CHAPTER 1

INTRODUCTION

This chapter discusses about software reliability, motivation behind this work. It also talks about objective of this study. Followed by the organization of the thesis.

1.1. SOFTWARE RELIABILITY

The IEEE Standard Glossary of Software Engineering Terminology definition of Software Reliability is “The ability of the software to perform its required function under stated conditions for a stated period of time” [1]. With the rapid growth and increasing complexity of the software, the software reliability is hard to achieve. Reliability is one of the important and crucial aspect and attribute of the software quality.

According to ANSI, Software Reliability is defined as “The probability of failure free software operation for a specified period of time in a specified environment” [32]. Software Reliability Growth Models (SRGM) has been used for predicting and estimating number of errors remaining in software. The primary goal of Software Reliability Modeling is to find out the probability of a system failing in given time interval or the expected time span between successive failures.

Software reliability is an extremely significant customer oriented attribute of software quality. Software reliability can be defined as the ability of a software system to execute according to its required functionalities under stipulated circumstances for a stipulated period of time. Software has become a crucial part of a large number of industrial, military and commercial systems. Since the software applications pervade almost every aspect of today's living, any failure in a software system affects us.

A significant issue in manufacturing high performance software is that it should fulfill the demands of the customer. A software engineer should be able to use machine learning to measure software reliability. Hence, the developers try to measure how reliable the software they have created is, and match the current number of failures experienced by the software with its past failure history. If a software experiences fewer failures as the time goes by, the software is said to be becoming more reliable. Software reliability estimation is exceedingly useful in forecasting the software failures along the time. In this way, the reliability a software system exhibits in the future can be judged to a great accuracy.

ML techniques have proved to be successful in predicting better results than statistical methods and can be used for prediction of software failures more accurately and precisely. These techniques envisages past failure data as input and quite less assumption is required for modeling of the software's with complex phenomena.

ML is an approach which is focused on learning automatically and allows computers to evolve and predict the system behavior based on past and the present failure data. Thus it is quite natural

for software practitioners and researchers to know that which particular method tends to work well for a given failure dataset and up to what extent quantitatively [2-6].

An adequate definition of software reliability prediction is:

> A forecast of how reliable an executable software system will be at some point in the future based on metrics (data) available now.

The prediction of software reliability is very useful because the ability to estimate the reliability of a software system would enable project management to better perform product assurance. Another important reason is the readiness for the release of the software can be evaluated. Thirdly, for achieving the software reliability goals for the system while minimizing cost and/or schedule for the project estimating the reliability of a software system is very important.

The biggest irony of the software industry is that the largest cost associated with any software product over its lifetime is actually its maintenance cost, i.e., the expenses incurred after the deployment of the software. The number of failures that might occur per unit time after the deployment of the software can be known to a great accuracy using software reliability estimation. The maintenance cost can thus be optimized to an extent if the reliability of the software is estimated beforehand.

1.2. MOTIVATION OF THE WORK

For the state of the art research, efforts are constantly being made to assess and improve the software reliability. If problems/ failures are discovered later, they may become inevitable and at times may prove to be costly to address. This shows that software reliability is indispensable part of software quality and is of prime importance.

A software engineer should be able to use machine learning to measure software reliability. Hence, the developers measure how reliable the software they have created is, and match the current number of failures experienced by the software with its past failure history. If a software experiences fewer failures as the time goes by, the software is said to be becoming more reliable. Software reliability estimation is exceedingly useful in forecasting the software failures along the time. In this way, the reliability a software system exhibits in the future can be judged to a great accuracy.

1.3. STUDY OBJECTIVE

Business applications which are critical in nature require reliable software, but developing such software's is a key challenge which our software industry faces today. With the increasing complexity of the software these days, achieving software reliability is hard to achieve. Software Reliability Growth Models (SRGM) has been used for predicting and estimating number of errors remaining in the software. The primary goal of Software Reliability Modeling is to find

out the probability of a system failing in given time interval or the expected time span between successive failures.

In this study, we attempt to empirically assess the use of Adaptive Neuro Fuzzy Inference System (ANFIS) for predicting the software failures. Other Machine Learning methods used for predicting software reliability are Feed Forward Back propagation Neural Network (FFBPNN), General Regression Neural Network (GRNN), Support Vector Machines (SVM), Multilayer Perceptron (MLP), Bagging, Cascading Forward Back propagation Neural Network (CFBPNN), Instance Based Learning (IBK), Linear Regression (Lin Reg), M5P, RepTree, M5Rules. Although ANN, SVM etc has been previously used in literature [14] but for the first time ANFIS has been applied to cumulative week failure dataset. In this work, in order to make the most realistic and efficient comparison we have also analyzed the same data sets for above mentioned Machine Learning techniques. The background of using ANFIS was that if it had proven empirically to predict the software failures with least errors in comparison to other above mentioned techniques, then it may possibly be used as a sound alternative to other mentioned existing techniques for software reliability predictions. Also other above mentioned Machine Learning techniques were empirically analyzed for the first time together on five different types of data sets taken altogether.

1.4. ORGANIZATION OF THE THESIS

The thesis is organized as follows:

Chapter 2 discusses the previous work done in the field of software reliability prediction. This includes the extensive study of various techniques that have been used in predicting the software reliability using machine learning methods in the literature so far. It also highlights some of the most relevant works in the direction of field of work presented in the thesis.

Chapter 3 comprises of the empirical data collection and the experiment design employed in this thesis. It gives a comprehensive knowledge about the dependent and the independent variables used in our study. This chapter is dedicated to a profound about the cross validation technique which is being employed in our experiment and tools used for empirical assessment.

Chapter 4 focuses on the research methodology used in our study. We exemplify the use of training and the testing data sets for conducting experiments. We also briefly discuss about the various machine learning techniques employed in this work.

Chapter 5 highlights the statistical efficacy measures used. It comprises of the experiment design employed in our work.

Chapter 6 presents a detailed analysis of the results obtained. In this chapter we compare and assess the results of the experiments. It highlights the comparative analysis of predictions for

various datasets using machine learning methods based on correlation coefficient, MARE, MRE and MSE.

Chapter 7 deals with the conclusions and future work. It also mentions about the discussions and summary of the results.

CHAPTER 2

LITERATURE SURVEY

Several Machine Learning techniques have been proposed and applied in the literature for software reliability modeling and forecasting. Some of the methods are Genetic Programming, Gene Expression Programming, Artificial Neural Network, Decision Trees, Support Vector Machines, Feed Forward Neural Network, fuzzy models, Generalized neural Network etc [13-19, 21, 27].

Karunanithi et al. (1992) [16] carried out analysis of detailed study to explain the use of connectionist models in the reliability growth prediction for the software's. Cai et al. (1991) [21] focused on the development of fuzzy software reliability models instead of probabilistic software reliability models as he says that reliability is fuzzy in nature.

The main advantage of artificial neural network and other machine learning techniques over is that it requires only past failure data as inputs, and very less assumption required for modeling

complex failure phenomena of software. ML is an approach concerned with the design and development of algorithms which allow computers to evolve the system behavior based on present and the past failure data of the software. Thus machine learning techniques are focused on learning automatically, recognizing complex patterns and making intelligent decisions based on past failure data. So that a machine is able to learn whenever it changes its structure, program, or data based on its inputs or in response to the external information in such a manner that it's expected future performance improves significantly (1995) [28]. Thus it is quite natural for software practitioners and researchers to know that which particular method tends to work well for a given failure dataset and up to what extent quantitatively (2006, 2009, 2010) [2-6].

Alternatively, few traditional statistical methods such as maximum likelihood estimation, least square estimation, linear regression analysis and logistic regression have also been applied for reliability estimation and prediction of soft wares [34-35].

Due to large computation and low learning rate of prediction model, the ML techniques using fuzzy inference system is found to be more effective than classical machine learning (Mueller and Lemke 1999) [39].

Ho et al. (2003) [22] carried out a comprehensive study of connectionist models and their applicability to software reliability prediction and inferred that these are better as compared to traditional modes. Su and Huang (2006) [23] had applied neural network for predicting software

reliability. Madsen et al., 2006 [24] focused on the application of Soft Computing techniques for software reliability prediction.

Pai and Hong (2006) [25] performed experiments using support vector machines (SVMs) for forecasting software reliability. Despite of recent advances in this field, it was observed that different models have varied predictive reliability capabilities.

The applications of support vector machines based ML approach in place of traditional statistical techniques has shown a remarkable improvement in the prediction reliability of the software in the recent years (Xingguo and Yanhua 2007) [38]. Support vector machines represents state of the art because of their generalization performance, ease of usability and rigorous theoretical foundations that practically can be used for modeling of the complex software failure behavior. The design of SVM is based on the extraction of a subset of the training data that serves as support vectors and therefore represents a stable characteristic of the data. The major limitation of SVMs is the increasing computational and storage requirement with respect to the number of training examples (Chen et al. 2008) [37]. Ping and Hong (2006) investigated the capabilities of SVMs for the prediction of software reliability with the help of simulated annealing algorithms (SA) [36].

CHAPTER 3

RESEARCH BACKGROUND

This chapter summarizes the details about the empirical data which is being collected for this study. It briefs about the independent and the dependent variables which are being used for conducting the experiment. It also mentions about the cross validation techniques which is being employed for creating training and test datasets. It highlights the tools which are used for conducting experiments for empirical assessment of the ML techniques used.

3.1. EMPIRICAL DATA COLLECTION

In this study we have used software failure data from various projects given in (Project Data) [7, 9]. We also use the data collected from Tandem Computers Software Data Project. This set of failure data, is from two of four major releases of software products at Tandem Computers (Project Data) (release 1 & 2) [8]. Other data sets include Telecommunication System Data. This data set was reported by (Project Data) (phase 1 & 2) [11] based on system test data for a telecommunication system. System test data consisting of two releases (Phases 1 and 2). In both

tests, automated test and human-involved tests are executed on multiple test beds. Also we have used data from the project On-Line Data Entry IBM Software Package. The data reported by (Project Data) [12] are recorded from testing an on-line data entry software package developed at IBM. All the datasets contain weeks as testing time input and failures as outputs. The different datasets which we have taken for analysis contains data for different no. of weeks which is being summarized below:

Dataset	No. of weeks
Project Data	**28**
Project Data(release2)	**19**
Project Data(release1)	**20**
Project Data(phase1)	**21**
Project Data(phase2)	**21**

The Waikato Environment for Knowledge Analysis (Weka) version 3.6.0 is a comprehensive and is a freely open source suite which is used to conduct the experiments. Also MATLAB tool version 7.9.0 is used to implement the various above mentioned techniques using 10 cross fold validation [20].

3.2. DEPENDENT & INDEPENDENT VARIABLES

The "dependent variable" represents the output or effect, or is tested to see if it is the effect. The "independent variables" represent the inputs or causes, or are tested to see if they are the cause. An independent variable is also known as a "predictor variable and a dependent variable is also known as a "response variable".

The dependent variable which we have used in our study is Failure Rate and the independent variable used is time interval in terms of weeks. As the number of faults changes, the failure rate changes accordingly. In our study, we predict the dependent variable based on the number of failures to be detected using various machine learning techniques. For the corresponding failures, testing time in terms of weeks is chosen to be the independent variable.

3.3. CROSS FOLD VALIDATION

We performed the comparative analysis based on predicting failures envisaging above mentioned machine learning methods using 10 cross fold validation. We have divided entire dataset into two parts: training and testing data set. The training data set is applied to the Machine Learning models for predicting software reliability. The training and testing dataset selection is being employed using k fold cross validation procedure where the entire data set is divided randomly into k subsets (k=10) and every time one of the k subsets is used as training data and the remaining (k-1) subsets are being used as testing data set so as to validate the prediction model for software reliability i.e. it envisages the application of 10-fold cross validation method which

divides the data set into 10-folds (9 parts are used for training and one part is used for validation taken randomly 10 times and results are recorded for each of the 10 runs) required for training and testing the model. Thus cross validation is used so as to maximize the utilization of failure past data sets by repeatedly resampling the same data set randomly by reordering it and then splitting it into ten folds of equal length [29].

3.4. TOOLS USED FOR EMPIRICAL ASSESSMENT

The Waikato Environment for Knowledge Analysis (Weka) version 3.6.0 is a comprehensive and is a freely open source suite which is used to conduct the experiments. Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Weka is open source software issued under the GNU General Public License.

Also MATLAB tool version 7.9.0 is used to implement the various above mentioned techniques using 10 cross fold validation [20]. MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java. You can use MATLAB for a range of applications, including signal processing and communications,

image and video processing, control systems, test and measurement, reliability prediction, computational finance, and computational biology. MATLAB provides tools to acquire, analyze, and visualize data, enabling you to gain insight into your data in a fraction of the time it would take using spreadsheets or traditional programming languages. With the MATLAB language, you can write programs and develop algorithms faster than with traditional languages because you do not need to perform low-level administrative tasks such as declaring variables, specifying data types, and allocating memory. In many cases, the support for vector and matrix operations eliminates the need for for-loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code. We have developed codes for implementing few of the machine learning algorithms using command line interpreter of MATLAB.

CHAPTER 4

RESEARCH METHODOLOGY

This chapter briefs about the training and the testing data taken for performing empirical analysis. All the datasets contain weeks as testing time input and failures as outputs. It also discusses about the various machine learning techniques being used in this study.

4.1. TRAINING AND TESTING DATASET

In this study, we have used software failure data from various projects given in (Project Data) [7-9, 11-12]. All the datasets contain weeks as testing time input and failures as outputs. Project Data [7] reports failures for 28 weeks. Project Data [8] reports failures for 19 weeks. Project Data [9] reports inter failure times and cumulative failures for 22 weeks whereas Project Data [12] reports inter failure times and cumulative failures for 15 weeks. Project Data [10] reports failures for 20 weeks. Whereas Project Data [11] (phase 1 & 2) reports failures for 21 weeks.

The dependent variable which we have used in our study is Failure Rate and the independent variable used is time interval in terms of weeks. In this study, we predict the dependent variable

based on the number of failures to be detected using various machine learning techniques. For the corresponding failures, testing time in terms of weeks is chosen to be the independent variable.

The training data set is applied to the Machine Learning models for predicting software reliability. The training and testing dataset selection is being employed using k fold cross validation procedure where the entire data set is divided randomly into k subsets (k=10) and every time one of the k subsets is used as training data and the remaining (k-1) subsets are being used as testing data set so as to validate the prediction model for software reliability. Emphasis is laid on the fact that every time unique subsets are being chosen for training and validation so that none of the subsets get repeated while training the model or predicting failures. Here, 10 cross validation is used where nine parts are used for training and one part is used for validation taken randomly10 times and results are recorded for each of the 10 runs. This process is applied to each of the five datasets taken into consideration.

Thus cross validation is used so as to maximize the utilization of failure past data sets by repeatedly re-sampling the same data set randomly by reordering it and then splitting it into ten folds of equal length [29].

4.2. MACHINE LEARNING TECHNIQUES USED

A brief description of the Techniques used is as follows:

4.2.1. Adaptive Neuro Fuzzy Inference System (ANFIS)

The FIS structure was generated using genfis1 function from the Matlab Fuzzy logic Toolbox [20]. The basic steps of the FIS model are: (i) identification of input variables (failure time) and output variable (cumulative failures) (ii) development of fuzzy profile of the input/output variables (iii) defining relationships between input and output variables using fuzzy inference system. Thus FIS is capable of making decisions under uncertainty which can be applied for reliability prediction when applied to unknown failure datasets [30]. The adaptive neuro-fuzzy learning techniques work in a similar manner to that of the neural networks. Fuzzy Logic Toolbox calculates the membership function parameters in such a manner that these parameters best permit the associated FIS (Fuzzy Inference System) to follow the input/output data.

The adaptive network based fuzzy inference system (ANFIS) is a fuzzy inference system executed in an adaptive network and it can establish an input-output relation through the back propagation process with an artificial intelligence style (if then rules of fuzzy inference). ANFIS is a hybrid of two intelligent system models. It combines the low-level computational power of a neural network with the high level reasoning capability of a fuzzy inference system. In the aspect of modeling, ANFIS can easily establish nonlinear functions and it can forecast time sequence of no qualitative relations. The easiest way to understand how the ANFIS model operates is to consider it in two steps. First, the system is trained in a similar way to a neural network with a large set of input data [40]. Then, once trained, the system operates exactly as a fuzzy expert system. The ANFIS are fuzzy Sugeno models put in the framework of adaptive systems to facilitate learning and adaptation [40].

4.2.2 Feed Forward Back propagation Neural Network (FFBPNN)

It contains more than one layer of neurons, just like the cascade-forward back propagation neural networks. A methodology similar to the cascade forward neural networks is carried out in order to build the model and train it. Single layered feed-forward neural networks have one layer of sigmoid neurons which are then followed by an output layer of linear neurons. The input layer with transfer functions that may be sigmoid (or of any other type except linear) permit the network model to learn both nonlinear as well as linear relationships between input variable vectors and output variable vectors. . The first layer has weights coming from the input. Each subsequent layer has a weight coming from the previous layer. All layers have biases. The last layer is the network output. Each layer's weights and biases are initialized. Adaption is done, which updates weights with the specified learning function. Training is done with the specified number of epochs. Performance is measured according to the specified performance function like 'corr2', which measures the correlation coefficient between the actual and predicted outputs.

4.2.3. General Regression Neural Network (GRNN)

It is one of a kind of probabilistic neural network. A probabilistic neural network is quite beneficial because of its ability to adapt to the basic function the data follows even when only a few training data samples are available. It is used for function approximation. A generalized regression neural network consists of a radial basis layer along with a special linear layer. A special parameter called spread is associated with it whose value generally lies close to 1. A large spread leads to a bigger area from the input vector where the input layer will respond with a number of significant outputs. GRNN as proposed by Donald F. Specht, falls into the category of

probabilistic neural networks. The use of a probabilistic neural network is especially advantageous due to its ability to converge to the underlying function of the data with only few training samples available. GRNN helps to predict behavior of systems based on few training samples, predict smooth multi-dimensional curves and interpolate between training samples.

4.2.4. Support Vector Machine (SVM)

This technique was being propounded as a method for classifying data, and is called as support vector regression (SVR). Application of this technique yields a model which is usually affected only by a subset of the data used while training the model. This is made possible since the cost function used for creating the model has no impact of the training points lying beyond the margin. A version of SVM for regression was proposed in 1996 by Vladimir N. Vapnik, Harris Drucker, Christopher J. C. Burges, Linda Kaufman and Alexander J. Smola. This method is called support vector regression (SVR). The model produced by support vector classification (as described above) depends only on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by SVR depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction

4.2.5. Multilayer Perceptron (MLP)

It is a type of FFBPNN. It maps vectors of input data to a vector of appropriate outputs and contains more than one layer of nodes where each of the layers is completely connected to the

next layer. In this type of networks, except for the input nodes, each node has a nonlinear activation function like sigmoid function associated with it.

A multilayer perceptron (MLP) is the most widely used feed-forward neural networks and one of the most popular training algorithms for feed-forward neural networks is the back-propagation algorithm. The back-propagation learning algorithm is developed for multilayer perceptrons is a form of gradient descent. As such, it is vulnerable to local minima in weight space. A number of techniques for forcing back-propagation networks out of a local minimum have been proposed in the literature. One of the simplest is adding a momentum term to the back-propagation formula, which adds an additional vector to the current weight update. The feed forward neural network architecture consists of a number of elements called neurons. These neurons are grouped together to form a layer. Each neuron has a number of inputs and a single output. Each input has an assigned factor or parameter called the weight.

4.2.6. Bagging (Bootstrap Aggregating)

It is a machine learning ensemble. This method is being employed to improve the accuracy as well as stability of the machine learning algorithms which are generally used in statistical regression and/or classification. It reduces the variance and helps avoiding over fitting issue. Usually, it is used with decision tree methods, but it can be applied to other types of methods as well. Bagging is a special case of the model averaging approach. Given a standard training set D of size n, bagging generates m new training sets D_i, each of size $n' < n$, by sampling from D uniformly and with replacement. By sampling with replacement, some observations may be repeated in each D_i. If $n'=n$, then for large n the set D_i is expected to have

the fraction (1 - 1/*e*) (≈63.2%) of the unique examples of *D*, the rest being duplicates. This kind of sample is known as a bootstrap sample. The *m* models are fitted using the above *m* bootstrap samples and combined by averaging the output (for regression) or voting (for classification).

4.2.7. Cascading Forward Back propagation Neural Network (CFBPNN)

These are a type of neural networks consisting of more than one layer of neurons. The weights of first layer originate from the input. The weights of each of the subsequent layer originate from the input and all layers prior to the layer in question. Each layer has biases. The last layer is called the network output. Using the neural network toolbox of MATLAB, weights and biases of each of the layers are initialized. Adaption is a process which updates weights with the stipulated learning function. First, adaption is done to create the model. Then the model is trained using the stipulated number of epochs. Performance is computed according to the stipulated performance function. These are similar to feed-forward networks (discussed later), but include a weight connection from the input to each layer, and from each layer to the successive layers. These networks have a weight connection from input to every successive layer and from every layer to all the following layers. The advantage of the additional connections is that they sometimes improve the speed at which the model is trained. For example, a three-layer network has connections from layer 1 to layers 2, layer 2 to layer 3, and layer 1 to layer 3. The three-layer network also has connections from the input to all three layers. The additional connections might improve the speed at which the network learns the desired relationship.

4.2.8. Instance Based K nearest neighbor (IBk)

It is a technique incorporated in WEKA. It employs k-nearest neighbor algorithm in order to classify data. It is one of the types of lazy learning. It stores the instances in memory while performing training and then compares new instances with these stored instances. It predicts the failure by looking at the k nearest neighbors of a test instance. : IBk in WEKA is a technique that employs the k-nearest neighbor algorithm to classify data. In pattern recognition, the *k*-nearest neighbor algorithm (*k*-NN) is a non-parametric method for classifying objects based on closest training examples in the feature space. *k*-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The *k*-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbors (*k* is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of its nearest neighbor.

The same method can be used for regression, by simply assigning the property value for the object to be the average of the values of its *k* nearest neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

4.2.9. Linear Regression (Lin Reg)

This technique envisages creation of the simplest model using a single input variable and a single output variable. A more complex model would comprise of dozens of input variables. In this

model, there are some independent variables between which a linear relationship is found out to yield result in the form of a dependent variable. To create the model, we use WEKA tool. Regression is one of the easiest techniques to use. This model can be made of one input variable and one output variable. It can get more complex than that, including dozens of input variables. In effect, regression models all fit the same general pattern. There are a number of independent variables, which, when taken together, produce a result — a dependent variable. The regression model is then used to predict the result of an unknown dependent variable, given the values of the independent variables. To create the model, we use WEKA tool, built for data mining tasks. WEKA is started, and then the Explorer is chosen. Explorer screen starts, with the Preprocess tab selected. First the Open File button is clicked and the data file is selected. After data preprocessing, Classify tab is clicked. The first step is that the model we want to build is selected, so that WEKA knows how to work with the data, and how to create the appropriate model. In order to do so and create a regression model, the Choose button is clicked, and then the functions branch is expanded. The Linear Regression leaf is then selected. This tells WEKA that a regression model has to be built.

4.2.10. M5P

It is a technique based on Quinlan's M5 algorithm. Based on linear regression at the nodes this technique merges a schematic decision tree with a possibility of functions. M5P generates M5 model trees, combining a conventional decision tree with the incorporation of linear regression functions at the leaves. Initially in order to build a tree, a decision-tree induction algorithm is

executed first. A splitting criterion is then applied. The splitting criterion, along each branch, minimizes intra-subset variation in class values. This splitting procedure is applied till the class values of each of the instance that reaches a node vary slightly or when just a few instances are left. Then the backward pruning from each of the leaf is being applied. If an inner node is pruned, that node is then converted into a leaf having a regression plane.

4.2.11. Reduced Error Pruning Tree (REPTree)

It is a fast decision tree learner which builds a decision or a regression tree and performs the pruning with back over fitting. It sorts values for numeric attributes only. Missing values are dealt with using other methods such as C4.5's method. REPTree produces a suboptimal tree under the constraint that a sub-tree can only be pruned if it does not contain a sub-tree with a lower classification error than itself.

4.2.12. M5Rules

M5Rules: this technique is also based on M5 algorithm. For problems based on regression, this technique uses divide and conquer method to generate a decision list. With every iteration, it builds a model tree and converts the best leaf into a separate rule.

CHAPTER 5

EMPIRICAL STUDY

This chapter briefs about the efficacy measures (correlation coefficient, MARE, MRE, MSE) used in this study. It also highlights about the flow diagram for the experiment design which is being employed in this study.

5.1. EFFICACY MEASURES USED

Statistical Efficacy Measures used for evaluating performance criteria are as follows:

5.2.1. Correlation Coefficient

Correlation Coefficient measures the intensity of relationship or agreement of predictions with the actual class. This statistics shows that how closely actual and predicted values are correlated with each other i.e. the correlation coefficient will tell you how strongly the two variables are associated. The correlation coefficient tells how accurately we can predict one variable from another, and it ranges from -1.00 to 0.00 to 1.00. A correlation coefficient of 0.00 means that we cannot predict one variable from the other. As the correlation coefficient gets larger (approaches

1.00) or smaller (approaches -1.00), the ability to predict one variable from another improves. When the value is 1.00 (or -1.00) then you can predict one variable from another with perfect accuracy. Whether the value is positive or negative refers to the nature of the relationship. Negative correlation coefficients are obtained when as one variable increases the other decreases.

The correlation coefficient lies between -1 to +1. The positive linear relationship grows stronger as correlation coefficient nears 1, and the negative linear relationship grows stronger as correlation coefficient nears -1. No linear relationship is there if the correlation coefficient is 0. The higher the correlation the better the reliability.

5.2.1. Mean Absolute Relative Error

Mean Absolute Relative Error is a measure of the relative differences between the actual values and the values of the output the model is predicting and is computed as the mean of the absolute difference of corresponding actual value and the corresponding predicted value divided by the corresponding actual value.

$$MARE = \frac{1}{n}\sum_{i=1}^{n} \frac{|\text{Actual Value} - \text{Predicted Value}|}{\text{Actual Value}}$$

5.2.2 Mean Relative Error

Mean Relative Error is another measure of the relative differences between the actual values and the values of the output the model is predicting and is computed as the mean of the difference of corresponding actual value and the corresponding predicted value divided by the corresponding actual value.

$$MRE = \sum_{i=1}^{n} \frac{\text{Actual Value} - \text{Predicted Value}}{\text{Actual Value}}$$

5.2.3. Mean Squared Error

Mean Squared Error is the mean of the square of the difference of corresponding actual value and the corresponding predicted value, i.e., mean of the square of errors.

$$MSE = \sum_{i=1}^{n} \frac{(\text{predicted} - \text{actual})^2}{n}$$

Table 5.1: Efficacy Measures used for evaluating performance criteria

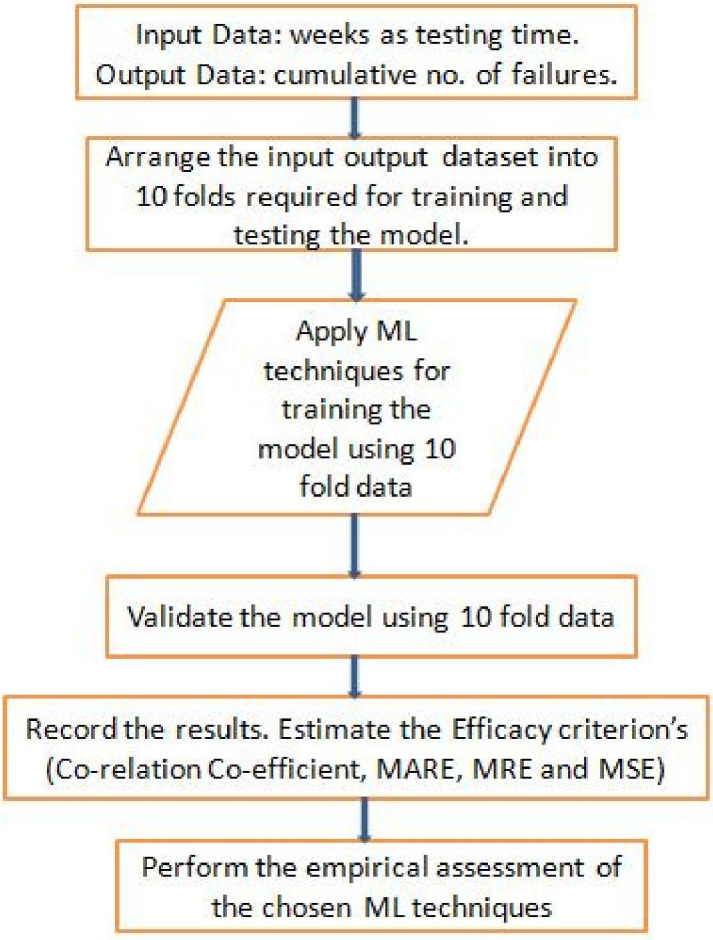
Efficacy Measure	**Definition**
Correlation Coefficient	Correlation Coefficient measures the intensity of relationship or agreement of predictions with the actual class. This statistics shows that how closely actual and predicted values are correlated with each other. The correlation coefficient lies between -1 to +1. The positive linear relationship grows stronger as correlation coefficient nears 1, and the negative linear relationship grows stronger as correlation coefficient nears -1. No linear relationship is there if the correlation coefficient is 0.
MARE	Mean Absolute Relative Error is a measure of the relative differences between the actual values and the values of the output the model is predicting and is computed as the mean of the absolute difference of corresponding actual value and the corresponding predicted value divided by the corresponding actual value. $MARE = \frac{1}{n} \sum_{i=1}^{n} \frac{\lvert \text{Actual Value} - \text{Predicted Value} \rvert}{\text{Actual Value}}$
MRE	Mean Relative Error is another measure of the relative differences between the actual values and the values of the output the model is predicting and is computed as the mean of the difference of corresponding actual value and the corresponding predicted value divided by the corresponding actual value. $MRE = \sum_{i=1}^{n} \frac{\text{Actual Value} - \text{Predicted Value}}{\text{Actual Value}}$
MSE	Mean Squared Error is the mean of the square of the difference of corresponding actual value and the corresponding predicted value, i.e., mean of the square of errors.

	$MSE = \sum_{i=1}^{n} \frac{(predicted - actual)^2}{n}$

5.2. EXPERIMENT DESIGN

The precise view of overall training and prediction process is being illustrated through a flow diagram in Figure 5.1.

Figure 5.1: Overview of software Reliability Prediction Process



In order to conduct this experiment, foremost requisite is the selection of independent and dependent variables. The dependent variable which we have used in our work is Failure Rate and the independent variable used is time interval in terms of weeks. In this study, we predict the dependent variable based on the number of failures to be detected using various machine learning techniques. For the corresponding failures, testing time in terms of weeks is chosen to be the independent variable.

Next step envisages the application of 10-fold cross validation method which divides the data set into 10-folds (9 parts are used for training and one part is used for validation taken randomly 10 times) required for training and testing the model. Then ML techniques are applied for training the model using training data. After this, we validate the models trained above using the testing data. Next step includes recording of failures predicted by applying above mentioned ML techniques. Then we estimate the statistical efficacy measures for all the chosen datasets.

Finally, we perform the empirical assessment of the chosen ML techniques. Based on the comparative analysis, we found that ANFIS produces better results as compared to all other mentioned techniques in terms of predicting software failures for various chosen datasets.

CHAPTER 6

RESULT ANALYSIS

In this section, we present the summary of results obtained for predicting reliability using five data sets which we have taken for comparison using ML techniques in terms of efficacy measures. Efficacy criterions which we have used for model evaluation for software reliability prediction are correlation coefficient between actual and predicted values, MARE, MRE and MSE. Software reliability prediction is phenomena that predict the future failure trends based on the past failure data. These results are then collected and used for analyzing and deciding the appropriate time to release the software.

Software reliability prediction is phenomena that predict the future failure trends based on the past failure data. These results are then collected and used for analyzing and deciding the appropriate time to release the software.

The performance criteria which we have taken in our study are correlation coefficient between actual and the predicted values, MARE, MRE and MSE. Based on these, we obtained the following results for various datasets which we have taken for analysis in our work. The results are being summarized in the following tables [6.1-6.9].

Sometimes, MARE is expressed in %. However, this paper follows the definition given in Table 1 and does not express MARE in % [31].

6.1. Results of Correlation Coefficient

Below is the comparative analysis for failure predictions on five different datasets taken from industrial software's by applying aforesaid machine learning techniques to them based on correlation coefficient.

Table 6.1 shows the results of correlation coefficients on five datasets taken for reliability analysis in this work using 11 ML techniques and one statistical method (Lin Reg) method. The results show that the correlation coefficient is above 0.99 in most of the predictions for ANFIS shows that the actual and the predicted values are very close.

Table 6.1 : Summary of correlation coefficient predictions for different datasets.

Project Data	ANFIS	GRNN	FFBPNN	CFBPNN	BAGGING	IBK	LINEAR REGRESSION	M5P	M5RULES	MULTILAYERED PERCEPTRO	REP TREE	SVM
Project Data	0.999	0.998	0.978	0.99	0.969	0.992	0.891	0.967	0.984	0.994	0.967	0.866
Project Data(release2)	0.999	0.998	0.997	0.984	0.976	0.986	0.966	0.982	0.988	0.998	0.945	0.955
Project Data(release1)	0.999	0.998	0.984	0.987	0.965	0.99	0.961	0.982	0.994	0.998	0.951	0.96
Project Data(phase1)	0.997	0.986	0.978	0.986	0.98	0.978	0.982	0.985	0.984	0.995	0.937	0.984
Project Data(phase2)	0.998	0.997	0.987	0.993	0.98	0.983	0.985	0.986	0.994	0.997	0.94	0.983

And we observed that ANFIS yields remarkable results as compared to other mentioned techniques. It produces correlation coefficient nearest to +1 which depicts that it shows positive correlation coefficient as compared to all other techniques. ANFIS results shows with correlation coefficient values nearer to 1 for various datasets taken into consideration clearly demonstrate that how closely actual and predicted values are correlated with each other. The higher the correlation the better the reliability. Hence we can say that ANFIS predicts failure more accurately than other mentioned techniques. Following the same pattern, the next in the hierarchy lies the GRNN and MLP which also produces positive Correlation Coefficient nearer to 1 and thus can also be used for predicting reliability efficiently and accurately after ANFIS.

6.2. Results of Mean Absolute Relative Error

Table 6.2 shows the values of MARE on five datasets taken for reliability analysis in this work using 11 ML techniques and and one statistical method (Lin Reg) method. The results show that the MARE is between the ranges of 0.025 to 1.5 in most of the predictions for ANFIS is quite lowest in comparison to the MARE's is being calculated by other mentioned techniques.

Table 6.2: Summary of MARE for different datasets.

Project Data	ANFIS	GRNN	FFBPNN	CFBPNN	BAGGING	IBK	LINEAR REGRESSION	M5P	M5RULES	MULTILAYERED PERCEPTRO	REP TREE	SVM
Project Data	0.145	0.293	0.442	0.151	1.031	0.194	1.34	0.832	0.188	0.155	0.832	2.166
Project Data(release2)	0.027	0.053	0.108	0.21	0.266	0.105	0.161	0.109	0.117	0.031	0.256	0.223
Project Data(release1)	0.031	0.059	0.194	0.11	0.162	0.075	0.125	0.083	0.067	0.032	0.158	0.131
Project Data(phase1)	0.14	0.55	0.638	0.219	0.397	0.147	0.246	0.222	0.213	0.145	0.449	0.214
Project Data(phase2)	0.073	0.09	0.249	0.142	0.268	0.141	0.177	0.161	0.141	0.074	0.318	0.194

MARE is a quantity which is used to measure how close forecasts or predictions are to the eventual outcomes. It is one of a number of ways of comparing forecasts with their eventual outcomes. Absolute error is the difference between the magnitude of the true value and the observed value. They are negatively oriented errors, lower values are better. From the above results, we found that ANFIS produces lowest MARE scores as compared to rest of the methods which again proves it to be better.

6.3. Results of Mean Relative Error

Table 6.3 shows the values of MRE on five datasets taken for reliability analysis in this work using 11 ML techniques and and one statistical method (Lin Reg) method. The results show that the error found in most of the predictions for ANFIS is quite lowest in comparison to the errors is being calculated by other mentioned techniques.

Table 6.3: Summary of MRE for different datasets.

Project Data	ANFIS	GRNN	FFBPNN	CFBPNN	BAGGING	IBK	LINEAR REGRESSION	M5P	M5RULES	MULTILAYERED PERCEPTRO	REP TREE	SVM
Project Data	0.025	-0.249	-0.077	-0.005	-0.974	-0.115	-1.191	-0.755	-0.114	-0.074	-0.755	-2.112
Project Data(release2)	-0.012	-0.035	0.066	0.107	-0.22	-0.012	-0.074	-0.04	-0.011	0.006	-0.154	-0.116
Project Data(release1)	-0.015	-0.045	0.036	-0.064	-0.093	-0.015	-0.046	-0.028	-0.023	-0.084	-0.103	-0.058
Project Data(phase1)	0.019	-0.458	0.239	-0.121	-0.307	-0.045	0.077	0.076	0.062	0	-0.304	0.056
Project Data(phase2)	-0.008	-0.044	0.158	0.048	-0.201	-0.036	0.029	0.027	0.008	-0.002	-0.169	-0.013

MRE measures for a given project the difference between actual and predicted failure relative to the actual failures. MRE helps us to compare how incorrect a quantity is from the value considered to be true. From the results obtained above, we observed that ANFIS produces lowest MRE scores as compared to rest of the mentioned techniques.

6.4. Results of Mean Squared Error

Table 6.4 shows the values of MSE on five datasets taken for reliability analysis in this work using 11 ML techniques and one statistical method (Lin Reg) method. The results show that the error is between the ranges of 0.5 to 16.0 in most of the predictions for ANFIS is quite lowest in comparison to the MSE's is being calculated by other mentioned techniques.

Table 6.4: Summary of MSE for different datasets.

Project Data	ANFIS	GRNN	FFBPNN	CFBPNN	BAGGING	IBK	LINEAR REGRESSION	M5P	M5RULES	MULTILAYERED PERCEPTRO	REP TREE	SVM
Project Data	15.838	22.077	231.55	116.534	438.628	85.902	1061.5	352.199	178.3	70.021	352.199	1636.6
Project Data(release2)	1.176	2.717	11.107	67.484	111.949	38	88.726	47.438	12.774	4.069	151.623	118.156
Project Data(release1)	1.658	4.857	43.193	39.131	61.866	15.875	61.834	28.861	12.774	3.286	91.562	64.69
Project Data(phase1)	0.39	2.792	2.607	1.649	3.401	3.321	2.681	2.277	2.381	0.852	10.663	2.781
Project Data(phase2)	0.548	0.618	4.431	3.043	9.001	6.81	5.912	5.516	4.784	1.109	24.313	7.557

The mean squared error (MSE) of an estimator is one of many ways to quantify the difference between values implied by an estimator and the true values of the quantity being predicted. From the above comparison, we found that for various industrial datasets taken into consideration, ANFIS yields lowest MSE scores as compared to other mentioned methods.

6.5. Results of Correlation Coefficient predictions for cumulative vs. interfailure time's data.

These performance criteria were also being observed for the Cumulative as well as Inter Failure times' data sets. Table 6.5 shows the results of correlation coefficients on dataset given in Project Data [9] taken for reliability analysis in this work using 9 ML techniques and one statistical method (Lin Reg) method. The results show that the correlation coefficient is above 0.9 for ANFIS prediction which shows that the actual and the predicted values are very close. It also depicts that the cumulative failures yields high correlation coefficients within the ranges of 0.8 to 0.9 in comparison to the correlation coefficient's being calculated for interfailure time's data.

Table 6.5: Summary of correlation coefficient predictions for cumulative vs. inter failure time's data.

ML TECHNIQUES	INTERFAILURE DATA	CUMMULATIVE DATA
ANFIS	0.307	0.995
GRNN	0.36	0.979
FFBPNN	0.324	0.939
CFBPNN	0.158	0.975
BAGGING	0.126	0.946
IBK	-0.147	0.951
LINEAR REGRESSION	0.264	0.92
MULTILAYERED PERCEPTRON	0.333	0.975
REP TREE	-0.093	0.856
SVM REG	0.392	0.87

From the above results, we found that cumulative failures produce more accurate and precise results as compared to inter failure time's data. Cumulative failures yields correlation coefficient nearer to 1 unlike inters failure time's failures whose correlation coefficient lie more close to -1. The positive linear relationship grows stronger as correlation coefficient nears 1, and the negative linear relationship grows stronger as correlation coefficient nears -1. It shows that how closely actual and predicted values are correlated with each other. Also we observed that ANFIS yields correlation coefficient nearer to 1 i.e. 0.9948 as compared to rest of the other mentioned techniques.

6.6. Results of MARE predictions for cumulative vs. interfailure time's data.

Table 6.6 shows the results of MARE on dataset given in Project Data [9] taken for reliability analysis in this work using 9 ML techniques and one statistical method (Lin Reg) method. The

results show that the value of MARE is 0.28 for ANFIS prediction. It also depicts that the cumulative failures yields low MARE within the ranges of 0.28 to 1.38 in comparison to the MARE's being calculated for interfailure time's data.

Table 6.6: Summary of MARE predictions for cumulative vs. inter failure time's data.

MARE	INTERFAILURE DATA	CUMMULATIVE DATA
ANFIS	91.9628	0.2849
GRNN	104.925	0.3141
FFBPNN	88.7611	1.1917
CFBPNN	94.6268	0.5044
BAGGING	73.1508	1.1231
IBK	152.9859	0.3175
LINEAR REGRESSION	61.497	1.504
MULTILAYERED PERCEPTRON	91.7452	1.3844
REP TREE	84.2668	2.2547
SVM REG	32.0544	0.7983

From above, we observed that cumulative failures yields better results than inter failure time's data. They produce lowest MARE scores as compared to inter failure time's data. Out of all, ANFIS gives lowest MARE score (28.49%) for cumulative failures which again proves it to be the best method amongst the other mentioned methods.

6.7. Results of Correlation Coefficient predictions for cumulative vs. interfailure time's data.

Table 6.7 shows the results of correlation coefficients on dataset given in Project Data [12] taken for reliability analysis in this work using 9 ML techniques and one statistical method (Lin Reg)

method. The results show that the correlation coefficient is above 0.9 for ANFIS prediction which shows that the actual and the predicted values are very close. It also depicts that the cumulative failures yields high correlation coefficients within the ranges of 0.74 to 0.99 in comparison to the correlation coefficient's being calculated for interfailure time's data.

Table 6.7: Summary of correlation coefficient predictions for cumulative vs. inter failure time's data.

ML TECHNIQUES	INTERFAILURE DATA	CUMMULATIVE DATA
ANFIS	0.7088	0.9989
GRNN	0.8974	0.9955
FFBPNN	0.6448	0.9826
CFBPNN	0.6352	0.984
BAGGING	0.7719	0.918
IBK	0.6532	0.9712
LINEAR REGRESSION	0.8676	0.9764
MULTILAYERED PERCEPTRON	0.8068	0.9969
REP TREE	0.6092	0.743
SVM REG	0.817	0.9791

From the above results, we found that cumulative failures produce more accurate and precise results as compared to inter failure time's data. Cumulative failures yields correlation coefficient nearer to 1 unlike inters failure time's failures whose correlation coefficient lie more close to -1. The positive linear relationship grows stronger as correlation coefficient nears 1, and the negative linear relationship grows stronger as correlation coefficient nears -1. It shows that how closely actual and predicted values are correlated with each other. Also we observed that ANFIS yields correlation coefficient nearer to 1 i.e. 0.9989 as compared to rest of the other mentioned techniques.

6.8. Results of MARE predictions for cumulative vs. interfailure time's data.

Table 6.8 shows the results of MARE on dataset given in Project Data [12] taken for reliability analysis in this work using 9 ML techniques and one statistical method (Lin Reg) method. The results show that the value of MARE is 0.04 for ANFIS prediction. It also depicts that the cumulative failures yields low MARE within the ranges of 0.04 to 0.48 in comparison to the MARE's being calculated for interfailure time's data.

Table 6.8: Summary of MARE predictions for cumulative vs. inter failure time's data.

MARE	INTERFAILURE DATA	CUMMULATIVE DATA
ANFIS	0.7983	0.0464
GRNN	0.2157	0.1219
FFBPNN	0.6565	0.4501
CFBPNN	0.6718	0.4887
BAGGING	0.5064	0.2246
IBK	0.2881	0.1858
LINEAR REGRESSION	0.3864	0.1879
MULTILAYERED PERCEPTRON	0.1747	0.1232
REP TREE	0.3925	0.3496
SVM REG	0.2893	0.1875

From above, we observed that cumulative failures yields better results than inter failure time's data. They produce lowest MARE scores as compared to inter failure time's data. Out of all, ANFIS gives lowest MARE score (4.64%) for cumulative failures which again proves it to be the best method amongst the other mentioned methods.

6.9. Results of Pred(0.25) for MRE.

PRED (x) can be described as: = (k / *n*) where *k* is the number of projects in which MRE <= x and n is the total number of projects. The estimation accuracy is directly proportional to PRED (x). Using the above formula, we have calculated Pred(0.25) for various datasets taken into consideration by applying aforesaid mentioned machine learning techniques to them.

Table 6.9 shows the pred(0.25) for different datasets. It also depicts that most of the values of MRE lies below 0.25. Hence the results are very high.

Table 6.9: Summary of Prediction values Pred(0.25) for MRE.

Project Data	ANFIS	GRNN	FFBPNN	CFBPNN	BAGGING	IBK	LINEAR REGRES	MLP	M5P	REPTree	M5Rules	SVM
Project Data	0.964	0.9642	0.9285	0.9642	0.9642	0.9642	1	0.9642	1	1	0.9642	1
Project Data(release2)	1	1	0.9473	0.9473	1	0.9473	1	1	1	0.8947	0.9473	1
Project Data(release1)	1	1	0.95	1	1	1	1	1	1	0.95	1	1
Project Data(phase1)	0.905	1	0.9047	1	1	0.9523	0.9047	0.9047	0.9	0.9523	0.9523	0.9047
Project Data(phase2)	1	1	0.8095	0.9523	1	0.9523	0.9047	0.9523	0.9	0.9047	0.9047	0.9047

Figures below depicts the graphical representation of the analysis of the performances of above mentioned Machine Learning techniques using 10 Fold Cross Validation for predicting Software Reliability.

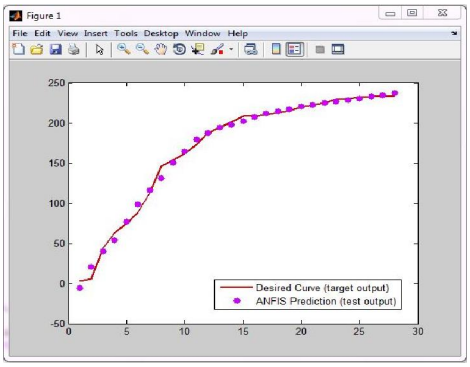


Figure 6.1. ANFIS failure prediction vs. actual failures for Project Data [7].

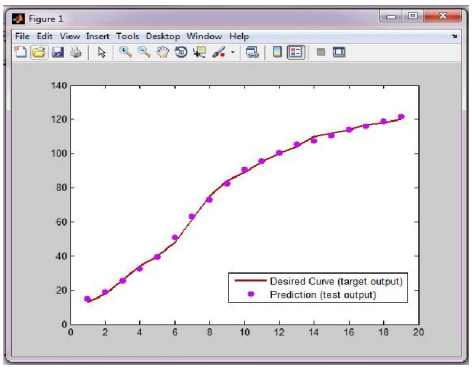


Figure 6.2. ANFIS failure prediction vs. actual failures for Project Data [8].

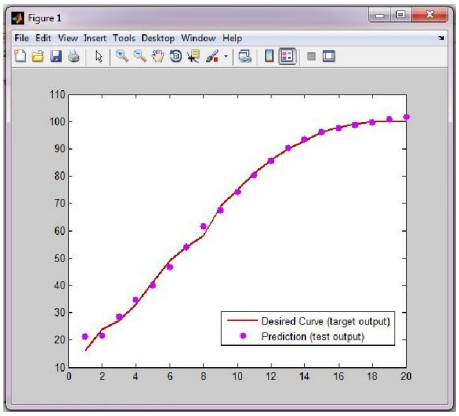


Figure 6.3. ANFIS failure prediction vs. actual failures for Project Data [10].

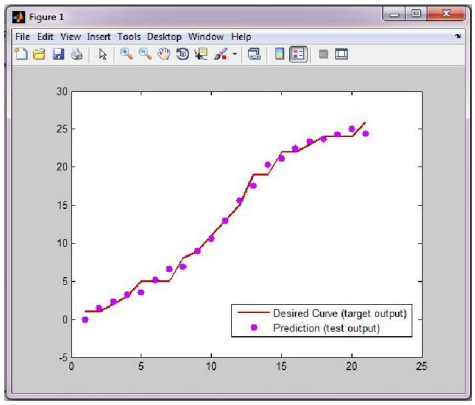


Figure 6.4. ANFIS failure prediction vs. actual failures for Project Data [11](phase1).

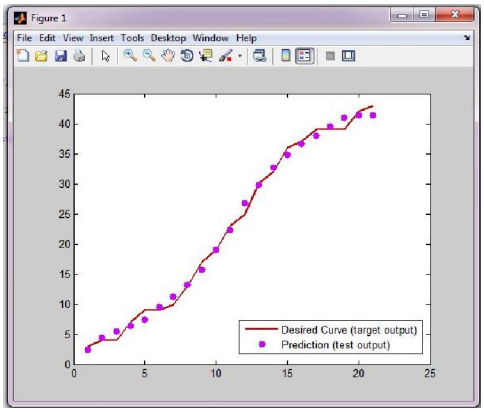


Figure 6.5. ANFIS failure prediction vs. actual failures for Project Data [11](phase2).

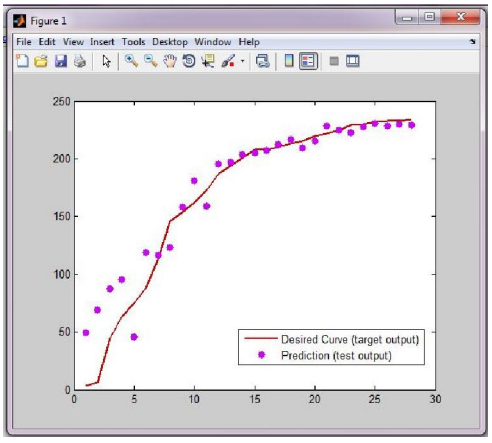


Figure 6.6. Bagging failure prediction vs. actual failures for Project Data [7].

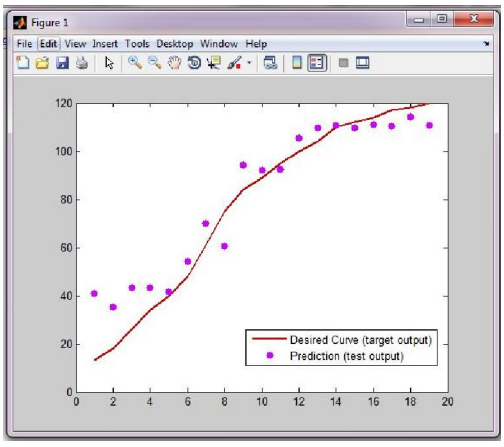


Figure 6.7. Bagging failure prediction vs. actual failures for Project Data [8].

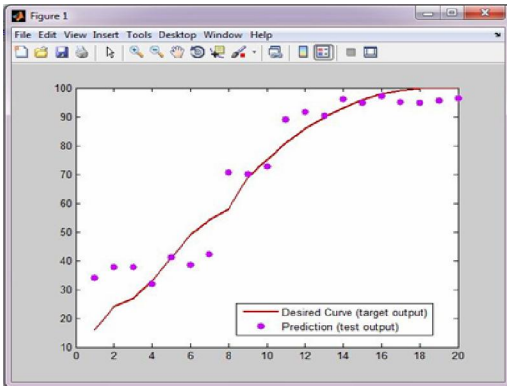


Figure 6.8. Bagging failure prediction vs. actual failures for Project Data [10].

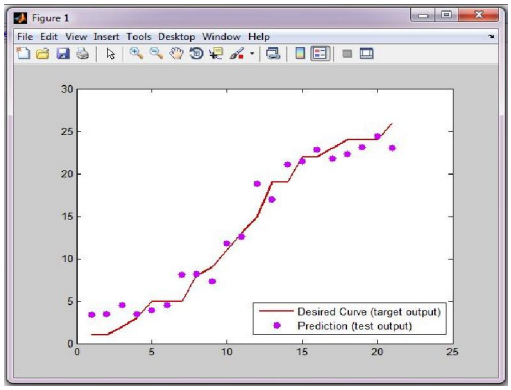


Figure 6.9. Bagging failure prediction vs. actual failures for Project Data [11]([phase1).

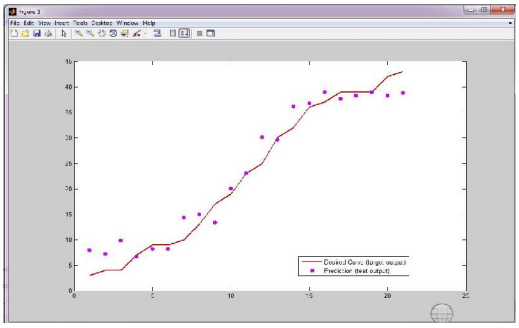


Figure 6.10. Bagging failure prediction vs. Actual failures for Project Data [11](phase2).

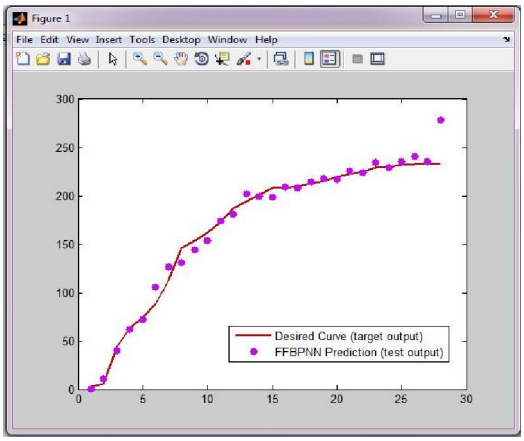


Figure 6.11. CFBPNN failure prediction vs. actual failures for Project Data [7].

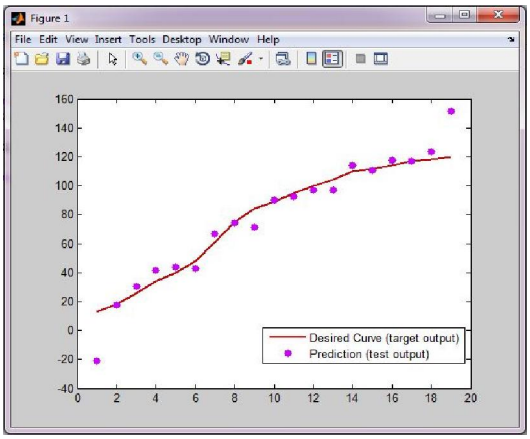


Figure 6.12. CFBPNN failure prediction vs. actual failures for Project Data [8].

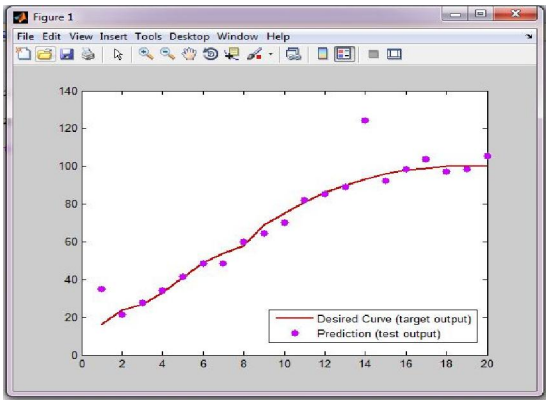


Figure 6.13. CFBPNN failure prediction vs. vs.actual actual failures for Project Data [10].

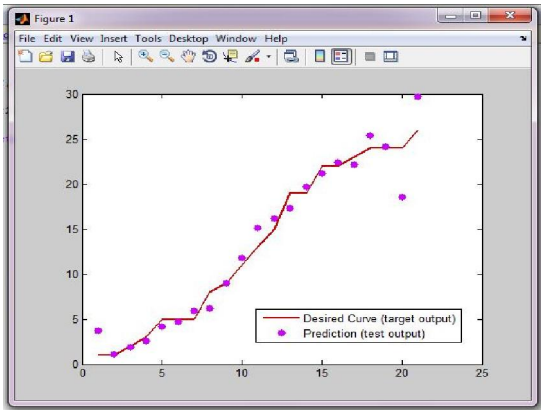


Figure 6.14. CFBPNN failure prediction failures for Project Data [11](phase1).

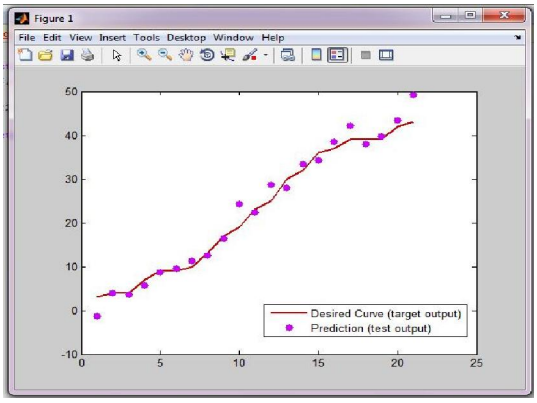


Figure 6.15. CFBPNN failure prediction vs. actual failures for Project Data [11].

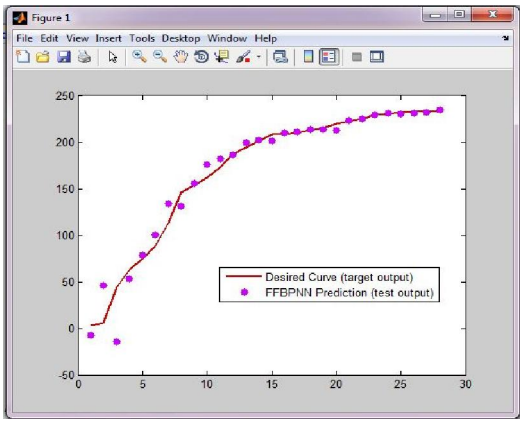


Figure 6.16. FFBPNN failure prediction vs. actual failures for Project Data [7].

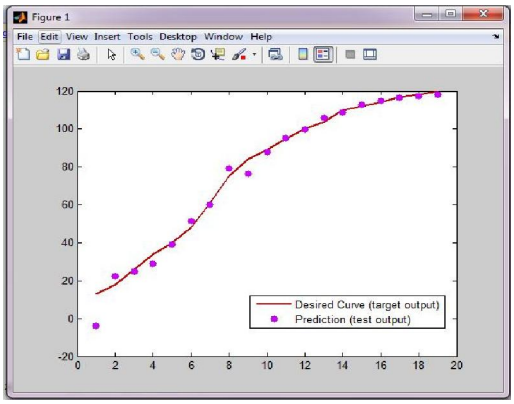


Figure 6.17. FFBPNN failure prediction vs. actual failures for Project Data [8].

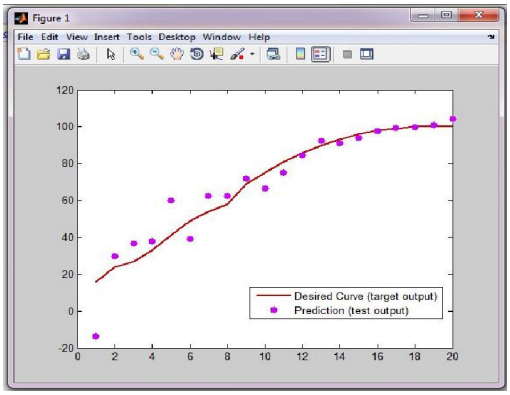


Figure 6.18. FFBPNN failure prediction vs. vs. actual failures for Project Data [10].

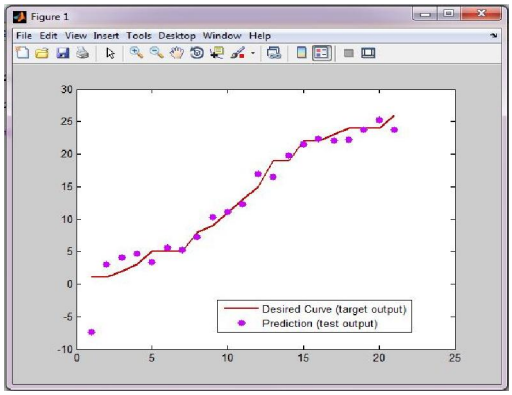


Figure 6.19. FFBPNN failure prediction Actual failures for Project Data [11](phase1).

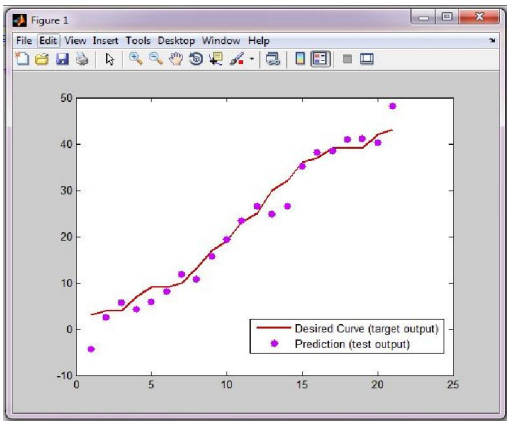


Figure 6.20. FFBPNN failure prediction vs. actual failures for Project Data [11](phase2).

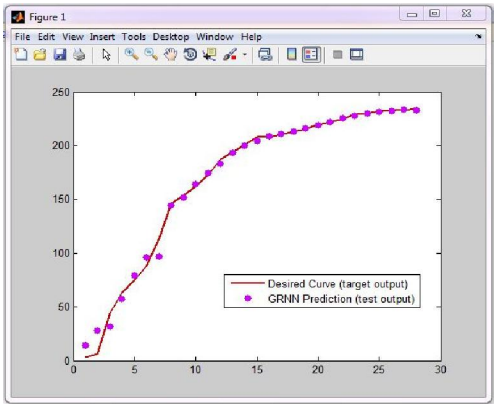


Figure 6.21. GRNN failure prediction vs. actual failures for Project Data [7].

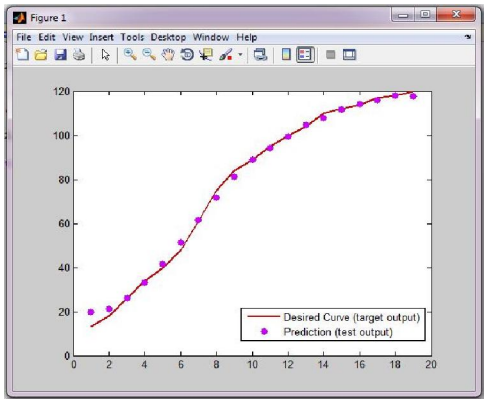


Figure 6.22. GRNN failure prediction vs. actual failures for Project Data [8].

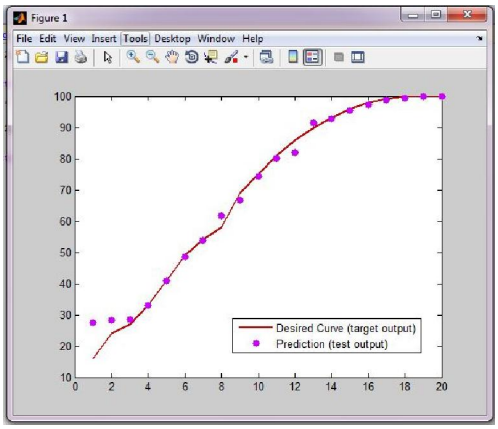


Figure 6.23.GRNN failure prediction vs. actual failures for Project Data [10].

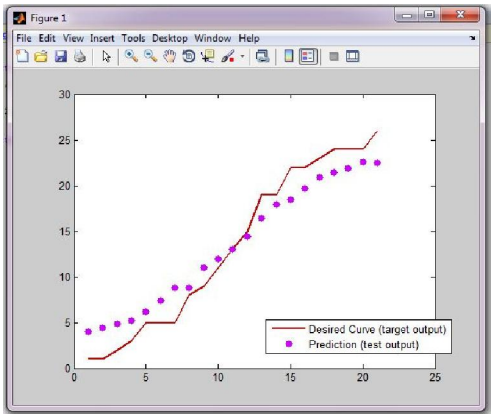


Figure 6.24. GRNN failure prediction vs. actual failures for Project Data [11](phase1).

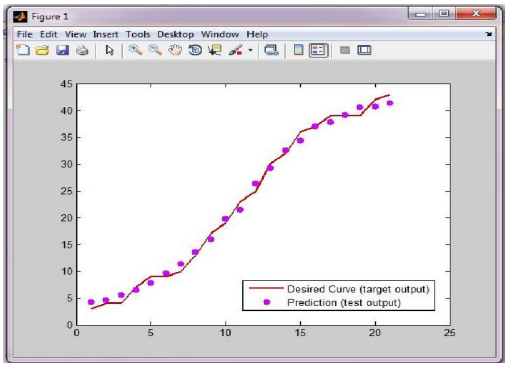


Figure 6.25. GRNN failure prediction vs. actual failures for Project Data [11](phase2).

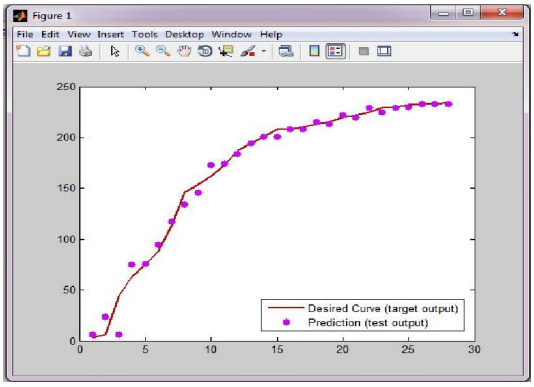


Figure 6.26. IBK failure prediction vs. actual failures for Project Data [7].

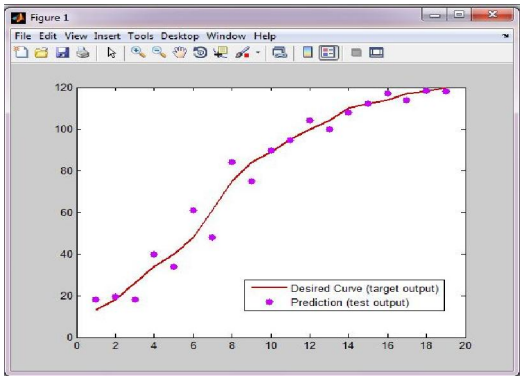


Figure 6.27. IBK failure prediction vs. actual failures for Project Data [8].

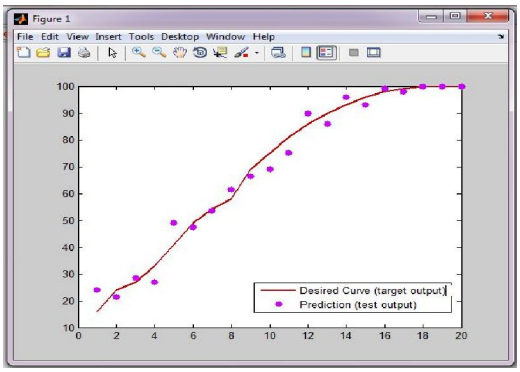


Figure 6.28. IBK failure prediction vs. actual failures for Project Data [10].

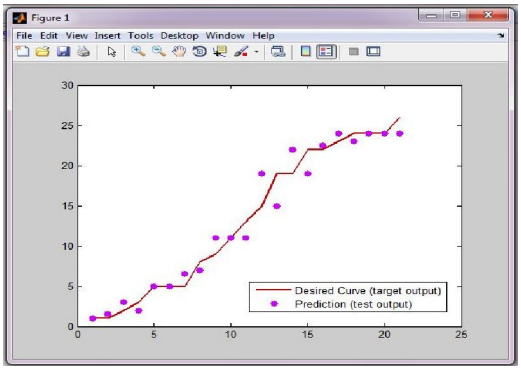


Figure 6.29. IBK failure prediction vs. actual failures for Project Data [11](phase1).

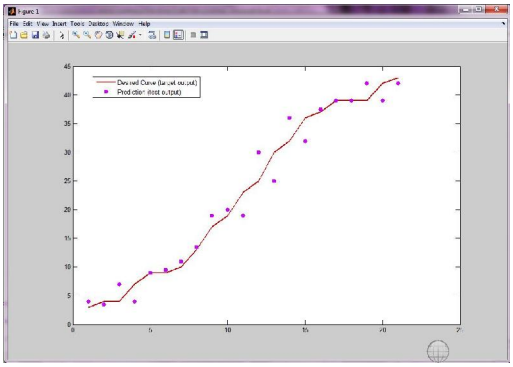


Figure 6.30. IBK failure prediction vs. actual failures for Project Data [11](phase2).

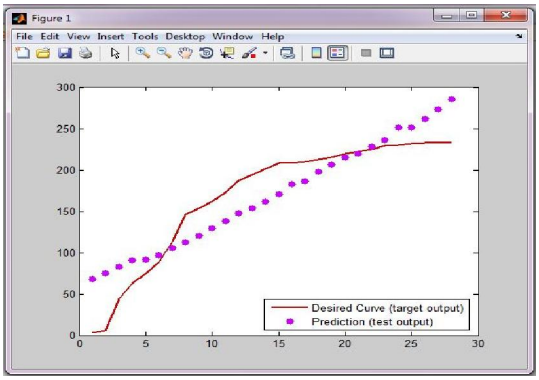


Figure 6.31. LinReg failure prediction vs. actual failures for Project Data [7].

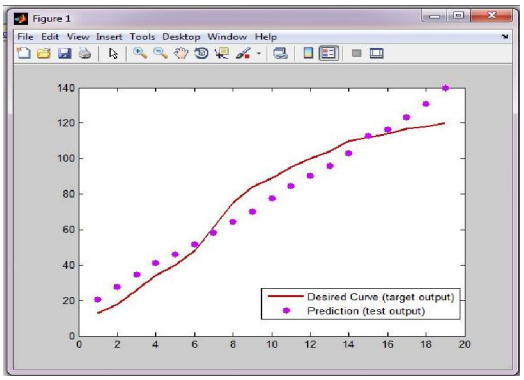


Figure 6.32. LinReg failure prediction vs. actual failures for Project Data [8].

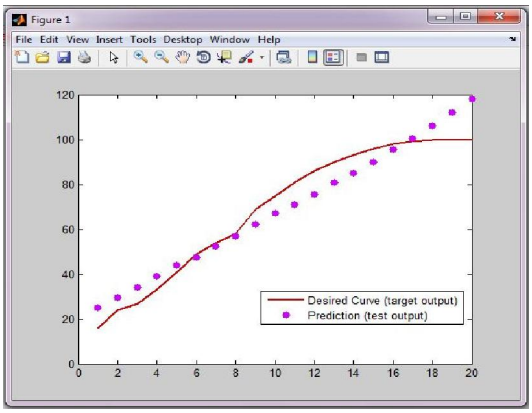


Figure 6.33. LinReg failure prediction vs. actual failures for Project Data [10].

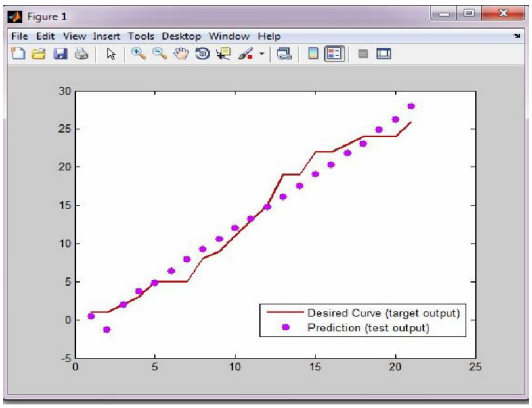


Figure 6.34. LinReg failure prediction vs. actual failures for Project Data [11](phase1).

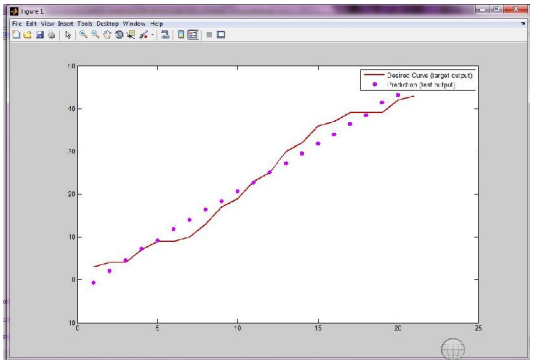


Figure 6.35. LinReg failure prediction vs. actual failures for Project Data [11](phase2).

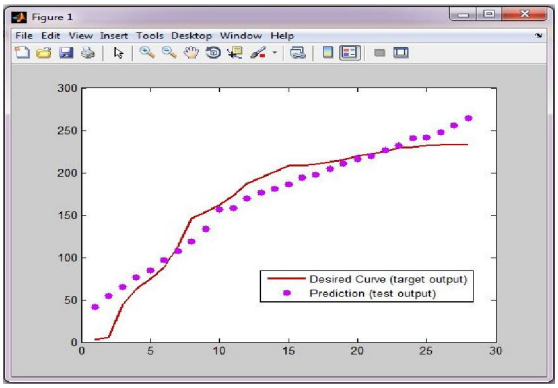


Figure 6.36. M5P failure prediction vs. actual failures for Project Data [7].

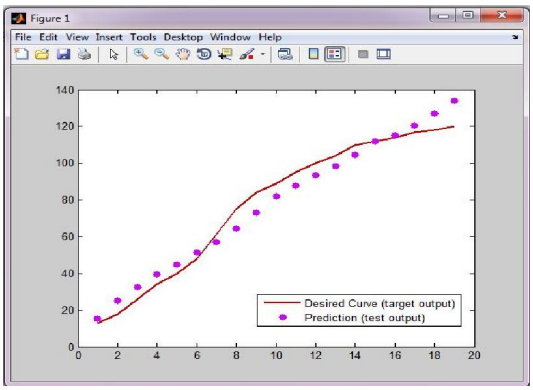


Figure 6.37. M5P failure prediction vs. actual failures for Project Data [8].

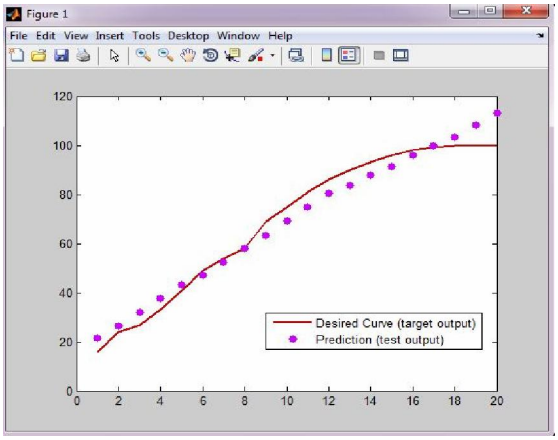


Figure 6.38. M5P failure prediction vs. actual failures for Project Data [10].

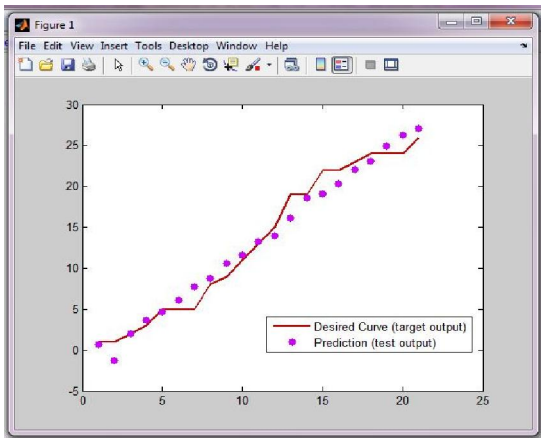


Figure 6.39. M5P failure prediction vs. actual failures for Project Data [11](phase1).

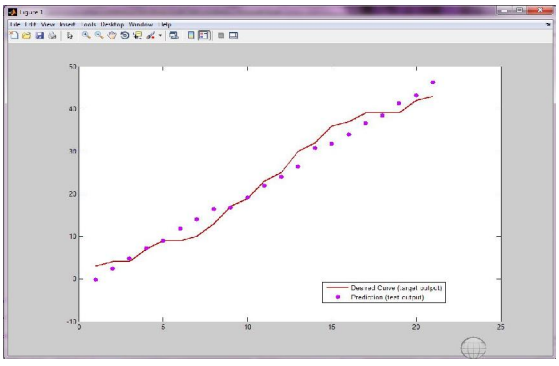


Figure 6.40. M5P failure prediction vs. actual failures for Project Data [11](phase2).

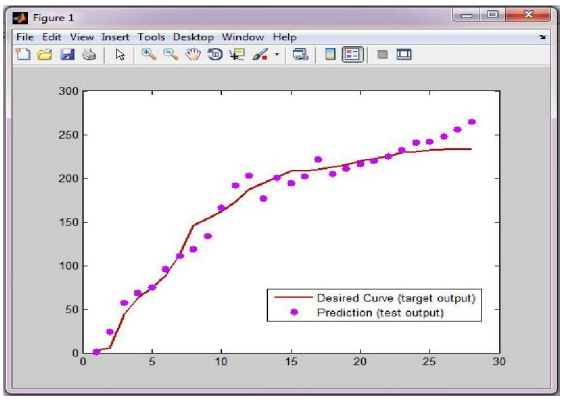


Figure 6.41. M5Rules failure prediction vs. actual failures for Project Data [7].

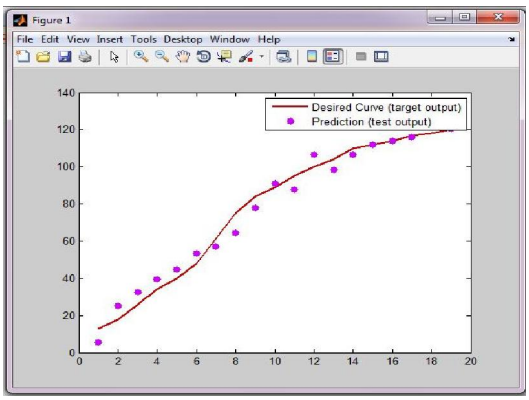


Figure 6.42. M5Rules failure prediction vs. actual failures for Project Data [8].

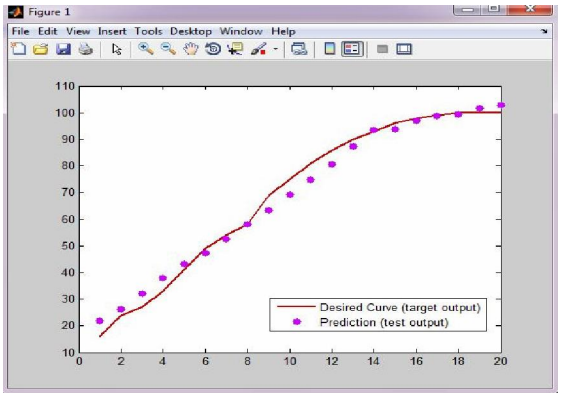


Figure 6.43. M5Rules failure prediction vs. actual failures for Project Data [10].

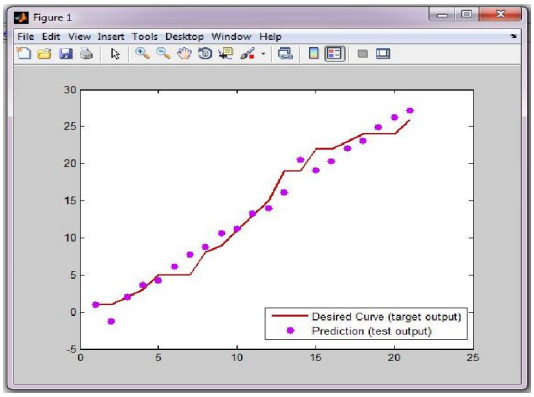


Figure 6.44. M5Rules failure prediction vs. actual failures for Project Data [11](phase1).

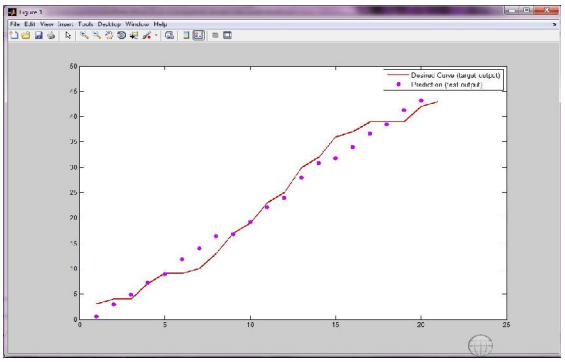


Figure 6.45. M5Rules failure prediction vs. actual failures for Project Data [11](phase2).

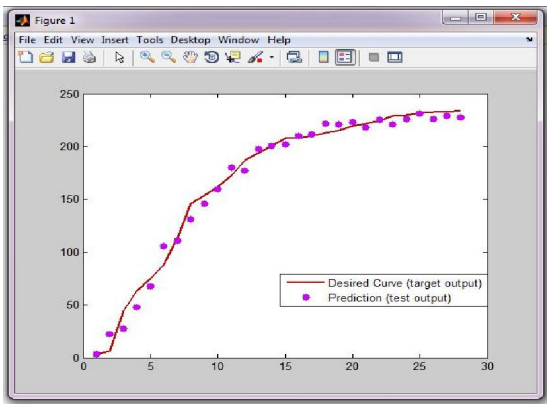


Figure 6.46. MLP failure prediction vs. actual failures for Project Data [7].

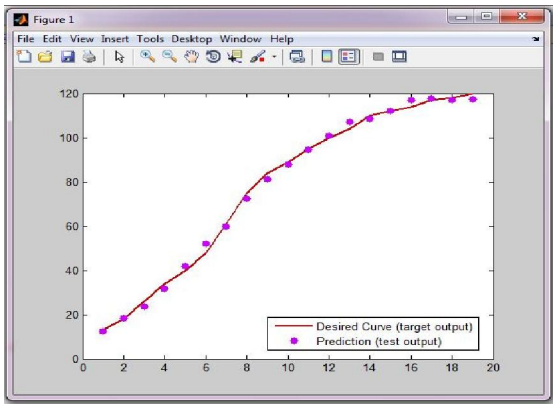


Figure 6.47. MLP failure prediction vs. actual failures for Project Data [8].

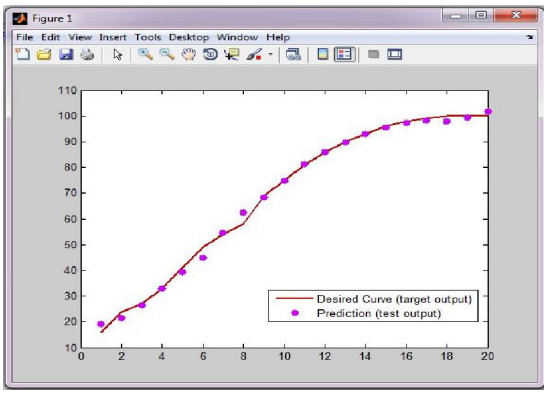


Figure 6.48. MLP failure prediction vs. actual failures for Project Data [10].

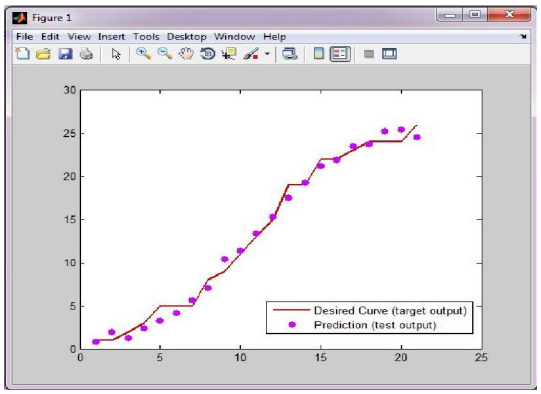


Figure 6.49. MLP failure prediction vs. actual failures for Project Data [11](phase1).

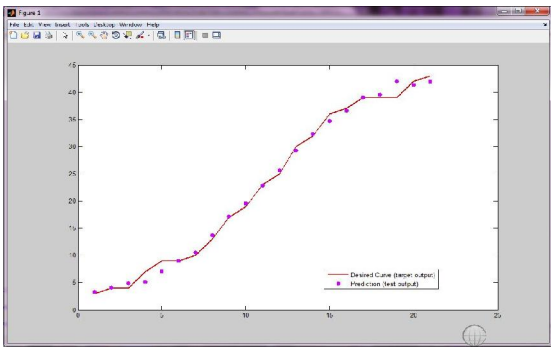


Figure 6.50. MLP failure prediction vs. actual failures for Project Data [11](phase2).

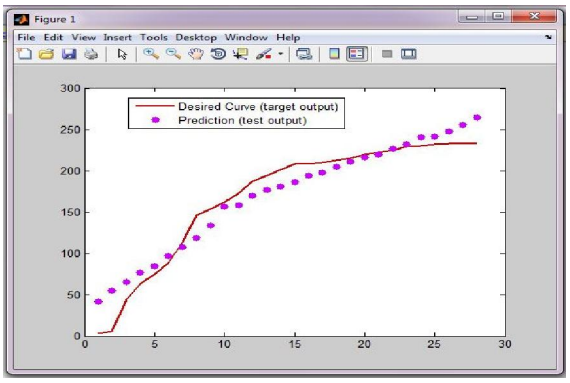


Figure 6.51. REPTree failure prediction vs. actual failures for Project Data [7].

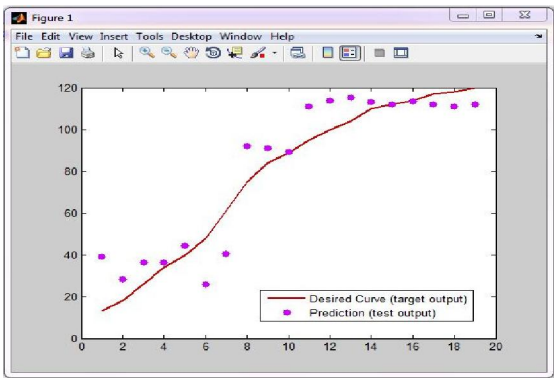


Figure 6.52. REPTree failure prediction vs. actual failures for Project Data [8].

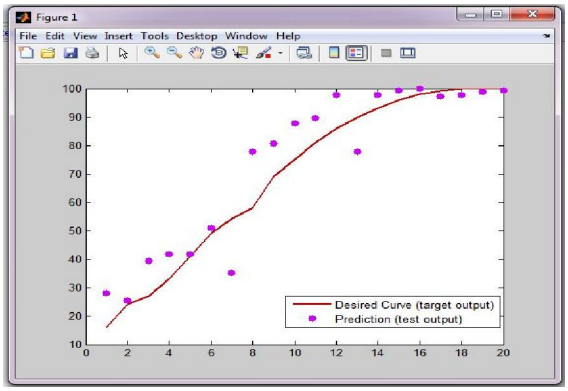


Figure 6.53. REPTree failure prediction vs. actual failures for Project Data [10].

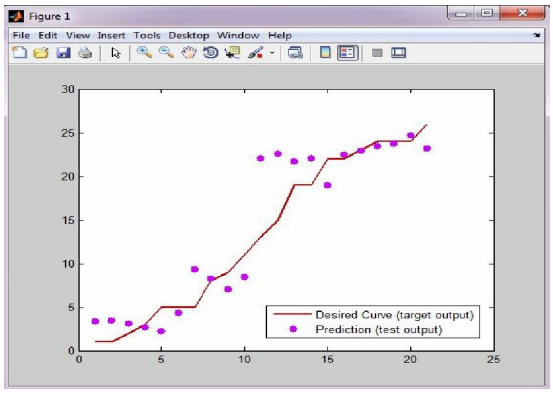


Figure 6.54. REPTree failure prediction vs. actual failures for Project Data [11](phase1).

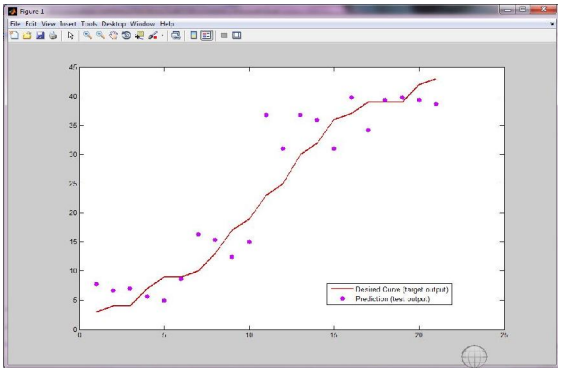


Figure 6.55. REPTree failure prediction vs. actual failures for Project Data [11](phase2).

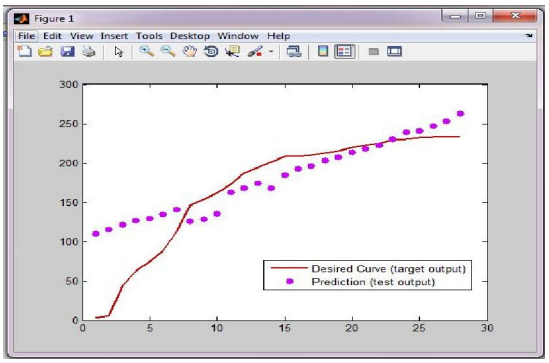


Figure 6.56. SVM failure prediction vs. actual failures for Project Data [7].

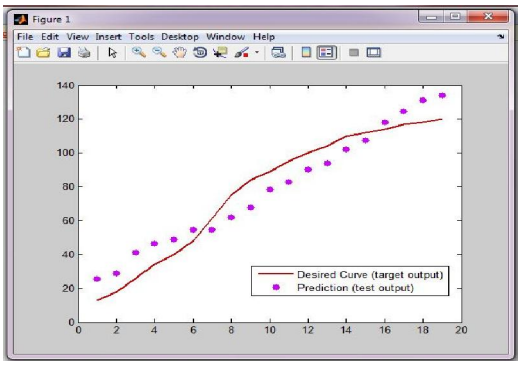


Figure 6.57. SVM failure prediction vs. actual failures for Project Data [8].

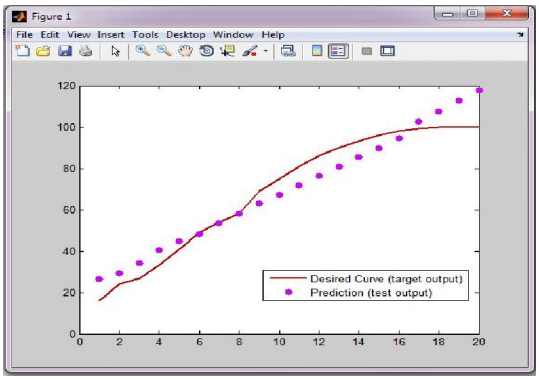


Figure 6.58. SVM failure prediction vs. actual failures for Project Data [10].

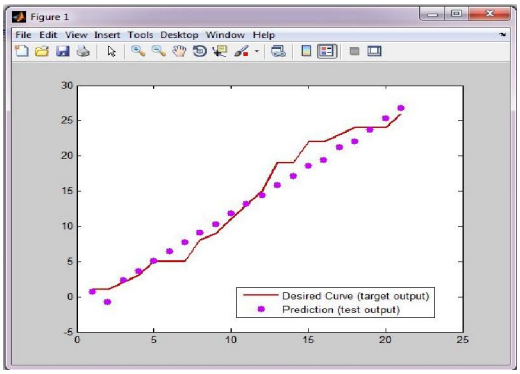


Figure 6.59. SVM failure prediction vs. actual failures for Project Data [11](phase1).

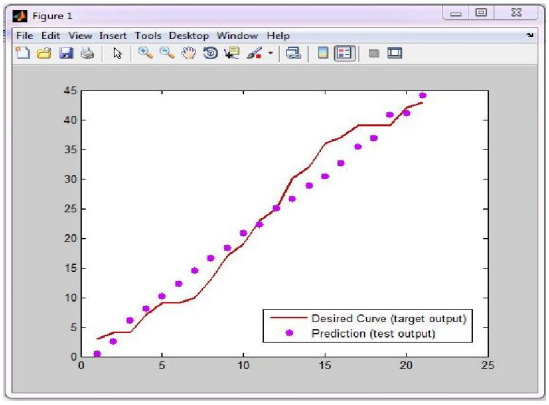


Figure 6.60. SVM failure prediction vs. actual failures for Project Data [11](phase2).

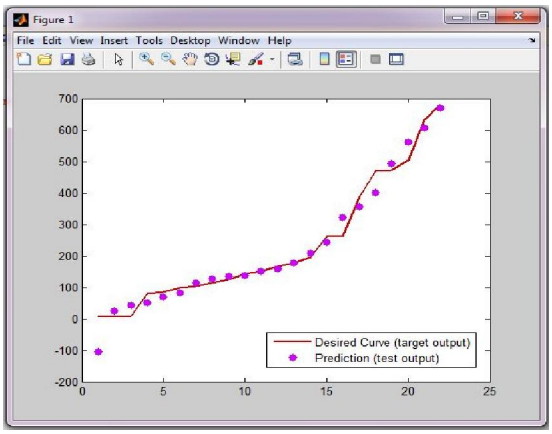


Figure 6.61. ANFIS cumulative failure Prediction vs. actual failures Data [9].

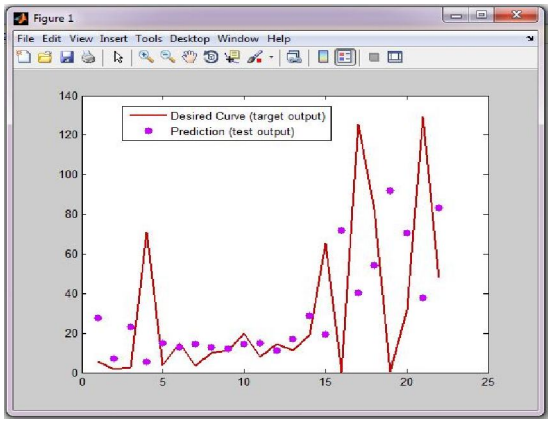


Figure 6.62. ANFIS interfailure time's prediction vs. actual failures for Project for Project Data [9].

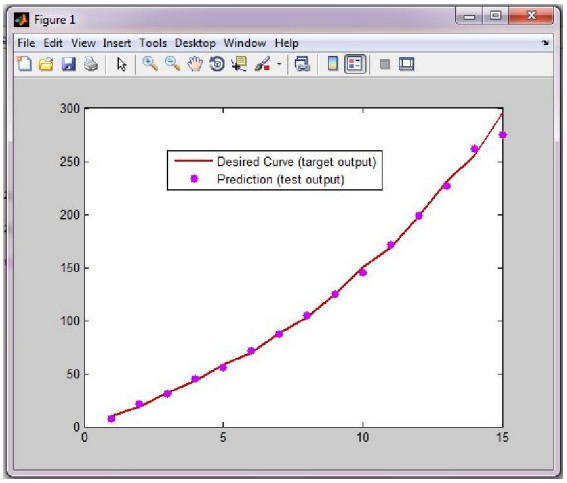


Figure 6.63. ANFIS cumulative failure prediction vs. actual failures for Project Data [12].

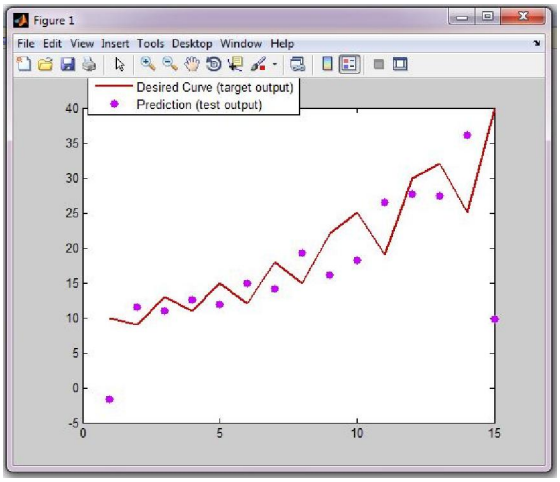


Figure 6.64. ANFIS interfailure time's Prediction vs. actual failures for Project Data [12].

Figures 6.65 depicts the graphical representation of the analysis of the performances of above mentioned ML techniques for predicting Software Reliability based on MARE for the industrial datasets taken into consideration.

Figure 6.65: Comparison of MARE for five different datasets.

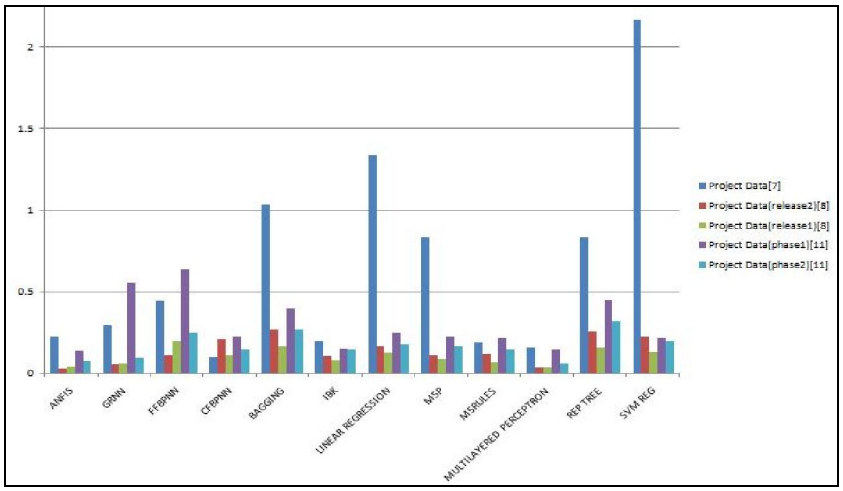


Figure 6.66 depicts the graphical representation of the analysis of the performances of above mentioned ML techniques for predicting Software Reliability based on correlation coefficient for the industrial datasets taken into consideration.

Figure 6.66: Comparison of Correlation Coefficient for five different datasets.

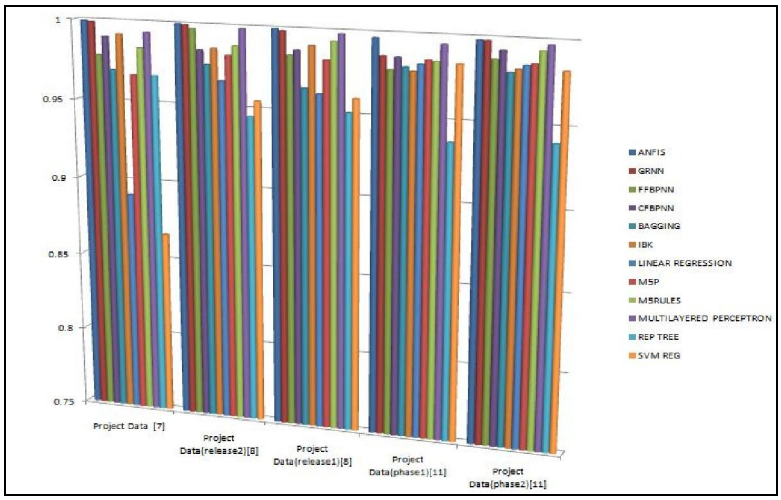


Figure 6.67 depicts the graphical representation of the analysis of the performances of above mentioned ML techniques for predicting Software Reliability based on MRE for the industrial datasets taken into consideration.

Figure 6.67: Comparison of MRE for five different datasets.

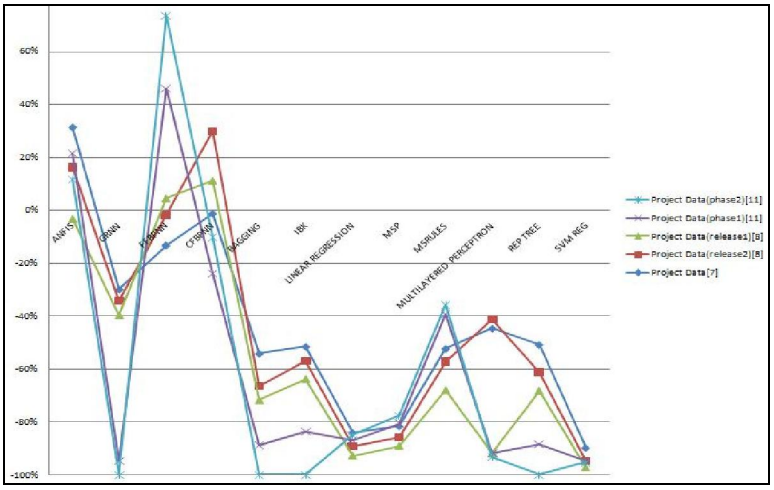
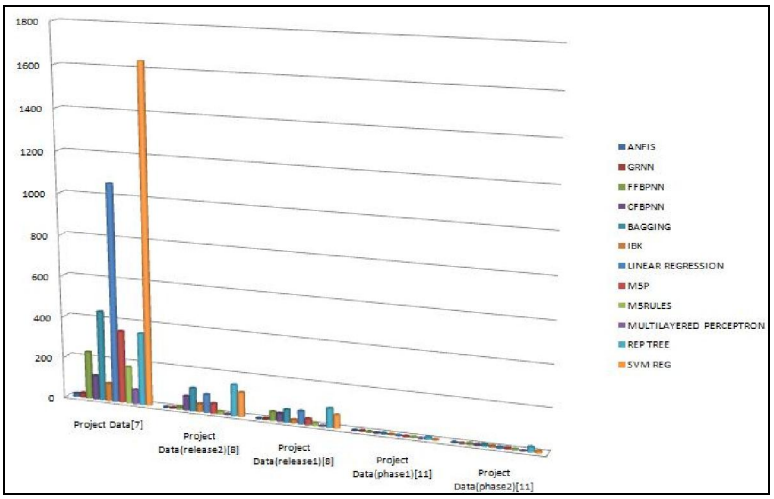


Figure 6.68 depicts the graphical representation of the analysis of the performances of above mentioned ML techniques for predicting Software Reliability based on MSE for the industrial datasets taken into consideration.

Figure 6.68: Comparison of MSE for different datasets.



CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This chapter briefly summarizes the results obtained from this study. It also includes the conclusion and the future scope.

7.1. SUMMARY OF RESULTS

In our study, we have applied machine learning techniques namely Adaptive Neuro Fuzzy Inference System (ANFIS), Feed Forward Back propagation Neural Network (FFBPNN), General Regression Neural Network (GRNN), Support Vector Machine(SVM), Multilayer Perceptron (MLP), Bagging, Cascading Forward Back propagation Neural Network (CFBPNN), Instance Based Learning (IBK), Linear Regression (Lin Reg), M5P, Reduced Error Pruning Tree (RepTree), M5Rules for predicting software reliability based on past failures of software products. The performance of above mentioned machine learning techniques have been evaluated using five different types of data sets being extracted from industrial data to predict the failure intensity of the software's in use. We have empirically proved that ANFIS outperformed the model predicted using other mentioned machine learning techniques for all the industrial

datasets being taken into consideration in predicting reliability in terms of the various statistical efficacy measures applied. Apart from this, GRNN shows very appreciating and encouraging results and can also be used for reliability prediction. We may also infer that GRNN and MLP follow Adaptive Neuro Fuzzy Inference in predicting reliability. GRNN and MLP produce positive Correlation Coefficient nearer to 1 and thus can also be used for predicting reliability efficiently and accurately after ANFIS. Also that cumulative data produces always better results as compared to inter failure time's data. This is the reason that cumulative failure data is always chosen for failure prediction experiments.

Based on the results obtained from rigorous experiments being conducted, we have made following observations/conclusions regarding our study:

i. ANFIS yields better results as compared to other techniques in predicting failures in terms of statistical efficacy measures undertaken.

ii. ANFIS produces Correlation Coefficient nearest to +1 which depicts that it shows positive correlation coefficient as compared to all other techniques. It also clearly demonstrate that how closely actual and predicted values are correlated with each other. The higher the correlation the better the reliability. Hence we can say that ANFIS predicts failure more accurately than other mentioned techniques.

iii. The techniques GRNN and MLP follow ANFIS in predicting reliability. GRNN also showed encouraging and sound. GRNN and MLP produce positive correlation

coefficient nearer to 1 and thus can also be used for predicting reliability efficiently and accurately after ANFIS.

iv. From the above results, we also found that ANFIS produces lowest MARE, MRE, MSE scores as compared to rest of the techniques which again proves it to be better in terms of predicting failures. It shows that ANFIS yields least failure discrepancy between the actual and the predicted failures. Hence we can conclude that ANFIS predicts reliability more precisely and accurately.

v. From the results obtained, we also infer that cumulative failures produce more accurate and precise results as compared to inter failure time's data. Cumulative failures yields correlation coefficient nearer to 1 unlike inters failure time's failures whose correlation coefficient lie more close to -1. The positive linear relationship grows stronger as correlation coefficient nears 1, and the negative linear relationship grows stronger as correlation coefficient nears -1. It shows that how closely actual and predicted values are correlated with each other. Also we observed that ANFIS yields correlation coefficient nearer to 1 (i.e. 0.9948 and 0.9989) for cumulative failures (Project Data [9, 12]) as compared to rest of the other mentioned techniques.

vi. From above, we observed that cumulative failures yields better results than inter failure time's data. They produce lowest MARE scores as compared to inter failure time's data. Out of all, ANFIS gives lowest MARE score (28.49%, 4.64%) for cumulative failures (Project Data [9, 12]) which again proves it to be the best method amongst the other mentioned techniques.

7.2. CONCLUSION

Software reliability is one amongst the important facet of the software quality. Presence of faults/failures makes the software unreliable. Software Reliability is dynamic and stochastic in nature so we may say that reliability is a probabilistic measure that assumes that the occurrence of failure of software is a random phenomenon [32]. In this paper, we have applied machine learning techniques namely ANFIS, FFBPNN, GRNN, SVM, MLP, Bagging, CFBPNN, IBK, Lin Reg, M5P, RepTree, M5Rules for predicting software reliability based on past failures of software products. The performance of above mentioned Machine Learning techniques have been evaluated using five different types of data sets being extracted from industrial data to predict the failure intensity of the software's in use. We have empirically proved that ANFIS outperformed the model predicted using other mentioned ML techniques for all the datasets being taken into consideration in predicting reliability in terms of the various statistical efficacy measures applied. For each of the five datasets taken into consideration, results show that the correlation coefficient is above 0.99 in most of the predictions for ANFIS which signifies that the actual and the predicted values are very close. Also we found that the MARE is between the ranges of 0.025 to 1.5 in most of the predictions for ANFIS and is quite lowest in comparison to the MARE's is being calculated by other mentioned techniques. The results also depicts that the MSE ranges between 0.5 to 16.0 in most of the predictions for ANFIS and MRE is quite lowest in comparison to the other mentioned techniques. Apart from this, GRNN shows very appreciating and encouraging results and can also be used for reliability prediction. We may also infer that GRNN and MLP follow ANFIS in predicting reliability. Also we observed that

cumulative data produces always better results as compared to inter failure time's data. The result shows that the cumulative failures yield high correlation coefficients within the ranges of 0.74 to 0.99 in comparison to the correlation coefficient's being calculated for interfailure time's data which signifies that the actual and the predicted values are very close. The results also depicts that the cumulative failures yields low MARE within the ranges of 0.04 to 1.38 in comparison to the MARE's being calculated for interfailure time's data. This is the reason that cumulative failure data is always chosen for failure prediction experiments.

7.3. FUTURE WORK

For further work, more techniques like DENFIS (Dynamic Neuro Fuzzy Inference System), GMDH (Group Method of Data Handling), and PNN (Probabilistic Neural Networks) can be applied to the data. More datasets can be collected and validated by applying various other machine learning techniques for failure predictions. Further research is planned in an attempt to combine above mentioned models with other machine learning techniques so as to develop prediction models which can predict the reliability of software more accurately with least precision errors.

REFERENCES

[1] Standards Coordinating Committee of the IEEE Computer Society, IEEE Standard Glossary of Software Engineering Terminology, IEEE-STD-610.12-1990 (New York: IEEE, 1991).

[2] Aggarwal KK. , Singh Y., Kaur A., Malhotra R. (2006), "Investigating the effect of coupling metrics on fault proneness in object-oriented systems", Softw Qual Prof, 8(4):4–16.

[3] Goel B., Singh Y. (2009), " An empirical analysis of metrics", Softw Qual Prof, 11(3):35–45.

[4] Singh Y., Kumar P. (2010a), "A software reliability growth model for three-tier client–server system", Int J Comp Appl, 1(13):9–16, doi: 10.5120/289-451.

[5] Singh Y., Kumar P. (2010b), "Determination of software release instant of three-tier client server software system", Int J Softw Eng, 1(3):51–62.

[6] Singh Y., Kumar P. (2010c), "Application of feed-forward networks for software reliability prediction", ACM SIGSOFT, Softw Eng Notes, 35(5):1–6.

[7] Hu P. Q., Dai S. Y., Xie M. and Ng H. S.(Sept. 2006), "Early software reliability prediction with extended ANN model," Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC '06), Volume 2, pp. 234-239.

[8] Wood A. (1996), "Predicting software reliability," Computer, vol. 29, no. 11, pp. 69-77.

[9] Pai F. P. and Hong C. W.(2006), “Software reliability forecasting by support vector machines with simulated annealing algorithms, ”Journal of Systems and Software, vol. 79, no. 6, pp. 747-755.

[10] Zhou Y., Xu B., and Leung H. (2010), “On the ability of complexity metrics to predict fault-prone classes in object-oriented systems”, The Journal of Systems and Software, vol. 83,:660–674.

[11] Zhang X., Jeske DR., Pham H. (2002) "Calibrating software reliability models when the test environment does not match the user environment,“ Applied Stochastic Models in Business and Industry, vol. 18:87-99.

[12] Ohba M. (1984a), "Software reliability analysis models," IBM J Research Development, vol. 21(4).

[13] Malhotra R., Kaur A., Singh Y. (2011), “Empirical validation of object oriented metrics for predicting fault proneness at different severity levels using support vector machines”, Int J Syst Assur Eng Manag 1(3):269–281. doi:10.1007/s13198-011-0048-7.

[14] Xingguo L., Yanhua S. (2007), “An early prediction method of software reliability based on support vector machine”, In: Proceedings international conference on wireless communications, networking and mobile computing (WiCom’07), Hefei: 6075–6078.

[15] Jung Hua L. (2010),” Predicting software reliability with support vector machines”, In: Proceedings of 2nd international conference on computer research and development (ICCRD’10), Kuala Lumpur, Malaysia: 765-769.

[16] Karunanithi N., Whitley D., Malaiya Y. (1992), “Prediction of software reliability using connectionist models”, IEEE Trans Softw Eng 18(7):563–574.

[17] Singh Y., Kumar P. (2010d), "Prediction of software reliability using feed forward neural networks", In: Proceedings of computational intelligence and software engineering (CiSE'10), Wuhan, China: 1–5. doi:10.1109/CISE.2010.5677251.

[18] Singh Y., Kumar P. (2010c), "Application of feed-forward networks for software reliability prediction", ACM SIGSOFT, Softw Eng Notes, 35(5):1–6.

[19] Eduardo OC., Aurora TR., Silvia RV. (2010), "A genetic programming approach for software reliability modeling", IEEE Trans Reliab, 59(1):222–230.

[20] Matlab fuzzy logic toolbox: tutorials on fuzzy inference system and ANFIS using MatLab. Available at http://www.mathworks.com. Accessed 14 Feb 2011.

[21] Cai, K.Y., Wen, C.Y., Zhang, M.L., (1991), "A critical review on software reliability modeling. Reliability Engineering and System Safety", 32 (3), 357–371.

[22] Ho, S.L., Xie, M., Goh, T.N., (2003), "A study of connectionist models for software reliability prediction", Computers and Mathematics with Applications, 46 (7), 1037–1045.

[23] Su, Y.-S., Huang, C.-Y., (2006), "Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models", Journal of Systems and Software, 80 (4), 606–615.

[24] Madsen, H., Thyregod, P., Burtschy, B., Albeanu, G., Popentiu, F., (2006), "On using soft computing techniques in software reliability engineering", International Journal of Reliability, Quality and Safety Engineering 13 (1), 61–72.

[25] Pai, P.F., Hong, W.C., (2006), "Software reliability forecasting by support vector machines with simulated vector machines with simulated annealing algorithms", The Journal of Systems and Software 79, 747– 755.

[26] Tian, L., Noore, A., (2005b), "Evolutionary neural network modeling for software cumulative failure time prediction", Reliability Engineering and System Safety 87, 45–51.

[27] Specht D.F., (1991), "A general regression neural network", IEEE Transactions on Neural Networks 2 (6), 568–576.

[28] Aljahdali H. Sultan and Buragga A. Khalid. (2008), "Employing four ANNs Paradigms for Software Reliability Prediction: an Analytical Study", ICGST-AIML Journal, ISSN: 1687-4846, Volume 8, Issue II, September 2008.

[29] Kohavi R. (1995), "The power of decision tables", In: The eighth european conference on machine learning (ECML-95), Heraklion, Greece, pp 174–189.

[30] Kumar P., Singh Y. (2012), "An empirical study of software reliability prediction using machine learning techniques", Int J Syst Assur Eng Manag, (July-Sept 2012) 3(3):194–208 DOI 10.1007/s13198-012-0123-8.

[31] Koten van C. and Gray R. A. (2005), "An application of Bayesian network for predicting object-oriented software maintainability", The Information Science Discussion Paper, Series Number 2005/02 March 2005 ISSN 1172-6024.

[32] Quyoum A., Dar Din UdM., Quadr K. M. S. (2010), "Improving Software Reliability using Software Engineering Approach- A Review", International Journal of Computer Applications (0975 – 8887), Volume 10– No.5, November 2010.

[33] Malhotra R., Negi A. (2013), "Reliability modeling using Particle Swarm Optimization", The Society for Reliability Engineering, Quality and Operations Management (SREQOM), India and The Division of Operation and Maintenance, Lulea University of Technology, Sweden 2013, Int J Syst Assur Eng Manag, DOI 10.1007/s13198-012-0139-0.

[34] Kohavi R. (1995), “The power of decision tables”, In: The eighth european conference on machine learning (ECML-95), Heraklion, pp 174–189.

[35] Phillip S. (2003), “ DTReg predictive modeling software “,available at http://www.dtreg.com, Accessed 8 Jan 2011.

[36] Ping PF., Hong WC. (2006), “Software reliability forecasting by support vector machines with simulated annealing algorithms”, J Syst Softw, 79(6):747–755.

[37] Chen KC., Shiang Y., Liang TZ. (2008), “A study of software reliability growth from the perspective of learning effects”, Reliab Eng Syst Saf, 93(10):1410–1421.

[38] Xingguo L., Yanhua S. (2007), “An early prediction method of software reliability based on support vector machine”, In: Proceedings international conference on wireless communications, networking and mobile computing, (WiCom’07), Hefei: 6075–6078.

[39] Mueller JA., Lemke F. (1999), “Self-organizing data mining: an intelligent approach to extract knowledge from data, Dresden, Berlin.

[40] Jang., J-S.R. and Sun., C-T. (1995), “Neuro-fuzzy Modelling and Control”, *Proceedings of IEEE*, 83 (3), 378-406.