SOFTWARE DEFECT PREDICTION USING ARTIFICIAL NEURAL NETWORK

Supervised By

Dr. Imran Mahmud
Associate Professor and Head

Department Of Software Engineering
Daffodil International University

Submitted By

Ahasanul Haque
Id: 191-35-2704

Department Of Software Engineering
Daffodil International University

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Software Engineering

APPROVAL

This thesis titled on "**Software Defect Prediction Using Artificial Neural Networks**", submitted by **Ahasanul Haque (ID: 191-35-2704)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

DECLARATION

I announce hereby that I am rendering this study document under Dr. Imran Mahmud, Associate Professor and Head, Department of Software Engineering, Daffodil International University. I there for state that this work or any portion of it was not proposed here therefore for Bachelor's degree or any graduation.

Supervised by:

[illegible]

Dr. Imran Mahmud
Associate Professor and Head
Department Of Software Engineering
Daffodil International University

Submitted by:

Ahasanul Haque

Ahasanul Haque
Id: 191-35-2704
Department Of Software Engineering
Daffodil International University

ACKNOWLEDGEMENT

First and foremost, I offer my heartfelt appreciation and gratitude to Almighty God for His divine gift, which has enabled us to successfully finish the final year proposal.

I really grateful and wish our profound our indebtedness to Dr. Imran Mahmud, Associate Professor and Head, Department of Software Engineering, Daffodil International University, Dhaka. Deep Knowledge & keen interest of my supervisor in the field of “Artificial Neural Networks” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to other faculty members and the staff of the SWE department of Daffodil International University.

Finally, I must respectfully thank our parents for their unwavering love and patience.

ABSTRACT

Defect severity assessment is the most crucial step in major companies and organizations where the complexity of the software is growing at an exponential rate. One of most active research areas in software engineering is software defect prediction (SDP). SDP is a technique for envisioning software defects. Early defect discovery during the software development life cycle (SDLC) results in early repairs and ultimately on-time delivery of maintainable software, which pleases the client and fosters his confidence in the development team. By consistently predicting bugs, removing bugs, and identifying defect modules, the software industry aims to improve the quality of its products. The requirement for high-quality and affordable software that can be maintained is growing as the need for automated online software systems rises daily. Predicting software defects is one of the major goals of the quality assurance process, which improves quality while reducing costs by reducing overall testing and maintenance activities.

Artificial Neural Networks (ANN), one of the commonly utilized machine learning approaches, are used in the majority of suggested frameworks and models for defect prediction. Using Artificial Neural Networks (ANN) approach, Laverberg-Marquardt (LM) and Bayesian Regularization (BR) we can simply predict software defect. To use the MATLAB simulation tool, a framework is built and used to the NASA software dataset being considered for performance study. By considering the performance I can select which ANN method algorithm is more perfect in predicting software defect. This study will benefit the researchers and serve as a benchmark for future improvements, analyses, and evaluations. An experimental investigation demonstrates that the suggested approach can offer superior performance for predicting software defects.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATION

SDP = software defect prediction

ANN = Artificial Neural Networks

SDLC = software development life cycle

GSD = global software development

LM = Laverberg-Marquardt

BR = Bayesian Regularization

MSE = mean squared error

CHAPTER 1

INTRODUCTION

1.1 Background

The world is becoming more digital than we anticipated because of the exponential increase in the use of computers over the past two decades, which has changed nearly everything around us. The development of software solutions to automate daily life tasks and procedures has risen significantly. Our lifestyles are now more easy that they were before thanks to these automated processes (Gao, K, 2011)The discipline of software development is expanding quickly both nationally and internationally as a result of technical advancements, and it is currently known as global software development (GSD), offshore software development, or development by outsourcing. A corporate-level method that has been used for the past 20 years and is becoming more established is software out sourcing . A GSD outsourcing approach called +e software outsourcing allows for the development of high-quality software at a cheaper cost. Despite its continued growth, the software development sector still faces considerable entrance obstacles due to a lack of a clearly defined model of operation (Iftikhar,2021)

The requirement for high-quality, affordable software that can be managed is being driven by the increasing need for automated software systems. One of the key components of the quality assurance process, software defect prediction (SDP) lowers the overall testing and maintenance costs while improving quality. Early defect detection during the software development life cycle (SDLC) results in early adjustments and ultimately timely delivery of reliable software, delighting the customer and promoting his trust in the development team (Khan, 2022). In the previous two decades, "the defect prediction approaches have accomplished the considerable adoption in the software industry." By identifying the software modules where problems are more likely to occur in advance, this action can enhance the quality of the software (Zhang, 2015). A well-known supervised learning method for dealing with prediction issues in a variety of areas of software engineering, such as estimation cost, effort and predict defect, is the Artificial Neural Network (ANN). (Laradji, 2015) A key element of artificial intelligence, artificial neural networks (ANNs), were first introduced in the early 1950s. It has acquired popularity in recent years as a result of its capacity to resolve a significant number of challenging real-world issues in the data mining and machine learning fields. Regression and classification challenges are two categories of issues that ANNs are most adept at solving (Ozyildirim,2013).Due to its flexible structure and ability to simulate any complicated non - linear behavior, ANNs are effective data-driven modeling tools that are frequently used for the numerical simulations and identification of nonlinear systems. Adaptive computing models called neural networks are made up of tightly coupled processing units called neurons. Figure-1 depicts an ANN's basic architecture. There are three layers that make up the ANN structure: input layer, output layer, and hidden layer (s).

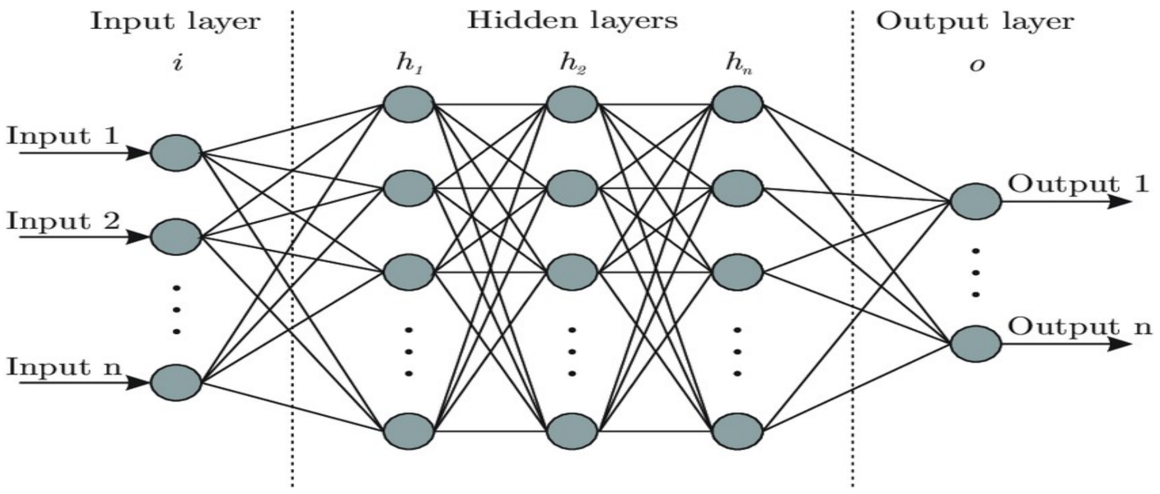


Fig.1.1: Architecture of ANN (Bre, 2018)

There is a connection between the nodes in the input layer and the hidden layer, and subsequently between the hidden layer nodes and the output layer nodes. The input layer is where the input data are sent into the neural network (Jayanthi, 2019). These parallel operating architectures' basic quality is their capacity for self-learning and knowledge discovery. The following characteristics of a neural network are intrinsic to it:

- ✓ Simple design and flexibility
- ✓ Knowledge discovery and the capacity for self-improvement (Arora, 2016).

In the world of software, ANNs are a widely used and acknowledged technique for addressing a variety of issues, including software effort estimation, cost estimation, optimization issues, and software fault prediction (Ahmad,2017). Recent research has concentrated on the use of artificial neural networks (ANNs) for developing fault prediction models in SDP. These studies have demonstrated that ANNs perform superior to conventional SDP models. Another benefit of ANNs is that they satisfy the requirements of the SDP models in that they are created using software metrics and do not require specialized knowledge (Arora, 2016). The purpose of this study is to identify the learning algorithm that, among the two ANN training algorithm variations, Levenberg-Marquardt (LM) and Bayesian Regularization (BR), performed the best for software fault prediction. This paper is very necessary for our software industry in Bangladesh and world. In this study, I offer a critical examination of recent literature on artificial neural networks' usage in predicting software defects that was published between 2000 and 2022.

1.2 Problem around the World

Nearly everything around us has changed as a result of the exponential growth in the usage of computers over the past two decades, and the world is becoming more digital than we originally anticipated. To automate the operations and procedures relevant to our daily lives, more software solutions are being developed today. Our lives are now more comfortable than they were before thanks to these automated processes. Software with poor quality can result in incorrect and unexpected outcomes (Khan, 2022). Today, it is getting harder to generate high-quality software products at lower costs due to the growing complexity of current software systems (Kim, 2011). In the previous two decades, "the defect prediction approaches have achieved the significant acceptance in the software industry." By identifying the software modules where problems are more likely to occur in advance, this action can enhance the quality of the software (Goyal, 2022). Due to the mass production of software programs, the quality of software is still an unresolved problem that results in subpar performance for both commercial and personal applications. Software testing, which assists in identifying and attempting to fix software program flaws or bugs, was thus developed to address this problem. A significant number of software applications are generated each year as a result of growing demand from modern technology-based industries and business applications, but software quality is still an unrecognized problem during this growth. Software applications have recently grown in importance for daily tasks and corporate purposes (Manjula, 2019). Software defect in the major problem in the software around the world. There are various factors that contribute to cross-prediction approaches' poor efficacy: First off, the characteristics of each software project as determined by software metrics vary. Due to the individual capacity of software creators, feature distributions can differ greatly. Machine learning techniques will have a lot of difficulties dealing with this mismatch since they presuppose that training and testing data should be in the same feature space and subject to similar data distribution. Second, each project's class imbalance may make predictions less accurate. The majority class of data will have an impact on the prediction outcome. The majority of software files used in actual software projects won't have any defects (Wu, R , 2011)As a result, the ratio of files with flaws to files without may influence the prediction result. Thirdly, the classifier may be misled by the project's outlier cases.. Real-world software projects will demonstrate all of these flaws. A good cross-project defect prediction approach should therefore consider all of these flaws (Cao ,2015)Because there are so many software applications, it is difficult for researchers to identify, locate, and discover software defects. In this area of software defect identification and prediction, defect density is another difficult task. The software industry cannot be provided Software on time because the cost is rising daily. Because of this, the software company experiences losses every day. Their business was almost to shut down at one time (Jayanthi, 2019).

1.3 Problem in Bangladesh

In the current setting, the size and consequently the complexity of the software are growing at an exponential rate. This results in the bugs getting into the software, which causes functional issues (Myers 2011). A software system or product must be built up through a series of predictable processes in order to produce timely, high-quality outcomes. A software process, to put it more properly, is a structure for the tasks needed to create high-quality software (Pressman 2005). An error, flaw, bug, mistake, failure, or fault in a computer program or system that could produce an incorrect or unexpected result or prevent the software from performing as intended is known as a software defect. A project team strives to produce high-quality software that has few or no flaws. To improve software quality, high risk components within the project should be identified as soon as possible. Costs in terms of quality and time are always associated with software problems.

In addition, one of the most time- and money-intensive software operations is finding and fixing errors. While it is not realistic to get rid of every flaw, it is possible to lessen the severity of flaws and their detrimental effects on project (Rawat, 2012). Bangladesh, which has a population of more than 135 million, is one of the biggest developing nations in the world. Bangladesh's software sector has advanced significantly in recent years. There are now more than three hundred (300) registered software companies working in Bangladesh, according to BASIS (the Bangladesh Association of Software and Information Services). These software companies face a lot of risk during this time when they start a project, build it and finally hand it over to their (Bernard ,2006)

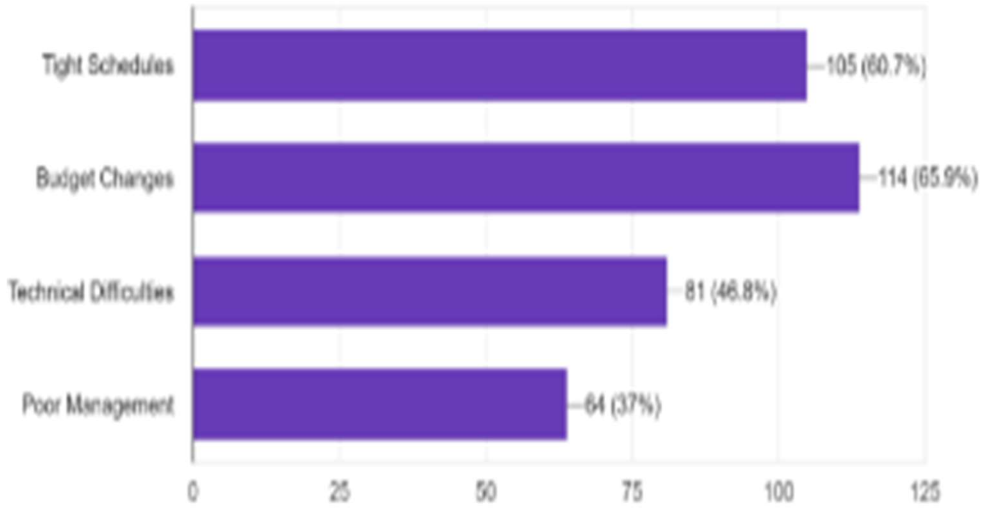


Figure. 1.2 Feedback about Project Failure (Rahman, 2022).

This survey's findings show that specific risk effect areas are frequently encountered by Bangladesh's IT industries. Such as-Tight schedules (60.7%) Budget changes (65.9%),Technical Difficulties(46.8%),Poor management(37%) (Rahman, 2022).

One of the key components of the quality assurance process, software defect prediction lowers the overall testing and maintenance costs while improving quality. Early defect identification in the life cycle of software development (SDLC). One of the primary concerns of software developers today is the timely delivery of high-quality software. Delivering high-quality maintainable software at lower costs is becoming more challenging, nevertheless, as a result of the increasing complexity of software systems. By employing methods for software fault prediction, this problem can be resolved. One of the popular supervised machine learning techniques to anticipate faults at the beginning of the SDLC is the artificial neural network (ANN) (Khan, 2022).

1.4 Research Model

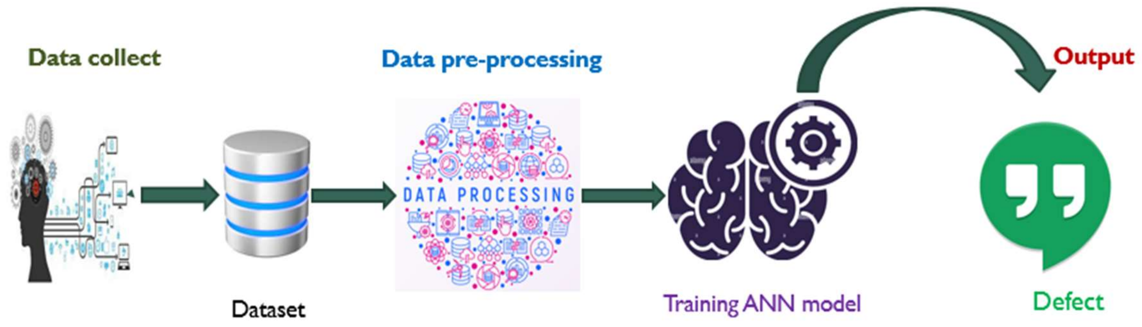


Figure. 1.3: Research model

1.5 Research Question

In this thesis, I has a core research questions which are given below:

RQ1. Does ANN help to Predict Defect of the software?

1.6 Research Objective

To answer the research questions, I has following objectives.

RO1. To implement ANN to Predict Defect from the software.

1.7 Organization of the Chapter

In Chapter 1, I have discussed software defect prediction process, global problem, Bangladesh problem, research model, research question, research objective. To discuss more about the thesis, the rest of the chapters are organized like:

In Chapter 2, I will describe the literature review that reflected previous work on software defect prediction. In chapter 3, which is research methodology. In this section, I will discuss about artificial neural network and its algorithm, data collection, model implementation process, performance evaluation metrics. Following Chapter 3, I will describe Chapter 4 which is Results and Discussion. In this section I will discuss the experimental setup, analysis of results, discussion of results. Finally in Chapter 5, I will discuss the conclusion and recommendations. This sector have be the total summary of my work. Here I discussed what work I will do in the future for the betterment of the work.

CHAPTER 2

LITERATURE REVIEW

Software Defect Prediction Using Neural Networks. Researchers in (Jindal,2014)The Radial Basis Function Network (RBF network), an artificial neural network with a single layer topology, is the framework that is being given. As suggested by the name, this network's activation functions are radial functions. As a result, the network offers a linear model of radial functions that have been applied to the inputs. In this study, we extract data from the PITS (Project and Issue Tracking System) database that NASA engineers frequently use. The suggested program would initially use a number of text mining algorithms to retrieve the pertinent data from the PITS database. Following extraction, the tool will use machine learning techniques to forecast the defect severities. The software researchers and developers will be able to focus their testing efforts on the more problematic sections of the software with the aid of the forecast of defect severity. Area Under the Curve (AUC) and susceptibility from Receiver Operating Characteristics (ROC) analysis are indeed the performance metrics employed in the paper. The model has already successfully detected problems of very low impact, as evidenced by the model's AUC, which has a highest value of 0.83.According to this pattern, the model may be able to forecast problems with high levels of severity with greater accuracy than defects with lower severity levels

Software Defect Prediction via Convolutional Neural Network. (Li, 2017)Is presented at Convolutional neural networks were used in a software defect prediction technique known as DP-CNN (Defect Prediction via Convolutional Neural Network), that preserved the semantic and factors – of the source code by generating discriminant information from the programs' Abstract Syntax Trees (ASTs). To benefit from both nonlinear features and hand-crafted elements, the CNN learnt components are combined with conventional defect prediction features. Hence, test all five models, including our suggested DP, on the seven projects. –CNN. Presented DP-CNN, a system for precise software defect prediction that automatically produces semantically and structural characteristics from source code and combines conventional features. Traditional, DBN, DBN+, CNN, and DP-CNN are the models with the maximum accuracy, going from lowest to most accurate. Most precisely, DPCNN enhances CNN by 2%, DBN by 12%, and Traditional by 16%, correspondingly. While it is advantageous to combine conventional features, DP-CNN is the most efficient.

Software Defect Prediction via Transfer Learning-Based Neural Network. This researchers (Cao, 2015) suggested the Transfer Component Analysis Neural Network as a useful software defect prediction technique (TCANN). TCANN is divided into three sections, each of which aims to address each of the three issues described above. Firstly, a strategy based on the Inter Quartile Range (IQR) is suggested for removing noise from datasets.

Secondly, the feature distribution disparities between both the target and the source data are minimized using the transfer component analysis method. Thirdly, a dynamic sampling neural network is suggested as a solution to the issue of minority class in the training dataset. Precision, Recall, F-measure, and AUC are used to evaluate the performance of the suggested technique. Within-project and cross-project performance assessment are two types of defect prediction that are carried out. The goal of a dynamic sampling neural network is to build a system for training ANNs dynamically. First off, this technique does not destroy any samples in the event of information loss. On a project-specific dataset, we do tests to assess the effectiveness of the recommended strategy. In order to create training samples, we randomly select 70% of each dataset. And test specimens will be drawn from the remaining 30% of the data. A typical ratio for training and testing datasets is this experimental configuration. Each project in the dataset would be evaluated separately for the experiment, and the test outcomes will be compared to the real value in accordance. The suggested method demonstrated great result for software fault prediction in both intra- and inter-project scenarios.

Improved Approach for Software Defect Prediction using Artificial Neural Networks. (Sethi, 2016) In this study, the outcomes of a basic fuzzy inference system technique are contrasted with those of a neural network-based method for predicting software defects. The suggested method, it is discovered that now the neural network-based training process offers superior and efficient outcomes on a variety of parameters. SDP is a method for anticipating the flaws that give a software module dependability. Several organizations, including Blackberry, IBM, and Google, use SDP. A simplified dataset of base research is taught as the dataset for a feed-forward, back-propagation network that use the MATLAB ANN tool. Comparing the suggested method to the conventional strategy, superior outcomes are being achieved.

Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization (Jin, 2015). The use of hybrid artificial neural networks (ANN) and quantum particle swarm optimization (QPSO) in software fault-proneness prediction is presented in this study. Software applications are categorized into fault-prone or non-fault-prone categories using ANN, and dimensionality is reduced using QPSO. The suggested method has brought attention to the relationship between software metrics and defective software modules. NASA software applications are utilized to provide the data sets used to test the performance of the suggested method. The performance in terms of AUC, number of chosen software indexes, and Mean Calculation Time. We compare the accuracy of the predictions by listing their AUC values. The maximum AUC scores of the new ANNs built on PSO (i.e., PSO + ANN) and QPSO (i.e., QPSO + ANN) over 30 independent runs on four datasets, correspondingly, are shown at the similar time.

Software defect prediction techniques using metrics based on neural network classifier. This paper (Jayanthi, 2019) ANN and the feature reduction technique were combined in a suggested method to software defect prediction. It provides a hybrid feature minimization and artificial neural network technique for predicting software defects. The upgraded PCA approach for feature reduction and its mathematical modeling are covered in the initial part of the paper, while the combined execution of the neural network and the suggested PCA method is covered in the second part. The proposed method is examined using the MATLAB tool and NASA software data sets. When compared to earlier investigations, the proposed approaches increase defect predictive performance while requiring fewer attributes. It employed attribute selection to demonstrate that classifiers still provided equivalent or even higher reliability than the total number of attributes used, even when the number of attributes was reduced by almost 80%. KC1, JM1, PC3, and PC4 are the four separate datasets used for performance monitoring. 2096 instances, 325 defects, or 15.5% of the dataset (KC1) have a flaw, and we used the suggested hybrid model for software defect. The prediction and outcome demonstrate that the suggested strategy provides 86.91% performance for the KC1 dataset, 83.03% performance for the JM1 dataset, 89% accuracy for PC3 and 93.64% performance for PC4.

A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. (Miholca, 2018) In this paper, they created a brand-new supervised classification technique called HyGRAR for such prediction of software defects. A non-linear hybrid approach called HyGRAR uses artificial neural networks and progressive relational mining of association rules to distinguish among software entities with defects and those without defects. Ten open-source data sets showed off the HYGRAR classifier's outstanding performance. To compare HyGRAR approach between AUC. The p-value was below = 0.1 in 92% of the cases (10 out of 11 analogies for the Ar data sets and 13 out of 14 analogies again for JEdit, Ant, and Tomcat data sets), indicating that HyGRAR performed on generally higher than other methods on average (in terms of AUC) at a significance level of 0.1.

Deep Neural Network-Based Hybrid Approach for Software Defect Prediction. (Manjula, 2019) In this research, a hybrid strategy that combined GA and DNN was proposed. After being enhanced by applying an adaptable auto-encoder, that offered a greater representation n of a few software characteristics, DNN too was added. They employ the MATLAB tool and NASA software data sets. The suggested methodology outperformed other state-of-the-art procedures when their performances were compared, according to the correlative analysis of the data.

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Artificial Neural Network

Artificial neural networks (ANNs) are computer models that draw inspiration from biological networks, particularly the brain's neural network. It attempts to mimic neurological processes carried out by the human brain, such as speech recognition, pattern recognition, and facial recognition, to name just a few (Jafarzadeh-Ghoushchi, 2015). Artificial Neural Network (ANN), one of the commonly used machine learning methods. As it continues to iterate using both forward and backward propagation, the network picks up lessons from previous examples. This procedure is carried out until the result satisfies the desired accuracy and conformance to the response given. Each example includes a set of inputs and related outputs, often known as responses, and the network modifies itself using internal connections called weights Awolusi(Awolusi, 2019).

Artificial Neural Network (ANN) is a widely accepted supervised learning approach to deal with the prediction problems in multiple domains of software engineering such as effort estimation, cost estimation, and defect prediction (Khan, 2022). ANN trained algorithms Laverberg-Marquardt (LM) and Bayesian Regularization (BR) are used to predict software defect. In this paper I am apply this two algorithm to predict software defect. Description of those algorithm is below-

3.1.1 Levenberg-Marqaurdt (LM) Algorithm: The Levenberg-Marquardt algorithm (LMA), a well-liked trust region approach, is employed to determine the lowest value of a function (either linear or nonlinear) above a set of parameters. In essence, an internal function, like a quadratic function, is used to model a trustworthy area of the objective function. For the majority of nonlinear least square issues, LM is employed as a solution. It was developed as a bridge here between technique of gradient descent and the Gauss Newton technique. (Moré, 1998). When training NN, the Newton technique approximates the inaccuracy in the initial order statement whereas the LM approximates the second order statement. We train this LM algorithm in MATLAB tools.

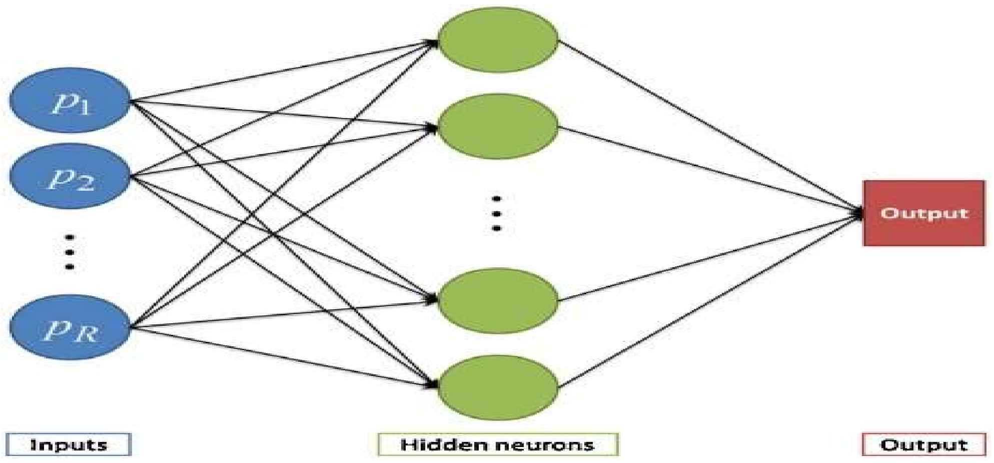


Figure 3.1 LM and BR Model (Nguyen-Truong, 2015)

3.1.2 Bayesian Regularization (BR) Algorithm: Through the mathematical method of "bayesian regularization," a nonlinear regression can be transformed into such a statistical issue that's also "well-posed," much like a ridge regression**.** (MacKay , 1998) Training Bayesian NN is so much more reliable. The optimization of a network design, the effort put into validating an idea, selecting a validating set, and determining a model's robustness are all addressed by BR. Because it is challenging to overtrain and overfit, BR learning algorithms are superior to other training functions

3.2 Data Collection

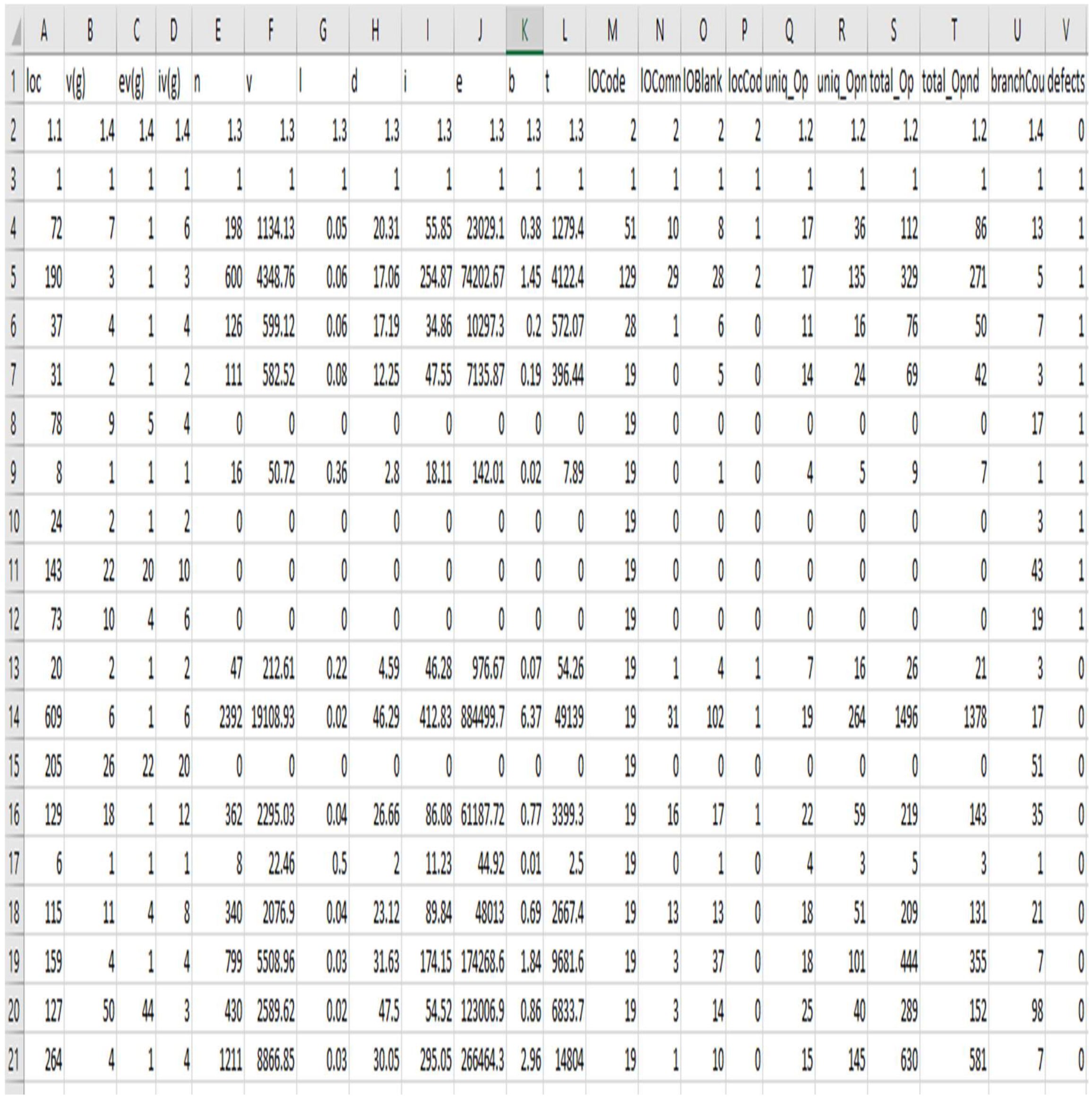
In this study we use two data sets to predict software defect using artificial neural networks. one data sets from the PROMISE repository were used for the study: KC2. These software datasets include a few broad attributes, that are shown in Table 3.1 along with the attribute's name and more specific information. A sample data set is display in Table 3.2. Description of the defect data sets' characteristics is there.

Table 3.1: PROMISE software defect prediction attribute details

NO	Attribute name	Description of attribute
1	LOC	Counts total number of line in the module
2	Iv(g)	Design complexity analysis (McCabe)
3	Ev(g)	McCabe essential complexity
4	N	Number of operators present in the software module
5	v(g)	Cyclomatic complexity measurement (McCabe)
6	D	Measurement difficulty
7	B	Estimation of effort
8	L	Programme length
9	V	Volume
10	I	Intelligence in measurement
11	E	Measurement effort
12	LoComment	Line of comments in software module
13	LoBlank	Total number of blank lines in the module
14	uniq_Op	Total number of unique operators
15	uniq_Opnd	Total number of unique operand
16	T	Time estimator
17	BranchCount	Total number of branch in the software module
18	total_Op	Total number of operators
19	total_Opnd	Total number of operators
20	LocodeandComment	Total number of line of code and comments
21	Defects	Information regarding defect whether defect is present or not

Based on this feature we fix the data set. The majority of recent studies use software metrics to pinpoint fault-prone modules, and their findings indicate that these measures are quite helpful for predicting Defect-proneness classes. Lines of code, cyclomatic complexity, essential complexity, design complexity, are some of the measures used. Length, volume, program length, difficulty, intelligence, effort, effort estimate, programming time, lines of code, lines of comment, lines of blank, lines of comment and code, unique operators, unique operands, total operators, and total operands are some of the Halstead metrics.

Table 3.2: Sample data set (KC2)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	loc	v(g)	ev(g)	iv(g)	n	v	l	d	i	e	b	t	lOCode	lOComn	lOBlank	locCod	uniq_Op	uniq_Opn	total_Op	total_Opnd	branchCou	defects
2	1.1	1.4	1.4	1.4	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	2	2	2	2	1.2	1.2	1.2	1.2	1.4	0
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	72	7	1	6	198	1134.13	0.05	20.31	55.85	23029.1	0.38	1279.4	51	10	8	1	17	36	112	86	13	1
5	190	3	1	3	600	4348.76	0.06	17.06	254.87	74202.67	1.45	4122.4	129	29	28	2	17	135	329	271	5	1
6	37	4	1	4	126	599.12	0.06	17.19	34.86	10297.3	0.2	572.07	28	1	6	0	11	16	76	50	7	1
7	31	2	1	2	111	582.52	0.08	12.25	47.55	7135.87	0.19	396.44	19	0	5	0	14	24	69	42	3	1
8	78	9	5	4	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	17	1
9	8	1	1	1	16	50.72	0.36	2.8	18.11	142.01	0.02	7.89	19	0	1	0	4	5	9	7	1	1
10	24	2	1	2	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	3	1
11	143	22	20	10	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	43	1
12	73	10	4	6	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	19	1
13	20	2	1	2	47	212.61	0.22	4.59	46.28	976.67	0.07	54.26	19	1	4	1	7	16	26	21	3	0
14	609	6	1	6	2392	19108.93	0.02	46.29	412.83	884499.7	6.37	49139	19	31	102	1	19	264	1496	1378	17	0
15	205	26	22	20	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	51	0
16	129	18	1	12	362	2295.03	0.04	26.66	86.08	61187.72	0.77	3399.3	19	16	17	1	22	59	219	143	35	0
17	6	1	1	1	8	22.46	0.5	2	11.23	44.92	0.01	2.5	19	0	1	0	4	3	5	3	1	0
18	115	11	4	8	340	2076.9	0.04	23.12	89.84	48013	0.69	2667.4	19	13	13	0	18	51	209	131	21	0
19	159	4	1	4	799	5508.96	0.03	31.63	174.15	174268.6	1.84	9681.6	19	3	37	0	18	101	444	355	7	0
20	127	50	44	3	430	2589.62	0.02	47.5	54.52	123006.9	0.86	6833.7	19	3	14	0	25	40	289	152	98	0
21	264	4	1	4	1211	8866.85	0.03	30.05	295.05	266464.3	2.96	14804	19	1	10	0	15	145	630	581	7	0

3.3 Model Implementation Process

Following this proposed model we predict software defect. In MATLAB, an artificial neural network is trained using the Levenberg-Marquardt (LM) and Bayesian Regularization (BR) methods. Neurons made up hidden layer in the network architecture. Data sets from the PROMISE library were used to compare the built models to predict software defect. Mean squared error (MSE) and Neuros regression (R) values, processing time, performance, and gradient have all been computed after implementing the LM and BR neural network training methods.

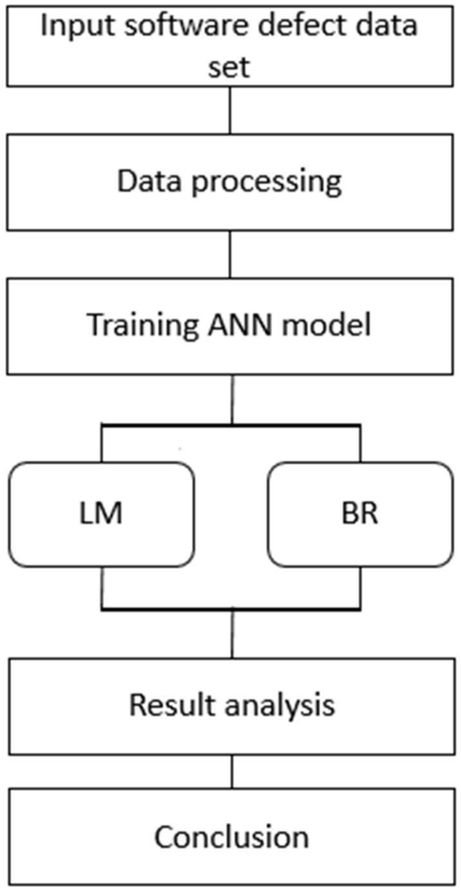


Fig 3.2: Research model

3.4 Performance Evaluation Measures

Defect prediction models created with levenberg-marquardt and Bayesian-regularization training techniques are experimentally evaluated by employing the following performance measures for the datasets included.

Error based measures: class of ANN functions are error based optimization functions. MSE is used to compare the results.

R value: The regression analysis can be used to assess a modeling technique's robustness. To determine how much the method fits the data, the R2 value of the test data is calculated R2 > 0.9 is typically considered to be a good fit.

Accuracy: The number of modules that are successfully predicted is how accurate a system is. We compare the accuracy of two algorithms LM and RM to see which algorithm is more accurate for predicting software defects.

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP}$$

CHAPTER 4
RESULTS AND DISCUSSION

In the MATLAB tool 2019a, the research was carried out. LM and BR training methods' are implemented in MATLAB. The number of neurons in the input layer is equal to the number of characteristics in the data set, and there is only one neuron in the bottom layer.

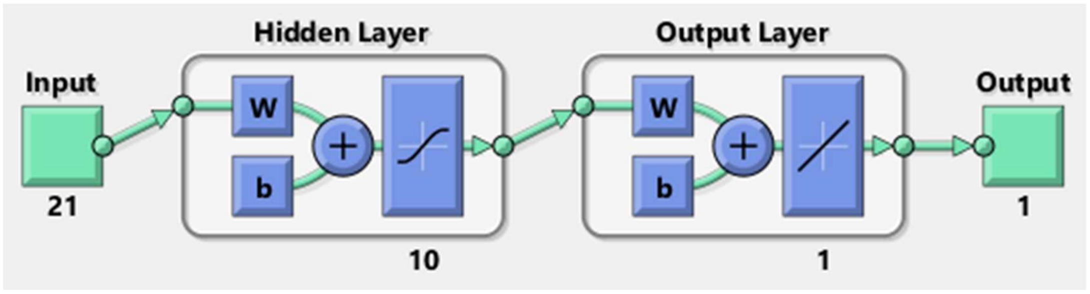


Fig 4.1: Neural network training

4.1 Experimental Setup

Using the LM and BR training of neural networks algorithms we are computed mean squared error (MSE), regression R values, epoch, processing, gradient and performance. Additionally provided are performance plots, training state plots, error histograms, and regression plots. The degree of difference between both the actual and predicted values of the dependent variable is determined by the mean squared error (MSE). The quality of fit increases with decreasing MSE value and vice versa. The degree to which the fitted model can account for changes in the dependent variable as a result of changes in the independent variable is indicated by the coefficient of correlation (R). 21 input layer, 10 hidden layer, and 1 output layer are trained in neural networks based on LM and BR models.

Table 4.1: Comparison of LM and BR progress

	Levenberg–Marquardt	Bayesian Regularization
Epoch	16 iteration	208 iteration
Processing	0: 00 : 00	0: 00 : 01
Performance	3.62e-05	8.71e-16
Gradient	0.00301	9.34e-8
MU	0.000100	0.500
MSE (validation)	3.31888e-0	0.00000e-0
R (validation)	9.66828e-1	0.00000e-0
MSE (testing)	5.90684e-2	1.83978 e-3
R (testing)	9.64029e-1	9.98062e-1

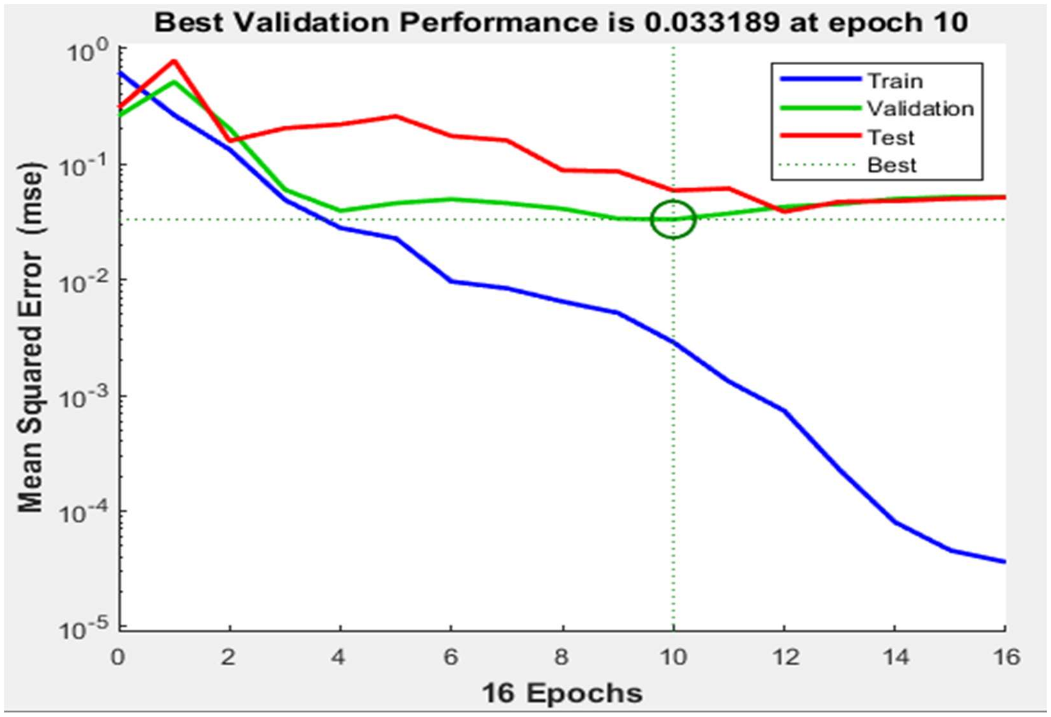


Figure 4.2: LM performance plot

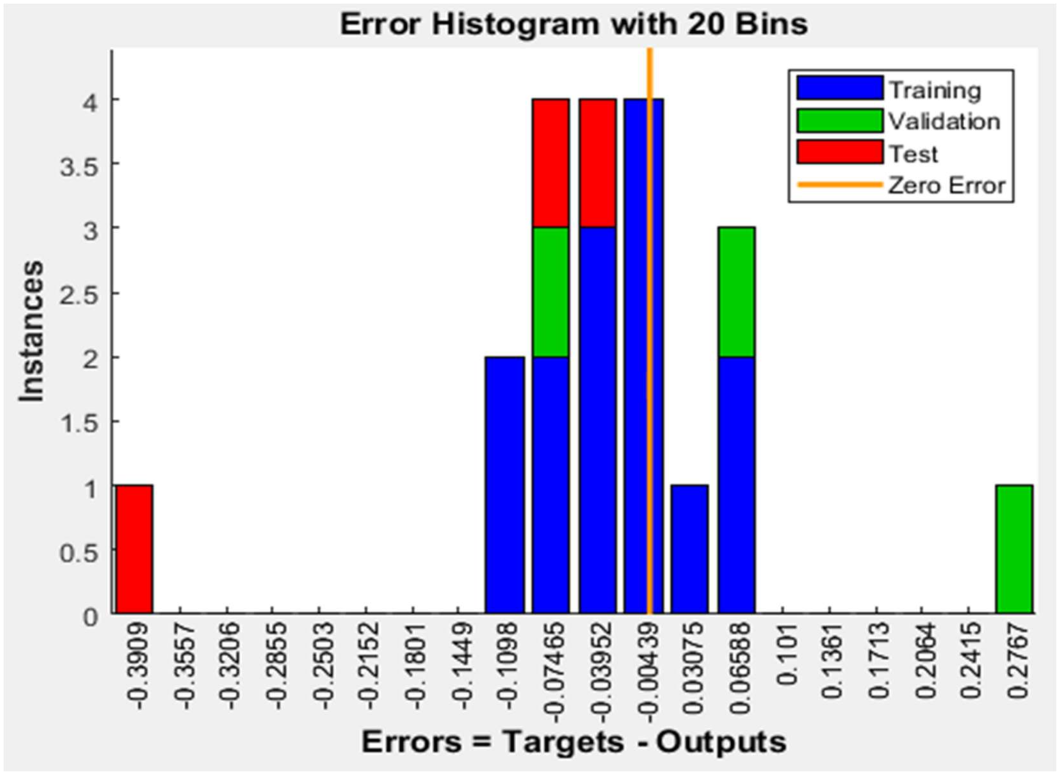


Figure 4.3: LM error histogram

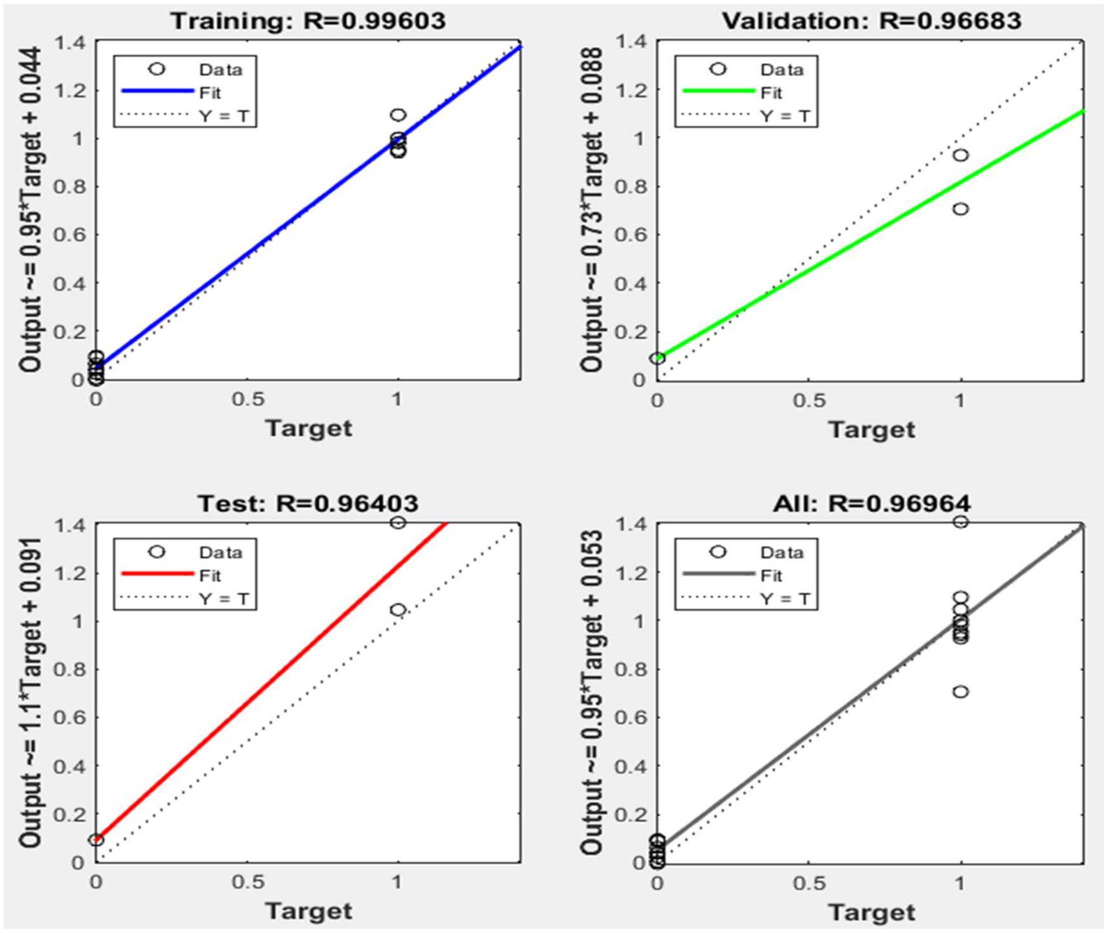


Figure 4.4: LM regression plot

Figure 4.2 displays performance plots that use the Levenberg-Marquardt method, which plots the mean squared error against 5 epoch values. The graph first exhibits a decreasing behavior between 0 and 10, so it follows a horizontal line trend for all values > 10.

Figure 4.3 displays an error histogram showing the distribution of errors during training, validation, and testing.

Figure 4.4 to help the analyst understand and improve the regression mode, the application automatically generates and presents the residuals when you perform the regression.

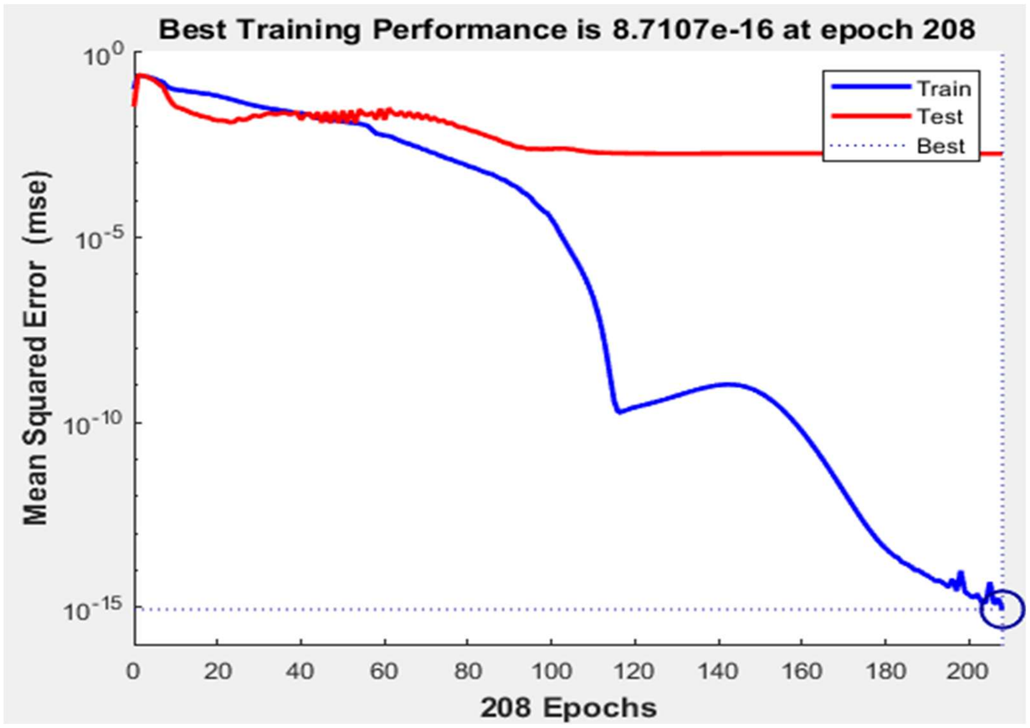


Figure 4.5: BR performance plot

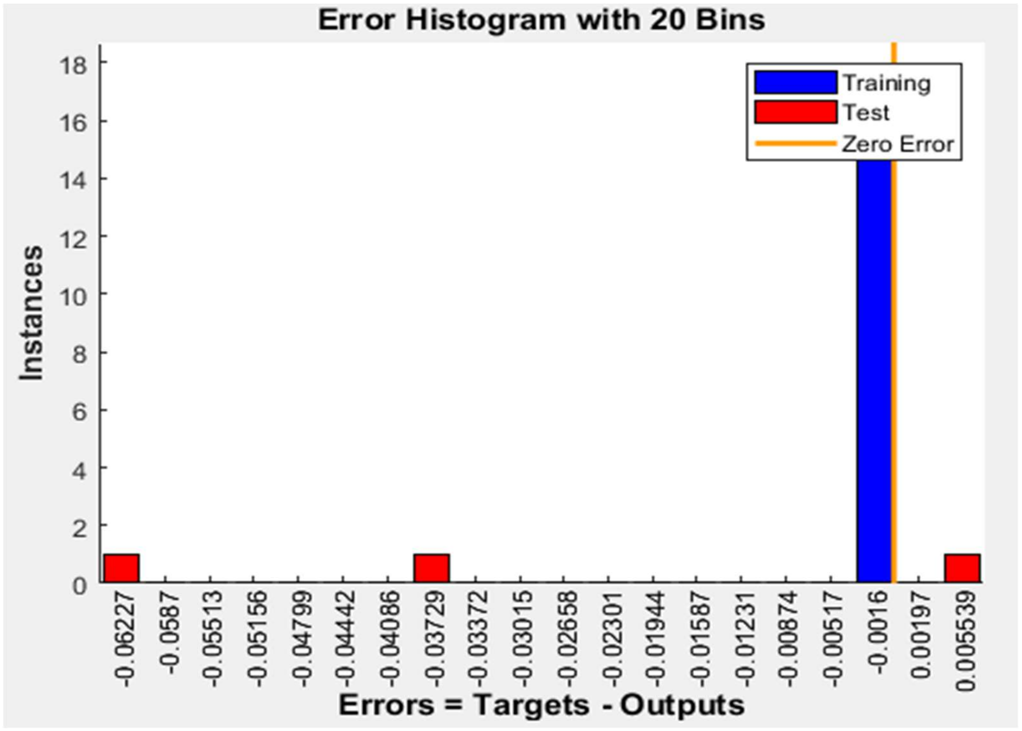


Figure 4.6: BR error histogram

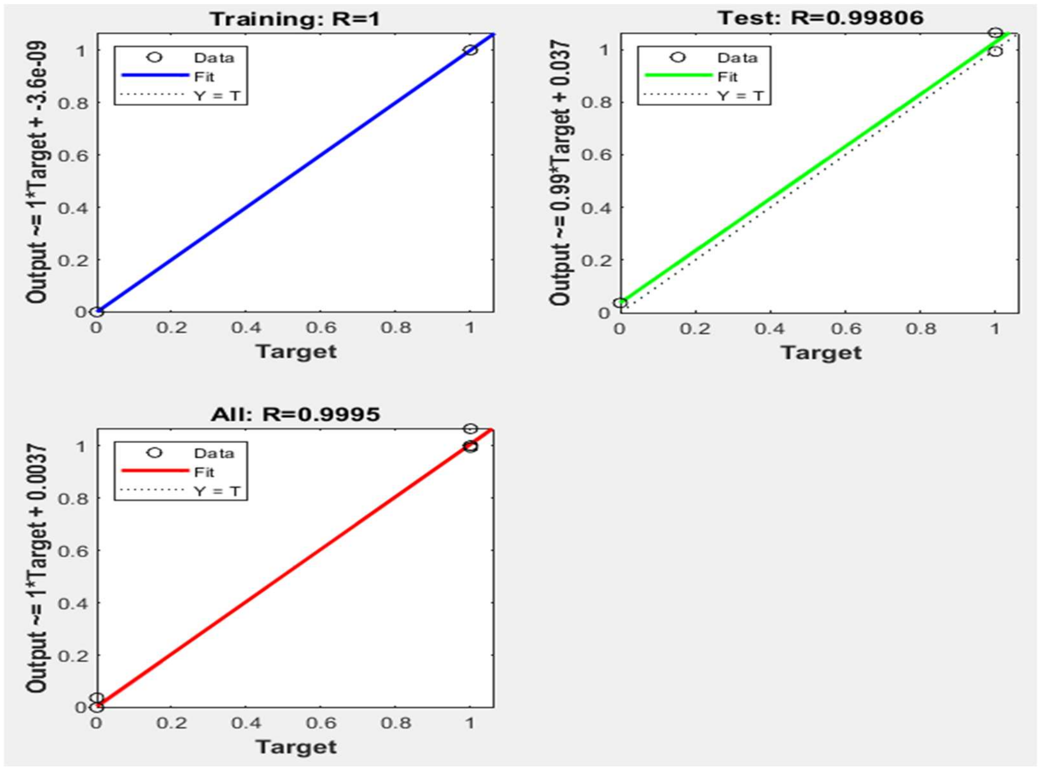


Figure 4.7: BR regression plot

Figure 4.5 shows the performance plot of the Bayesian technique that plots the mean squared error against the value of 208 epochs. The graphic displays the increasing behavior linearly from 0 to 200. It roughly follows a horizontal line pattern, with all values >200.

The error histogram in figure 4.6 illustrates how the training, validation, and testing errors of the Bayesian Regularization method are roughly symmetrically distributed.

The regression figure 4.7 contrasts the quality of the fit measure R for training, validation, and testing with targets on the horizontal plane and various functional structures along the vertical axis over the 4 specifications. R measure values in test and validation scenarios

4.2 Result Analysis

Based on BR and LM algorithm performance in MATLAB. The results obtained for MSC and R are shown in Table 4.2.

Table 4.2: MSG and R value

LM			**BR**	
MSE	R		MSE	R
1.58555e-2	9.69644e-1		2.75967e-4	9.99496e-1

Table 4.3: R Square value of test set

Algorithm	R square value
LM	0.99603
BR	1

Table 4.4: Accuracy

Algorithm	Accuracy
LM	83.33%
BR	91.18%

4.3 Result Discussion

From the implementation, Table 4.2 clearly shows the MSC value of BR algorithm is 2.75967e-4 and the MSC value of BR algorithm is 1.58555e-2. As MSC value of BR is less then MSC value of LR, so we say that the Bayesian regularization (BR) algorithm is the winner.

But the R2 values are completely the opposite. The higher the R^2 result, the more the fit. In Table 4.3, The R^2 value in LM performance is 99 percent and The R^2 value in BR performance is 100 percent. The value of R^2 is almost similar between LM and BR. Usually, R2 > 0.9 is usually treated as a good fit (Arora, 2016) This criterion showed that LM and BR were both reliable prediction approaches in defect datasets.

From the implementation, I got the best accuracy which was 91.18% in table 4.4. The Bayesian-based training function outperformed the LM methods on the accuracy parameters, performing over 90% of them on the dataset, as shown in table 4.4. This is similar to how they outperformed on the error scale.

In the area of software defect prediction, the Bayesian regularization-based Artificial Neural Networks (ANN) training algorithm surpasses the Levenberg-Marquardt-based approaches.so I can say that in term of two artificial neural network training algorithm LM and BR, the BR algorithm preforms best to predict software defect but the another algorithm LM is not bad. There is not much different between the performance of LM and the performance of BR.

So I can say that Using Artificial Neural Networks (ANN) model we can simply predict software defect.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATION

5.1 Conclusion

Because modern life is becoming more and more digital, there is an exponential rise in the demand for software systems, which now has led to a demand for higher-quality, more affordable software. Today, software developers are thought to be most concerned with timely delivery of high-quality software. Increasing complexity of software systems, it is not possible to deliver a high-quality maintainable software at a low cost in a timely manner. This problem can be resolved by employing methods for predicting software defects. ANN is the best approach to predicting software defect at the early stage of SDLC.

This problem is solved in this paper using Artificial Neural Networks (ANN) approach. Laverberg-Marquardt (LM) and Bayesian Regularization (BR), two common ANN-based training techniques, are empirically compared in this paper's performance evaluation for the purpose of predicting software defects. Using the command-line interface of MATLAB, an artificial neural network was created. Data sets from the promise repository are used to run the tests. The Ann models compared depending on confusion matrix, MSE, RMSE, R2 values. A comprehensive comparison revealed that BR performed better than LM, as measured by MSE, R2 value and accuracy. To get us better results in predicting software defects. We can use Laverberg-Marquardt (LM) which is an ANN-based training algorithm. To predict software defect Artificial Neural Networks (ANN) is a best approach.

The Artificial Neural Networks (ANN) training methods fall under the category of optimization algorithms where it is essential to optimize the weights in order to obtain the optimal performances. Additionally, studies have demonstrated the use of nature-inspired search-based optimization algorithms in the realm of software engineering

5.2 Recommendations for Future Works

It has been determined that ANN offers significant potential for software defect prediction, particularly when applied in a hybrid fashion following integration with other methods like future selection or other preprocessing methods. Additionally, it has been observed that the majority of studies have focused on preprocessing techniques to enhance the functionality of ANN classifiers.

The prediction of software defects may be further studied by concentrating on machine learning methods apart from ANNs. Additionally, specific data sets can be examined while keeping in mind the reported accuracy of recently presented methodologies. The issue of predicting software defects can also be studied using contemporary feature selection and ensemble techniques. In future Using Convolutional Neural Network (CNN) for software Defect Prediction. It makes use of deep learning to generate features efficiently

REFERENCES

Ahmad, M., & Aftab, S. (2017). Analyzing the performance of SVM for polarity detection with different datasets. International Journal of Modern Education and Computer Science, 9(10), 29.

Arora, I., & Saha, A. (2016, December). Comparison of back propagation training algorithms for software defect prediction. In 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I) (pp. 51-58). IEEE.

Awolusi, T. F., Oke, O. L., Akinkurolere, O. O., Sojobi, A. O., & Aluko, O. G. (2019). Performance comparison of neural network training algorithms in the modeling properties of steel fiber reinforced concrete. Heliyon, 5(1), e01115.

Bre, F., Gimenez, J. M., & Fachinotti, V. D. (2018). Prediction of wind pressure coefficients on building surfaces using artificial neural networks. Energy and Buildings, 158, 1429-1441.

Bernard, W., & Sazzad, H. (2006). Software Process Improvement in Bangladesh. Software Engineering Research and Practice, (1).

Cao, Q., Sun, Q., Cao, Q., & Tan, H. (2015, October). Software defect prediction via transfer learning based neural network. In 2015 First international conference on reliability systems engineering (ICRSE) (pp. 1-10). IEEE.

Gao, K., & Khoshgoftaar, T. M. (2011, July). Software Defect Prediction for High-Dimensional and Class-Imbalanced Data. In *SEKE* (pp. 89-94).

Goyal, J., & Ranjan Sinha, R. (2022). Software defect-based prediction using logistic regression: review and challenges. In Second International Conference on Sustainable Technologies for Computational Intelligence (pp. 233-248). Springer, Singapore

Iftikhar, A., Alam, M., Ahmed, R., Musa, S., & Su'ud, M. M. (2021). Risk prediction by using artificial neural network in global software development. *Computational intelligence and neuroscience*, *2021*.

Jindal, R., Malhotra, R., & Jain, A. (2014, October). Software defect prediction using neural networks. In Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization (pp. 1-6). IEEE.

Jayanthi, R., & Florence, L. (2019). Software defect prediction techniques using metrics based on neural network classifier. Cluster Computing, 22(1), 77-88.

Jin, C., & Jin, S. W. (2015). Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization. Applied Soft Computing, 35, 717-725.

Khan, M. A., Elmitwally, N. S., Abbas, S., Aftab, S., Ahmad, M., Fayaz, M., & Khan, F. (2022). Software Defect Prediction Using Artificial Neural Networks: A Systematic Literature Review. *Scientific Programming*, *2022*.

Kim, S., Zhang, H., Wu, R., & Gong, L. (2011, May). Dealing with noise in defect prediction. In 2011 33rd international conference on software engineering (ICSE) (pp. 481-490). IEEE.

Laradji, I. H., Alshayeb, M., & Ghouti, L. (2015). Software defect prediction using ensemble learning on selected features. Information and Software Technology, 58, 388-402.

Li, J., He, P., Zhu, J., & Lyu, M. R. (2017, July). Software defect prediction via convolutional neural network. In 2017 IEEE international conference on software quality, reliability and security (QRS) (pp. 318-328). IEEE.

Manjula, C., & Florence, L. (2019). Deep neural network based hybrid approach for software defect prediction using software metrics. Cluster Computing, 22(4), 9847-9863.

Myers, G. J., Sandler, C., & Badgett, T. (2011). The art of software testing. John Wiley & Sons.

Miholca, D. L., Czibula, G., & Czibula, I. G. (2018). A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. Information Sciences, 441, 152-170.

Moré, J. J. (1978). The Levenberg-Marquardt algorithm: implementation and theory. In Numerical analysis (pp. 105-116). Springer, Berlin, Heidelberg.

MacKay, D. J. (1992). A practical Bayesian framework for backpropagation networks. Neural computation, 4(3), 448-472.

Nguyen-Truong, H. T., & Le, H. M. (2015). An implementation of the Levenberg–Marquardt algorithm for simultaneous-energy-gradient fitting using two-layer feed-forward neural networks. Chemical Physics Letters, 629, 40-45.

Ozyildirim, B. M., & Avci, M. (2013). Generalized classifier neural network. Neural Networks, 39, 18-26.

Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave macmillan.

Rawat, M. S., & Dubey, S. K. (2012). Software defect prediction models for quality improvement: a literature study. *International Journal of Computer Science Issues (IJCSI)*, *9*(5), 288.

Rahman, T., Saha, S. K., Sohel, M. S. R., Maula, M. T., Bowmik, A., & Nabil, R. H. (2022). Risk Identification and Analysis in Software Development in Bangladesh IT Industry: A Hybrid Model. *AIUB Journal of Science and Engineering (AJSE)*, *21*(1), 37-44.

Sethi, T. (2016, September). Improved approach for software defect prediction using artificial neural networks. In *2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)* (pp. 480-485). IEEE.

Wu, R., Zhang, H., Kim, S., & Cheung, S. C. (2011, September). Relink: recovering links between bugs and changes. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering* (pp. 15-25).

Zhang, Y., Lo, D., Xia, X., & Sun, J. (2015, July). An empirical study of classifier combination for cross-project defect prediction. In *2015 IEEE 39th Annual computer software and applications conference* (Vol. 2, pp. 264-269). IEEE.