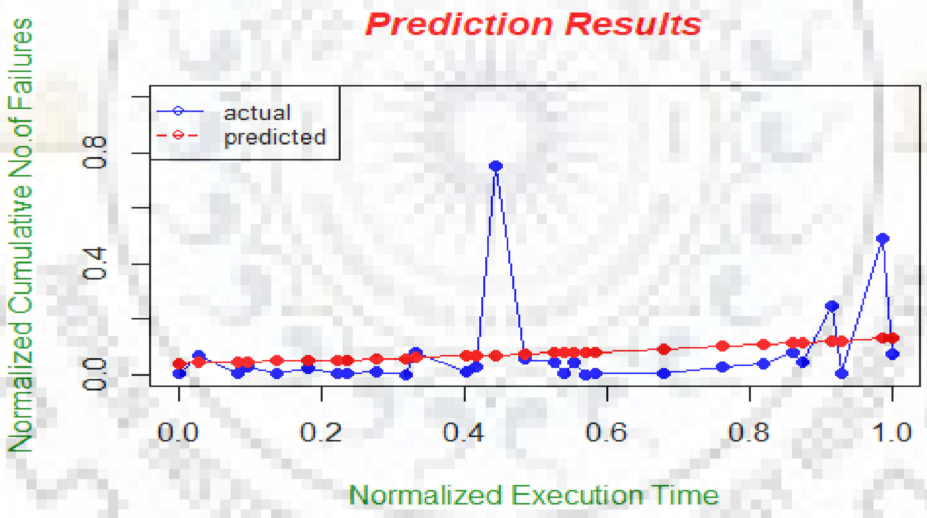


Figure 4.11: Interpretation of FFNN for dataset5

We can see clear difference among the datasets- dataset 5 and Main. Among these two Main dataset is giving best performance by giving less error rate. As there is a close overlapped plot among actual and predicted lines which shows less error rate compared to earlier datasets.

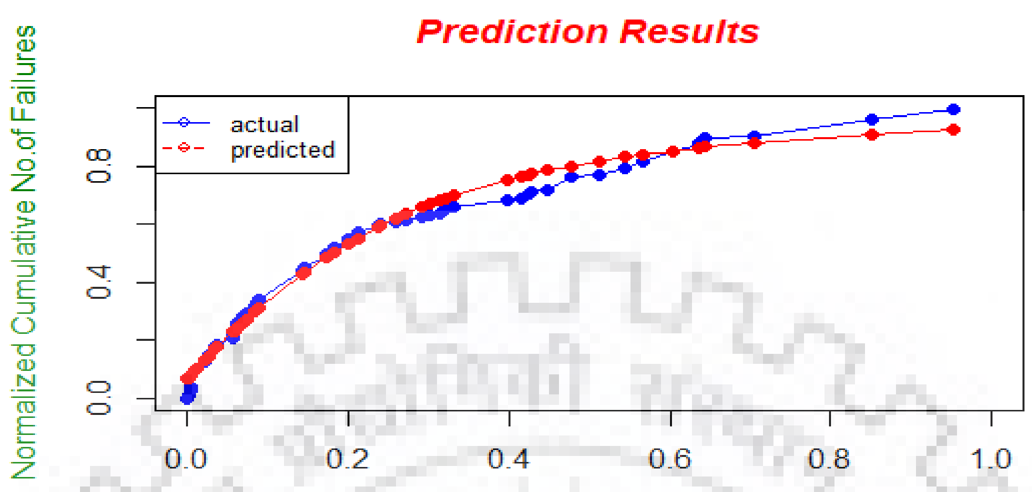


Figure 4.12: Interpretation of FFNN for dataset Main

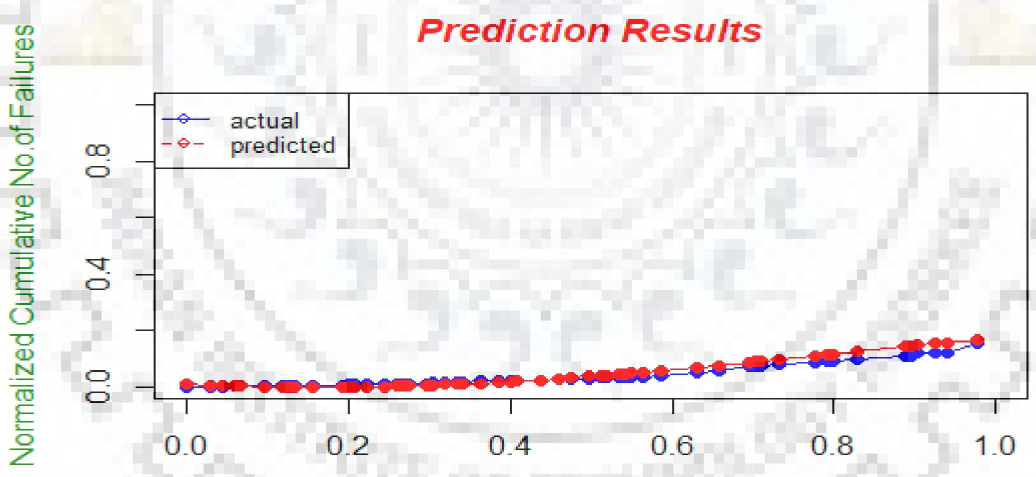


Figure 4.13: Interpretation of FFNN for dataset Musa DB1

Musa DB1 is the standard dataset which has best validation performance which clearly visible through figure 17. This plot has less difference among actual and predicted data. Same performance is observed in datasets musa DB2 and DATA7 also.

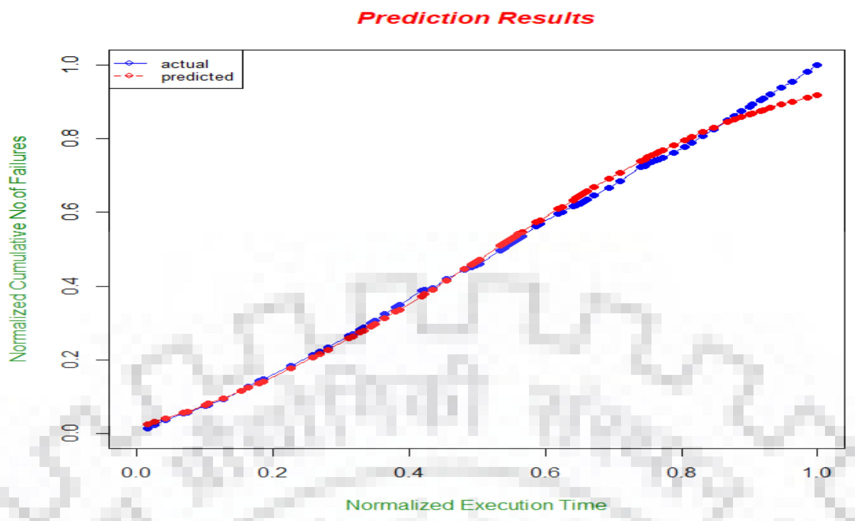


Figure 4.14: Interpretation of FFNN for dataset Musa DB2

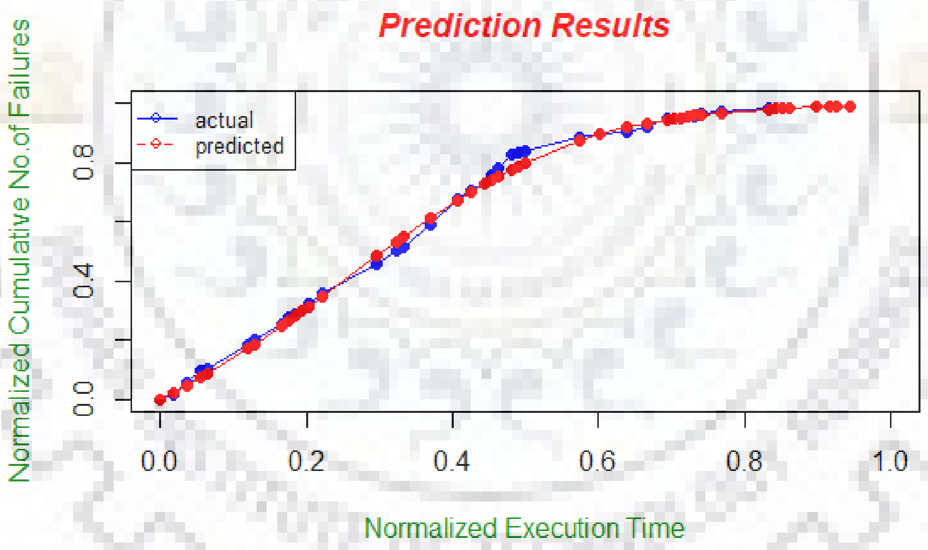


Figure 4.15: Interpretation of FFNN for dataset DATA7

The above given plots show the different datasets performance using FFNN with back propagation learning algorithm. The graphs are plotted against normalized execution time and normalized cumulative number of failures. The red colored spots represent predicted data and blue colored dots represent actual data. Through the graph we can observe the error among actual and predicted data.

We used 10 datasets as mentioned in section 4.1. We observed the error variation among all the datasets with FFNN technique. This result is compared with other methods used and observations are noted.

4.4.3 Generalized Regression Neural Network:

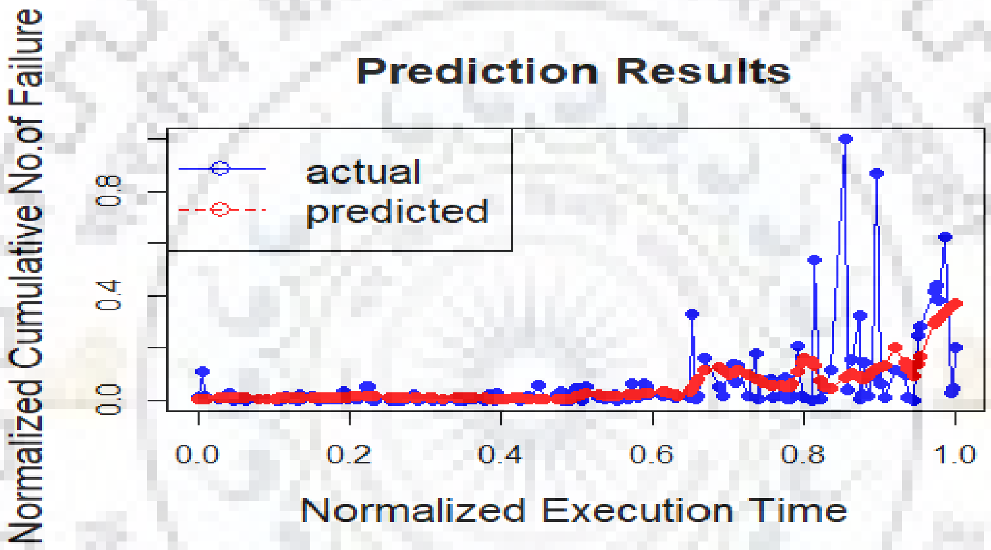


Figure 4.16: Interpretation of GRNN for dataset CSR1

Here the same 10 datasets mentioned in section 4.1 is again tested for GRNN model to see the performance variation of all datasets compared to FFNN and error enhancement is observed.

For the dataset CSR1 GRNN is having better prediction data than FFNN. Due to the optimization function activation we can have the error rate reduction. And the input data is trained in fraction of time when compared to FFNN.

Datasets CSR1 and dataset 1 both have same error prediction results for the model GRNN with relatively less error rate than FFNN.

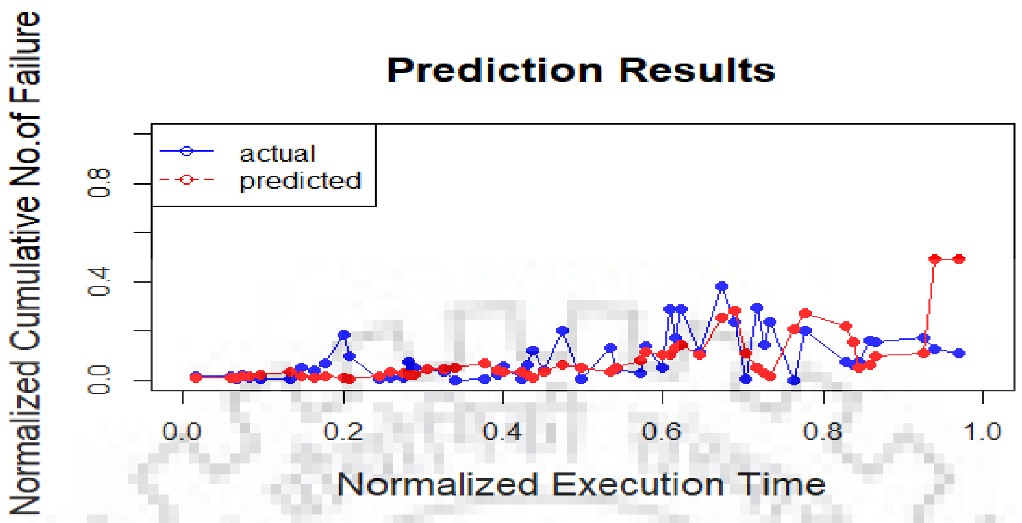


Figure 4.17: Interpretation of GRNN for dataset1

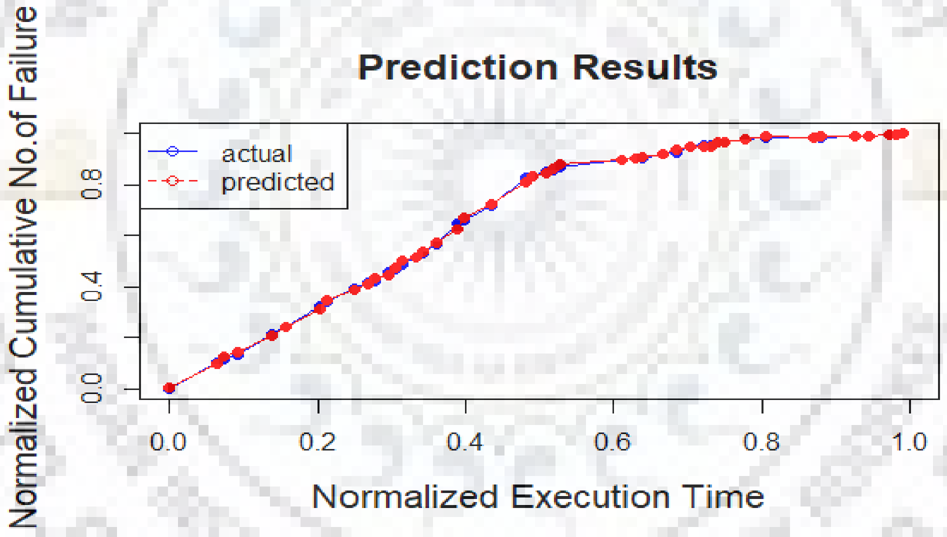


Figure 4.18: Interpretation of GRNN for DATA7

Data 7 performance is shown in figure 22. This plot shows the similarity among actual and predicted data which means very less error rate. Coming to dataset2 in figure 23, the separation of actual and predicted data is more which leads to more error rate i.e., difference of training and testing data. Data 7 gives improved performance than dataset2.

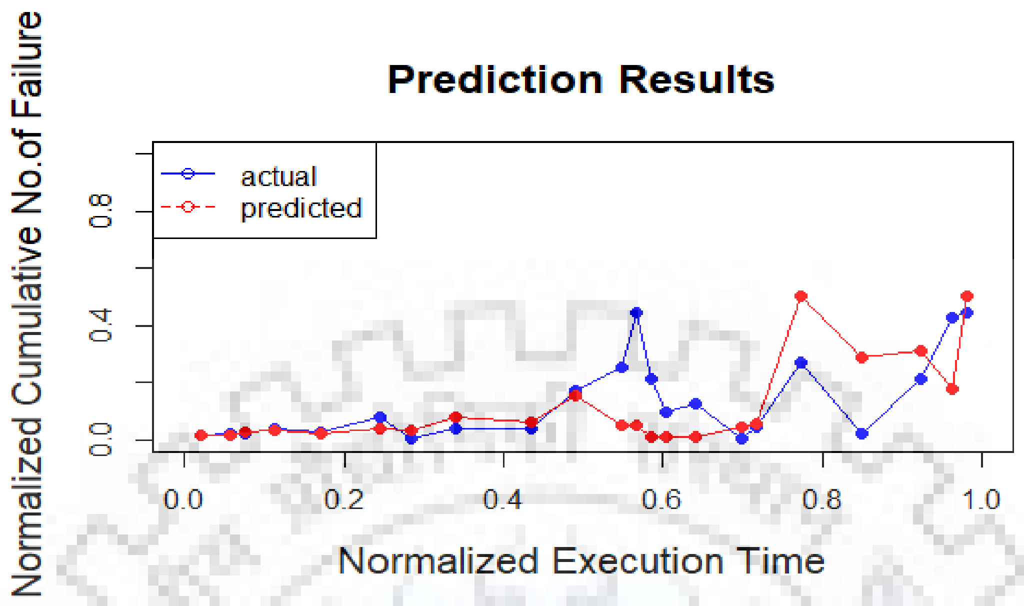


Figure 4.19: Interpretation of GRNN for dataset2

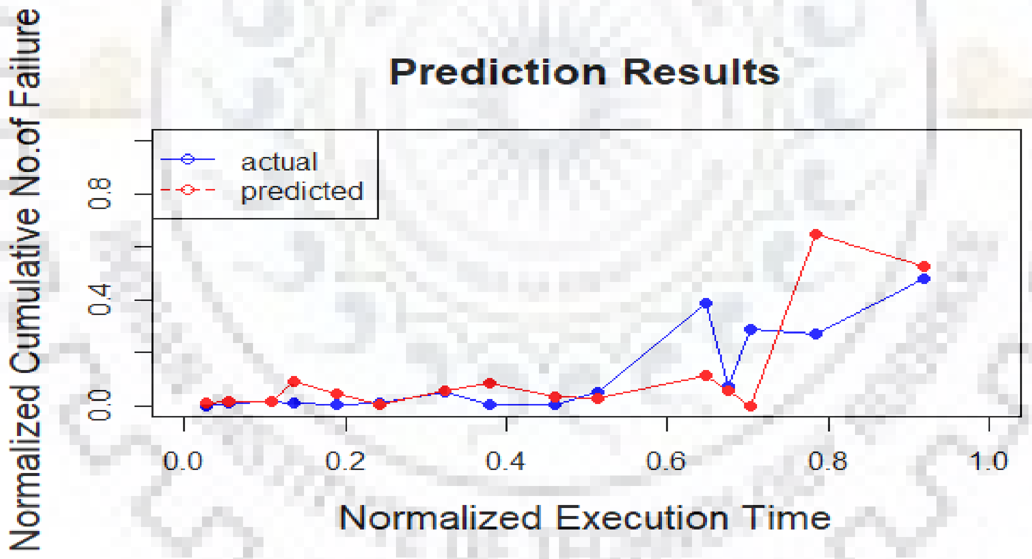


Figure 4.20: Interpretation of GRNN for dataset3

Dataset3 in figure 24 is similar to dataset2. Both have almost same error rate which is more compared to DATA7. Dataset 4 from figure 25 is again producing error rate more than DATA 7 performance results. We can have fast formation of training data irrespective of data type.

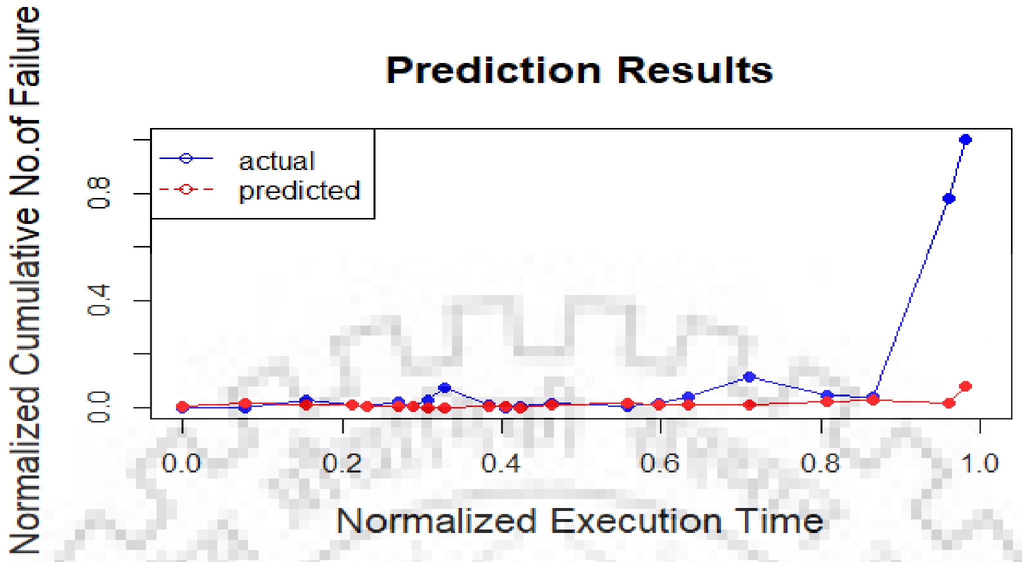


Figure 4.21: Interpretation of GRNN for dataset4

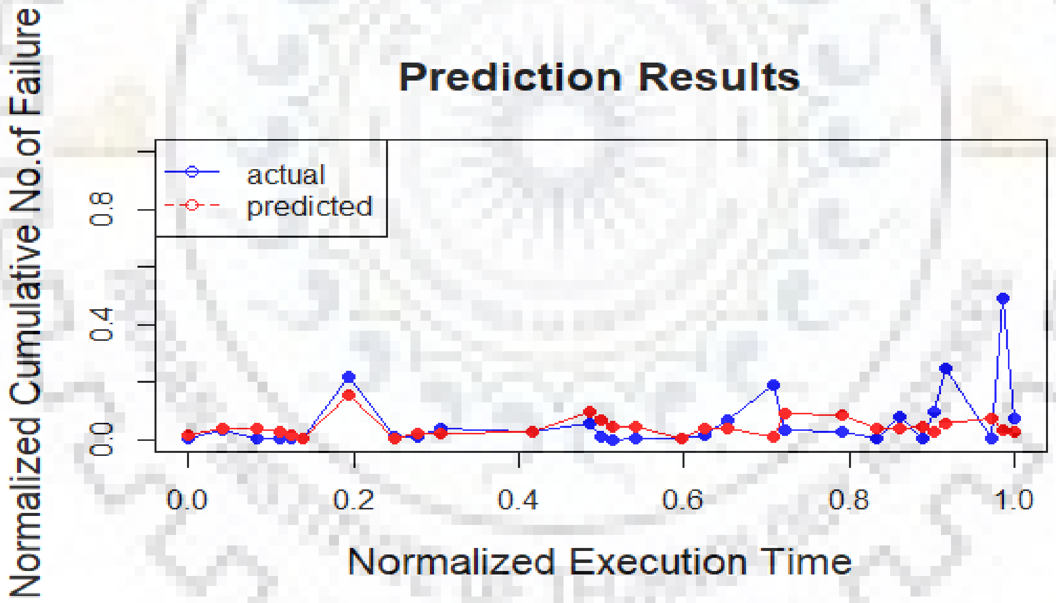


Figure 4.22: Interpretation of GRNN for dataset5

Dataset5 plot in figure 26 has a parallel move of both red and blue colored lines. This clearly shows the partial error reduction compared to earlier datasets in model GRNN.

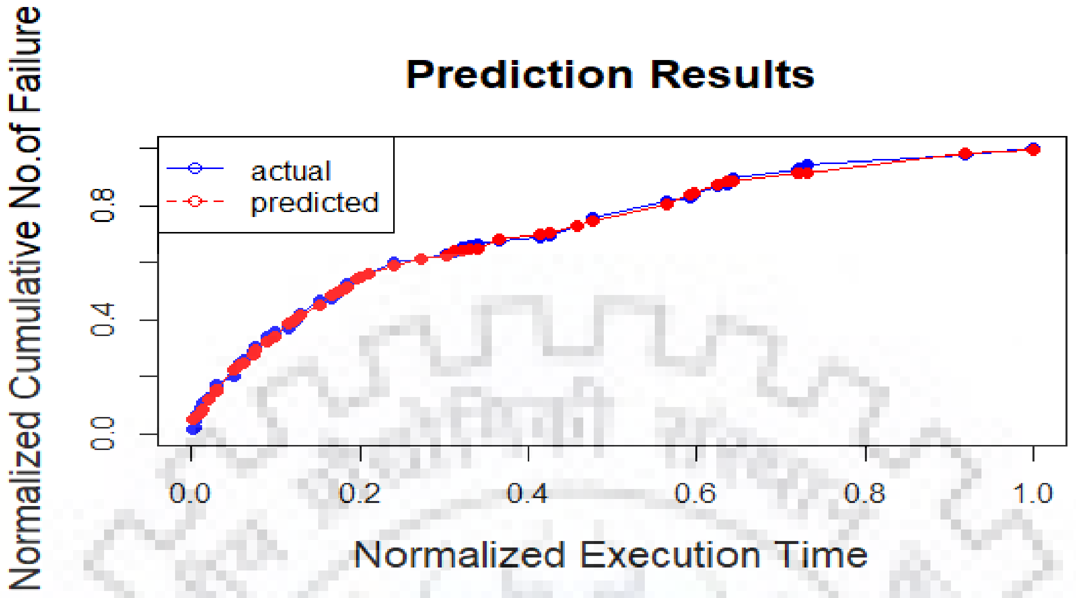


Figure 4.23: Interpretation of GRNN for dataset Main

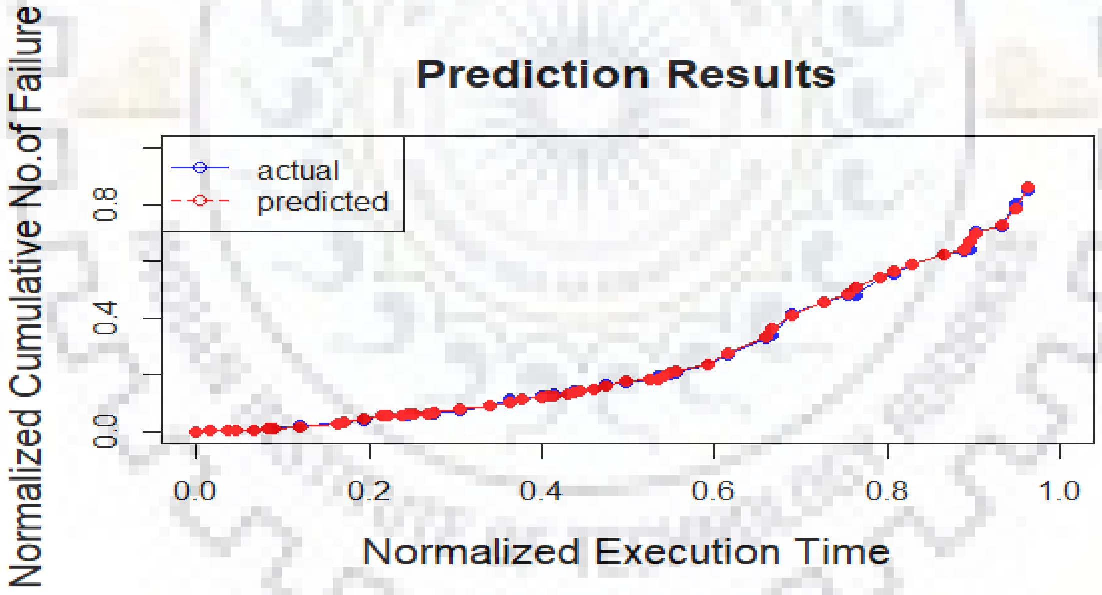


Figure 4.24: Interpretation of GRNN for dataset Musa DB1

Main dataset in figure 27, Musa DB1 in figure 28 and Musa DB2 in figure 29 has the best performance among rest of the datasets. Where, the difference between actual and predicted data gives very less error rate with best validation prediction results.

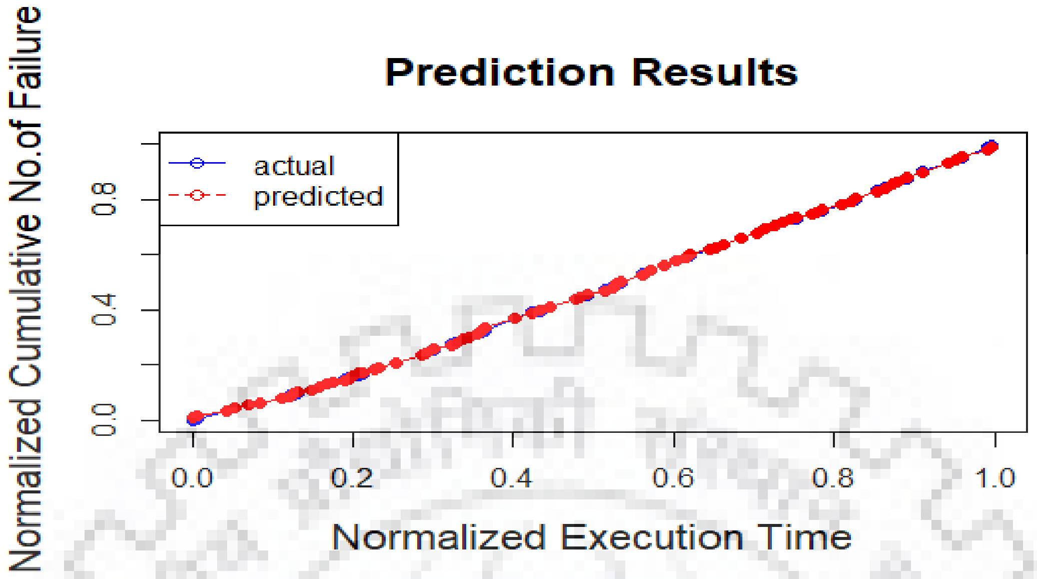


Figure 4.25: Interpretation of GRNN for dataset Musa DB2

4.4.4 Radial Basis Function Network:

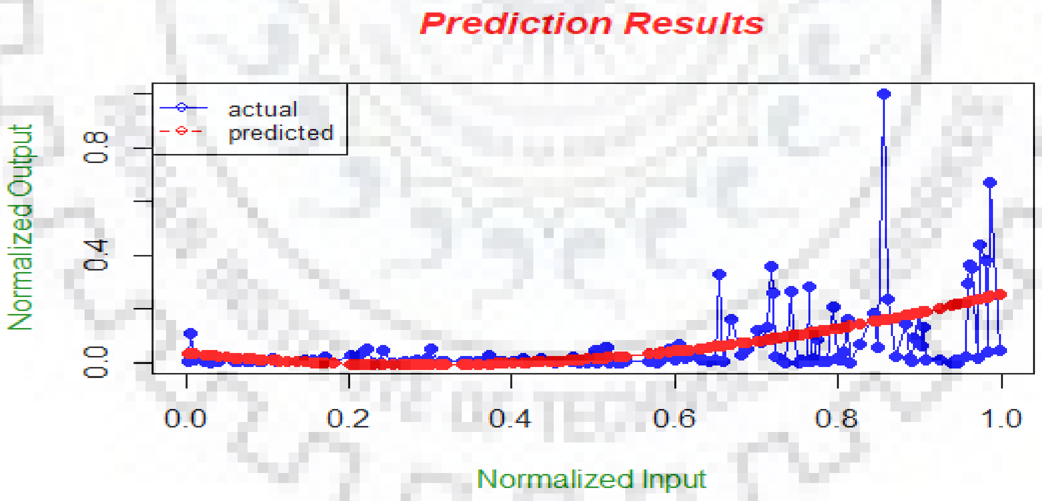


Figure 4.26: Interpretation of RBFN for dataset CSR1

CSR1 is showing better noise controlled prediction results in RBFN. This also displays relatively less error rate when observed with other two models discussed earlier.

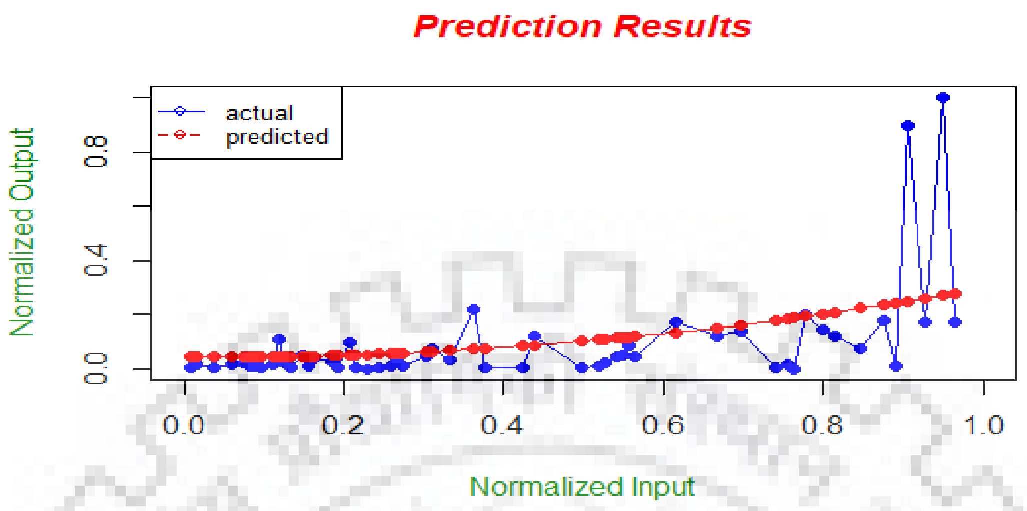


Figure 4.27: Interpretation of RBFN for dataset1

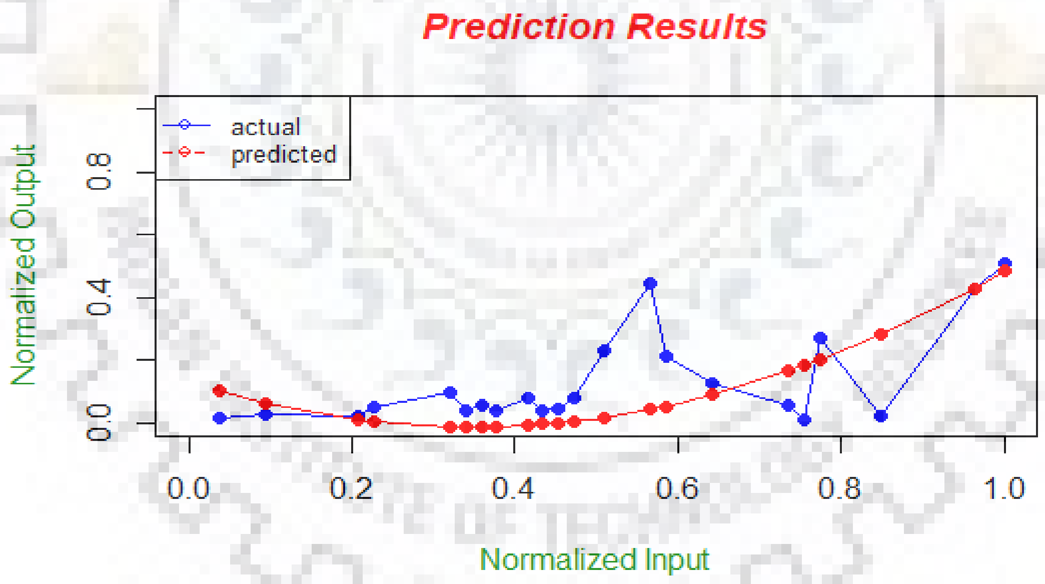


Figure 4.28: Interpretation of RBFN for dataset2

In both figures 31 and 32 the datasets performance is mostly similar and produces little more error rate when compared to FFNN, GRNN.

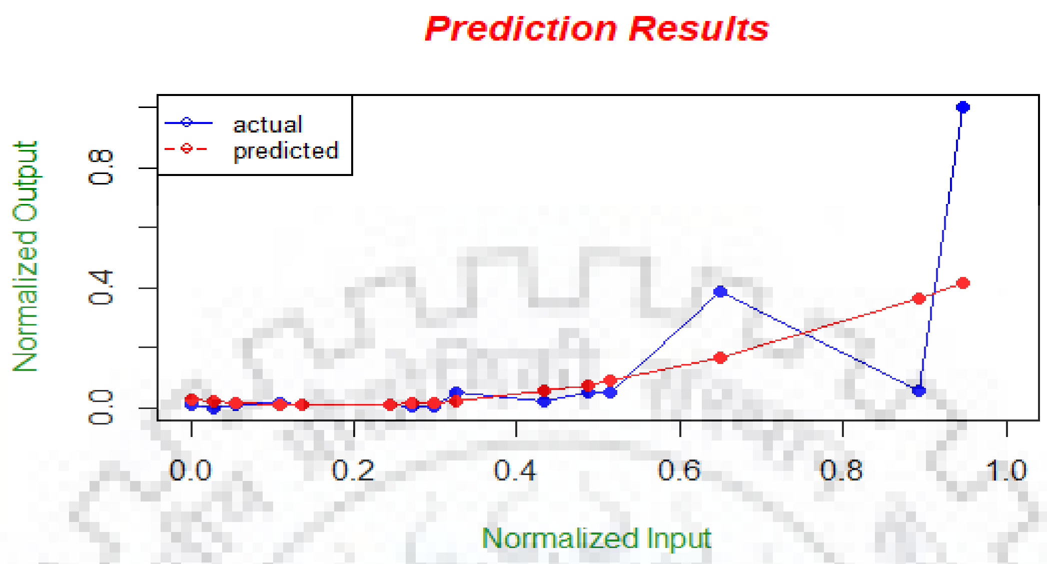


Figure 4.29: Interpretation of RBFN for dataset3

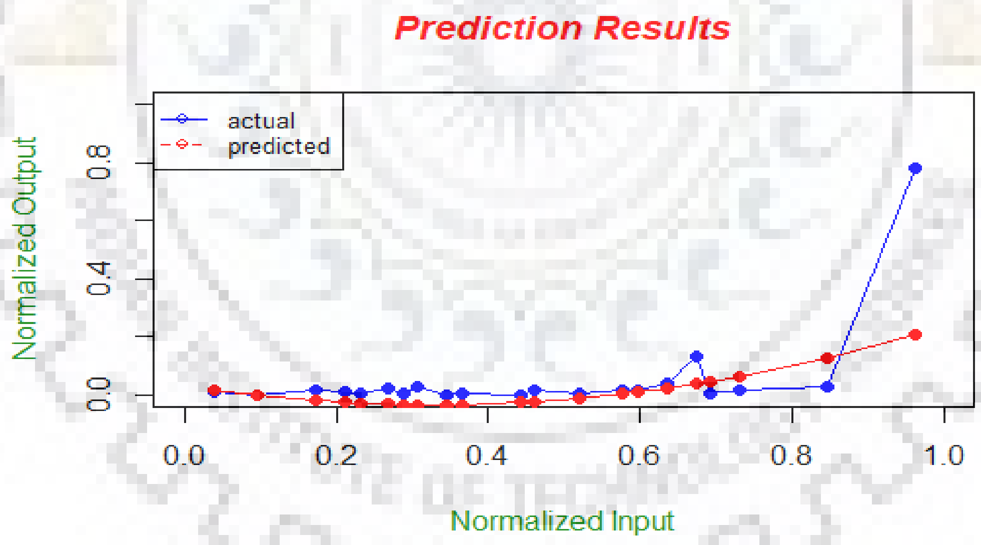


Figure 4.30: Interpretation of RBFN for dataset4

Figures 33 and 34 have prediction results of dataset3, dataset4. Among them dataset3 is having more error rate than dataset4. RBFN gave best performance results for dataset1 to dataset5, where these are the more noisy datasets among the rest. FFNN, GRNN gave high error rate for these datasets. Thus RBFN has optimized activation function for these better results.

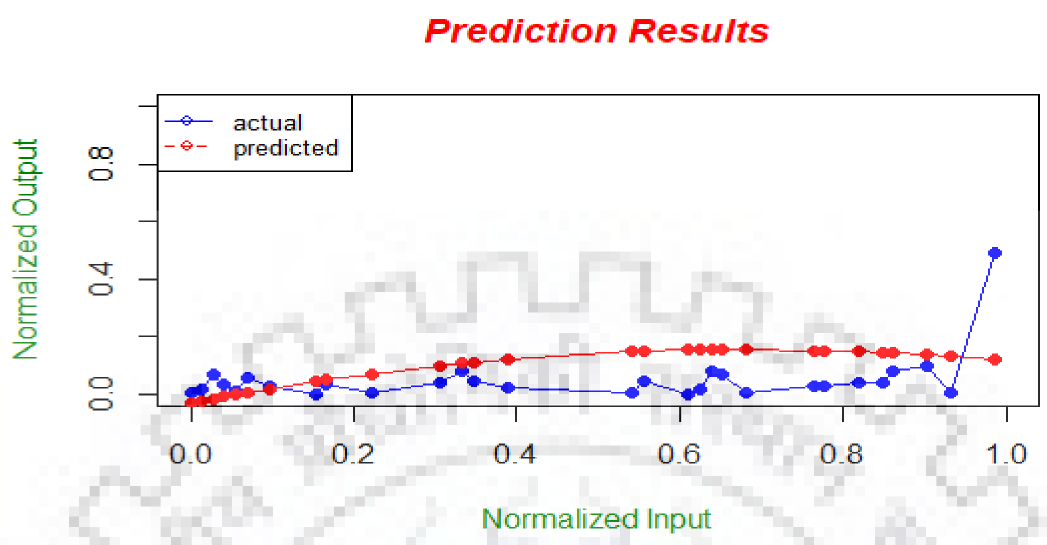


Figure 4.31: Interpretation of RBFN for dataset5

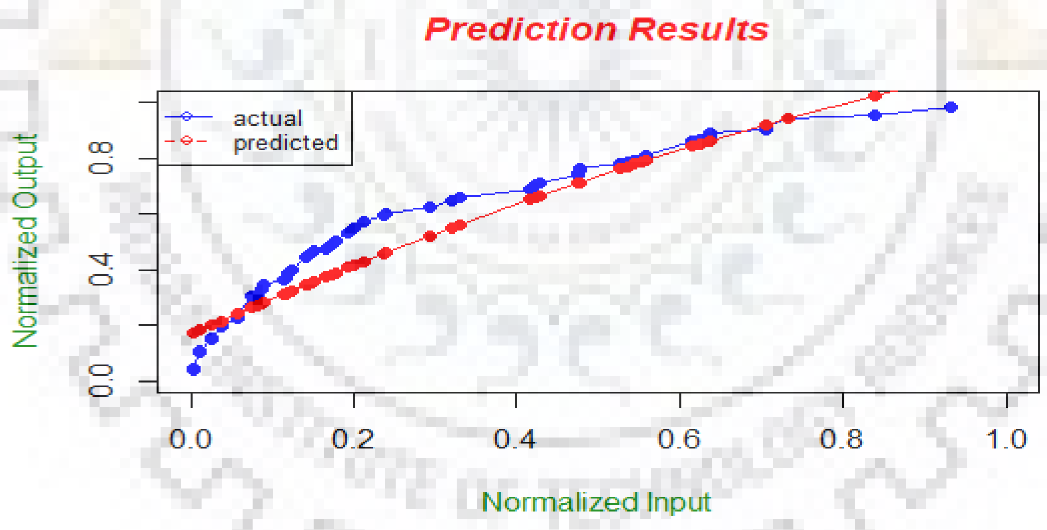


Figure 4.32: Interpretation of RBFN for dataset Main

Main dataset in figure 36 is having similar performance just like in FFNN and GRNN. Musa DB1 and DB2 also have nearest error rates among all three models. DATA7 comes in same way that has comparative prediction results

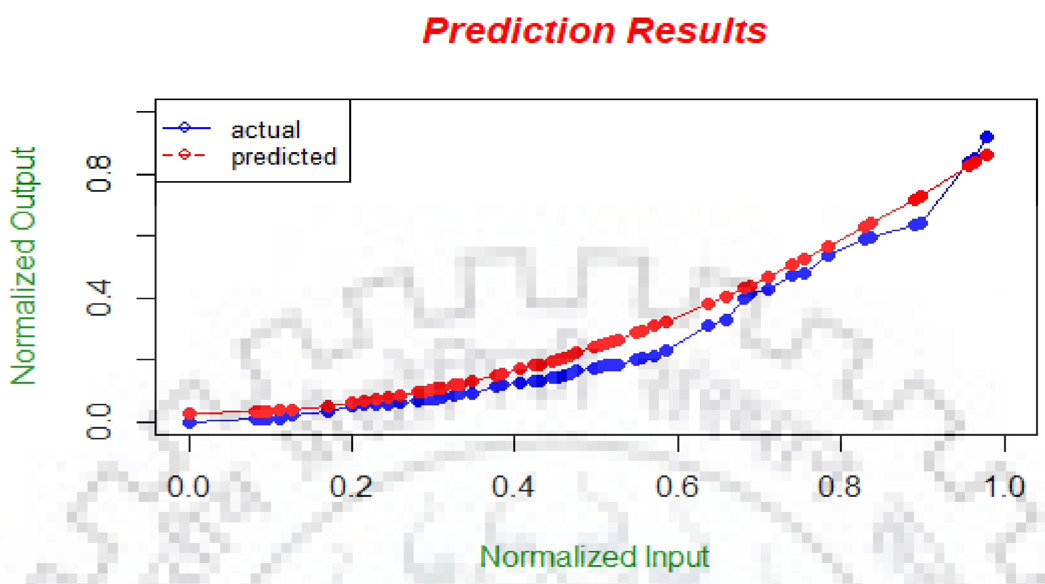


Figure 4.33: Interpretation of RBFN for dataset Musa DB1

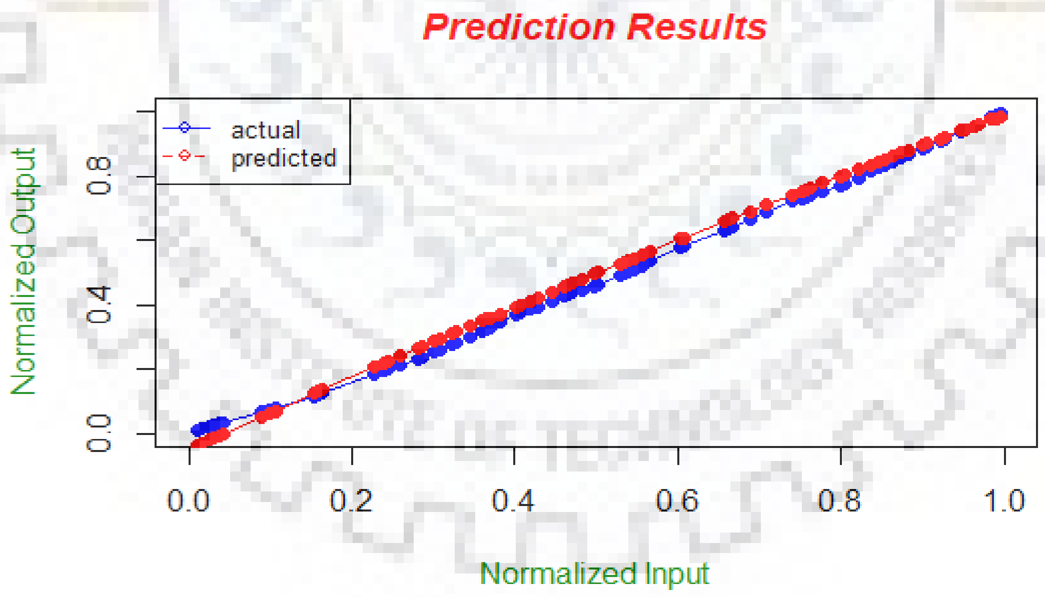


Figure 4.34: Interpretation of RBFN for dataset Musa DB2

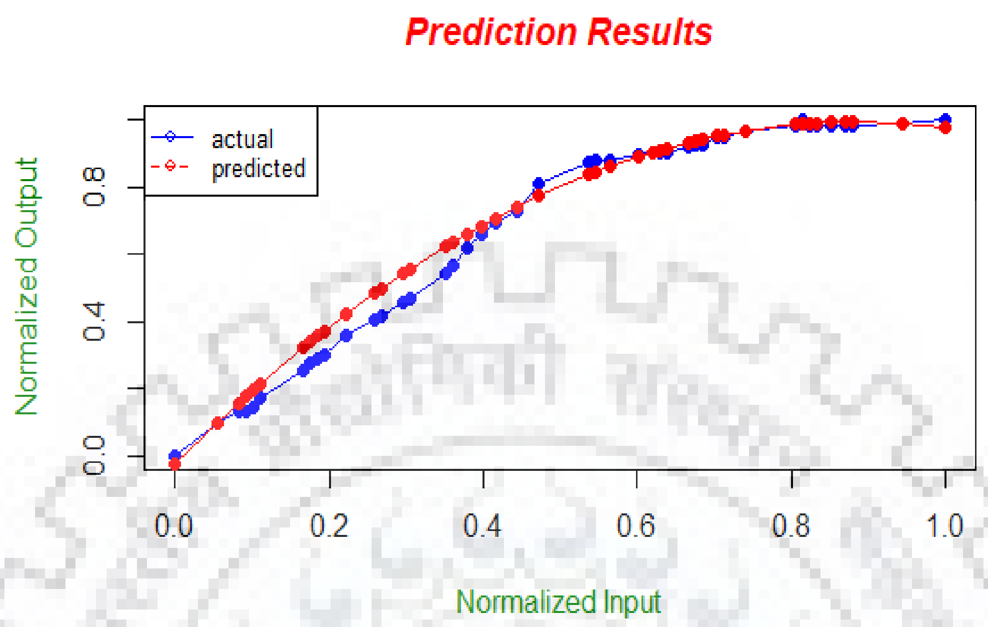


Figure 4.35: Interpretation of RBFN for DATA7

The graphs shown under FFNN are plots between normalized cumulative number of failures and normalized execution time. This shows the accuracy or closeness of actual data (blue in color) and predicted data (red in color). With basic 10 datasets we have the performance calculation. This shows more accuracy than traditional methods. According to error also there is better performance in FFNN compared to analytical methods.

GRNN with function optimization to FFNN has the plots between normalized cumulative number of failures and normalized execution time. We can observe the closeness of actual and predicted data provided in the plots. This model shows optimization and accuracy without back propagation learning algorithm. This will work more efficient with small datasets.

RBFN non-linear classifier is also tested with variety of datasets to see the accuracy level among the neural networks. The plots are between normalized input and normalized output. This model clears the noise in the input data and makes better accuracy among actual and predicted data. It shows efficient results with small datasets.

Out of these results we observe that neural network varies with the type of input dataset and the relation between input and output. Accordingly we have to design the network architecture by modifying the activation function and increasing the hidden neurons count.

4.5 Results and Discussion

Analytical Models	**Average Relative Error**
Logarithmic [8]	16.23
Exponential	17.93
Inverse Polynomial	18.45
Power	26.42
Delayed S-shape	25.61

Table 2: Comparison with Analytical models [6]

The above table shows the error value of standard analytical traditional models. These values are compared with the three methods. The error variation is observed among all the models. With the use of neural networks we have lots of error reduction when compared to traditional models. Table 2 shows the comparison among FFNN, GRNN and RBFN models. We can observe that there is a slight variation in error value and optimization can be achieved. Among all the models RBFN shows the better error reduction due to its ability to reduce noise in input data and function optimization nature.

The graph shows the performance variation of three models with 10 different datasets. From table 3 we can clearly see the average relative error rate among three proposed models. Based on these values we have a plot which displays the better performance model among all the three models.

Datasets	**FFNN Average Relative Error**	**GRNN Average Relative Error**	**RBFN Average Relative Error**
CSR1	4.7137	3.8103	2.8126
DATA7	0.7668	0.3623	0.5014
Main	0.9131	0.8004	1.2204
Musadataset1	1.4342	0.3326	0.5048
Musadataset2	0.8837	0.2014	0.2394
DB1	7.8229	6.7563	2.3809
DB2	6.9438	6.3405	2.8639
DB3	8.4658	6.9671	2.3516
DB4	6.0469	5.8719	1.9340
DB5	6.2307	4.9549	3.0551

Table 3: Comparison of Errors among proposed models with different datasets

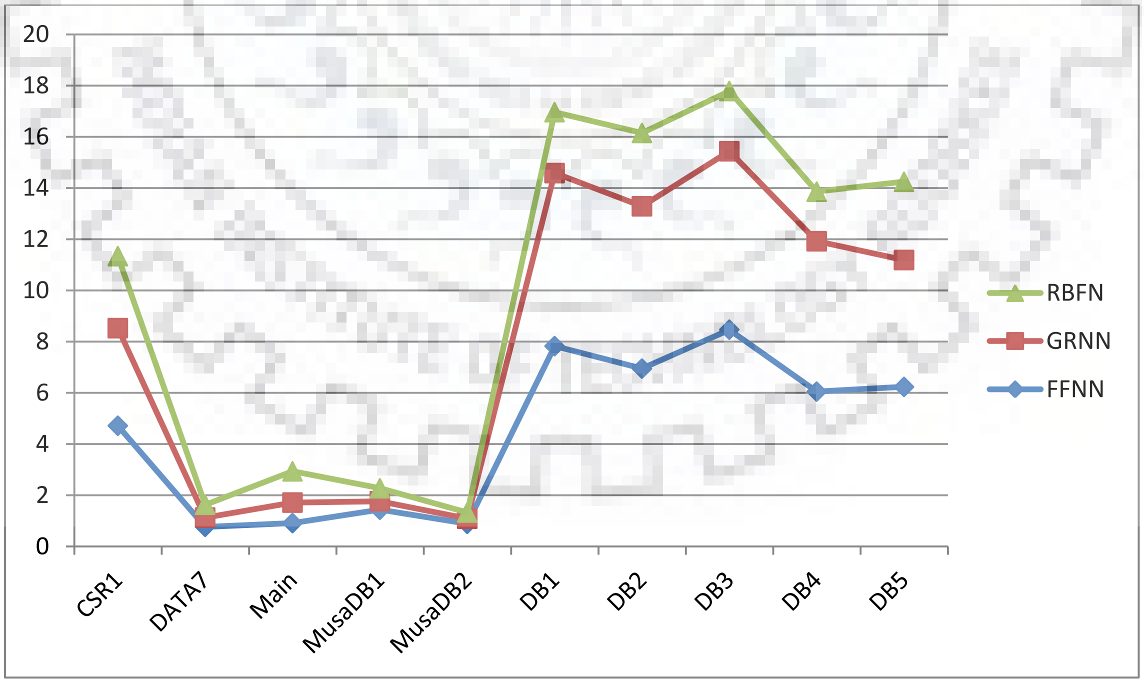


Figure 4.36: Comparison among proposed models

CHAPTER 5

CONCLUSION AND FUTURE WORK

We have observed positive results with the execution o feed forward with back propagation learning algorithm, generalized regression neural network and radial basis function network. From the results section we can have clarity that NN accomplish superior results in terms of minimum error value during prediction in contrast to traditional analytical models. Thus we can finalize that neural networks will be the best option for prediction of software reliability. Through the plots provided in earlier sections we can easily observe the variation between existing models and NN methods. In the connectionism networks we have arbitrarily initialized the weights to communication links. Due to this nature we have varied interpretations and variety of responses with same dataset; hence network behavior also changes accordingly. This final concludes that the functionality of NN models is mostly influenced by the essence or complexion of datasets provided. Larger the datasets better the performance of NN models. All the three models proposed here are certainly well suited with various regularized datasets.

In the other models without back propagation the input data noise is reduced and accuracy is established. Among all the FFNN with back propagation is more accurate though others are fast in execution time. All these methods are a step enhancements in the basic feed forward neural network for better and efficient accuracy in reliability prediction compared to all analytical models.

We can also have the combination of various artificial neural networks like fuzzy logic [20]. With real time critical systems we can have base model Markov chain and Petri net design to enhance the neural performance accuracy. The further work will be based on the replacement of neural hidden layers and extension of NN with Petri nets and fuzzy logics.

CHAPTER 6

REFERENCES

[1] A.Wood, “Software Reliability Growth Models”, Tandem Computers 10300, Tandem Technical Report 96.1, Part Number 130056, ©Tandem Computers, 1996.

[2] J. D. Musa, “Software Reliability Data,” Data & Analysis Centre for Software, January 1980.

[3] R. Iyer and I. Lee, “Measurement-based analysis of software reliability,” Handbook of Software Reliability Engineering, McGraw-Hill, pp. 303 – 358, 1996.

[4] J. D. Musa and K. Okumoto, “A Logarithmic Poisson Execution Time Model for Software Reliability Measurement,” in ICSE, EEE Press Piscataway. NJ, USA: Proceedings of the 7th International Conference on software Engineering, pp. 230–238, 1984.

[5] N. Karunanithi et al, “Prediction of Software Reliability Using Neural Networks,” in Proceedings IEEE International Symposium on Software Reliability Engineering. Austin, TX: IEEE, pp. 124–130, May 1991.

[6] T. M. Khoshgoftaar et al, “A Neural Network Approach for Predicting Software Development Faults.” Research Triangle Park, NC: Proceedings of Third International Symposium on Software Reliability Engineering, pp. 83–89, October 1992.

[7] Y. Singh and P. Kumar, “Prediction of Software Reliability Using Feed Forward Neural Networks”, in Computational Intelligence and Software Engineering (CiSE), I. Conference, Ed. IEEE, pp. 1–5, 2010.

[8] R. Sitte, “Comparison of software-reliability-growth predictions: neural networks vs. parametric-recalibration”, Reliability, IEEE Transactions, vol. 48, no. 3, pp. 285–291, September 1999.

[9] M. K. Bhuyan et al, “Software Reliability Assessment using Neural Networks of Computational Intelligence Based on Software Failure Data”, Baltic J. Modern Computing, Vol. 4 No. 4, pp. 1016–1037, 2016.

[10] S. Ramasamy and I. Lakshmanan, “Application of Artificial Neural Network for Software Reliability Growth Modeling with Testing Effort”, Indian Journal of Science and Technology, Vol 9(29), DOI: 10.17485/ijst/2016/v9i29/90093, August 2016.

[11] R. Shadmehr and D. Z. DSArgenio, "A Comparison of a Neural Network Based Estimator and Two Statistical Estimators in a Sparse and Noisy Data Environment", in IJCNN, vol. 1, Washington D.C, pp. 289–292, June 1990.

[12] T. M. Khoshgoftaar et al, "A Neural Network Approach For Predicting Software Development Faults.", Research Triangle Park, NC: Proceedings of Third International Symposium on Software Reliability Engineering, pp. 83–89, October 1992.

[13] M. M. T. Thwin and T. S. Quah, Eds., "Application of Neural Network for Predicting Software Development Faults using Object-Oriented Design Metrics", vol. 5. Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02), November 2002.

[14] N. Karunanithi and D. Whitley, "Prediction of Software Reliability Using Feed forward and Recurrent Neural Nets", in Neural Networks, 1992. IJCNN, vol. 1. Baltimore, MD: IEEE, pp. 800–805, June 1992.

[15] L. Tian and A. Noore, "Software Reliability Prediction Using Recurrent Neural Network with Bayesian Regularization", International Journal of Neural Systems, vol. 14, no. 3, pp. 165–174, June 2004.

[16] N. RajKiran and V. Ravi, "Software Reliability Prediction using Wavelet Neural Networks", in International Conference on Computational Intelligence and Multimedia Applications, vol. 1. Sivakasi, Tamil Nadu: IEEE, pp. 195 – 199, December 2007.

[17] J. H. Lo, "The Implementation of Artificial Neural Networks Applying to Software Reliability Modeling", Control and Decision Conference, 2009. CCDC '09, Chinese, pp. 4349 – 4354, June 2009.

[18] L. Zhao et al, "Software reliability growth model based on fuzzy wavelet neural network", in 2nd International Conference on Future Computer and Communication (ICFCC), vol. 1. Wuhan: IEEE, pp. 664– 668, May 2010.

[19] P. Werbos, "Generalization of Back propagation with Application to Recurrent Gas Market Model", Neural Network, vol. 1, pp. 339–356, 1988.

[20] R. G. Al gargoor and N. N. Saleem, "Software Reliability Prediction Using Artificial Techniques", IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 4, No 2, July 2013.

[21] M. K. Bhuyan et al, “A Survey of Computational Intelligence Approaches for Software Reliability Prediction”, ACM SIGSOFT Software Engineering Notes, Vol. 39, No 2, March 2014.

[22] D. F. Specht, “A General Regression Neural Network”, IEEE TRANSACTIONS ON NEURAL NETWORKS. VOL. 2. NO. 6. NOVEMBER 1991.

[23] https://minds.wisconsin.edu/bitstream/handle/1793/7779/ch2.pdf?sequence=14, General Regression Neural Network

[24] http://mccormickml.com/2013/08/15/radial-basis-function-network-rbfn-tutorial/, Radial basis function network.