THE COMPARISON OF CLASSIFICATION METHODS ON SOFTWARE DEFECT DATA SETS

HNIN YI SAN

M.C.Sc. DECEMBER 2018

THE COMPARISON OF CLASSIFICATION METHODS ON SOFTWARE DEFECT DATA SETS

By

Hnin Yi San

B.C.Sc. (Honours)

A dissertation submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (M.C.Sc.)

University of Computer Studies, Yangon

December 2018

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to **Prof. Dr. Mie Mie Thet Thwin**, the Rector of the University of Computer Studies, Yangon, for her kind permission to develop this thesis.

My sincere thanks and regards go to **Dr. Thi Thi Soe Nyunt**, Professor, Head of Faculty of Computer Science, University of Computer Studies, Yangon, for her kind management throughout the completion of this thesis.

I would like to express my heartfelt thanks to my supervisor, **Dr. Khine Khine Oo,** Professor, Faculty of Information Science, University of Computer Studies, Yangon, for her encouragement, patient and invaluable supervision in compiling the materials for my thesis.

I also deeply thank **Daw Aye Aye Khine**, Associate Professor and Head of the Department, the Department of Language, for editing my thesis from the language point of view.

Finally, I especially thank my parents, all of my friends and their suggestions, support and generous help rendered to me during the development of thesis. Their love and concerned encouragement has strengthened me through the studies.

Moreover, I would like to thank all the staffs, teachers from the University of Computer Studies, Yangon for their support.

ABSTRACT

Nowadays, it is difficult for us to imagine a life without devices that is controlled by $^{\text{software}}$. Software quality has become the main concern during the software development. Software quality is a field of study and practice that describes the desirable attributes of software products. Software quality prediction is a process of utilizing software metrics such as code-level measurements and defect data to build classification models that are able to estimate the quality of program modules. The major problem that affects the quality of datasets is high dimensionality and class imbalanced. A more useful and efficient mechanism is k Nearest Neighbor method, where Nearest Neighbor classify classes of testing dataset based on *k* nearest neighbor of training dataset. Another mechanism is Class Based Weighted k Nearest Neighbor with BINER Algorithm for classifying classes of testing dataset. By using BINER Algorithm, it narrows down the training dataset range instead of whole training dataset that has the maximum likelihood of occurrence and then CBW k-NN classifies classes of testing dataset based on this range. This thesis is the comparison of two classification methods by classifying classes of testing dataset focuses on NASA MDP (PC1, CM1 and JM1) datasets. The comparison results of two methods based on Accuracy, Reliability, Mean Absolute Error and Root Mean Squared Error.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1
INTRODUCTION

Software industry has been trying to find out ways for developing software product within time and according to the needs of customers. Software quality is software functional and structural quality in software engineering. Functional quality reveals functional requirements whereas structural quality highlights non-functional requirements. Software quality and reliability have become the main concern during the software development. The desirable attributes of software products are software quality that is a field of study and practice. The performance of the software products must be perfect without any defects. A subset of software metrics, software quality metrics depend on the quality aspects of product, process, and project.

Software life cycle is a human activity, so it is difficult to create software without defects but is possible to prevent the injection of defects. In software life cycle, software requirements, software design, software coding and software testing are the important aspects for detecting software defects [14]. During the stages of software requirements, software design and software testing, software defects can evoke and affect in software source codes. Therefore, prediction software defect is the most popular method for software industry to detect software defect by using software source codes. The abstract expressions of software source codes complexity are size and complexity metrics which are line of code, design and cyclomatic complexity and so on.

A large portion of the project budget is the finding and fixing the defects after delivery. Therefore, detection software defect before delivery can wake up the success of project quality and cost. A software defect is an incorrect or unexpected result and unintentional outcomes by using software and is also called an error, fault, flaw, or failure in a computer system or program [3]. The software defect is the main software quality characteristic. During software development and maintenance, the most expensive activity is the finding and correcting software defects. Therefore, the key element for a creative and successful software project is the development high quality software within the assigned time and budget. A panel at IEEE Metrics 20022 also concluded that manual software reviews can find only 60 percent of defects. In the software engineering field, software defect prediction has been an important research topic, especially to solve the inefficiency and ineffectiveness of existing manufacturing approach of software testing and reviews.

The reduction number of faults in the delivered code is the main goal of software developers to create the better software. Therefore, they need to fix the faults as early as possible, in order to ensure the reliability of software systems. Moreover, the earlier an error can identify, the better and more cost effectively can be fixed. Therefore, the need and high demand in software industry is to predict software defects across the stages of software development process. The machine learning is becoming an important field of computer science. The machine learning is associated the number of core algorithms for pattern recognition and data mining. The classification, clustering and prediction are the useful machine learning algorithms. Classification algorithms have been successfully used in several areas and different applications have their own related issues. The amount of data in our lives appears the increasing more and more and there is no end in sight.

1.1 Objectives of the Thesis

The primary objectives of this thesis are to find out the way that can identify the software defects and compare classification methods on imbalanced software defect datasets. The other objectives are

- To provide the effective method for classification software defect datasets
- To study basic concepts of classification methods with the purpose of to be applied as software defect classification techniques
- To identify defects and non-defects using k Nearest Neighbor and Class Based Weighted k Nearest Neighbor with BINER Algorithm
- To classify class label on imbalanced software defect datasets using both methods
- To provide comparison of two methods of their abilities of software defect classification using Accuracy, Reliability, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE)

1.2 Motivation of the Thesis

In the data mining field, a series of challenges have recently developed and rapidly triggered from academic to the resulting needs of real-life applications and applied science. This thesis is concerned with classification tasks and related issues. They are not complete and not relevant records and redundant pieces of the information, imbalanced class distribution and error costs of software [18].

Today, usage of software is increasing very rapidly so it becomes very difficult to create software without defect. It is necessary for developers to classify features of software defect by using module metrics of the better software. Therefore, developers need to know characteristics of defect for developing software. In the field of machine learning, classification has been a valuable and energetic of research. In classification, the goal is to classify class value for testing data and to separate a given lots of data items into groups. Classification can develop the performance of retrieval on features of software.

1.3 Organization of the Thesis

This thesis is mainly composed of five chapters. Chapter 1 introduces the basic information about the thesis and motivation including the scope and objectives. Chapter 2 describes the literature of the classification methods that pointed out some systems offering important knowledge and background history for the understanding of the following chapters. Chapter 3 presents the theoretical background of the system and about software defect datasets. Chapter 4 describes overview design and implementation of the system that implemented by using PHP Hypertext Processor Language (PHP) and includes the experimental result of the thesis. Chapter 5 discusses conclusion and the directions for further development and deals with problems for future research.

CHAPTER 2
BACKGROUND THEORY

This chapter presents data mining with functionalities and application area, data preparation and cleaning, classification methods, classification rules and challenges and evaluation methods. Firstly, this chapter describes the meaning of data mining that is important to analyze a large amount of data. Secondly, it explains how to use data mining methods in many application areas. Thirdly, it describes about a several of classification methods and classification rules and challenges. Finally, it presents evaluation methods in order to compare and evaluate between classification methods with calculation equations.

2.1 Data Mining

Data mining is about solving problems by analyzing data already presented in databases. In highly competitive business growth, the customer-centered, service-oriented economy, data is the raw material is need. It is also defined as the process of discovering patterns in data that must be automatic or semiautomatic. The patterns discovered must be meaningful that lead to an economic advantage. The nontrivial predictions on new data are used to make useful patterns. The data patterns are used by economists, statisticians, forecasters, and communication engineers to seek automatically, identified, validated, and used for prediction.

Data mining refers to extracting or “mining” knowledge from large amounts of data. The synonym for another popularly used term for data mining is, Knowledge Discovery from Data, or KDD. Knowledge discovery consists of an iterative sequence of the following steps:

1. Data cleaning- it is use to remove inconsistent and noise data.
2. Data integration- it is use to combine multiple data sources.
3. Data selection- it is use to retrieve data relevant from the database to the analysis task.
4. Data transformation- it is use to transform or consolidate into forms by performing summary or aggregation actions.
5. Data mining- it is use to extract data patterns by applying intelligent methods.

6. Pattern evaluation- it is use to identify truly interesting patterns representing knowledge based on some incorrect measures.
7. Knowledge presentation- it is use to present the mined knowledge to the user based on visualization and knowledge representation techniques.

2.1.1 Data Mining Functionalities

The functions of data mining are used to identify the kind of patterns to be found in data mining tasks. In general, data mining tasks can be classified into two categories such as descriptive and predictive. Descriptive mining tasks characterize the general properties of the data in the database. Predictive mining tasks perform inference on the current data in order to make predictions [23].

Descriptive Function

The descriptive function deals with the general properties of data in the database. The list of descriptive functions −

(i) Class/Concept Description
(ii) Mining of Frequent Patterns
(iii) Mining of Associations
(iv) Mining of Correlations
(v) Mining of Clusters

(i) Class/Concept Description

Class/Concept refers to the data associated with the classes or concepts. For example, in a company, the classes of items for sales include computer and printers, and concepts of customers include big spenders and budget spenders. These descriptions can be derived by the following two ways −

- **Data Characterization** − It refers to summarizing data of class under study that is called as target class.
- **Data Discrimination** − It refers to the classification of a class with some predefined group or class.

(ii) Mining of Frequent Patterns

Frequent patterns are those patterns that occur frequently in transactional data. The list of kind of frequent patterns are-

- **Frequent Item Set** − It refers to a set of items that frequently appear together, e.g. milk and bread.
- **Frequent Subsequence** − A sequence of patterns that occur frequently such as purchasing a camera is followed by memory card.
- **Frequent Sub Structure** − Substructure refers to different structural forms, such as graphs, trees, or lattices, which may be combined with item-sets or subsequences.

(iii) Mining of Association

Associations are used in retail sales to identify data patterns that are frequently purchased together. This process refers to the process of determining association rules and uncovering the relationship among data.

(iv) Mining of Correlations

Correlation is a kind of additional analysis that performs to uncover interesting statistical correlations between associated-attribute-value pairs or between two item sets to analyze positive, negative pair or no effect on each other.

(v) Mining of Clusters

Cluster refers to a group of similar kind of objects. Cluster analysis analyses group of objects that are very similar to each other in same cluster but are highly different from the objects in other clusters.

Classification and Prediction

Classification is the process of finding a model that describes the data classes or concepts. The purpose of classification is to use this model to predict the class of objects whose class label is unknown. This derived model is based on the analysis of sets of training data i.e. the data object whose class label is well known. The derived model can be presented as Classification (IF-THEN) Rules, Decision Trees, Mathematical Formulae and Neural Networks.

Prediction is used to predict missing data or unavailable numerical data values rather than class labels. Regression Analysis is generally used for prediction. Prediction can also be used for identification of distribution trends based on available data. Outlier Analysis refers to the data objects that do not comply with the general behavior or model of the data available. Evolution Analysis refers to the description and model regularities or trends for objects whose behavior changes over time.

2.1.2 Application Areas of Data Mining

Data mining is a process that analyzes a large amount of data to find new and hidden information that improves business efficiency. Many industries have been implementing data mining to their critical business processes to gain competitive advantages and help business grows [21].

In **Sales/Marketing**, data mining supports businesses to understand the hidden patterns inside historical purchasing transaction data [22]. It can help planning and launching new marketing campaigns in a prompt and cost-effective way. Retail companies identify customer's behavior buying patterns by using data mining techniques. Data mining is used for market basket analysis to provide information on what product combinations were purchased together and in what sequence. In addition, it encourages customers to purchase related products that they may have been missed or overlooked.

In **Banking/ Finance**, several data mining techniques as distributed data mining have been researched, modeled and developed to help credit card fraud detection [22]. To help the bank process retaining credit card customers, data mining is applied. By analyzing the past bank information data, data mining can help banks to predict customers that likely to change their credit card affiliation. So, they can plan and launch different special offers to recall those customers. Data mining can identify credit card spending by customer groups.

In **Education**, there is a new emerging field, called Educational Data Mining (EDM) concerns with developing methods that discover knowledge from educational environments data originating. The goals of EDM are identified as predicting students' future learning behavior, studying the effects of educational support, and advancing scientific knowledge about learning methods. Data mining can use by an institution to take accurate decisions and can also predict the results of the student focus on what to teach and how to teach. Learning pattern of the students can be captured and used to develop techniques to teach them by using data mining.

In **Research Analysis**, history data shows that researchers have witnessed revolutionary changes in research. Data mining is helpful in data cleaning, data pre-processing and integration of databases. By using data mining, researchers can find any similar data from the database that might bring any change in the research. Data visualization and visual data mining provide them with a clear view of the data.

In **Lie Detection**, this filed includes text mining also. Apprehending a criminal is easy whereas bringing out the truth from him is difficult. Law enforcement can use mining techniques to investigate crimes, monitor communication of suspected terrorists. This process seeks to find meaningful patterns in data which is usually unstructured text. The data samples collected from previous investigations are compared and a model for lie detection is created.

In **Telecommunication**, today the telecommunication industry is one of the most emerging industries providing various services such as fax, pager, cellular phone, internet messenger, images, e-mail, web data transmission, etc. Due to the development of new computer and communication technologies, the telecommunication industry is rapidly expanding. This is the reason why data mining becomes very important to help and understand the business. Data mining in telecommunication industry helps in identifying the telecommunication patterns, catch fraudulent activities, make better use of resource, and improve quality of service. The list of examples for which data mining improves telecommunication services are multidimensional analysis of telecommunication data, fraudulent pattern analysis, identification of unusual patterns, multidimensional association and sequential patterns analysis, mobile telecommunication services, and use of visualization tools in telecommunication data analysis.

2.2 Data Preparation and Cleaning

Data pre-processing is the important step in the data mining process. The phrase “Garbage In, Garbage Out” is particularly applicable to data mining and machine learning. Data collecting methods are often loosely controlled, resulting in out-of-range values such as Income value is (-100), impossible data combinations such as Gender is Male and Pregnant is Yes, missing values, etc. Analyzing data has been carefully separated for such problems to prevent producing misleading results. Thus, the demonstration and quality of data is first and foremost before processing an analysis. The irrelevant and redundant information present noisy and unreliable data for knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can be sizeable amount of processing time. The product of data pre-processing is the final training dataset. Data pre-processing is one of the most critical steps in a data mining process to prepare and transform the initial dataset [25]. It is an important issue for both data warehousing and data mining, as real-world data tends to be incomplete, noisy, and inconsistent. It includes data cleaning, data integration, data transformation, and data reduction.

Data cleaning can be used to fill missing values, smooth noisy data, identify outliers, and correct data inconsistencies. Data integration combines data from multiples sources to form a coherent data store. Data transformation conform the data into appropriate forms for mining methods. Data reduction techniques can be used to obtain a reduced representation of the data, while minimizing the loss of information content. They are data cube aggregation, dimension reduction, data compression, numerosity reduction, and discretization. Noisy data is a random error or variance in a measured variable. The data smoothing techniques are binning methods, clustering, combined computer and human inspection, regression. Binning methods smooth a sorted data value by consulting the neighborhood, or values around it. Data can be smoothed by fitting the data to a function, such as with regression. Using regression to find a mathematical equation to fit the data helps smooth out the noise.

2.3 Classification Methods

Classification is the process of finding a model (or function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. The derived model is based on the analysis of a set of training data (i.e., data objects whose class label is known). Classification predicts categorical (discrete, unordered) labels.

The classification learning is sometimes called supervised learning because the method operates under supervision by being provided with the actual outcome for each of the training examples. The success of classification learning can be decided by trying out the concept description that is learned on an independent set of test data. The success rate on test data gives an objective measure of how well the concept has been learned. In many practical data mining applications, success is measured more subjectively in terms of how acceptable the learned description. Many classification and prediction methods have been proposed by researchers in machine learning, pattern recognition, and statistics.

2.3.1 Decision Tree (C4.5)

Decision Tree is the process of learning a tree from pre-classified training examples [16]. A decision tree is like a flowchart tree structure, where each internal node called non-leaf node denotes a test on an attribute. Each branch represents an outcome of the test, and each leaf node or terminal node holds a class label. The topmost node in a tree is the root node. Decision tree algorithms transform from the raw data to rule based mechanism.

C4.5 is an improved version of ID3 (Iterative Dichotomizer 3 algorithm), an inductive learning method developed by John Ross Quinla at 1989. C4.5 can accept input values as both symbolic and numeric, and generates a classification tree for output. It employs a splitting procedure which recursively partitions a set of examples into disjointed subsets. C4.5 accepts both continuous and discrete features, handles incomplete data points, solves over-fitting problem by bottom-up technique and can be applied different weights that comprise the training data. For example, in the training phase, the gain ratio of each attribute is adjusted by a factor which depends on the number of complete records (in that attribute) in the training set. Input/output (activation) functions are continuous and differentiable. The output is a classification tree where the leaves contain class assignments determined by majority rule.

A decision tree is a special case of a state-space graph. It is a tree in which each internal node corresponds to a decision that has a sub tree for each possible outcome of the decision. Decision trees can be used to model problems in which a series of decisions leads to a solution. Its programs construct a decision tree from a set of training cases and are used to improve the prediction and classification accuracy of the algorithm. It is widely applied in various areas since it is robust to data scales or distributions by comparing to other data mining techniques,

2.3.2 Naive Bayesian Classification

Naive is a statistical classifier that can predict class membership probabilities such as the probability a given example belongs to a particular class [10]. Naive Bayes classifier is a probabilistic classifier that produces probability estimates based on the Bayes theorem rather than predictions. For each class value, they estimate the probability that a given instance belongs to that class by using a small amount of training data to estimate. It assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. Bayesian classifiers have also exhibited high accuracy and speed when applied to large database.

The Naive Bayes classifier technique is based on Bayes' theorem and is particularly appropriate when the dimensionality of the feature space is high. For example, a vector $x=(x_1,x_2,\ldots,x_n)$ of *n* features is associated with each observation and Naive Bayes learns the class conditional probabilities $p(x_i|y_i)$ of each categorical variable i, $i=1,2,\ldots,n$, given the class label y_i. A new observation with feature vector x is classified by using the Bayes' rule to compute the posterior probability of each class y_i given the vector of attributes. The basic assumption of Naive Bayes' classifier is that the variables are conditionally independent given the class label.

2.3.3 Neural Network (NN)

A neural network (NN) can be defined as reasoning model based on the human brain [11]. A NN consists of a number of interconnected processors called neurons. Firstly, a neuron receives input signals from its input links, computes an output signal and transmits this signal through its output links. An input signal can be raw data or the outputs from other neurons. The output signal can be either a final solution to the problem or an input to other neurons. A NN is set through repeated adjustments of these weights. A neural network model, the branch of artificial intelligence is generally referred to as Artificial Neural Networks (ANNs). ANN constructs the system to execute task, instead of programming computational system to do definite tasks.

Neural Networks are capable of predicting new observations from existing observations. The neurons within the network work together, in parallel, to produce an output function. Since the computation is performed by the collective neurons, a neural network can still produce the output function even if some of the individual neurons are malfunctioning (the network is robust and fault tolerant). Neural Networks (NN) are important data mining tool used for classification and clustering [21]. It is an attempt to build machine that can mimic brain activities and be able to learn. Basic NN consists of three layers such as input, output and hidden layer. Each layer can have number of nodes and nodes are connected from input layer to hidden layer and hidden layer's nodes are connected to the nodes from output layer. Those connections represent weights between nodes. Back Propagation Neural Network (BPNN), one of the most popular NN algorithms need a very large number of training samples and need a lot of time to gradually approach good values of the weights.

2.3.4 Support Vector Machine (SVM)

The concept of decision planes to define decision boundaries is Support Vector Machine (SVM) that supports both regression and classification. A decision plane is the one that separates between a set of objects having different class membership [17]. SVM performs classification task by constructing hyper plane in a multidimensional space that separates cases of different class labels. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyper plane.

SVM was first proposed by Vapnik at 1995 as learning systems for binary classification [11]. It is trained using an algorithm from optimization theory and statistical learning theory to derive a separating hyper plane in a high dimensional feature space. SVMs are based on a nonlinear mapping of the problem data into a higher dimension feature space. However, the learning algorithm may be inefficient and SVMs may be difficult to implement as a large number of 17 parameters is required. In addition, small training samples can result in over fitting, with poor generalization ability. The original model proposed by Vapnik was a linear classifier, but other types were later proposed in order to improve the accuracy of the original model. The main difference of the new models compared to the initial model is the function used to map the data into a higher dimensional space. New functions were proposed, namely: polynomial, Radial Basis Function (RBF) and sigmoid. All these functions transform the original data into a higher dimensional space and then the linear classifier is used subsequently.

2.3.5 Genetic Algorithms

Genetic Algorithms attempt to incorporate ideas of natural evolution [6]. Genetic algorithms are used to discover classification rules for data that can be used for predictions. The genetic algorithms are adaptive techniques that can be successfully used to solve complex search and optimization problems. They are based on the principles of genetics and Darwin's natural selection theory ("the one that is best endowed, survives"). In data mining, genetic algorithms have been effectively used in order to determine classification rules and to search for appropriate cluster centers, to select the attributes of interest in predicting the value of a target attribute and so on. By using some hybrid algorithms, classification of instances was performed such as Genetic Algorithms and Particle Swarm Optimization, respectively Naive Bayes and k-Nearest Neighbors. A few applications in which genetic algorithms were effectively applied to solve classification problems are prints' classification, heart disease classification, classification of emotions on the human face.

The fitness functions of the genetic algorithms used for mining classification rules may contain metrics concerning predictive accuracy, rule comprehensibility as well as rule interestingness [19]. Diverse studies suggest genetic algorithms with fitness functions that take into consideration in different ways. Genetic algorithms are a type of optimization algorithm, meaning they are used to find the maximum or minimum of a function [5]. These algorithms are far more efficient and powerful than random and exhaustive search algorithms. In data mining, the advantage of Genetic algorithm becomes more obvious when the search space of a task is large. Genetic algorithm is a search technique used in computing to find exact or approximate solution to optimization and search problems.

2.4 Classification Rules and Challenges

The process of assigning each element in a population to one of the pre-defined classes is defined by classification rules. A perfect classification process is such that every element in the dataset is assigned to the class it really belongs to (High Accuracy classification). An imperfect classification process is such that some errors appear like false negatives or false positives. Statistical analysis is then applied to analyze the efficiency of the classification algorithm [17].

2.5 Evaluation Methods

Classification methods can be compared and evaluated according to following criteria:

Accuracy- The accuracy of a classifier refers to the ability to correctly predict the class label of new or previously unknown data.

Speed- This refers to the computational costs involved in generating and using the given classifier or predictor.

Robustness: This is the ability of the classifier or predictor to make correct predictions given noisy data or data with missing values.

Scalability: This refers to the ability to construct the classifier or predictor efficiently given large amounts of data.

Interpretability: This refers to the level of understanding and insight that is provided by classifier or predictor.

The reliable estimate of predictor accuracy is measured in terms of error. For example, D^T be a test set of the form $(X_1, y_1), (X_2,y_2), \ldots , (X_d, y_d)$, where the X_i are the *n*-dimensional test tuples with associated known values, y_i, for a response variable, *y*, and *d* is the number of tuples in D^T. The accuracy of a predictor is estimated by computing an error based on the difference between the predicted value and the actual known value of y for each of test tuples, X. Loss functions measure error between actual value, y_i and predicted value, y_i'. The most common loss functions can be executed in Equation 2.1 and 2.2.

$$\text{Absolute error} : |y_i -, y_i'| \tag{2.1}$$

$$\text{Squared error} : (y_i -, y_i')^2 \tag{2.2}$$

Based on the above, the test error (rate), or generalization error, is the average loss over the test set. The most popular evaluation metric to measure the prediction accuracy is Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

$$Mean\ Absolute\ Error\ (MAE) = \frac{\sum_{i=1}^{d} |y_i - y_i'|}{d} \tag{2.3}$$

$$Mean\ Squared\ Error\ (MSE) = \frac{\sum_{i=1}^{d} (y_i - y_i')^2}{d} \tag{2.4}$$

$$Root\ Mean\ Squared\ Error\ (RMSE) = \sqrt{\frac{\sum_{i=1}^{d} (y_i - y_i')}{d}} \tag{2.5}$$

The Mean Squared Error exaggerates the presence of outliers, while the mean absolute error does not. Root Mean Squared Error can accomplish by taking the square root of the Mean Squared Error. This is useful in that it allows the error measured to be of the same magnitude as the quantity being predicted. MAE, MSE and RMSE can range values from 0 to ∞. The lower the error values, the better the model is. If the square root of the mean squared error is taken, this is useful to allow the error measured of the same magnitude as the quantity being predicted.

CHAPTER 3
METHODS OF THE PROPOSED SYSTEM

This chapter presents the background theory of k nearest neighbor method, class based weighted k-NN method, BINER algorithm and software defect datasets and imbalanced dataset. Firstly, this chapter describes about software defects that is important to develop and maintain software. Secondly, it explains how to applied data mining methods for software defect detection. Thirdly, it describes about the nature of software defect dataset and their characteristics. Finally, it presents background theories such as k-NN method and CBW k-NN method with BINER algorithm with flowchart figures.

3.1 Software Defect

During software development and maintenance, the costs of finding and correcting software defects have been the most expensive activity. A panel at IEEE Metrics 20022 also decided that manual software reviews can find only 60 percent of defects [23]. Therefore, software defect prediction became an important research topic in the software engineering field. Especially, it is to solve the inefficiency and ineffectiveness of existing industrial approach of software testing and reviews. The accurate prediction of defect prone software modules can help direct test effort, reduce costs, improve the software testing process, and identify refactoring candidates that are predicted as fault-prone. Software fault prediction approaches are much more efficient and effective to detect software faults compared to software reviews.

Various machine learning classification algorithms have been applied for software defect prediction, including Logistic Regression, Decision Trees, Neural Networks and Naive-Bayes. The software defect prediction remains a largely unsolved problem that is the comparisons and benchmarking result of the defect prediction using machine learning classifiers. There is a need of accurate defect prediction model for large-scale software system. Two common aspects of data quality that can affect classification performance are class imbalanced and noisy attributes of data sets. Software defect datasets have an imbalanced nature with very few defective modules compared to defective ones. Imbalanced can lead to a model that is not practical in software defect prediction, because most instances can be predicted as non-defect prone.

3.2 Software Defect Dataset

The real time defect datasets are taken from the NASA's MDP (Metric Data Program) data repository, created by NASA MDP [13, 20]. For example, PC1 dataset which is collected from flight software for an earth orbiting satellite coded in C programming language, containing 1109 modules. And CM1 dataset which is collected from NASA spacecraft instrument, containing 402 modules and JM1 dataset which is collected from Real-time predictive ground system, containing 1096 modules. These datasets are coded in C programming language. All these datasets varied in percentage of defect modules, with PC1 dataset containing the least number of defect modules.

The metrics in NASA MDP datasets describe vary in size and complexity, programming languages, development processes, etc. [20]. When reporting a fault prediction modeling experiment, it is important to describe the characteristics of the datasets. Each dataset contains twenty-one software metrics, which describe product's size, complexity and some structural properties. Also the product metrics and product module metrics available in dataset which can also be use are the product requirement metrics are as follows:

- Module
- Action
- Conditional
- Continuance
- Imperative
- Option
- Risk_Level
- Source
- Weak_Phrase

The product module metrics are as follows:

Table 3.1 List of Product Module Metrics

Number	Module Metrics Name
1.	Module
2.	Loc_Blank
3.	Branch_Count
4.	Call_Pairs
5.	LOC_Code_and_Comment
6.	LOC_Comments
7.	Condition_Count
8.	Cyclomatic_complexity
9.	Cyclomatic_Density
10.	Decision_Count
11.	Edge_Count
12.	Essential_Complexity
13.	Essential_Density
14.	LOC_Executable
15.	Parameter_Count
16.	Global_Data_Complexity
17.	Global_Data_Density
18.	Halstead_Content
19.	Halstead_Difficulty
20.	Halstead_Effort
21.	Halstead_Error_EST
22.	Halstead_Length
23.	Halstead_Prog_Time
24.	Halstead_Volume
25.	Normalized_Cyclomatic_Complexity
26.	Num_Operands
27.	Num_Operators
28.	Num_Unique_Operands
29.	Num_Unique_Operators
30.	Number_Of_Lines
31.	Pathological_Complexity
32.	LOC_Total

The attribute nature taken from NASA MDP software projects are shown in Table 3.1. Among of NASA MDP datasets, this thesis uses three datasets. They are PC1, CM1 and JM1. The PC1, Flight software for earth orbiting satellite used C language has 1109 modules and 22 attributes. The CM1, NASA spacecraft instrument, also used C language has 402 modules and 22 attributes. The JM1, Real-time predictive ground system, also used C language has 1096 modules and 22 attributes.

These three datasets have 22 attributes that was measured based on metrics of McCabe and Halstead [2]. The McCabe metrics are a collection of four software metrics. They are essential complexity, cyclomatic complexity, design complexity and LOC, Lines of Code. The Halstead falls into three groups such as the base measures, the derived measures, and lines of code measures. The following is explanation of 22 attributes.

1. **loc** (McCabe's line count of code) – it is straightforward to measure line of code because it counts blanks, comments, etc.
2. **v(g)** (McCabe "cyclomatic complexity") – it measures the number of linearly independent paths through flowgraph of a given program. Its formula is v(g) = number of decision statements + 1.
3. **ev(g)** (McCabe "essential complexity") – it is the measure of the unstructuredness of a program. Its formula is **ev(G) = v(G) –** ***m*** where ***m*** is number of one-entry one-exit program's flowgraph.
4. **iv(g)** (McCabe "design complexity") – it is the cyclomatic complexity of a module's reduced flowgraph. According to McCabe, this complexity measurement reflects the modules calling patterns to its immediate subordinate modules.
5. **N** (Halstead total operators + operands) – it sums number of total occurrence operators and operands in program.
6. **V** (Halstead "volume") – it is a count of the number of mental comparisons required to generate a program. Its formula is V=N x $\log_2(n)$ where V is the Volume, N is the number of words in the program and $\log_2(n)$ is the minimum number of bits required to represent all unique words in the program.
7. **L** (Halstead "program length") – it represents a program written at the highest possible level. Its formula is $L = \frac{1}{D}$.

8. **D** (Halstead “difficulty") – it is related to the difficulty of the program to write or understand. Operands and operators that are used repeatedly can tend to increase the Volume and the program Difficulty. Its formula is $D = \frac{n1}{2} \times \frac{N2}{n2}$.
9. **I** (Halstead "intelligence") – The intelligence content is correlated highly with the potential volume. Its formula is $I = L \times V$.
10. **E** (Halstead "effort") – The effort measure translates into actual coding time by selecting each word to be used in the implementation. Its formula is $E = D \times V$.
11. **B** (Halstead “number of delivered bugs") – Halstead's delivered bugs (B) is an estimate for the number of errors in the implementation. Its formula is $B = \frac{V}{3000}$.
12. **T** (Halstead's time estimator) – it is an estimate of the amount of time it took a programmer to write a program. Its formula is $T = \frac{E}{18}$.
13. **lOCode** (Halstead's line count) – it measures line of code in a program.
14. **lOComment** (Halstead's count of lines of comments) – it measures count of comment line in a program.
15. **lOBlank** (Halstead's count of blank lines) – it measures count of blank line in a program.
16. **lOCodeAndComment** – It measures count of code with comment in a program.
17. **uniq_Op** (unique operators) – it counts number of unique operators (not count duplicated) in program.
18. **uniq_Opnd** (unique operands) – it counts number of unique operands (not count duplicated) in program.
19. **total_Op** (total operators) – it counts number of total operators (count duplicated) in program.
20. **total_Opnd** (total operands) – it counts number of total operands (count duplicated) in program.
21. **branchCount** – it counts number of branches for program.

22. **Defects** (false or true) – it is class value attribute that describe non-defect or defect for a program.

Imbalanced Data

Building data mining models with unreliable or abnormal datasets can be a significant challenge to classifier construction [7]. Numerous studies dealing with classification problem shows training dataset's errors presence lower than testing data's predictive accuracy. There are many different dimensions of data quality that included class noise or labeling errors, attribute noise, and missing values. The occurrence of class imbalance is another commonly encountered challenge in data mining applications. Imbalanced data set problem occurs in classification, where the number of instances of one class is much lower than the instances of the other classes.

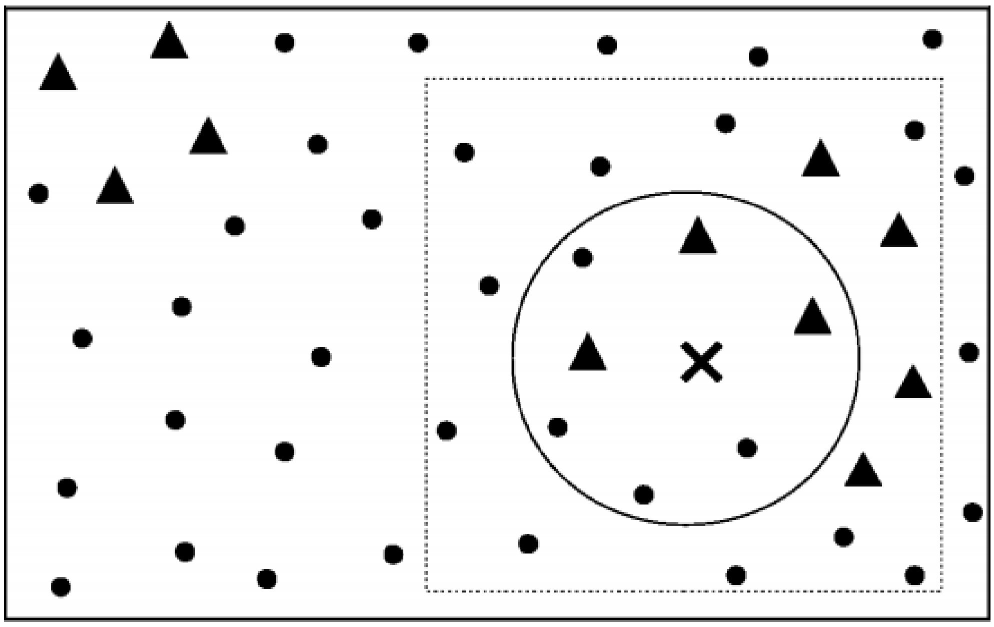


Figure 3.1 Two Classes' Imbalance Solve

Figure 3.1 shows an example of two-class imbalance problem to classify new query instance. In this figure, the majority class "**A**" represented circles and the minority class "**B**" represented triangles. The new data instance is **cross** symbol. The data instance has been classified as the majority class "A" by a regular k-NN algorithm using threshold (k) value 7. But, if the algorithm had taken into account the imbalance class distribution around the neighborhood of the data instances. Finally, the new data instance has been classified as minority class "B", which is the desired class.

3.3 k Nearest Neighbor Classifier

k Nearest Neighbor Classifier (k-NN) is a method for classifying objects based on closest training dataset [10], [17]. The k-NN is an instance-based algorithm for approximating real valued or discrete-valued target functions, assuming instances correspond to points in an n-dimensional Manhattan space. The target function value for a new query is estimated from the known values of the k nearest training examples. When a new query instance is encountered, a set of similar related instances is retrieved from memory and used to classify the new query instance. Instance-based methods can use more complex, symbolic representations for instances. The most basic instance-based method is the k-Nearest-Neighbor algorithm. The nearest neighbors of an instance are defined in terms of the standard Manhattan distance. Then the distance between two instances x_i and x_j is defined to be d (x_i, x_j), where in nearest-neighbor learning the target function may be either discrete-valued or real-valued.

Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it [24]. The training tuples are described by *n* attributes. Each tuple represents a point in an n-dimensional space. In this way, all of the training tuples are stored in an n-dimensional pattern space. When given an unknown tuple, a k-nearest-neighbor classifier searches the pattern space for the k training tuples that are closest to the unknown tuple. These k training tuples are k "nearest neighbors" of unknown tuple.

"Closeness" is defined in terms of a distance metric, such as Manhattan distance [10]. The Manhattan distance between two points or tuples, say, $X_1 = (x_{11}, x_{12}, : : : , x_{1n})$ and $X_2 = (x_{21}, x_{22}, : : : , x_{2n})$, is

$$dist(X_1, X_2) = \sum_{i=1}^{n} |x_{1i} - x_{2i}| \tag{3.1}$$

For each data point in the target dataset, the distance metric between target data and all training data are calculated and sorted. The threshold value (k) has to eliminate all distance values depend on threshold value (k) and taken into account based on classes of selected distance values to classify target data. If the threshold value k = 1, then the 1-Nearest Neighbor Algorithm where x_i is the training instance nearest to x. For larger values of k, the algorithm assigns the most common value among the k nearest training examples. The advantages of k Nearest Neighbor algorithm [4] are

- very fast training
- Simple and easy to learn
- Robust to noisy training data
- easy to implement to classify data point
- Effective if training data is large

Figure 3.2 is k Nearest Neighbor algorithm, is a method for classifying objects based on closest training dataset. For each data in the testing dataset, it calculates distance metrics between testing data and all training data by using Equation 3.1. And, it sorts all distance values and eliminates distance values depend on threshold value (k). Then, it taken into account based on classes of selected distance values to classify testing data.

```
Algorithm 1: k Nearest Neighbor Algorithm
Input: X, C, k, x
Output: class label for testing data x
1. For all distances between testing data and training dataset do
        $dist(X_j, x) = \sum_{i=1}^{n} |X_{ji} - x_i|$
2. End for
3. Sort d ($X_j$, x) by ascending order
4. For iteration <= k-value do
        $D_x^k = d(X_j, x)$
5. iteration++
6. End for
7. Taken into account imbalanced classes in training data
8. Classify testing data by $D_x^k$
9. Output class label of x
```

Figure 3.2 k Nearest Neighbor Algorithm

Table 3.2 reveals the meaning of various symbols used in k Nearest Neighbor algorithm. This table is used to form better understanding of the algorithm.

Table 3.2 List of Symbols Used in the k Nearest Neighbor Algorithm

Symbol	Meaning
X	Training dataset
X_j	Each data of Training dataset
C	Class labels of X
k	Threshold value
x	Each data of testing dataset
d(X, x)	Manhattan distance between one testing data and training dataset
D_x^k	Eliminated Distance values based on *k* value

3.4 Class Based Weighted k-NN

One obvious refinement to the k- Nearest Neighbor Algorithm is to weigh the contribution of each of the k neighbors according to their distance to the query point x, giving greater weight to closer neighbors [12, 24]. The approximates discrete-valued target functions, we might weigh the vote of each neighbor according to the inverse square of its distance from x. Class based Weighted k Nearest Neighbor classifier calculate a weight is assigned to each of the class based on how its instances are classified in the neighborhood of query instance. The only disadvantage of considering all examples is that our classifier can run more slowly. If all training examples are considered when classifying a new query instance, it calls the algorithm a global method

The distance-weighted k-Nearest Neighbor Algorithm is a highly effective inductive inference method for many practical problems [12, 15]. It is robust to noisy training data and quite effective when it is provided a sufficiently large set of training data. By taking the weighted average of the k neighbors nearest to the query point, it can smooth out the impact of isolated noisy training examples. The distance between neighbors can be dominated by the large number of irrelevant attributes that is sometimes referred to as the curse of dimensionality. Nearest-neighbor approaches are especially sensitive to this problem.

Distance value is to weigh each attribute differently when calculating the distance between two instances. This corresponds to stretching the axes in the Manhattan space, shortening the axes that correspond to less relevant attributes, and lengthening the axes that correspond to more relevant attributes. To see how, it is chosen to minimize the true classification error of the learning algorithm. Second, note that this true error can be estimated using cross validation. An algorithm is to select a subset of the available data to use as training that lead to the minimum error in classifying the remaining examples. By repeating this process, the estimate for these weighting factors can be made more accurate. The advantages of Class-Based Weighted k Nearest Neighbor algorithm are -

- Overcomes limitations of k-NN of assigning equal weight to k neighbors implicitly
- Uses all training samples not just k
- Defines the threshold value k (maximum) and (minimum)
- Does not need large memory according to partition of training dataset
- Makes the algorithm global one

Figure 3.3 is algorithm for Class Based Weighed k Nearest Neighbor with BINER algorithm. It finds the nearest range of training dataset where the testing data has the maximum likelihood of occurrence by using BINER algorithm as shown in Figure 3.4. Then, it calculates distance metrics between testing data and nearest range of training data by using Equation 3.1. And, it sorts all distance values and eliminates distance values depend on threshold value (*k*). Then, it calculates weight values for eliminated distances and total weight values for each class. Finally, it compares final weight values to classify testing data.

Algorithm 2: Class Based Weighted k Nearest Neighbor Algorithm

Input: X, C, k, x

Output: class label for testing data x

```
1. Find nearest range by using BINER algorithm
```

```
2. For all distances between testing data and sub range training dataset do
        $dist(X_j, x) = \sum_{i=1}^{n} |X_{ji} - x_i|$
3. End for
4. Sort d ($X_j$, x) by ascending order
5. For iteration <= k-value do
        $D_x^k = d(X_j, x)$
6. iteration++
7. End for
8. For all weight of $D_x^k$ do
        $w_i = \frac{d_k - d_i}{d_k - d_1}$
9. End for
10. Calculate total weight values of each class
11. In class based weighted factor, compute
        $w(c)=1/frequency[c]$
12. Multiply total weight and $w(c)$
13. Compare final total weight
14. Output class label of x
```

Figure 3.3 Class Based Weighted k Nearest Neighbor Algorithm

Table 3.3 reveals the meaning of various symbols used to form better understanding of Class Based Weighted k Nearest Neighbor.

Table 3.3 List of Symbols Used in the CBW k-NN Algorithm

Symbol	Meaning
k	Threshold value
x	Each data of testing dataset
X	Training dataset

X_j	Each data of Training dataset
C	Class labels of X
d(X, x)	Manhattan distance between one testing data and training dataset
D_x^k	Eliminated distance values based on *k* value
w_i	Weight value of eliminated distances
w(c)	class based weighted factor for each class

3.5 BINER Algorithm

Harshit Dubey proposed BINary search based Efficient Regression (BINER) which is a new efficient technique for regression [8].

BINER follows the same overall methodology as k-NN. Firstly, it finds the k nearest neighbors to the given query. Then, weighted mean of response variables in k nearest neighbors is given as output. The weights are kept inversely proportional to distance from the query. The intuition of BINER is that the query Q is expected to be similar to tuples whose response variable values are close to that of Q. Thus it is more beneficial to find nearest neighbors in a locality where tuple have nearby response variable values rather than the whole dataset. This guarantees that even if the tuples in the considered locality are not the global nearest neighbors (nearest neighbors of the query in the complete dataset), the value of predicted response variable can be more appropriate.

Like other k-NN based approaches, BINER has the following core assumption - tuples with similar X-values have similar response variable values. This assumption is almost always borne out in practice and is justified also by experiments. Instead of directly predicting the value of response variable, BINER narrows down the range in which the response variable has the maximum likelihood of occurrence and then interpolates to give the output. The data is hierarchically partitioned in the preprocessing step, and search for the partition in which the response has the maximum likelihood of occurrence is carried out at the runtime. It takes a single parameter k, the same as in k-NN and more than often outperforms the conventional state of art methods on a wide variety of datasets as illustrated by our experimental study.

The algorithm proceeds in two steps. First, it finds the range of tuples where the query Q has the maximum likelihood of occurrence. The term range (or locality), here, refers to consecutively indexed tuples in the dataset D and thus is characterized by two integers namely, start index and end index. Second, k-NN is applied to these few (compared to D) tuples and weighted mean of the K nearest neighbors in these ranges is quoted as output. To find the range in which the query has the maximum likelihood of occurrence, the dataset is sorted in, say, non-decreasing manner of response variable values and then the function BINER described below is invoked with Q as query, and range (0, n) where n is the number of tuples in D.

Figure 3.4 is BINER Algorithm to search nearest range of training dataset in Class Based Weighed k Nearest Neighbor method. Firstly, the training dataset is sorted in increasing order based on maximum value attribute and then the algorithm BINER is invoked with x as testing data, and range (0, n) where n is the number of records in Training dataset. The *getDistance( )* of testing data *x* from a range is calculated as shown in Equation 3.2.

$$getDistance(\,) = \sqrt{\sum \frac{(\mu_i - q_i)^2}{\sigma_i^2}} \tag{3.2}$$

$$\sigma_i^2 = \frac{1}{N}\left[\sum x_i^2 - \frac{1}{N}\left(\sum x_i\right)^2\right] \tag{3.3}$$

In Equation 3.2, q_i is the i^{th} attribute of the testing data, μ_i is the mean of i^{th} attribute values in all records in the range and σ_i is the standard deviation of values of the i^{th} attribute in the whole training dataset. In Equation 3.3, standard deviation equation, x_i is value of each attribute and N is the number of records in training dataset. *RangeMean*(s_i,e_i) is the mean values of each range of training dataset and *i* is number of range. The distance values of three ranges are compared by using *similar( )* function. The two distances, d_i and d_j are similar if *min*(d_i/d_j, d_j/d_i) is greater than *0.95* (selected by experimentations and it works well on most of the datasets). If the two smaller distances are similar, it returns the current range instead of selecting a sub range. If the two larger distances are similar or the three distances are not similar, it chooses the sub range of smallest distance. Then, that sub range calculate with BINER function again.

```
Algorithm 3: BINER Algorithm
Input: x, k, Range (start, end)
Output: Range (s, e)
while end – start > 2 * k do
r =end – start
s1 = start
e1 = start + r/2
$s_2$ = start +r/4
$e_2$ = start + 3r/4
$s_3$ = start + r/2
$e_3$ = end
$d_1$ = getDistance (RangeMean($s_1$, $e_1$), x)
$d_2$ = getDistance (RangeMean($s_2$, $e_2$), x)
$d_3$ = getDistance (RangeMean($s_3$, $e_3$), x)
if similar ($d_1$ , $d_2$, $d_3$ ) then
        return Range(start, end)
        break
else
        start = $s_i$
        end = $e_i$
end if
end while
return Range(start, end)
```

Figure 3.4 BINER Algorithm

Table 3.4 presents the various symbols used in BINER algorithm to form better understanding.

Table 3.4 List of Symbols Used in the BINER Algorithm

Symbol	**Meaning**
Range (start, end)	Whole Training dataset range
k	Threshold value
x	Each data of testing dataset
s_1, e_1	First range of three sub ranges
d_1	Distance value between testing data and first range
RangeMean(s_1,e_1)	Mean values for first range of training dataset
s_i, e_i	Selected sub range to calculate next sub ranges

BINER Algorithm Complexity

The algorithm divides the current range into 3 sub-ranges that each sub-range has half size of current range and considers one of them for subsequent processing. It can be observed that the function iterates O (log n) time. The function returns a range of size, say, R which is significantly smaller than n as confirmed by experimentations. Thus computational complexity of the algorithm becomes O (log n + R) and when the R << n it becomes logarithmic.

CHAPTER 4
SYSTEM DESIGN AND IMPLEMENTATION

This chapter presents overview design of the system, data pre-processing, problem formulation for k-NN and CBW k-NN, user interface design and experimental results of the system. Firstly, this chapter explains about system overview design with figure and detailed explanation. Secondly, it describes data pre-processing procedure for imbalanced datasets and clean datasets. Thirdly, it calculates problem formulation for k-NN and CBW k-NN to understand two methods clearly. Finally, it presents user interface design of the system with step by step detailed explanation figures and experimental results of the system.

4.1 Overview Design of the System

In overview design of the system (Figure 4.1), the main process is to classify target data that are defective or non-defective and compare two methods according to accuracy and reliability. Firstly, user input dataset file such as PC1, CM1 and JM1 and then the system checks dataset file and removes unnecessary data as data pre-processing algorithm. Then, the system stores clean data to database and selects data to divide training set and testing set. User input training and testing ratio and then the system divides training dataset and testing dataset by ratio value. And user input and the system checks k value to eliminate k nearest instances in training dataset for each testing data. User can choose two methods such as k-NN and Class Based Weighted k-NN with BINER Algorithm for classification to calculate step by step processes. The system displays to user classification results (defective or non-defective) according to chosen method (k-NN or Class Based Weighted k-NN with BINER Algorithm). Then, the system also compares classification results by calculating accuracy, reliability, and error rate methods (MAE and RMSE) for two methods. Finally, the system displays comparison results of two methods as accuracy, precision, recall, reliability, MAE and RMSE.

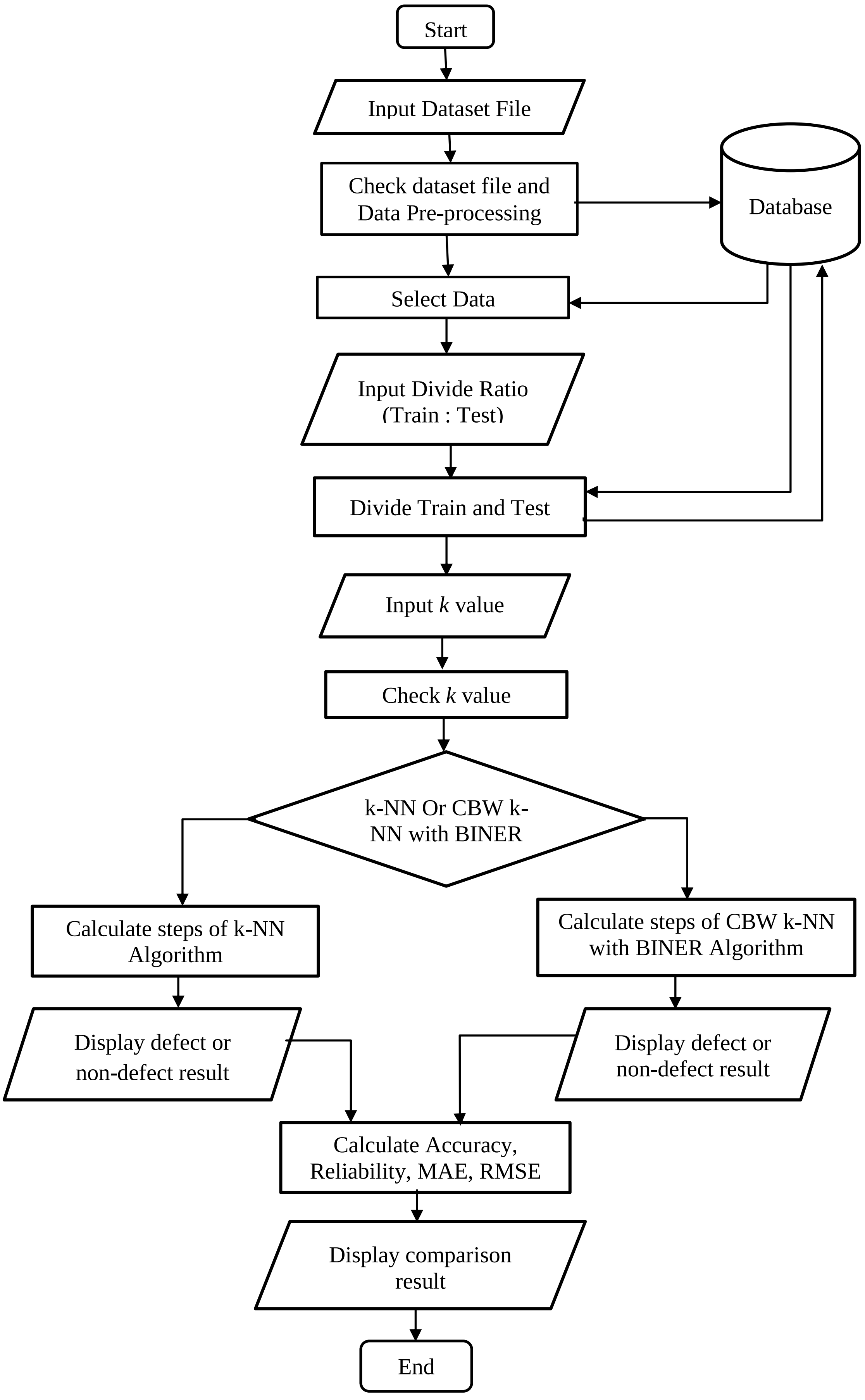

Start
Input Dataset File
Check dataset file and Data Pre-processing
Database
Select Data
Input Divide Ratio (Train : Test)
Divide Train and Test
Input *k* value
Check *k* value
k-NN Or CBW k-NN with BINER
Calculate steps of k-NN Algorithm
Calculate steps of CBW k-NN with BINER Algorithm
Display defect or non-defect result
Display defect or non-defect result
Calculate Accuracy, Reliability, MAE, RMSE
Display comparison result
End

Figure 4.1: Overview Design of the System

4.2 Data Pre-processing

For k Nearest Neighbor (k-NN) and Class-Based Weighted k-NN (CBW k-NN) methods, data preparation and cleaning procedure is used by software defect datasets. It checks imbalanced dataset file, removes unnecessary text in input data file and removes duplicated records. The detail process of data pre-processing is shown in Figure 4.2.

```
Algorithm: Data Preparation and Cleaning Algorithm
Input: upload dataset file
Upload dataset file
Read all instances in given file
Check the dataset file for imbalanced instances
For iteration <= length_of_file do
    Read each record of dataset
    Save each record to Array
    iteration++
End For
Check record in Array for duplication
Remove unnecessary text
Remove same record in Array
Save remaining record to database
```

Figure 4.2 Data Pre-processing Algorithm

4.3 Problem Formulation for k-NN

The k-NN, k Nearest Neighbor classifier, is the classification method to classify testing data based on training data. First, it needs to divide training and testing dataset by ratio value. Second, it calculates the distance value for each testing data point based on all training data points. Third, it eliminates calculated distance values according to k value. Finally, it classifies class value for each testing data point based on class value of eliminated training data points.

4.3.1 Dividing Dataset

In PC1 dataset, it is used to create flight software for earth orbiting satellite and total instances are 932 records. In detail calculation processes, it uses totally 30 instances for PC1 dataset. It divides training dataset and testing dataset by ratio 2:1 as number of training instances is 20 and number of testing instances is 10 are shown in Table 4.1 and 4.2.

Table 4.1 Training Dataset

No.	Instances	Class
1	16,4,1,3,59,283.63,0.07,13.5,21.01,3829.06,0.09,212.73,16,3,0,0,14,14,32,27,7	false
2	16,3,1,3,60,300,0.11,9.33,32.14,2800,0.1,155.56,16,0,0,8,14,18,36,24,5	true
3	15,4,3,2,86,457.69,0.07,14.73,31.08,6740.46,0.15,374.47,14,6,1,8,18,22,50,36,7	true
4	16,2,1,2,73,347.11,0.14,7.37,47.11,2557.63,0.12,142.09,16,0,0,8,8,19,38,35,3	false
5	14,4,1,2,69,353.92,0.09,10.88,32.54,3848.89,0.12,213.83,12,7,2,4,15,20,40,29,7	true
	………………………	
18	14,5,1,2,66,317.29,0.05,20,15.86,6345.71,0.11,352.54,12,4,2,6,16,12,36,30,9	false
19	19,2,1,2,71,351.75,0.08,12.66,27.79,4451.81,0.12,247.32,18,25,1,9,15,16,44,27,3	true
20	23,4,1,4,101,548.05,0.08,11.79,46.5,6459.19,0.18,358.84,23,0,0,12,15,28,57,44,7	true

Table 4.2 Testing Dataset

No.	Instances	Class
1	17,1,1,1,60,285.29,0.1,10.21,27.93,2914.07,0.1,161.89,17,18,0,11,13,14,38,22,1	false
2	16,1,1,1,54,244.27,0.11,9.17,26.65,2239.16,0.08,124.4,16,15,0,12,11,12,34,20,1	false
3	23,4,1,4,101,544.62,0.08,12.22,44.56,6656.52,0.18,369.81,23,0,0,12,15,27,57,44,7	true
4	29,2,1,2,140,718.1,0.1,9.93,72.35,7127.8,0.24,395.99,28,3,1,8,8,27,73,67,3	true
5	17,4,1,3,63,284.98,0.09,11.15,25.55,3178.67,0.09,176.59,17,0,0,0,10,13,34,29,7	false
6	8,3,1,3,48,214.05,0.08,12,17.84,2568.63,0.07,142.7,8,0,0,0,11,11,24,24,5	false
7	16,1,1,1,54,244.27,0.12,8.08,30.24,1972.97,0.08,109.61,16,12,0,11,10,13,33,21,1	false
8	28,6,1,5,86,477.69,0.07,13.73,34.79,6559.11,0.16,364.4,28,0,0,7,21,26,52,34,8	true
9	9,1,1,1,31,114.71,0.05,21.67,5.29,2485.46,0.04,138.08,9,0,0,2,10,3,18,13,1	false
10	12,3,1,3,49,230.32,0.09,11.67,19.74,2687.08,0.08,149.28,12,1,0,7,14,12,29,20,5	false

4.3.2 Calculation Distances

$$dist(X_j, x) = \sum_{i=1}^{n} |X_{ji} - x_i|$$

d(x_1, x)= |16-17| + 4-1| + |1-1| + ………. + |27-22|+ |7-1| = 1029.74

Table 4.3 is all distance values for first instance of testing and the whole training dataset.

Table 4.3 k-NN Distance Values Calculation

Instance No.	Distance Value
1	1029.74
2	180.21
3	4317.12
	…………………..
19	1735.26
20	4166.08

4.3.3 Sorting Distances

By ordering the instances according to the distance value, it is sorted distance values are shown in Table 4.4.

Table 4.4 Sorted Distance Values

Instance No.	Distance Value
2	180.21
13	478.23
4	523.14
9	586.39
	…………………..
20	4166.08
3	4317.12

4.3.4 Elimination the k Nearest Instances to x

In this PC1 dataset, total instance is 30 and train and test are divided by ratio 2:1 as number of training instances is 20 and number of testing instances is 10. Now, it is result for first instance of testing and eliminates distance value by *k* value **6**. The eliminated distance values are shown in Table 4.5.

4.3.5 Taking into Account the Result

In training dataset, number of class **true** values is **7** and number of class false values is **13**, so training's true and false ratio is 1 to 2. In class values of eliminated distances, number of class **true** values is **1** and number of class **false** values is **5** by *k* value **6** is shown in Table 4.5.

Table 4.5 Eliminated Distances and Class Values

Instance No.	Distance Value	class
2	180.21	true
13	478.23	false
4	523.14	false
9	586.39	false
15	729.62	false
12	842.05	false

Therefore, class label of first instance of testing dataset is '**True'** by using true and false ratio of training dataset (1:2). Then, all remaining testing instances calculate class value as the above calculation processes.

4.4 Problem Formulation for CBW k-NN

The CBW k-NN with BINER Algorithm, Class Based Weighted k-NN classifier, is the classification method to classify testing dataset based on calculated weight values of k Nearest Neighbor distances. First, BINER algorithm divides three ranges of training dataset. Second, BINER algorithm calculates mean value of each range and deviation value and then calculates three distance values for each range. Third, it calculates similar values based on three ranges of training dataset to find nearest training dataset range of each testing data. Finally, it calculates weight values and classifies class value for weighted distance values based on k Nearest Neightbor distances.

4.4.1 Dividing and Sorting Training Dataset

In detail calculation processes, it uses totally 30 instances for PC1 dataset. It divides training dataset and testing dataset by ratio 2:1 as number of training instances is 20 and number of testing instances is 10. Firstly, it is sort training dataset by ascending order to divide three ranges of training dataset by BINER algorithm. The sorted training dataset and testing dataset are shown in Table 4.6 and 4.7.

Table 4.6 Sorted Training Dataset

No.	Instances	Class
1	15,3,1,3,48,228.23,0.16,6.18,36.95,**1409.68**,0.08,78.32,15,15,0,7,10,17,27,21,5	false
2	10,3,1,1,31,142.13,0.09,10.83,13.12,**1539.78**,0.05,85.54,10,0,0,2,15,9,18,13,5	false
3	20,2,1,2,51,245.18,0.13,7.44,32.95,**1824.39**,0.08,101.36,20,0,0,2,11,17,28,23,3	false
	……………………….	
19	23,4,1,4,101,548.05,0.08,11.79,46.5,**6459.19**,0.18,358.84,23,0,0,12,15,28,57,44,7	true
20	15,4,3,2,86,457.69,0.07,14.73,31.08,**6740.46**,0.15,374.47,14,6,1,8,18,22,50,36,7	true

Table 4.7 Testing Dataset

No.	Instances	Class
1	17,1,1,1,60,285.29,0.1,10.21,27.93,2914.07,0.1,161.89,17,18,0,11,13,14,38,22,1	false
2	16,1,1,1,54,244.27,0.11,9.17,26.65,2239.16,0.08,124.4,16,15,0,12,11,12,34,20,1	false
3	23,4,1,4,101,544.62,0.08,12.22,44.56,6656.52,0.18,369.81,23,0,0,12,15,27,57,44,7	true
4	29,2,1,2,140,718.1,0.1,9.93,72.35,7127.8,0.24,395.99,28,3,1,8,8,27,73,67,3	true
5	17,4,1,3,63,284.98,0.09,11.15,25.55,3178.67,0.09,176.59,17,0,0,0,10,13,34,29,7	false
6	8,3,1,3,48,214.05,0.08,12,17.84,2568.63,0.07,142.7,8,0,0,0,11,11,24,24,5	false
7	16,1,1,1,54,244.27,0.12,8.08,30.24,1972.97,0.08,109.61,16,12,0,11,10,13,33,21,1	false
8	28,6,1,5,86,477.69,0.07,13.73,34.79,6559.11,0.16,364.4,28,0,0,7,21,26,52,34,8	true
9	9,1,1,1,31,114.71,0.05,21.67,5.29,2485.46,0.04,138.08,9,0,0,2,10,3,18,13,1	false
10	12,3,1,3,49,230.32,0.09,11.67,19.74,2687.08,0.08,149.28,12,1,0,7,14,12,29,20,5	false

4.4.2 Calculation Sub Range by BINER Function

Firstly, it is divided into three ranges of whole dataset such as Range 1(0, 10), Range 2(5, 15), and Range 3 (10, 20).

Table 4.8 Mean and Deviation Values

Attribute	Mean 1	Mean 2	Mean 3	Deviation
1	15.2	15.8	17.1	9.2275
2	2.6	3.3	4.3	1.6475
….	………..	….	………..	….
20	23.7	28.7	32.2	73.1475
21	4.2	5.6	7.6	6.59

And then it calculates sub range for each instance of testing dataset by using mean and deviation values are shown in Table 4.8. In distance equation, μ_i is mean values of i th attribute of sub range and σ_i^2 is standard deviation of i th attribute of training dataset.

$$d = \sqrt{\sum \frac{(\mu_i - q_i)^2}{\sigma_i^2}}$$

$$d_1 = \sqrt{\frac{(15.2-17)^2}{9.2275} + \frac{(2.6-1)^2}{1.6475} + \frac{(1-1)^2}{0.19} + ...+ \frac{(23.7-22)^2}{73.1475} + \frac{(4.2-1)^2}{6.59}}$$

$$d_1 = 4.0169$$

$$d_2 = \sqrt{\frac{(15.8-17)^2}{9.2275} + \frac{(3.3-1)^2}{1.6475} + \frac{(1-1)^2}{0.19} + ...+ \frac{(28.7-22)^2}{73.1475} + \frac{(5.6-1)^2}{6.59}}$$

$$d_2 = 4.2021$$

$$d_3 = \sqrt{\frac{(17.1-17)^2}{9.2275} + \frac{(4.3-1)^2}{1.6475} + \frac{(1.2-1)^2}{0.19} + ...+ \frac{(32.2-22)^2}{73.1475} + \frac{(7.6-1)^2}{6.59}}$$

$$d_3 = 5.7577$$

These distance values are used to calculate similar values for using equation of *min (d_i/d_j, d_j/d_i)* function.

$\min(d_1/d_2, d_2/d_1) = \min(0.9559, 1.0461) = \mathbf{0.9559}$

$\min(d_1/d_3, d_3/d_1) = \min(0.6977, 1.4334) = \mathbf{0.6977}$

$\min(d_2/d_3, d_3/d_2) = \min(0.7298, 1.3702) = \mathbf{0.7298}$

And then it compares three similar values to choose nearest range by using BINER function. In three similar values, d_1 and d_2 are similar because similar value 0.9559 is greater than 0.95. If the two smaller distances are similar, it chooses the current range instead of selecting a sub range as final range (0, 20). Therefore, training instances range is 1 to 20 for first instance of testing dataset.

4.4.3 Calculation $d(x_i, x)$ for Result Sub Range

Table 4.9 is distance values for first instance of testing dataset with the selected range of training dataset.

Table 4.9 Calculated Distance Values

Instance No.	Distance Value
1	1029.74
2	180.21
3	4317.12
	…………………..
19	1735.26
20	4166.08

4.4.4 Ordering $d(x_i, x)$ and Elimination the *k* Nearest Instances to *x*

By ordering the instances according to the distance value, it is sorted distance values and eliminated distances by k value **6** are shown in Table 4.10.

Table 4.10 Sorted and Eliminated Distances and Class Values

Instance No.	Distance Value	class
2	180.21	true
13	478.23	false
4	523.14	false
9	586.39	false
15	729.62	false
12	842.05	false

4.4.5 Calculation Class Based Weighted k-NN d(x_j, x)

Then, it calculates each weight value of eliminated distance values in above Table 4.10.

$$w_i = \frac{d_k - d_i}{d_k - d_1}$$ For class T=1, class F=2

$$w_{11} = \frac{842.05 - 180.21}{842.05 - 180.21} = 1$$

$$w_{22} = \frac{842.05 - 478.23}{842.05 - 180.21} = 0.5497$$

....

....

$$w_{26} = \frac{842.05 - 842.05}{842.05 - 180.21} = 0$$

4.4.6 Calculation for Total Weight Value of Each Class

It calculates total weight value of each class as w_1 is 'True' class and w_2 is 'False' class.

For class True, $w_1 = 1$

For class False, $w_2 = 1.5878$

4.4.7 Calculation for Class Based Weighted Factor

It calculates class based weighted factor of each class for unbalance dataset.

For class True, $w(c_T) = 1/7 = 0.1429$

For class False, $w(c_F) = 1/13 = 0.0769$

4.4.8 Calculation for Final Weight and Class Value

It calculates final weight values of each class and classifies class result according to final weights.

For class True, $total\_w_1 = 1 * 0.1429 \quad = 0.1429$

For class False, $total\_w_2 = 1.5878 * 0.076 \quad = 0.1221$

$total\_w_1 > total\_w_2$

Therefore, class label of first instance of testing dataset is **true** by comparing total weight values for each class. Then, the remaining testing instances calculate class values by using BINER function as the above calculation processes.

4.5 Implementation of the System

The system is implemented with the PHP and MYSQL as shown in Figure 4.3. In this page, user can view the four options in the navigation bar. User can click the desired link to use the system. In the navigation, it has home, file open, process and about system.

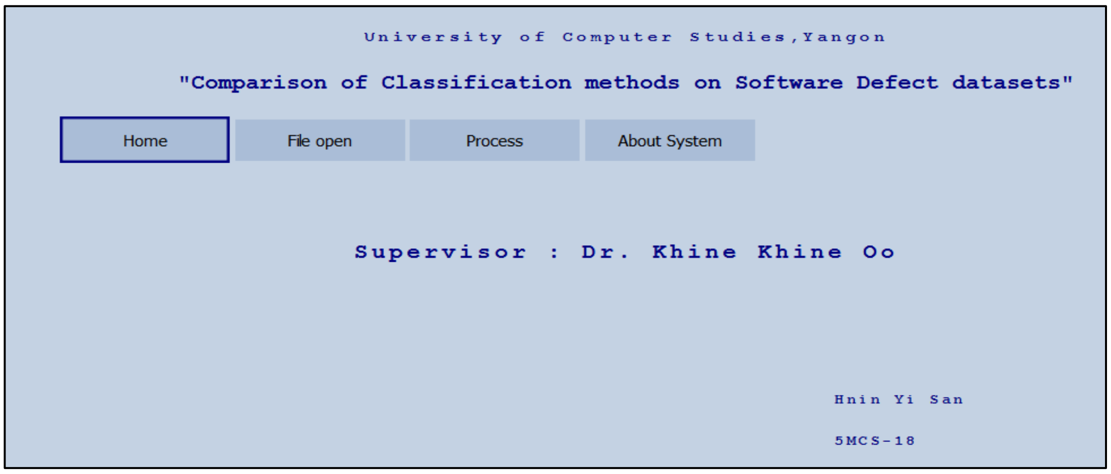
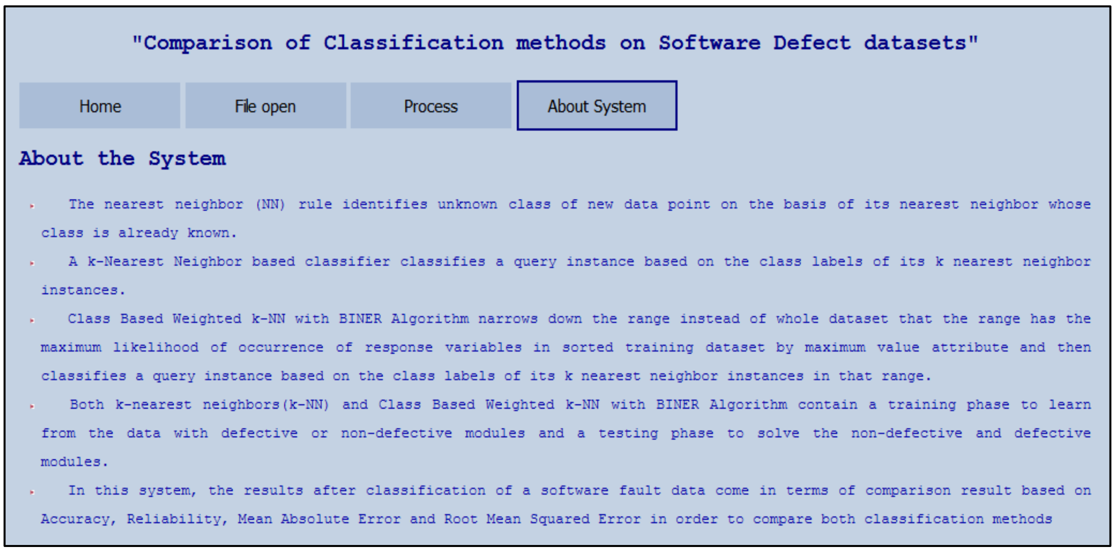

University of Computer Studies,Yangon
"Comparison of Classification methods on Software Defect datasets"
Home
File open
Process
About System
Supervisor : Dr. Khine Khine Oo
Hnin Yi San
5MCS-18

Figure 4.3 Home Page for the System

Figure 4.4 represents about system page of the system. In this page, user can view about the whole system.

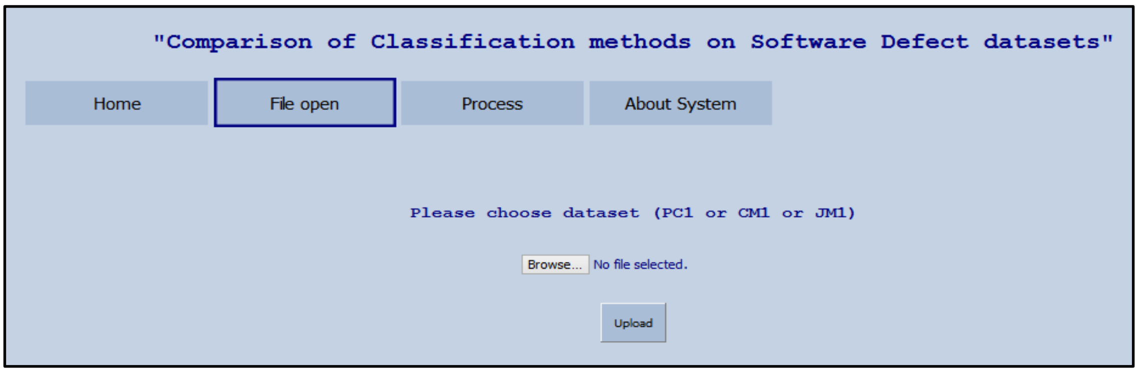

"Comparison of Classification methods on Software Defect datasets"
Home
File open
Process
About System

About the System

- The nearest neighbor (NN) rule identifies unknown class of new data point on the basis of its nearest neighbor whose class is already known.
- A k-Nearest Neighbor based classifier classifies a query instance based on the class labels of its k nearest neighbor instances.
- Class Based Weighted k-NN with BINER Algorithm narrows down the range instead of whole dataset that the range has the maximum likelihood of occurrence of response variables in sorted training dataset by maximum value attribute and then classifies a query instance based on the class labels of its k nearest neighbor instances in that range.
- Both k-nearest neighbors(k-NN) and Class Based Weighted k-NN with BINER Algorithm contain a training phase to learn from the data with defective or non-defective modules and a testing phase to solve the non-defective and defective modules.
- In this system, the results after classification of a software fault data come in terms of comparison result based on Accuracy, Reliability, Mean Absolute Error and Root Mean Squared Error in order to compare both classification methods

Figure 4.4 About Page for the System

Figure 4.5 represents the file open page of the system. In this page, user can upload three software defect datasets such as PC1, CM1 and JM1. The input file must be **.txt** format only. After choosing the input file, user can click the Upload button.

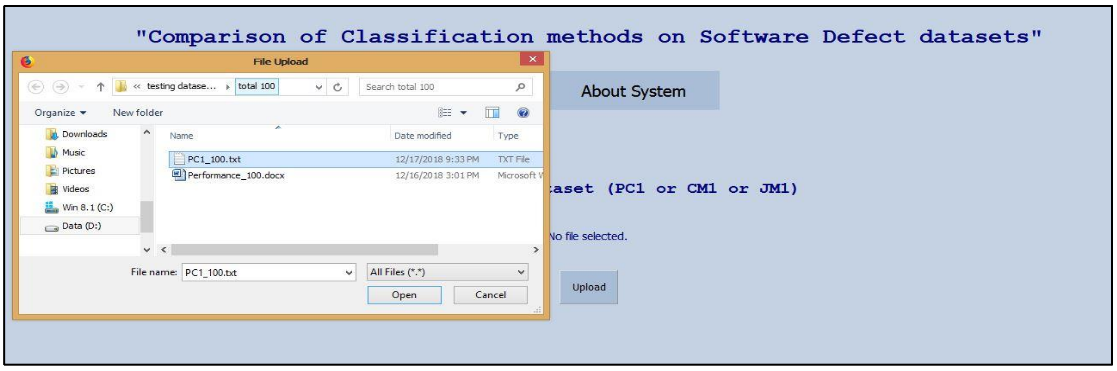

"Comparison of Classification methods on Software Defect datasets"

Please choose dataset (PC1 or CM1 or JM1)

No file selected.

Figure 4.5 File Open Page for the System

Figure 4.6 represents the selecting files of the system. In this page, user can select the data input file with only **.txt** format and then click upload button to check file format. By selecting the files of the software defect datasets, system checks file format and dataset is imbalanced dataset.


"Comparison of Classification methods on Software Defect datasets"

File Upload

« testing datase... › total 100

Name	Date modified	Type
PC1_100.txt	12/17/2018 9:33 PM	TXT File
Performance_100.docx	12/16/2018 3:01 PM	Microsoft W

File name: PC1_100.txt

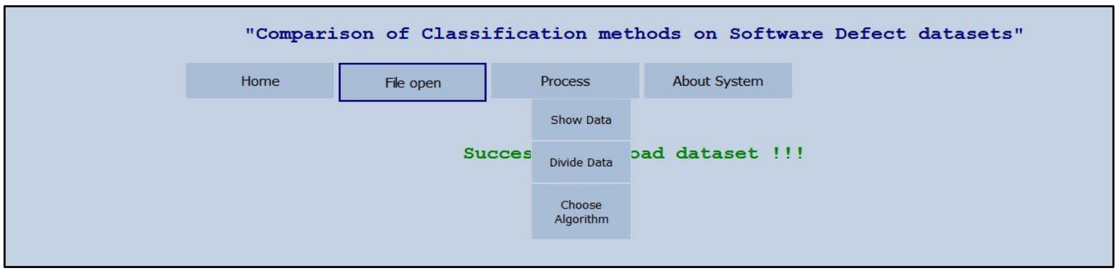
All Files (*.*)

aset (PC1 or CM1 or JM1)

No file selected.

Figure 4.6 Select File for the System

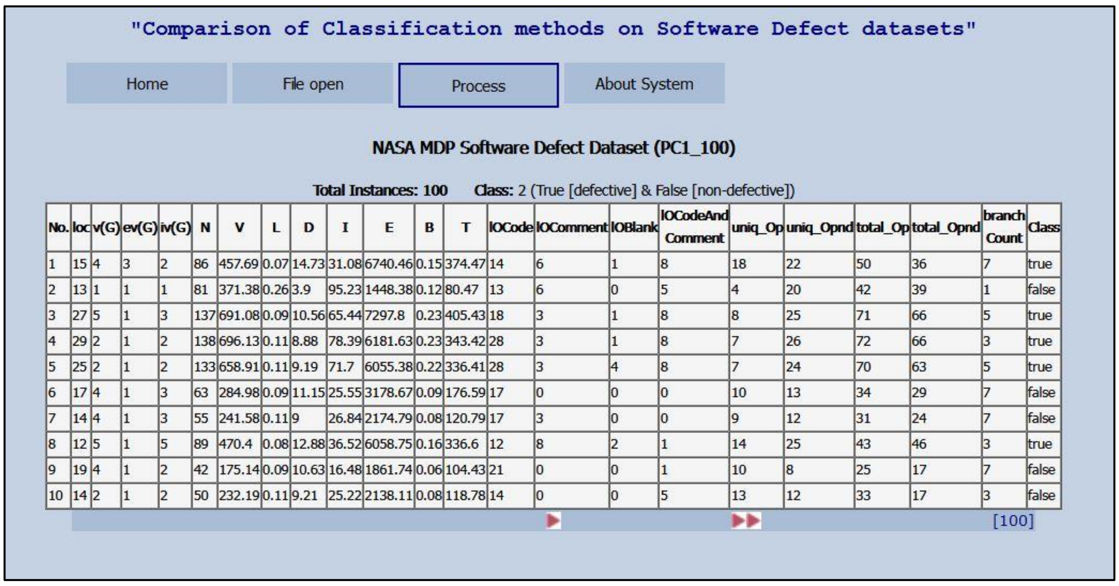
Figure 4.7 represents the process page of the system and the successful massage for uploading dataset file. In this page, user can view sub menus of process menu such as show data, divide data, and choose algorithm to choose sub menus.


"Comparison of Classification methods on Software Defect datasets"

Succes oad dataset !!!

Figure 4.7 Upload File Success for the System

Figure 4.8 represents the all data by uploading dataset. In this page, user can view all data of the uploaded dataset before dividing training and testing with table format and the remaining data by clicking next button and last button. User can also check total instances and dataset name of uploaded file.


"Comparison of Classification methods on Software Defect datasets"
Home
File open
Process
About System

NASA MDP Software Defect Dataset (PC1_100)

Total Instances: 100 **Class:** 2 (True [defective] & False [non-defective])

No.	loc	v(G)	ev(G)	iv(G)	N	V	L	D	I	E	B	T	lOCode	lOComment	lOBlank	lOCodeAnd Comment	uniq_Op	uniq_Opnd	total_Op	total_Opnd	branch Count	Class
1	15	4	3	2	86	457.69	0.07	14.73	31.08	6740.46	0.15	374.47	14	6	1	8	18	22	50	36	7	true
2	13	1	1	1	81	371.38	0.26	3.9	95.23	1448.38	0.12	80.47	13	6	0	5	4	20	42	39	1	false
3	27	5	1	3	137	691.08	0.09	10.56	65.44	7297.8	0.23	405.43	18	3	1	8	8	25	71	66	5	true
4	29	2	1	2	138	696.13	0.11	8.88	78.39	6181.63	0.23	343.42	28	3	1	8	7	26	72	66	3	true
5	25	2	1	2	133	658.91	0.11	9.19	71.7	6055.38	0.22	336.41	28	3	4	8	7	24	70	63	5	true
6	17	4	1	3	63	284.98	0.09	11.15	25.55	3178.67	0.09	176.59	17	0	0	0	10	13	34	29	7	false
7	14	4	1	3	55	241.58	0.11	9	26.84	2174.79	0.08	120.79	17	3	0	0	9	12	31	24	7	false
8	12	5	1	5	89	470.4	0.08	12.88	36.52	6058.75	0.16	336.6	12	8	2	1	14	25	43	46	3	true
9	19	4	1	2	42	175.14	0.09	10.63	16.48	1861.74	0.06	104.43	21	0	0	1	10	8	25	17	7	false
10	14	2	1	2	50	232.19	0.11	9.21	25.22	2138.11	0.08	118.78	14	0	0	5	13	12	33	17	3	false

[100]

Figure 4.8 Show All Data for the System

Figure 4.9 represents dividing data for training and testing set. In this page, user can divide training set and testing set for the all data of the dataset. If user clicks skip button, the system divides automatically **2:1** ratio for training and testing.

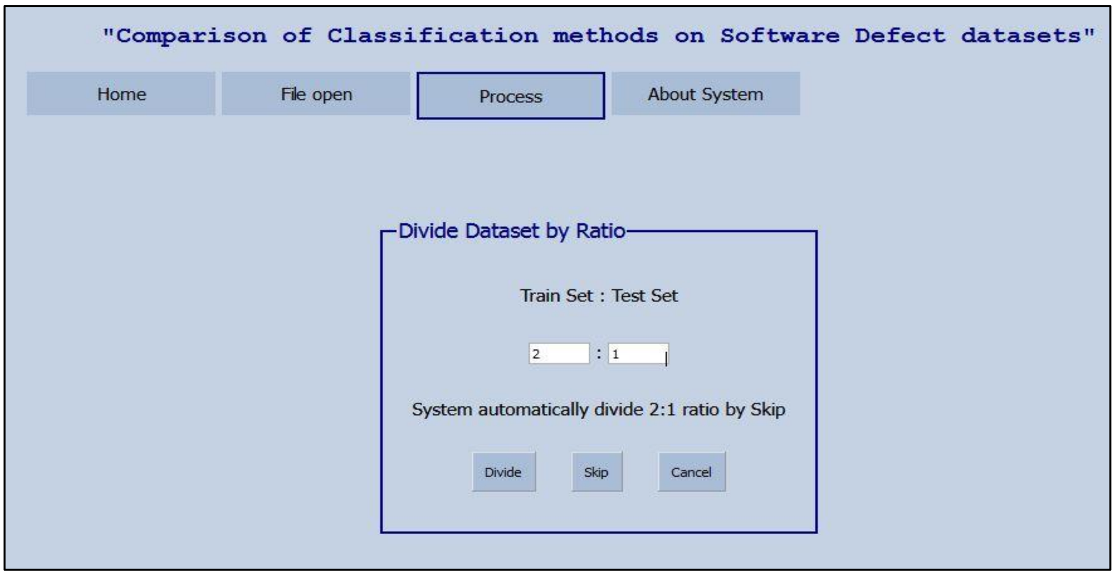
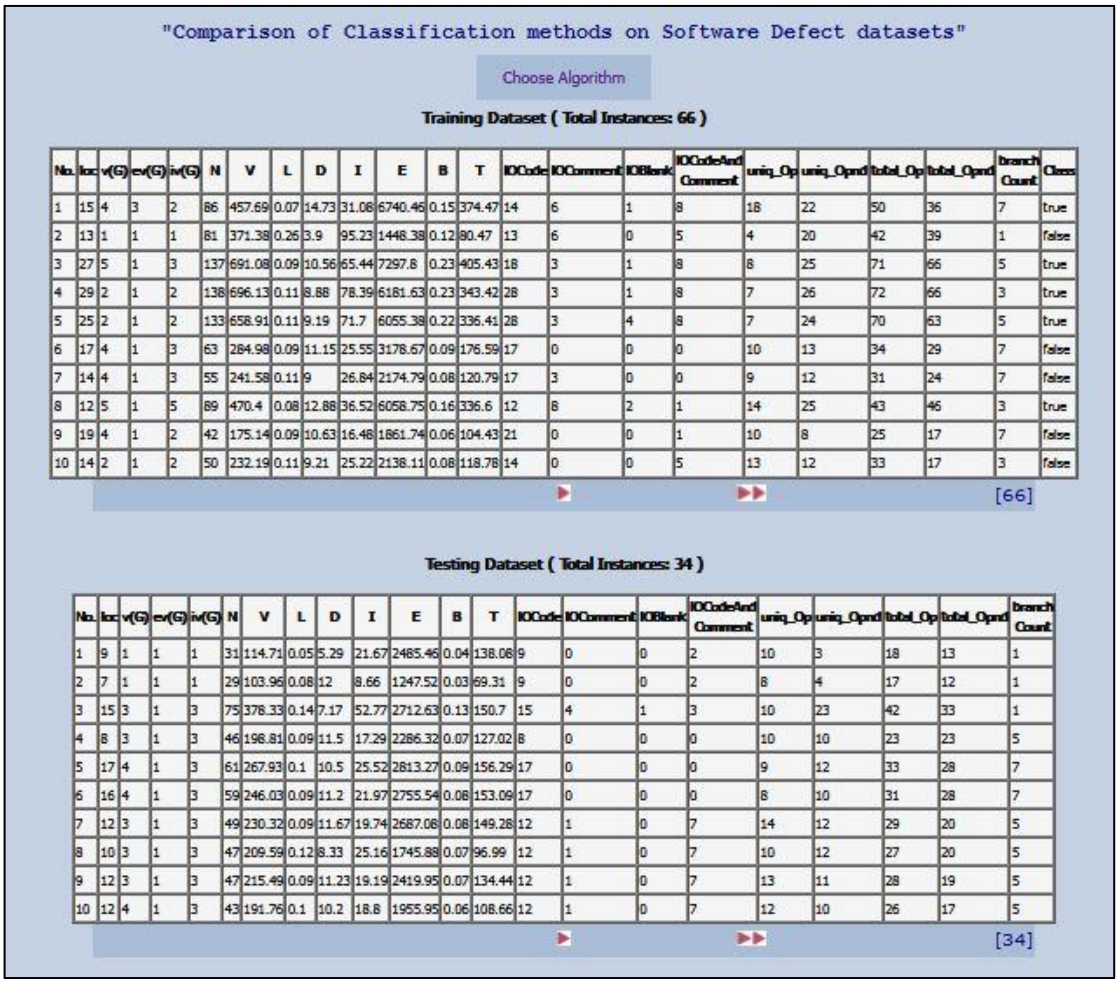

"Comparison of Classification methods on Software Defect datasets"
Home
File open
Process
About System
Divide Dataset by Ratio
Train Set : Test Set
2 : 1
System automatically divide 2:1 ratio by Skip
Divide
Skip
Cancel

Figure 4.9 Division Data for the System

Figure 4.10 represents the divided data information of the system. For example, the whole dataset instances is **100**, user divide **2:1** ratio, training dataset is

66 instances and testing dataset is **34** instances. Then, user clicks choose algorithm button to select k-NN algorithm or CBW k-NN algorithm.



"Comparison of Classification methods on Software Defect datasets"

Choose Algorithm

Training Dataset (Total Instances: 66)

No.	loc	v(G)	ev(G)	iv(G)	N	V	L	D	I	E	B	T	lOCode	lOComment	lOBlank	lOCodeAndComment	uniq_Op	uniq_Opnd	total_Op	total_Opnd	branchCount	Class
1	15	4	3	2	86	457.69	0.07	14.73	31.08	6740.46	0.15	374.47	14	6	1	8	18	22	50	36	7	true
2	13	1	1	1	81	371.38	0.26	3.9	95.23	1448.38	0.12	80.47	13	6	0	5	4	20	42	39	1	false
3	27	5	1	3	137	691.08	0.09	10.56	65.44	7297.8	0.23	405.43	18	3	1	8	8	25	71	66	5	true
4	29	2	1	2	138	696.13	0.11	8.88	78.39	6181.63	0.23	343.42	28	3	1	8	7	26	72	66	3	true
5	25	2	1	2	133	658.91	0.11	9.19	71.7	6055.38	0.22	336.41	28	3	4	8	7	24	70	63	5	true
6	17	4	1	3	63	284.98	0.09	11.15	25.55	3178.67	0.09	176.59	17	0	0	0	10	13	34	29	7	false
7	14	4	1	3	55	241.58	0.11	9	26.84	2174.79	0.08	120.79	17	3	0	0	9	12	31	24	7	false
8	12	5	1	5	89	470.4	0.08	12.88	36.52	6058.75	0.16	336.6	12	8	2	1	14	25	43	46	3	true
9	19	4	1	2	42	175.14	0.09	10.63	16.48	1861.74	0.06	104.43	21	0	0	1	10	8	25	17	7	false
10	14	2	1	2	50	232.19	0.11	9.21	25.22	2138.11	0.08	118.78	14	0	0	5	13	12	33	17	3	false

[66]

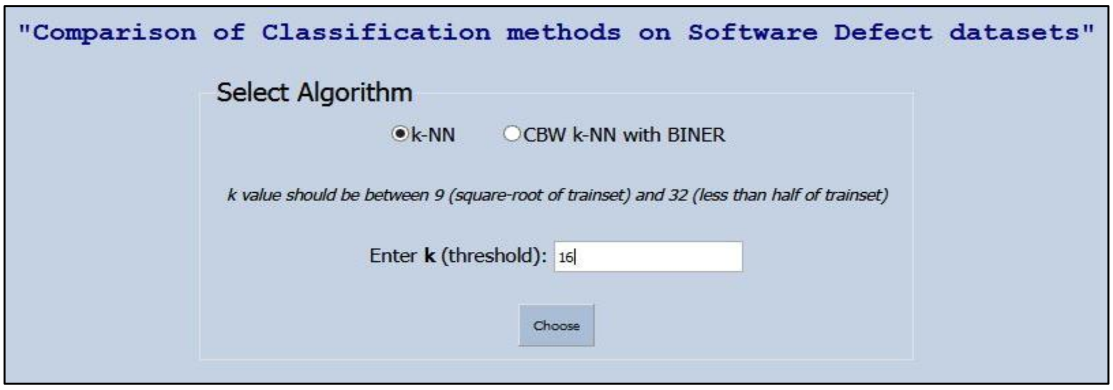
Testing Dataset (Total Instances: 34)

No.	loc	v(G)	ev(G)	iv(G)	N	V	L	D	I	E	B	T	lOCode	lOComment	lOBlank	lOCodeAndComment	uniq_Op	uniq_Opnd	total_Op	total_Opnd	branchCount
1	9	1	1	1	31	114.71	0.05	5.29	21.67	2485.46	0.04	138.08	9	0	0	2	10	3	18	13	1
2	7	1	1	1	29	103.96	0.08	12	8.66	1247.52	0.03	69.31	9	0	0	2	8	4	17	12	1
3	15	3	1	3	75	378.33	0.14	7.17	52.77	2712.63	0.13	150.7	15	4	1	3	10	23	42	33	1
4	8	3	1	3	46	198.81	0.09	11.5	17.29	2286.32	0.07	127.02	8	0	0	0	10	10	23	23	5
5	17	4	1	3	61	267.93	0.1	10.5	25.52	2813.27	0.09	156.29	17	0	0	0	9	12	33	28	7
6	16	4	1	3	59	246.03	0.09	11.2	21.97	2755.54	0.08	153.09	17	0	0	0	8	10	31	28	7
7	12	3	1	3	49	230.32	0.09	11.67	19.74	2687.06	0.08	149.28	12	1	0	7	14	12	29	20	5
8	10	3	1	3	47	209.59	0.12	8.33	25.16	1745.88	0.07	96.99	12	1	0	7	10	12	27	20	5
9	12	3	1	3	47	215.49	0.09	11.23	19.19	2419.95	0.07	134.44	12	1	0	7	13	11	28	19	5
10	12	4	1	3	43	191.76	0.1	10.2	18.8	1955.95	0.06	108.66	12	1	0	7	12	10	26	17	5

[34]

Figure 4.10 Divided Data for the System

Figure 4.11 represents to select algorithm for calculation class values. If user choose k-NN algorithm, system calculates the step-by-step processes of k-NN algorithm firstly. Then, this page shows minimum k value and maximum k value according to training dataset and user can input k values to calculate.



"Comparison of Classification methods on Software Defect datasets"

Select Algorithm

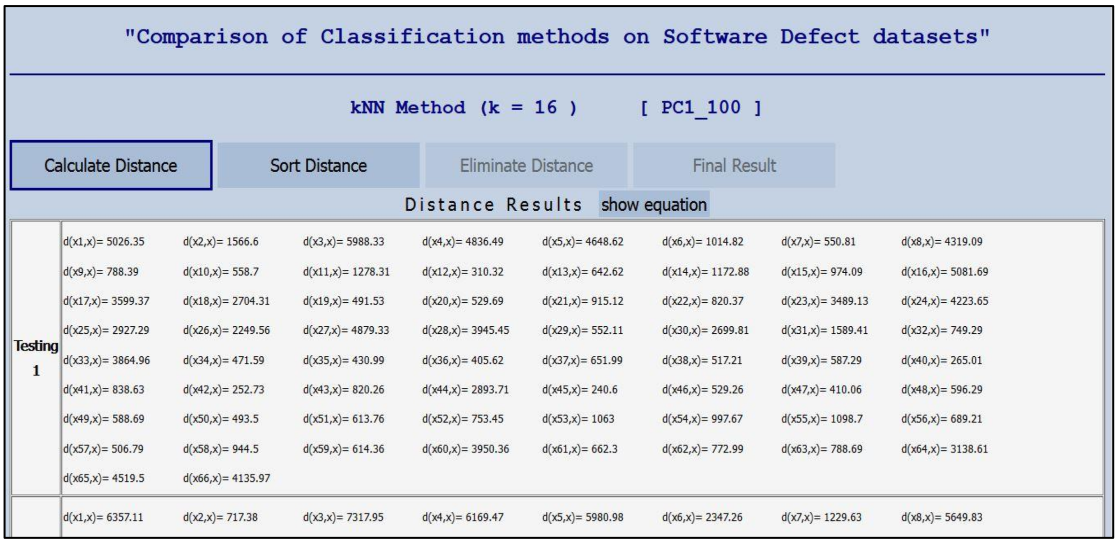
[✓ k-NN] CBW k-NN with BINER

k value should be between 9 (square-root of trainset) and 32 (less than half of trainset)

Enter **k** (threshold): 16

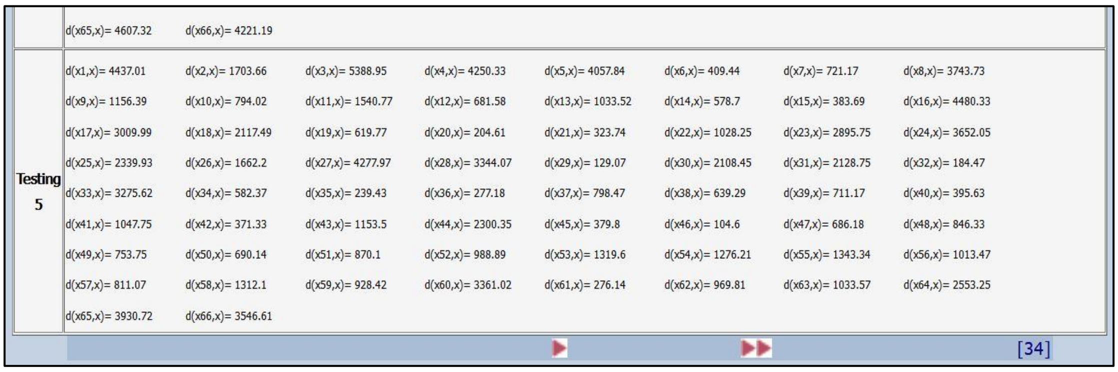
Figure 4.11 Choosing Algorithms for the System

Figure 4.12 and Figure 4.13 represent calculation distance page of the system. This system shows the calculated distance values for each tuple of testing dataset and user can click next and last button to view remaining distance values by showing 5 testing data per page. Then user can click show equation to view distance calculation formula.



Testing 1	d(x1,x)= 5026.35	d(x2,x)= 1566.6	d(x3,x)= 5988.33	d(x4,x)= 4836.49	d(x5,x)= 4648.62	d(x6,x)= 1014.82	d(x7,x)= 550.81	d(x8,x)= 4319.09
	d(x9,x)= 788.39	d(x10,x)= 558.7	d(x11,x)= 1278.31	d(x12,x)= 310.32	d(x13,x)= 642.62	d(x14,x)= 1172.88	d(x15,x)= 974.09	d(x16,x)= 5081.69
	d(x17,x)= 3599.37	d(x18,x)= 2704.31	d(x19,x)= 491.53	d(x20,x)= 529.69	d(x21,x)= 915.12	d(x22,x)= 820.37	d(x23,x)= 3489.13	d(x24,x)= 4223.65
	d(x25,x)= 2927.29	d(x26,x)= 2249.56	d(x27,x)= 4879.33	d(x28,x)= 3945.45	d(x29,x)= 552.11	d(x30,x)= 2699.81	d(x31,x)= 1589.41	d(x32,x)= 749.29
	d(x33,x)= 3864.96	d(x34,x)= 471.59	d(x35,x)= 430.99	d(x36,x)= 405.62	d(x37,x)= 651.99	d(x38,x)= 517.21	d(x39,x)= 587.29	d(x40,x)= 265.01
	d(x41,x)= 838.63	d(x42,x)= 252.73	d(x43,x)= 820.26	d(x44,x)= 2893.71	d(x45,x)= 240.6	d(x46,x)= 529.26	d(x47,x)= 410.06	d(x48,x)= 596.29
	d(x49,x)= 588.69	d(x50,x)= 493.5	d(x51,x)= 613.76	d(x52,x)= 753.45	d(x53,x)= 1063	d(x54,x)= 997.67	d(x55,x)= 1098.7	d(x56,x)= 689.21
	d(x57,x)= 506.79	d(x58,x)= 944.5	d(x59,x)= 614.36	d(x60,x)= 3950.36	d(x61,x)= 662.3	d(x62,x)= 772.99	d(x63,x)= 788.69	d(x64,x)= 3138.61
	d(x65,x)= 4519.5	d(x66,x)= 4135.97						
	d(x1,x)= 6357.11	d(x2,x)= 717.38	d(x3,x)= 7317.95	d(x4,x)= 6169.47	d(x5,x)= 5980.98	d(x6,x)= 2347.26	d(x7,x)= 1229.63	d(x8,x)= 5649.83

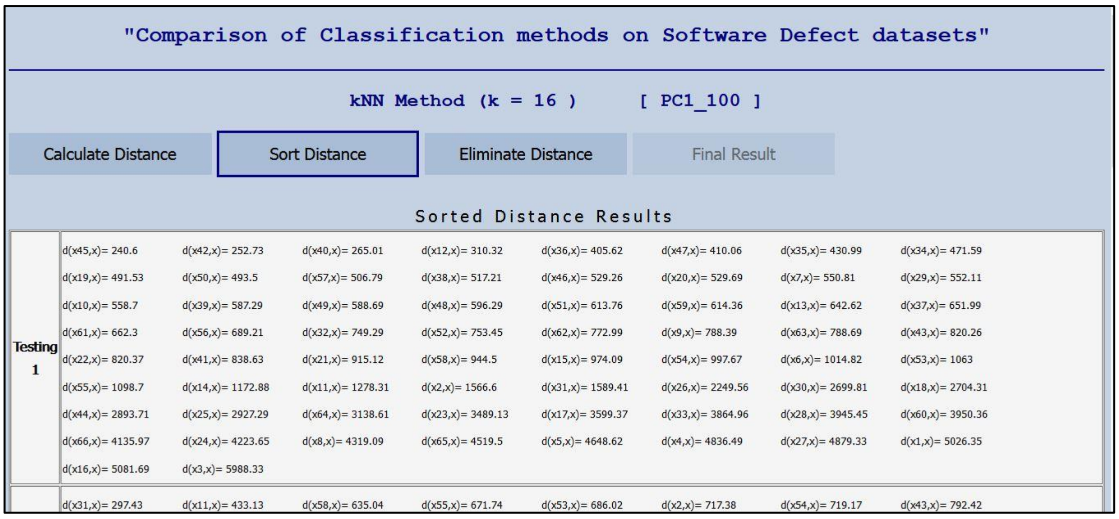
Figure 4.12 Calculation Distance for k-NN Method of Testing 1



	d(x65,x)= 4607.32	d(x66,x)= 4221.19						
Testing 5	d(x1,x)= 4437.01	d(x2,x)= 1703.66	d(x3,x)= 5388.95	d(x4,x)= 4250.33	d(x5,x)= 4057.84	d(x6,x)= 409.44	d(x7,x)= 721.17	d(x8,x)= 3743.73
	d(x9,x)= 1156.39	d(x10,x)= 794.02	d(x11,x)= 1540.77	d(x12,x)= 681.58	d(x13,x)= 1033.52	d(x14,x)= 578.7	d(x15,x)= 383.69	d(x16,x)= 4480.33
	d(x17,x)= 3009.99	d(x18,x)= 2117.49	d(x19,x)= 619.77	d(x20,x)= 204.61	d(x21,x)= 323.74	d(x22,x)= 1028.25	d(x23,x)= 2895.75	d(x24,x)= 3652.05
	d(x25,x)= 2339.93	d(x26,x)= 1662.2	d(x27,x)= 4277.97	d(x28,x)= 3344.07	d(x29,x)= 129.07	d(x30,x)= 2108.45	d(x31,x)= 2128.75	d(x32,x)= 184.47
	d(x33,x)= 3275.62	d(x34,x)= 582.37	d(x35,x)= 239.43	d(x36,x)= 277.18	d(x37,x)= 798.47	d(x38,x)= 639.29	d(x39,x)= 711.17	d(x40,x)= 395.63
	d(x41,x)= 1047.75	d(x42,x)= 371.33	d(x43,x)= 1153.5	d(x44,x)= 2300.35	d(x45,x)= 379.8	d(x46,x)= 104.6	d(x47,x)= 686.18	d(x48,x)= 846.33
	d(x49,x)= 753.75	d(x50,x)= 690.14	d(x51,x)= 870.1	d(x52,x)= 988.89	d(x53,x)= 1319.6	d(x54,x)= 1276.21	d(x55,x)= 1343.34	d(x56,x)= 1013.47
	d(x57,x)= 811.07	d(x58,x)= 1312.1	d(x59,x)= 928.42	d(x60,x)= 3361.02	d(x61,x)= 276.14	d(x62,x)= 969.81	d(x63,x)= 1033.57	d(x64,x)= 2553.25
	d(x65,x)= 3930.72	d(x66,x)= 3546.61						

Figure 4.13 Calculation Distance for k-NN Method of Testing 5

Figure 4.14 and Figure 4.15 represent sorted distance page of the system. This system shows the sorted distance values for each tuple of testing dataset according to the above distance values. Sorting order of distance values is lowest to highest to eliminate the nearest neighbor distance values.



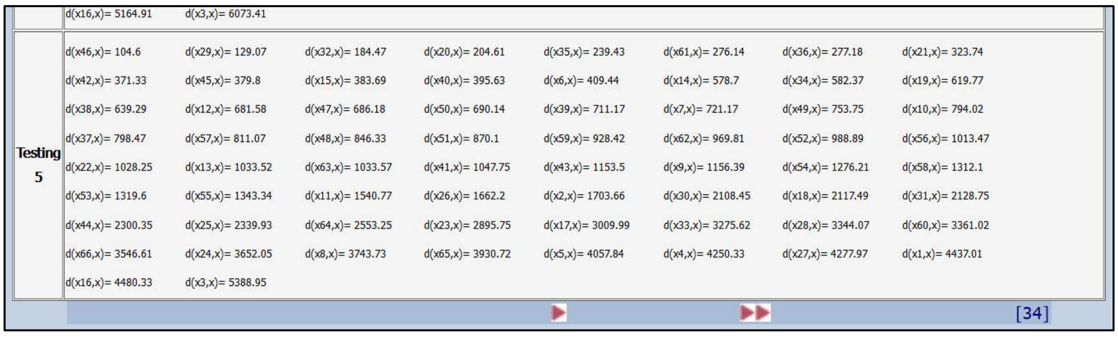
"Comparison of Classification methods on Software Defect datasets"

kNN Method (k = 16) [PC1_100]

Sorted Distance Results

Testing 1	d(x45,x)= 240.6	d(x42,x)= 252.73	d(x40,x)= 265.01	d(x12,x)= 310.32	d(x36,x)= 405.62	d(x47,x)= 410.06	d(x35,x)= 430.99	d(x34,x)= 471.59
	d(x19,x)= 491.53	d(x50,x)= 493.5	d(x57,x)= 506.79	d(x38,x)= 517.21	d(x46,x)= 529.26	d(x20,x)= 529.69	d(x7,x)= 550.81	d(x29,x)= 552.11
	d(x10,x)= 558.7	d(x39,x)= 587.29	d(x49,x)= 588.69	d(x48,x)= 596.29	d(x51,x)= 613.76	d(x59,x)= 614.36	d(x13,x)= 642.62	d(x37,x)= 651.99
	d(x61,x)= 662.3	d(x56,x)= 689.21	d(x32,x)= 749.29	d(x52,x)= 753.45	d(x62,x)= 772.99	d(x9,x)= 788.39	d(x63,x)= 788.69	d(x43,x)= 820.26
	d(x22,x)= 820.37	d(x41,x)= 838.63	d(x21,x)= 915.12	d(x58,x)= 944.5	d(x15,x)= 974.09	d(x54,x)= 997.67	d(x6,x)= 1014.82	d(x53,x)= 1063
	d(x55,x)= 1098.7	d(x14,x)= 1172.88	d(x11,x)= 1278.31	d(x2,x)= 1566.6	d(x31,x)= 1589.41	d(x26,x)= 2249.56	d(x30,x)= 2699.81	d(x18,x)= 2704.31
	d(x44,x)= 2893.71	d(x25,x)= 2927.29	d(x64,x)= 3138.61	d(x23,x)= 3489.13	d(x17,x)= 3599.37	d(x33,x)= 3864.96	d(x28,x)= 3945.45	d(x60,x)= 3950.36
	d(x66,x)= 4135.97	d(x24,x)= 4223.65	d(x8,x)= 4319.09	d(x65,x)= 4519.5	d(x5,x)= 4648.62	d(x4,x)= 4836.49	d(x27,x)= 4879.33	d(x1,x)= 5026.35
	d(x16,x)= 5081.69	d(x3,x)= 5988.33						
	d(x31,x)= 297.43	d(x11,x)= 433.13	d(x58,x)= 635.04	d(x55,x)= 671.74	d(x53,x)= 686.02	d(x2,x)= 717.38	d(x54,x)= 719.17	d(x43,x)= 792.42

Figure 4.14 Sorting Distance for k-NN Method of Testing 1

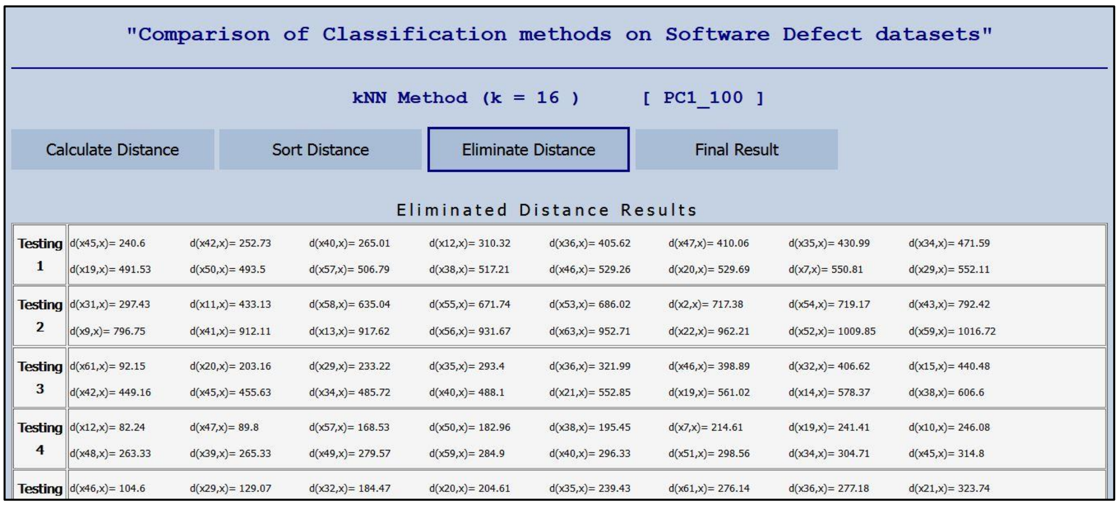


	d(x16,x)= 5164.91	d(x3,x)= 6073.41						
Testing 5	d(x46,x)= 104.6	d(x29,x)= 129.07	d(x32,x)= 184.47	d(x20,x)= 204.61	d(x35,x)= 239.43	d(x61,x)= 276.14	d(x36,x)= 277.18	d(x21,x)= 323.74
	d(x42,x)= 371.33	d(x45,x)= 379.8	d(x15,x)= 383.69	d(x40,x)= 395.63	d(x6,x)= 409.44	d(x14,x)= 578.7	d(x34,x)= 582.37	d(x19,x)= 619.77
	d(x38,x)= 639.29	d(x12,x)= 681.58	d(x47,x)= 686.18	d(x50,x)= 690.14	d(x39,x)= 711.17	d(x7,x)= 721.17	d(x49,x)= 753.75	d(x10,x)= 794.02
	d(x37,x)= 798.47	d(x57,x)= 811.07	d(x48,x)= 846.33	d(x51,x)= 870.1	d(x59,x)= 928.42	d(x62,x)= 969.81	d(x52,x)= 988.89	d(x56,x)= 1013.47
	d(x22,x)= 1028.25	d(x13,x)= 1033.52	d(x63,x)= 1033.57	d(x41,x)= 1047.75	d(x43,x)= 1153.5	d(x9,x)= 1156.39	d(x54,x)= 1276.21	d(x58,x)= 1312.1
	d(x53,x)= 1319.6	d(x55,x)= 1343.34	d(x11,x)= 1540.77	d(x26,x)= 1662.2	d(x2,x)= 1703.66	d(x30,x)= 2108.45	d(x18,x)= 2117.49	d(x31,x)= 2128.75
	d(x44,x)= 2300.35	d(x25,x)= 2339.93	d(x64,x)= 2553.25	d(x23,x)= 2895.75	d(x17,x)= 3009.99	d(x33,x)= 3275.62	d(x28,x)= 3344.07	d(x60,x)= 3361.02
	d(x66,x)= 3546.61	d(x24,x)= 3652.05	d(x8,x)= 3743.73	d(x65,x)= 3930.72	d(x5,x)= 4057.84	d(x4,x)= 4250.33	d(x27,x)= 4277.97	d(x1,x)= 4437.01
	d(x16,x)= 4480.33	d(x3,x)= 5388.95						

[34]

Figure 4.15 Sorting Distance for k-NN Method of Testing 5

Figure 4.16 represents eliminate distance page of the system by using k-NN method. If user clicks the eliminate distance button in navigation link, user can view the eliminated distance values for each tuple of testing dataset according to the above sorted distance values.



"Comparison of Classification methods on Software Defect datasets"

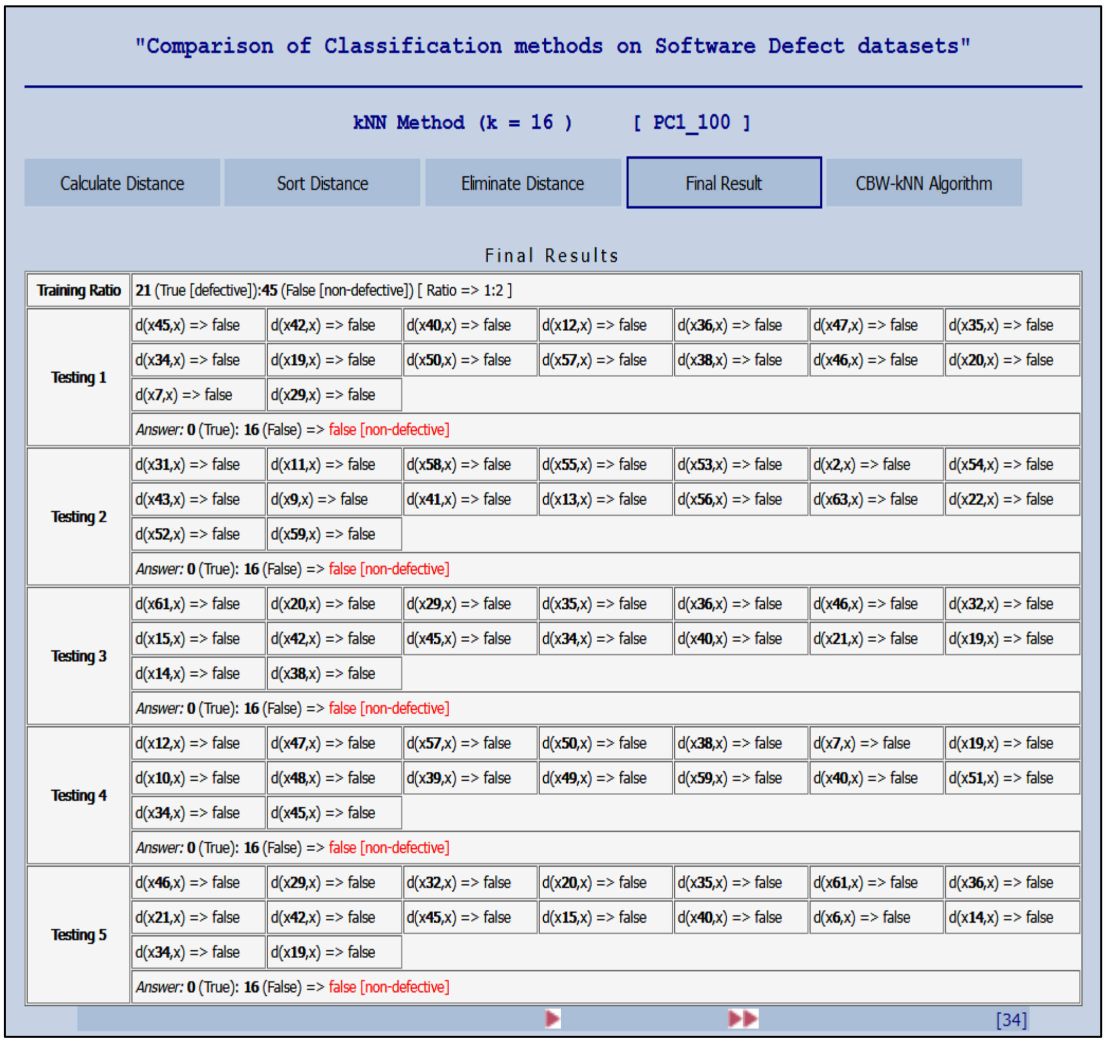
kNN Method (k = 16) [PC1_100]

Eliminated Distance Results

Testing 1	d(x45,x)= 240.6	d(x42,x)= 252.73	d(x40,x)= 265.01	d(x12,x)= 310.32	d(x36,x)= 405.62	d(x47,x)= 410.06	d(x35,x)= 430.99	d(x34,x)= 471.59
	d(x19,x)= 491.53	d(x50,x)= 493.5	d(x57,x)= 506.79	d(x38,x)= 517.21	d(x46,x)= 529.26	d(x20,x)= 529.69	d(x7,x)= 550.81	d(x29,x)= 552.11
Testing 2	d(x31,x)= 297.43	d(x11,x)= 433.13	d(x58,x)= 635.04	d(x55,x)= 671.74	d(x53,x)= 686.02	d(x2,x)= 717.38	d(x54,x)= 719.17	d(x43,x)= 792.42
	d(x9,x)= 796.75	d(x41,x)= 912.11	d(x13,x)= 917.62	d(x56,x)= 931.67	d(x63,x)= 952.71	d(x22,x)= 962.21	d(x52,x)= 1009.85	d(x59,x)= 1016.72
Testing 3	d(x61,x)= 92.15	d(x20,x)= 203.16	d(x29,x)= 233.22	d(x35,x)= 293.4	d(x36,x)= 321.99	d(x46,x)= 398.89	d(x32,x)= 406.62	d(x15,x)= 440.48
	d(x42,x)= 449.16	d(x45,x)= 455.63	d(x34,x)= 485.72	d(x40,x)= 488.1	d(x21,x)= 552.85	d(x19,x)= 561.02	d(x14,x)= 578.37	d(x38,x)= 606.6
Testing 4	d(x12,x)= 82.24	d(x47,x)= 89.8	d(x57,x)= 168.53	d(x50,x)= 182.96	d(x38,x)= 195.45	d(x7,x)= 214.61	d(x19,x)= 241.41	d(x10,x)= 246.08
	d(x48,x)= 263.33	d(x39,x)= 265.33	d(x49,x)= 279.57	d(x59,x)= 284.9	d(x40,x)= 296.33	d(x51,x)= 298.56	d(x34,x)= 304.71	d(x45,x)= 314.8
Testing	d(x46,x)= 104.6	d(x29,x)= 129.07	d(x32,x)= 184.47	d(x20,x)= 204.61	d(x35,x)= 239.43	d(x61,x)= 276.14	d(x36,x)= 277.18	d(x21,x)= 323.74

Figure 4.16 Elimination Distance for k-NN Method

If user clicks the final result button in navigation link, user can view the class values for each tuple of testing dataset according to the class values of above eliminated distance values. The result is shown in Figure 4.17.

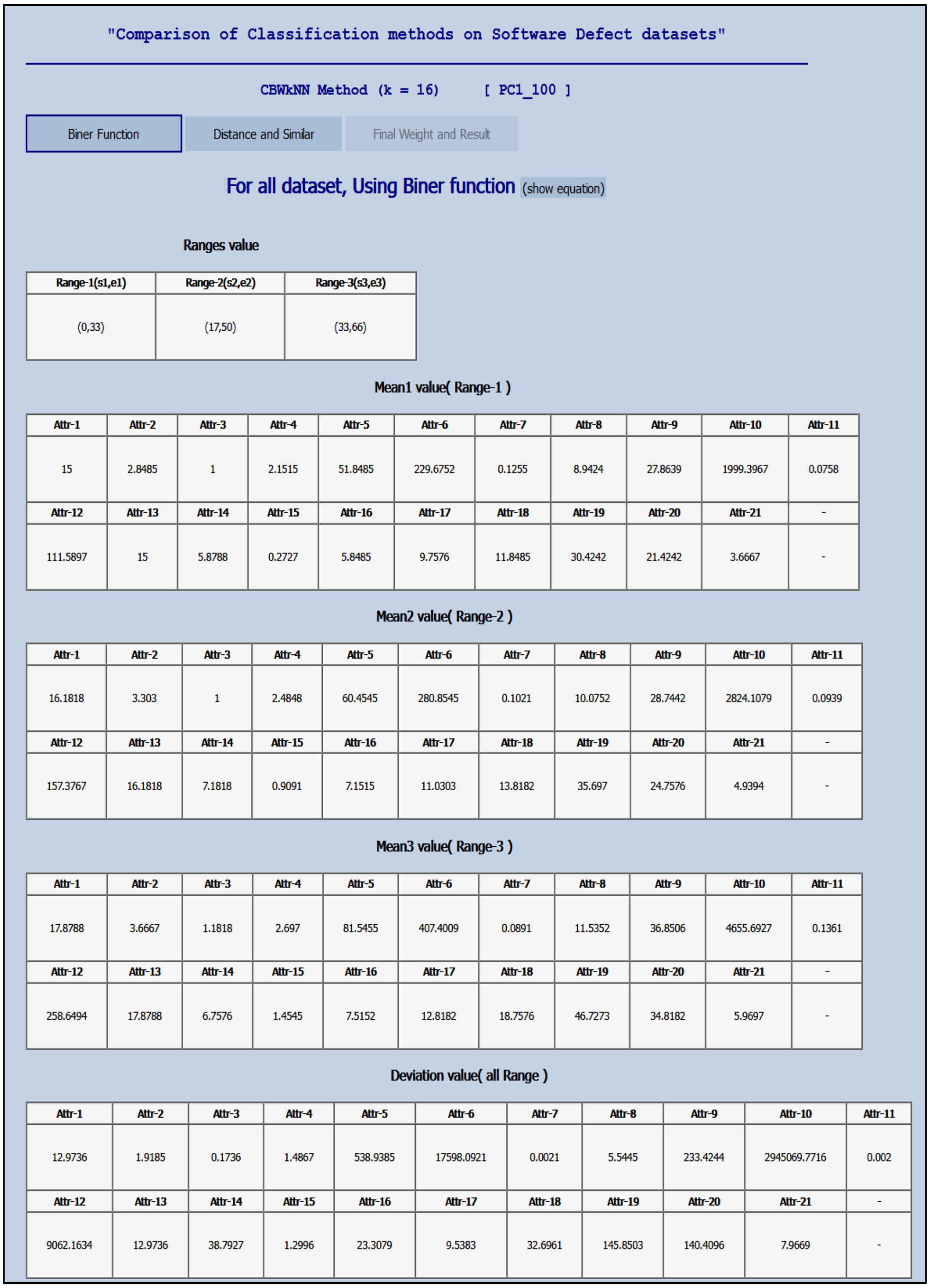

"Comparison of Classification methods on Software Defect datasets"
kNN Method (k = 16) [PC1_100]
Calculate Distance
Sort Distance
Eliminate Distance
Final Result
CBW-kNN Algorithm
[34]

Final Results

Training Ratio	**21** (True [defective]):**45** (False [non-defective]) [Ratio => 1:2]						
Testing 1	d(x**45**,x) => false	d(x**42**,x) => false	d(x**40**,x) => false	d(x**12**,x) => false	d(x**36**,x) => false	d(x**47**,x) => false	d(x**35**,x) => false
	d(x**34**,x) => false	d(x**19**,x) => false	d(x**50**,x) => false	d(x**57**,x) => false	d(x**38**,x) => false	d(x**46**,x) => false	d(x**20**,x) => false
	d(x**7**,x) => false	d(x**29**,x) => false					
	Answer: **0** (True): **16** (False) => false [non-defective]						
Testing 2	d(x**31**,x) => false	d(x**11**,x) => false	d(x**58**,x) => false	d(x**55**,x) => false	d(x**53**,x) => false	d(x**2**,x) => false	d(x**54**,x) => false
	d(x**43**,x) => false	d(x**9**,x) => false	d(x**41**,x) => false	d(x**13**,x) => false	d(x**56**,x) => false	d(x**63**,x) => false	d(x**22**,x) => false
	d(x**52**,x) => false	d(x**59**,x) => false					
	Answer: **0** (True): **16** (False) => false [non-defective]						
Testing 3	d(x**61**,x) => false	d(x**20**,x) => false	d(x**29**,x) => false	d(x**35**,x) => false	d(x**36**,x) => false	d(x**46**,x) => false	d(x**32**,x) => false
	d(x**15**,x) => false	d(x**42**,x) => false	d(x**45**,x) => false	d(x**34**,x) => false	d(x**40**,x) => false	d(x**21**,x) => false	d(x**19**,x) => false
	d(x**14**,x) => false	d(x**38**,x) => false					
	Answer: **0** (True): **16** (False) => false [non-defective]						
Testing 4	d(x**12**,x) => false	d(x**47**,x) => false	d(x**57**,x) => false	d(x**50**,x) => false	d(x**38**,x) => false	d(x**7**,x) => false	d(x**19**,x) => false
	d(x**10**,x) => false	d(x**48**,x) => false	d(x**39**,x) => false	d(x**49**,x) => false	d(x**59**,x) => false	d(x**40**,x) => false	d(x**51**,x) => false
	d(x**34**,x) => false	d(x**45**,x) => false					
	Answer: **0** (True): **16** (False) => false [non-defective]						
Testing 5	d(x**46**,x) => false	d(x**29**,x) => false	d(x**32**,x) => false	d(x**20**,x) => false	d(x**35**,x) => false	d(x**61**,x) => false	d(x**36**,x) => false
	d(x**21**,x) => false	d(x**42**,x) => false	d(x**45**,x) => false	d(x**15**,x) => false	d(x**40**,x) => false	d(x**6**,x) => false	d(x**14**,x) => false
	d(x**34**,x) => false	d(x**19**,x) => false					
	Answer: **0** (True): **16** (False) => false [non-defective]						

Figure 4.17 Final Result for k-NN Method

If user clicks the CBW k-NN Algorithm button to calculate the class values with CBW k-NN Algorithm. The user calculates the class values with k-NN algorithm and clicks the other algorithm, CBW k-NN and vice versa. For CBW k-NN algorithm, user calculates the partitions of three sub ranges after sorting the training dataset and mean and deviation values for each sub range. The result is shown in Figure 4.18.



"Comparison of Classification methods on Software Defect datasets"

CBWkNN Method (k = 16) [PC1_100]

For all dataset, Using Biner function (show equation)

Ranges value

Range-1(s1,e1)	Range-2(s2,e2)	Range-3(s3,e3)
(0,33)	(17,50)	(33,66)

Mean1 value(Range-1)

Attr-1	Attr-2	Attr-3	Attr-4	Attr-5	Attr-6	Attr-7	Attr-8	Attr-9	Attr-10	Attr-11
15	2.8485	1	2.1515	51.8485	229.6752	0.1255	8.9424	27.8639	1999.3967	0.0758
Attr-12	**Attr-13**	**Attr-14**	**Attr-15**	**Attr-16**	**Attr-17**	**Attr-18**	**Attr-19**	**Attr-20**	**Attr-21**	-
111.5897	15	5.8788	0.2727	5.8485	9.7576	11.8485	30.4242	21.4242	3.6667	-

Mean2 value(Range-2)

Attr-1	Attr-2	Attr-3	Attr-4	Attr-5	Attr-6	Attr-7	Attr-8	Attr-9	Attr-10	Attr-11
16.1818	3.303	1	2.4848	60.4545	280.8545	0.1021	10.0752	28.7442	2824.1079	0.0939
Attr-12	**Attr-13**	**Attr-14**	**Attr-15**	**Attr-16**	**Attr-17**	**Attr-18**	**Attr-19**	**Attr-20**	**Attr-21**	-
157.3767	16.1818	7.1818	0.9091	7.1515	11.0303	13.8182	35.697	24.7576	4.9394	-

Mean3 value(Range-3)

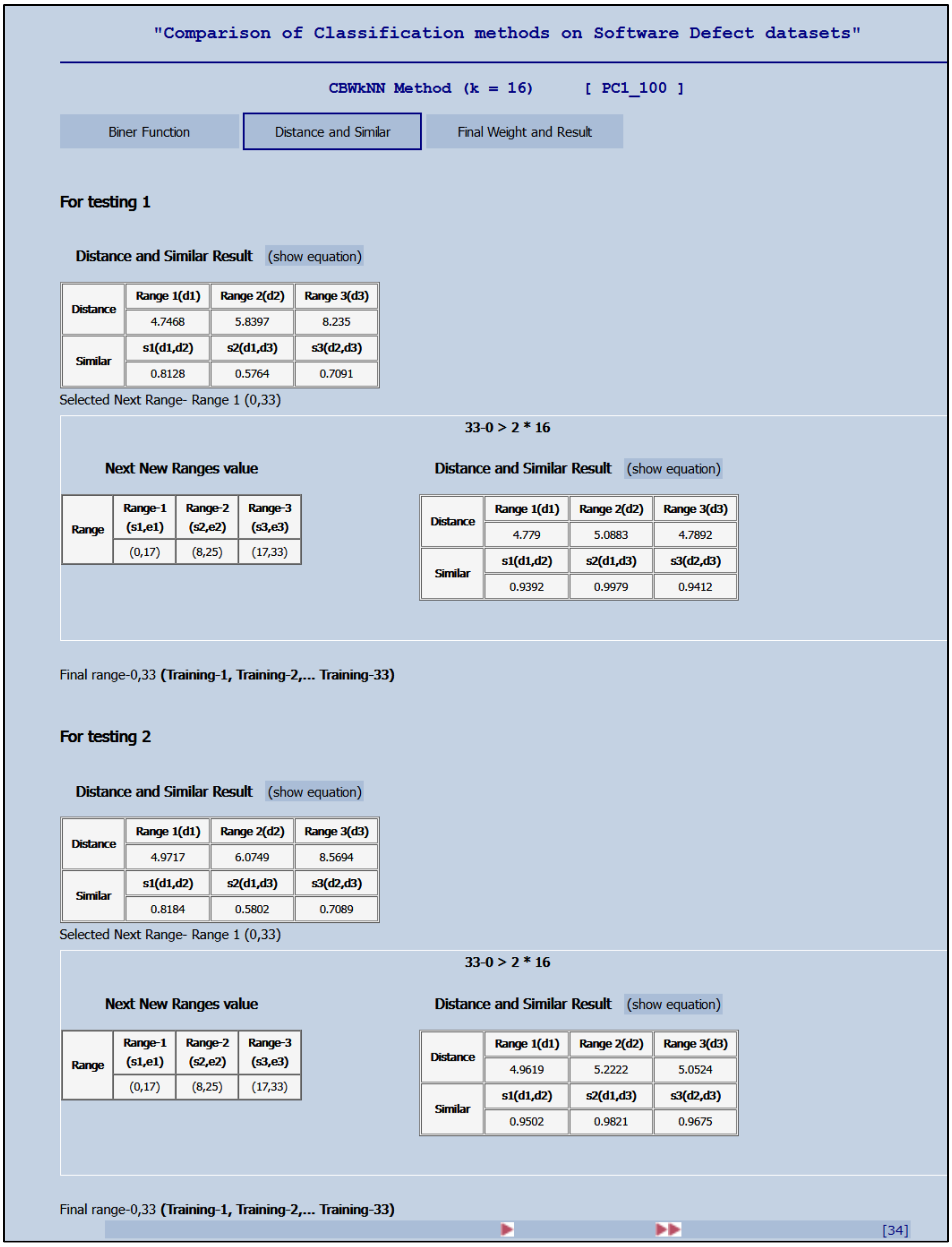
Attr-1	Attr-2	Attr-3	Attr-4	Attr-5	Attr-6	Attr-7	Attr-8	Attr-9	Attr-10	Attr-11
17.8788	3.6667	1.1818	2.697	81.5455	407.4009	0.0891	11.5352	36.8506	4655.6927	0.1361
Attr-12	**Attr-13**	**Attr-14**	**Attr-15**	**Attr-16**	**Attr-17**	**Attr-18**	**Attr-19**	**Attr-20**	**Attr-21**	-
258.6494	17.8788	6.7576	1.4545	7.5152	12.8182	18.7576	46.7273	34.8182	5.9697	-

Deviation value(all Range)

Attr-1	Attr-2	Attr-3	Attr-4	Attr-5	Attr-6	Attr-7	Attr-8	Attr-9	Attr-10	Attr-11
12.9736	1.9185	0.1736	1.4867	538.9385	17598.0921	0.0021	5.5445	233.4244	2945069.7716	0.002
Attr-12	**Attr-13**	**Attr-14**	**Attr-15**	**Attr-16**	**Attr-17**	**Attr-18**	**Attr-19**	**Attr-20**	**Attr-21**	-
9062.1634	12.9736	38.7927	1.2996	23.3079	9.5383	32.6961	145.8503	140.4096	7.9669	-

Figure 4.18 BINER Function for CBW k-NN Method

Figure 4.19 represents distance and similar page of the system. If user clicks the distance and similar button, system calculates the three distance values for each sub range and similar values for three distance values. Then, the system calculates final range of each tuple in testing dataset.



"Comparison of Classification methods on Software Defect datasets"

CBWkNN Method (k = 16) [PC1_100]

For testing 1

Distance and Similar Result (show equation)

Distance	**Range 1(d1)**	**Range 2(d2)**	**Range 3(d3)**
	4.7468	5.8397	8.235
Similar	**s1(d1,d2)**	**s2(d1,d3)**	**s3(d2,d3)**
	0.8128	0.5764	0.7091

Selected Next Range- Range 1 (0,33)

33-0 > 2 * 16

Next New Ranges value

Range	**Range-1 (s1,e1)**	**Range-2 (s2,e2)**	**Range-3 (s3,e3)**
	(0,17)	(8,25)	(17,33)

Distance and Similar Result (show equation)

Distance	**Range 1(d1)**	**Range 2(d2)**	**Range 3(d3)**
	4.779	5.0883	4.7892
Similar	**s1(d1,d2)**	**s2(d1,d3)**	**s3(d2,d3)**
	0.9392	0.9979	0.9412

Final range-0,33 **(Training-1, Training-2,... Training-33)**

For testing 2

Distance and Similar Result (show equation)

Distance	**Range 1(d1)**	**Range 2(d2)**	**Range 3(d3)**
	4.9717	6.0749	8.5694
Similar	**s1(d1,d2)**	**s2(d1,d3)**	**s3(d2,d3)**
	0.8184	0.5802	0.7089

Selected Next Range- Range 1 (0,33)

33-0 > 2 * 16

Next New Ranges value

Range	**Range-1 (s1,e1)**	**Range-2 (s2,e2)**	**Range-3 (s3,e3)**
	(0,17)	(8,25)	(17,33)

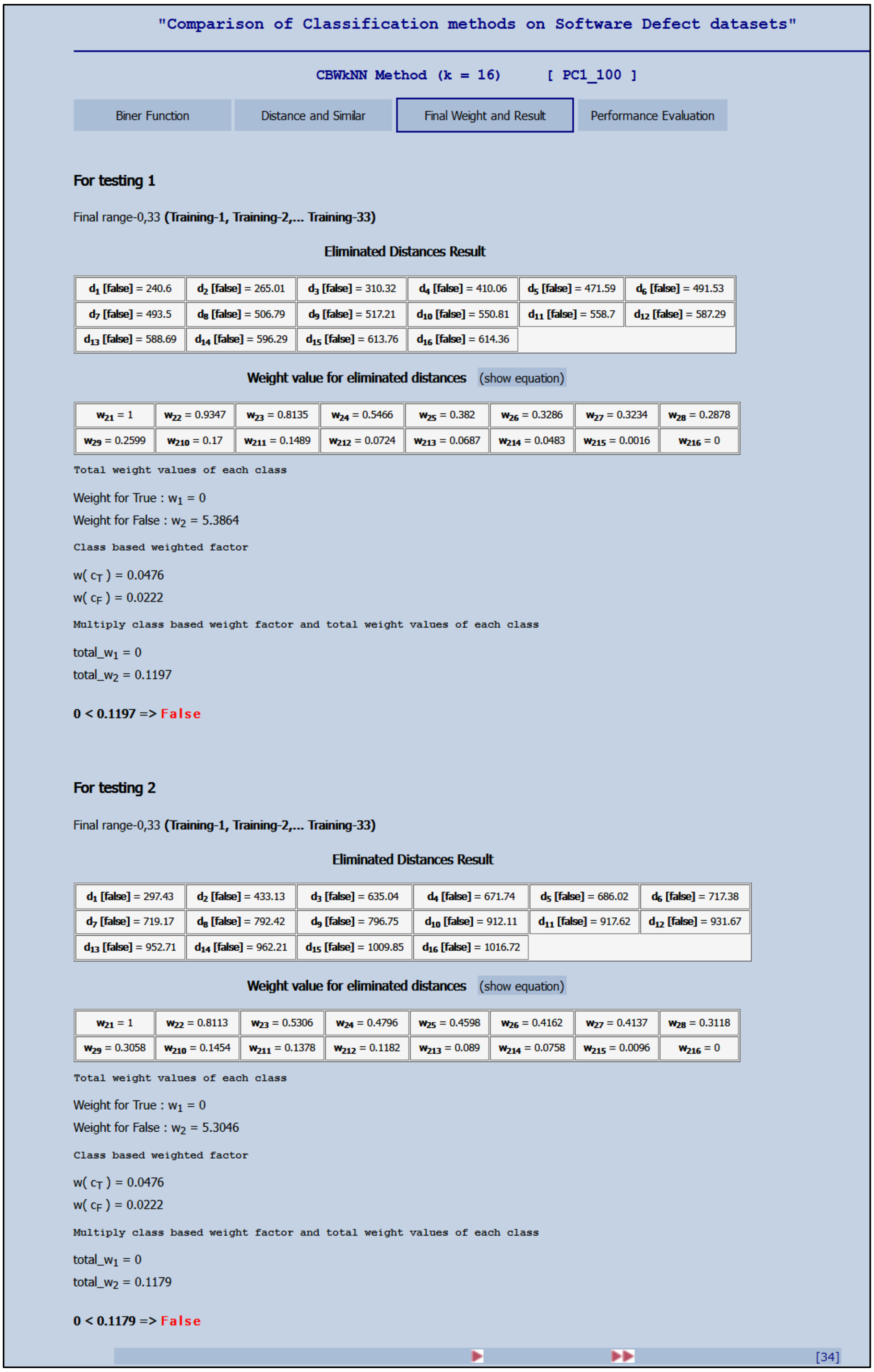
Distance and Similar Result (show equation)

Distance	**Range 1(d1)**	**Range 2(d2)**	**Range 3(d3)**
	4.9619	5.2222	5.0524
Similar	**s1(d1,d2)**	**s2(d1,d3)**	**s3(d2,d3)**
	0.9502	0.9821	0.9675

Final range-0,33 **(Training-1, Training-2,... Training-33)**

Figure 4.19 Calculation Distance and Similar Value for CBW k-NN Method

Figure 4.20 represents final weight and result page of the system. If user clicks final weight and result button, system calculates the eliminated distance values and final weight values and compares weight values to generate class values for each tuple in testing dataset.


"Comparison of Classification methods on Software Defect datasets"
CBWkNN Method (k = 16) [PC1_100]
Biner Function
Distance and Similar
Final Weight and Result
Performance Evaluation

For testing 1

Final range-0,33 **(Training-1, Training-2,... Training-33)**

Eliminated Distances Result

d_1 **[false]** = 240.6	d_2 **[false]** = 265.01	d_3 **[false]** = 310.32	d_4 **[false]** = 410.06	d_5 **[false]** = 471.59	d_6 **[false]** = 491.53
d_7 **[false]** = 493.5	d_8 **[false]** = 506.79	d_9 **[false]** = 517.21	d_{10} **[false]** = 550.81	d_{11} **[false]** = 558.7	d_{12} **[false]** = 587.29
d_{13} **[false]** = 588.69	d_{14} **[false]** = 596.29	d_{15} **[false]** = 613.76	d_{16} **[false]** = 614.36		

Weight value for eliminated distances (show equation)

w_{21} = 1	w_{22} = 0.9347	w_{23} = 0.8135	w_{24} = 0.5466	w_{25} = 0.382	w_{26} = 0.3286	w_{27} = 0.3234	w_{28} = 0.2878
w_{29} = 0.2599	w_{210} = 0.17	w_{211} = 0.1489	w_{212} = 0.0724	w_{213} = 0.0687	w_{214} = 0.0483	w_{215} = 0.0016	w_{216} = 0

Total weight values of each class

Weight for True : $w_1 = 0$

Weight for False : $w_2 = 5.3864$

Class based weighted factor

$w(c_T) = 0.0476$

$w(c_F) = 0.0222$

Multiply class based weight factor and total weight values of each class

$total\_w_1 = 0$

$total\_w_2 = 0.1197$

0 < 0.1197 => False

For testing 2

Final range-0,33 **(Training-1, Training-2,... Training-33)**

Eliminated Distances Result

d_1 **[false]** = 297.43	d_2 **[false]** = 433.13	d_3 **[false]** = 635.04	d_4 **[false]** = 671.74	d_5 **[false]** = 686.02	d_6 **[false]** = 717.38
d_7 **[false]** = 719.17	d_8 **[false]** = 792.42	d_9 **[false]** = 796.75	d_{10} **[false]** = 912.11	d_{11} **[false]** = 917.62	d_{12} **[false]** = 931.67
d_{13} **[false]** = 952.71	d_{14} **[false]** = 962.21	d_{15} **[false]** = 1009.85	d_{16} **[false]** = 1016.72		

Weight value for eliminated distances (show equation)

w_{21} = 1	w_{22} = 0.8113	w_{23} = 0.5306	w_{24} = 0.4796	w_{25} = 0.4598	w_{26} = 0.4162	w_{27} = 0.4137	w_{28} = 0.3118
w_{29} = 0.3058	w_{210} = 0.1454	w_{211} = 0.1378	w_{212} = 0.1182	w_{213} = 0.089	w_{214} = 0.0758	w_{215} = 0.0096	w_{216} = 0

Total weight values of each class

Weight for True : $w_1 = 0$

Weight for False : $w_2 = 5.3046$

Class based weighted factor

$w(c_T) = 0.0476$

$w(c_F) = 0.0222$

Multiply class based weight factor and total weight values of each class

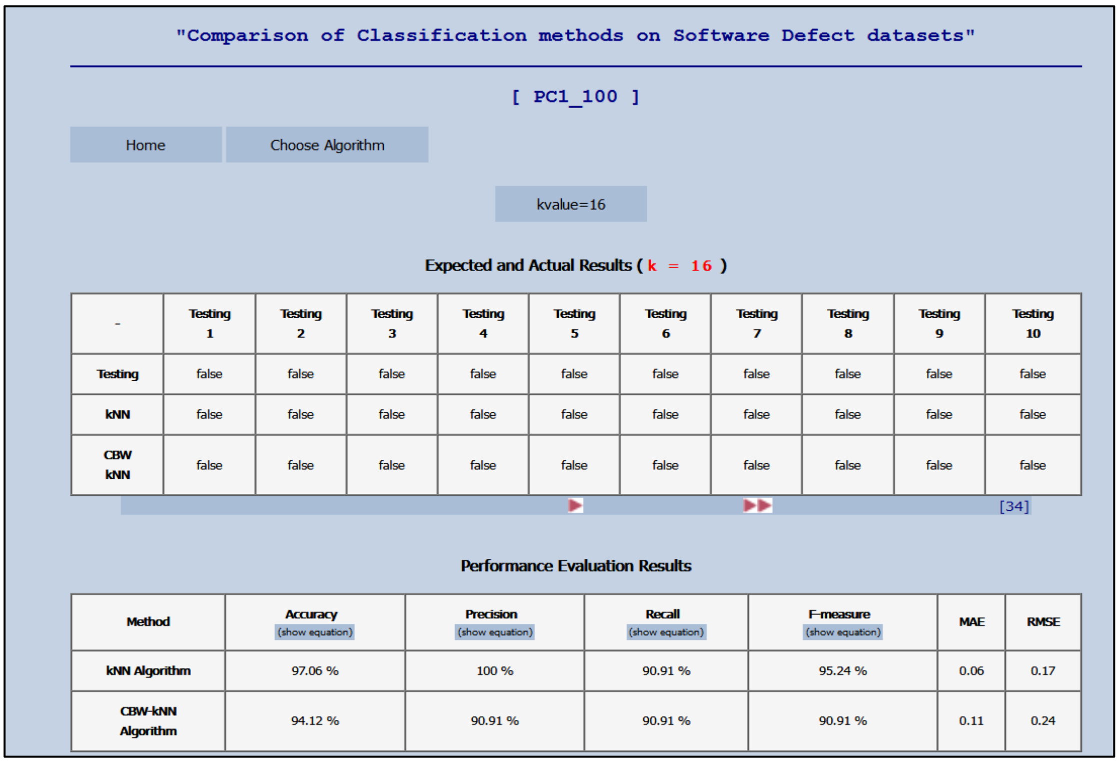
$total\_w_1 = 0$

$total\_w_2 = 0.1179$

0 < 0.1179 => False

Figure 4.20 Final Result for CBW k-NN Method

Figure 4.21 represents the performance evaluation page of the system for k value 16. This is the expected and actual class values result and performance evaluation results based on two method, k-NN and CBW k-NN algorithm. Then, user can click choose algorithm button to select algorithm and input k values for next calculation of these dataset.


"Comparison of Classification methods on Software Defect datasets"

[PC1_100]

Expected and Actual Results (k = 16)

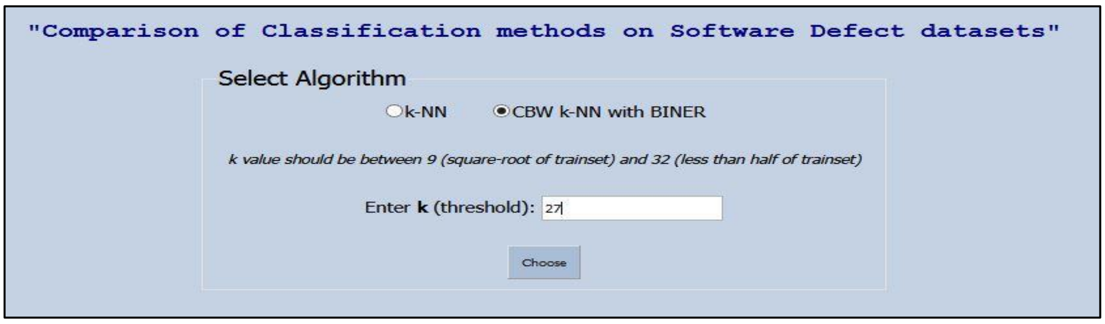
-	Testing 1	Testing 2	Testing 3	Testing 4	Testing 5	Testing 6	Testing 7	Testing 8	Testing 9	Testing 10
Testing	false	false	false	false	false	false	false	false	false	false
kNN	false	false	false	false	false	false	false	false	false	false
CBW kNN	false	false	false	false	false	false	false	false	false	false

Performance Evaluation Results

Method	Accuracy (show equation)	Precision (show equation)	Recall (show equation)	F-measure (show equation)	MAE	RMSE
kNN Algorithm	97.06 %	100 %	90.91 %	95.24 %	0.06	0.17
CBW-kNN Algorithm	94.12 %	90.91 %	90.91 %	90.91 %	0.11	0.24

Figure 4.21 Performance Evaluation Result for Both Methods

Figure 4.22 represents the select algorithm page of the system. User can input another k values and select algorithm to calculate another calculation.


"Comparison of Classification methods on Software Defect datasets"

Select Algorithm

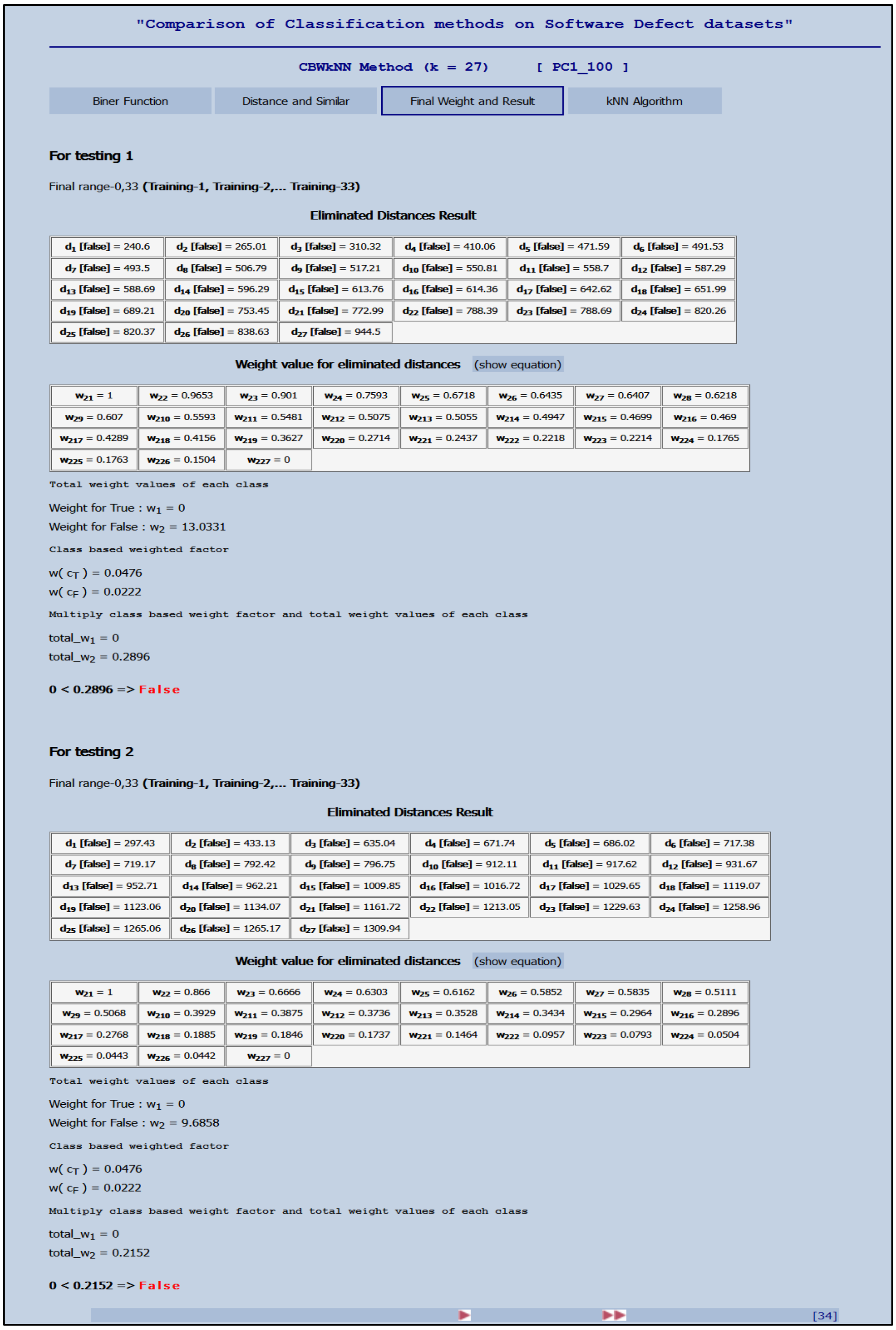
k-NN [✓ CBW k-NN with BINER]

k value should be between 9 (square-root of trainset) and 32 (less than half of trainset)

Enter **k** (threshold): 27

Figure 4.22 Choosing another k Value for Next Selection

If user clicks button in navigation link one-by-one, user can view the final class values for CBW k-NN algorithm for another k value. The result is shown in Figure 4.23.


"Comparison of Classification methods on Software Defect datasets"

CBWkNN Method (k = 27)   [PC1_100]

For testing 1

Final range-0,33 **(Training-1, Training-2,... Training-33)**

Eliminated Distances Result

d_1 [false] = 240.6	d_2 [false] = 265.01	d_3 [false] = 310.32	d_4 [false] = 410.06	d_5 [false] = 471.59	d_6 [false] = 491.53
d_7 [false] = 493.5	d_8 [false] = 506.79	d_9 [false] = 517.21	d_{10} [false] = 550.81	d_{11} [false] = 558.7	d_{12} [false] = 587.29
d_{13} [false] = 588.69	d_{14} [false] = 596.29	d_{15} [false] = 613.76	d_{16} [false] = 614.36	d_{17} [false] = 642.62	d_{18} [false] = 651.99
d_{19} [false] = 689.21	d_{20} [false] = 753.45	d_{21} [false] = 772.99	d_{22} [false] = 788.39	d_{23} [false] = 788.69	d_{24} [false] = 820.26
d_{25} [false] = 820.37	d_{26} [false] = 838.63	d_{27} [false] = 944.5			

Weight value for eliminated distances (show equation)

w_{21} = 1	w_{22} = 0.9653	w_{23} = 0.901	w_{24} = 0.7593	w_{25} = 0.6718	w_{26} = 0.6435	w_{27} = 0.6407	w_{28} = 0.6218
w_{29} = 0.607	w_{210} = 0.5593	w_{211} = 0.5481	w_{212} = 0.5075	w_{213} = 0.5055	w_{214} = 0.4947	w_{215} = 0.4699	w_{216} = 0.469
w_{217} = 0.4289	w_{218} = 0.4156	w_{219} = 0.3627	w_{220} = 0.2714	w_{221} = 0.2437	w_{222} = 0.2218	w_{223} = 0.2214	w_{224} = 0.1765
w_{225} = 0.1763	w_{226} = 0.1504	w_{227} = 0					

```
Total weight values of each class
```

Weight for True : $w_1 = 0$
Weight for False : $w_2 = 13.0331$

```
Class based weighted factor
```

w(c_T) = 0.0476
w(c_F) = 0.0222

```
Multiply class based weight factor and total weight values of each class
```

total_w_1 = 0
total_w_2 = 0.2896

0 < 0.2896 => False

For testing 2

Final range-0,33 **(Training-1, Training-2,... Training-33)**

Eliminated Distances Result

d_1 [false] = 297.43	d_2 [false] = 433.13	d_3 [false] = 635.04	d_4 [false] = 671.74	d_5 [false] = 686.02	d_6 [false] = 717.38
d_7 [false] = 719.17	d_8 [false] = 792.42	d_9 [false] = 796.75	d_{10} [false] = 912.11	d_{11} [false] = 917.62	d_{12} [false] = 931.67
d_{13} [false] = 952.71	d_{14} [false] = 962.21	d_{15} [false] = 1009.85	d_{16} [false] = 1016.72	d_{17} [false] = 1029.65	d_{18} [false] = 1119.07
d_{19} [false] = 1123.06	d_{20} [false] = 1134.07	d_{21} [false] = 1161.72	d_{22} [false] = 1213.05	d_{23} [false] = 1229.63	d_{24} [false] = 1258.96
d_{25} [false] = 1265.06	d_{26} [false] = 1265.17	d_{27} [false] = 1309.94			

Weight value for eliminated distances (show equation)

w_{21} = 1	w_{22} = 0.866	w_{23} = 0.6666	w_{24} = 0.6303	w_{25} = 0.6162	w_{26} = 0.5852	w_{27} = 0.5835	w_{28} = 0.5111
w_{29} = 0.5068	w_{210} = 0.3929	w_{211} = 0.3875	w_{212} = 0.3736	w_{213} = 0.3528	w_{214} = 0.3434	w_{215} = 0.2964	w_{216} = 0.2896
w_{217} = 0.2768	w_{218} = 0.1885	w_{219} = 0.1846	w_{220} = 0.1737	w_{221} = 0.1464	w_{222} = 0.0957	w_{223} = 0.0793	w_{224} = 0.0504
w_{225} = 0.0443	w_{226} = 0.0442	w_{227} = 0					

```
Total weight values of each class
```

Weight for True : $w_1 = 0$
Weight for False : $w_2 = 9.6858$

```
Class based weighted factor
```

w(c_T) = 0.0476
w(c_F) = 0.0222

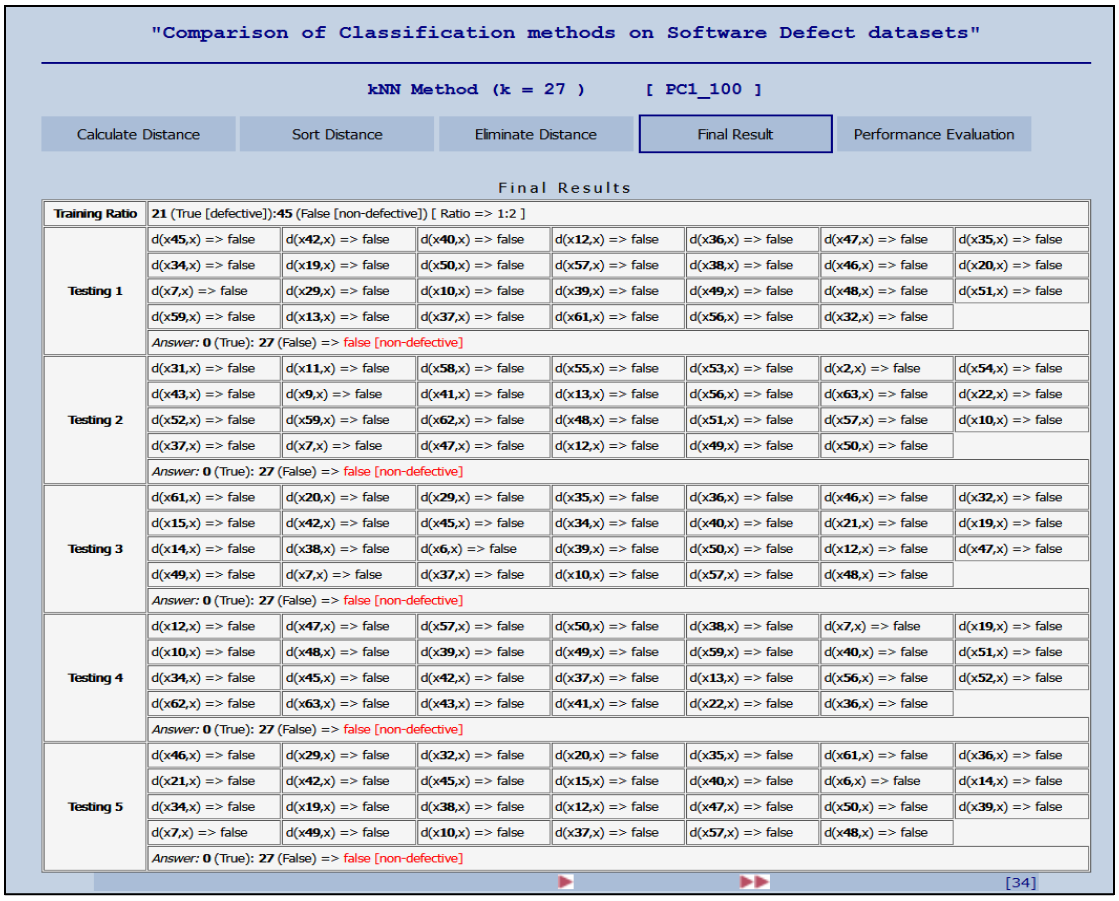
```
Multiply class based weight factor and total weight values of each class
```

total_w_1 = 0
total_w_2 = 0.2152

0 < 0.2152 => False

Figure 4.23 Final Result for CBW k-NN Method (Next Selection)

If user clicks the button in navigation link one-by-one, user can view the final class values for k-NN algorithm for another k value. The result is shown in Figure 4.24.

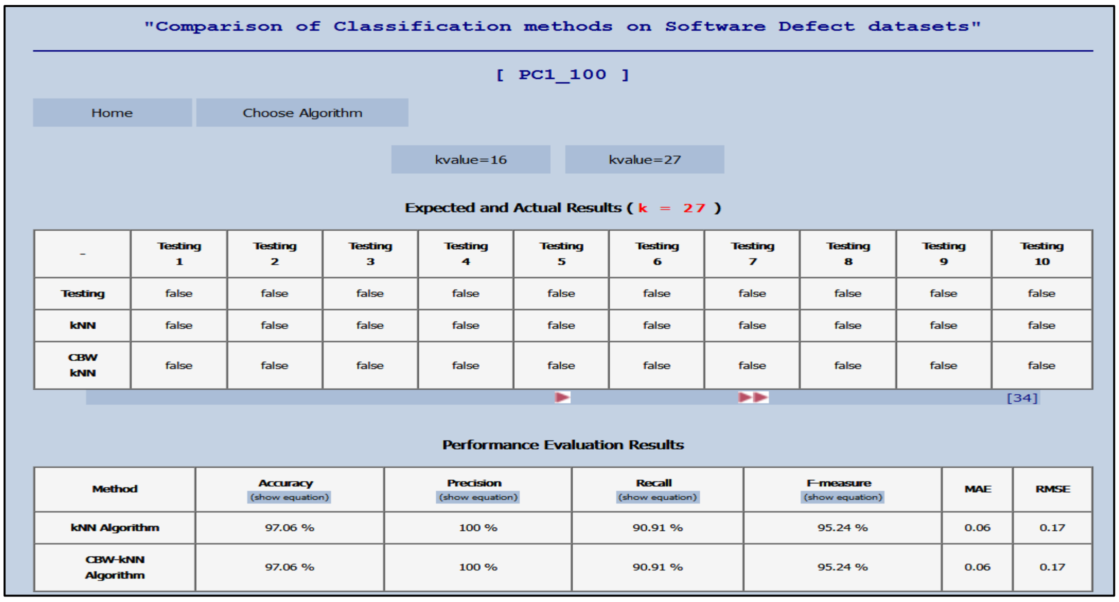


Final Results

Training Ratio	**21** (True [defective]):**45** (False [non-defective]) [Ratio => 1:2]						
Testing 1	d(x**45**,x) => false	d(x**42**,x) => false	d(x**40**,x) => false	d(x**12**,x) => false	d(x**36**,x) => false	d(x**47**,x) => false	d(x**35**,x) => false
	d(x**34**,x) => false	d(x**19**,x) => false	d(x**50**,x) => false	d(x**57**,x) => false	d(x**38**,x) => false	d(x**46**,x) => false	d(x**20**,x) => false
	d(x**7**,x) => false	d(x**29**,x) => false	d(x**10**,x) => false	d(x**39**,x) => false	d(x**49**,x) => false	d(x**48**,x) => false	d(x**51**,x) => false
	d(x**59**,x) => false	d(x**13**,x) => false	d(x**37**,x) => false	d(x**61**,x) => false	d(x**56**,x) => false	d(x**32**,x) => false	
	Answer: **0** (True): **27** (False) => false [non-defective]						
Testing 2	d(x**31**,x) => false	d(x**11**,x) => false	d(x**58**,x) => false	d(x**55**,x) => false	d(x**53**,x) => false	d(x**2**,x) => false	d(x**54**,x) => false
	d(x**43**,x) => false	d(x**9**,x) => false	d(x**41**,x) => false	d(x**13**,x) => false	d(x**56**,x) => false	d(x**63**,x) => false	d(x**22**,x) => false
	d(x**52**,x) => false	d(x**59**,x) => false	d(x**62**,x) => false	d(x**48**,x) => false	d(x**51**,x) => false	d(x**57**,x) => false	d(x**10**,x) => false
	d(x**37**,x) => false	d(x**7**,x) => false	d(x**47**,x) => false	d(x**12**,x) => false	d(x**49**,x) => false	d(x**50**,x) => false	
	Answer: **0** (True): **27** (False) => false [non-defective]						
Testing 3	d(x**61**,x) => false	d(x**20**,x) => false	d(x**29**,x) => false	d(x**35**,x) => false	d(x**36**,x) => false	d(x**46**,x) => false	d(x**32**,x) => false
	d(x**15**,x) => false	d(x**42**,x) => false	d(x**45**,x) => false	d(x**34**,x) => false	d(x**40**,x) => false	d(x**21**,x) => false	d(x**19**,x) => false
	d(x**14**,x) => false	d(x**38**,x) => false	d(x**6**,x) => false	d(x**39**,x) => false	d(x**50**,x) => false	d(x**12**,x) => false	d(x**47**,x) => false
	d(x**49**,x) => false	d(x**7**,x) => false	d(x**37**,x) => false	d(x**10**,x) => false	d(x**57**,x) => false	d(x**48**,x) => false	
	Answer: **0** (True): **27** (False) => false [non-defective]						
Testing 4	d(x**12**,x) => false	d(x**47**,x) => false	d(x**57**,x) => false	d(x**50**,x) => false	d(x**38**,x) => false	d(x**7**,x) => false	d(x**19**,x) => false
	d(x**10**,x) => false	d(x**48**,x) => false	d(x**39**,x) => false	d(x**49**,x) => false	d(x**59**,x) => false	d(x**40**,x) => false	d(x**51**,x) => false
	d(x**34**,x) => false	d(x**45**,x) => false	d(x**42**,x) => false	d(x**37**,x) => false	d(x**13**,x) => false	d(x**56**,x) => false	d(x**52**,x) => false
	d(x**62**,x) => false	d(x**63**,x) => false	d(x**43**,x) => false	d(x**41**,x) => false	d(x**22**,x) => false	d(x**36**,x) => false	
	Answer: **0** (True): **27** (False) => false [non-defective]						
Testing 5	d(x**46**,x) => false	d(x**29**,x) => false	d(x**32**,x) => false	d(x**20**,x) => false	d(x**35**,x) => false	d(x**61**,x) => false	d(x**36**,x) => false
	d(x**21**,x) => false	d(x**42**,x) => false	d(x**45**,x) => false	d(x**15**,x) => false	d(x**40**,x) => false	d(x**6**,x) => false	d(x**14**,x) => false
	d(x**34**,x) => false	d(x**19**,x) => false	d(x**38**,x) => false	d(x**12**,x) => false	d(x**47**,x) => false	d(x**50**,x) => false	d(x**39**,x) => false
	d(x**7**,x) => false	d(x**49**,x) => false	d(x**10**,x) => false	d(x**37**,x) => false	d(x**57**,x) => false	d(x**48**,x) => false	
	Answer: **0** (True): **27** (False) => false [non-defective]						

Figure 4.24 Final Result for k-NN Method (Next Selection)

Figure 4.25 represents performance evaluation results for both methods and then compares the evaluation results by selecting several calculated k values. If user wants to upload new dataset file, user clicks home button to go to the home page of the system and file open button to upload another dataset files.


"Comparison of Classification methods on Software Defect datasets"
[PC1_100]
Home
Choose Algorithm
kvalue=16
kvalue=27
[34]

Expected and Actual Results (k = 27)

-	Testing 1	Testing 2	Testing 3	Testing 4	Testing 5	Testing 6	Testing 7	Testing 8	Testing 9	Testing 10
Testing	false	false	false	false	false	false	false	false	false	false
kNN	false	false	false	false	false	false	false	false	false	false
CBW kNN	false	false	false	false	false	false	false	false	false	false

Performance Evaluation Results

Method	Accuracy (show equation)	Precision (show equation)	Recall (show equation)	F-measure (show equation)	MAE	RMSE
kNN Algorithm	97.06 %	100 %	90.91 %	95.24 %	0.06	0.17
CBW-kNN Algorithm	97.06 %	100 %	90.91 %	95.24 %	0.06	0.17

Figure 4.25 Performance Evaluation Result for Two k Values

4.6 Experimental Results

For experimental purpose, to demonstrate the comparison of two methods by using various thresholds (k) values. The experimental results are shown in Figure 4.21 and Figure 4.25. We have tested the k values 16 and 27. It measures the system performance by using Accuracy, F-measure, Precision, Recall, MAE and RMSE to compare output results with other methodologies on the same datasets.

Accuracy is also known as correct classification rate. Accuracy can be defined as the total number of correctly identified defects (true positive and true negative) divided by the total number of defects or non-defects [1], [9]. It is usually expressed as a percentage.

TP - a query instance is defective and classify as defective
FN - a query instance is defective and classify as non-defective
TN - a query instance is non-defective and classify as non-defective
FP - a query instance is non-defective and classify as defective

$$Accuracy = \frac{(TP+TN)}{TP+TN+FP+FN} * 100\% \tag{4.1}$$

Precision is also known as correctness. It is defined as the ratio of the number of modules correctly predicted as defective to the total number of modules predicted as defective. It is usually expressed as a percentage.

$$\Pr ecision = \frac{TP}{TP+FP} * 100\% \tag{4.2}$$

Recall is also known as defect detection rate. It is defined as the ratio of the number of modules correctly predicted as defective to the total number of modules that are actually defective. It is usually expressed as a percentage.

$$\mathrm{Re}call = \frac{TP}{TP+FN} * 100\% \tag{4.3}$$

The higher the Precision, the less effort wasted in testing and inspection. The higher the Recall, the fewer defective modules go undetected. F-measure combines precision and recall in a single efficiency measure by taking their harmonic mean. It is usually expressed as a percentage.

$$F - measure = \frac{2 * \Pr ecision * \operatorname{Re} call}{\Pr ecision + \operatorname{Re} call} * 100\% \tag{4.4}$$

MAE and RMSE can be used together to diagnose the variation in the errors in a set of test samples. MAE is the average over the verification sample of the absolute values of the differences between forecast (predict) and corresponding observed (actual) value. RMSE is calculated square of the difference between forecast and corresponding value and then averaged over the sample. Therefore, RMSE gives a relatively high weight to large errors more than MAE.

$$MAE = \frac{\sum_{i=1}^{m} \left| \chi_i - \overline{\chi} \right|}{m} \tag{4.5}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{m} \left(\chi_i - \overline{\chi} \right)^2}{m}} \tag{4.6}$$

Where i =1, 2, ….. , m number of test samples

x_i = value of ith test sample, $\overline{\chi}$ = mean value of x_i

Calculation of Performance Evaluation

The result of performance evaluation is different k values (16 and 27) as shown in Figure 4.21 and Figure 4.25. For k value (16), result of k-NN method is more accurate than result of CBW k-NN with BINER Algorithm. But result of k-NN method is the same as result of CBW k-NN with BINER Algorithm in k value (27). This is step by step detail calculation of all performance methods for k-NN and CBW k-NN method.

For **k** = **16, k-NN Method**

TP = 10 *FN = 1* *TN = 23* *FP = 0*

$$Accuracy = \frac{(TP + TN)}{TP + TN + FP + FN} * 100\% = 97.06\%$$

$$\Pr ecision = \frac{TP}{TP + FP} * 100\% = 100\%$$

$$\operatorname{Re} call = \frac{TP}{TP + FN} * 100\% = 90.91\%$$

$$F-measure=\frac{2\times \Pr ecision\times \operatorname{Re} call}{\Pr ecision+\operatorname{Re} call}*100\%=95.24\%$$

For **k=16** in k-NN method, $MAE=\frac{\sum_{i=1}^{m}\left|\chi_i-\overline{\chi}\right|}{m}$

Testing No. (i)	χ_i	$\left|\chi_i-\overline{\chi}\right|$ $(\Delta\chi_i)$
1	1	0.0294
2	1	0.0294
3	1	0.0294
……………..		
18	0	0.9706
……………..		
33	1	0.0294
34	1	0.0294

$$\overline{x}=\frac{\chi_1+\chi_2+......+\chi_m}{m}=0.9706$$

$$MAE=\overline{\Delta x}=\frac{\Delta\chi_1+\Delta\chi_2+......+\Delta\chi_m}{m}=0.06$$

The RMSE calculate as the above MAE calculation way according to RMSE formula.

For **k=16** in k-NN method,

$$RMSE=\overline{\Delta x}=\sqrt{\frac{\Delta\chi_1+\Delta\chi_2+......+\Delta\chi_m}{m}}=0.17$$

For **k** = **16, CBW k-NN with BINER Algorithm**

TP = 10 *FN = 1* *TN = 22* *FP = 1*

$$Accuracy=\frac{(TP+TN)}{TP+TN+FP+FN}*100\%=94.12\%$$

$$\Pr ecision=\frac{TP}{TP+FP}*100\%=90.91\%$$

$$\operatorname{Re} call=\frac{TP}{TP+FN}*100\%=90.91\%$$

$$F-measure = \frac{2 \times \Pr ecision \times \operatorname{Re} call}{\Pr ecision + \operatorname{Re} call} * 100\% = 90.91\%$$

For **k=16** in CBW k-NN with BINER Algorithm, $MAE = \frac{\sum_{i=1}^{m} \left| \chi_i - \overline{\chi} \right|}{m}$

Testing No. (i)	χ_i	$\left| \chi_i - \overline{\chi} \right|$ $(\Delta\chi_i)$
1	1	0.0588
2	1	0.0588
3	1	0.0588
……………..		
18	0	0.9412
19	0	0.9412
……………..		
33	1	0.0588
34	1	0.0588

$$\overline{x} = \frac{\chi_1 + \chi_2 + \ldots\ldots + \chi_m}{m} = 0.9412$$

$$MAE = \overline{\Delta x} = \frac{\Delta\chi_1 + \Delta\chi_2 + \ldots\ldots + \Delta\chi_m}{m} = 0.11$$

The RMSE calculate as the above MAE calculation way according to RMSE formula.

For **k=16** in CBW k-NN with BINER Algorithm,

$$RMSE = \overline{\Delta x} = \sqrt{\frac{\Delta\chi_1 + \Delta\chi_2 + \ldots\ldots + \Delta\chi_m}{m}} = 0.24$$

For **k** = **27, k-NN Method and CBW k-NN with BINER Algorithm**

TP = 10 *FN = 1* *TN = 23* *FP = 0*

$$Accuracy = \frac{(TP + TN)}{TP + TN + FP + FN} * 100\% = 97.06\%$$

$$\Pr ecision = \frac{TP}{TP + FP} * 100\% = 100\%$$

$$\mathrm{Re}call = \frac{TP}{TP+FN} * 100\% = 90.91\%$$

$$F - measure = \frac{2 \times \mathrm{Pr}ecision \times \mathrm{Re}call}{\mathrm{Pr}ecision + \mathrm{Re}call} * 100\% = 95.24\%$$

For **k=27** in k-NN method, $MAE = \frac{\sum_{i=1}^{m} \left| \chi_i - \overline{\chi} \right|}{m}$

Testing No. (i)	χ_i	$\left\lvert \chi_i - \overline{\chi} \right\rvert$ $(\Delta\chi_i)$
1	1	0.0294
2	1	0.0294
3	1	0.0294
……………..		
18	0	0.9706
……………..		
33	1	0.0294
34	1	0.0294

$$\overline{x} = \frac{\chi_1 + \chi_2 + ...+ \chi_m}{m} = 0.9706$$

$$MAE = \overline{\Delta x} = \frac{\Delta\chi_1 + \Delta\chi_2 + ...+ \Delta\chi_m}{m} = 0.06$$

The RMSE calculate as the above MAE calculation way according to RMSE formula.

For **k=27** in k-NN method,

$$RMSE = \overline{\Delta x} = \sqrt{\frac{\Delta\chi_1 + \Delta\chi_2 + ...+ \Delta\chi_m}{m}} = 0.17$$

For other testing result, it uses three datasets that are PC1, CM1 and JM1. It measures the system performance by using various kind of k value. For PC1 dataset, different threshold k values are 25, 32 and 50. For CM1 dataset, different threshold k values are 16, 25 and 38. For JM1dataset, different threshold k values are 27, 42 and 65. The performance range for evaluation methods is 0% to 100%. For three datasets, the threshold k value range is more increase, the accuracy and reliability is higher except MAE and RMSE. For demonstration purpose, the evaluation results have shown in Figure 4.26, 4.27 and 4.28.

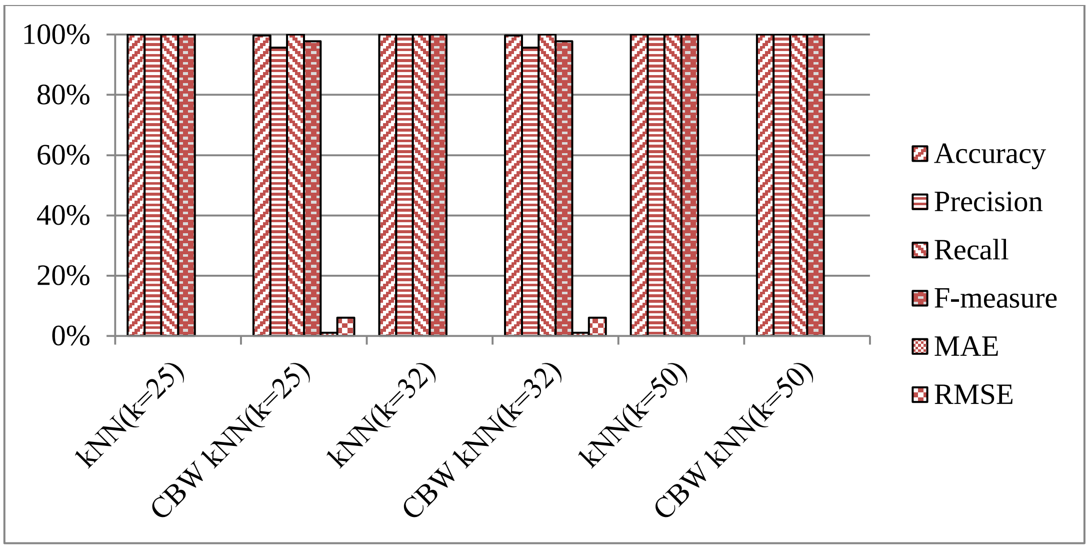


Figure 4.26 Performance Evaluation Result for PC1

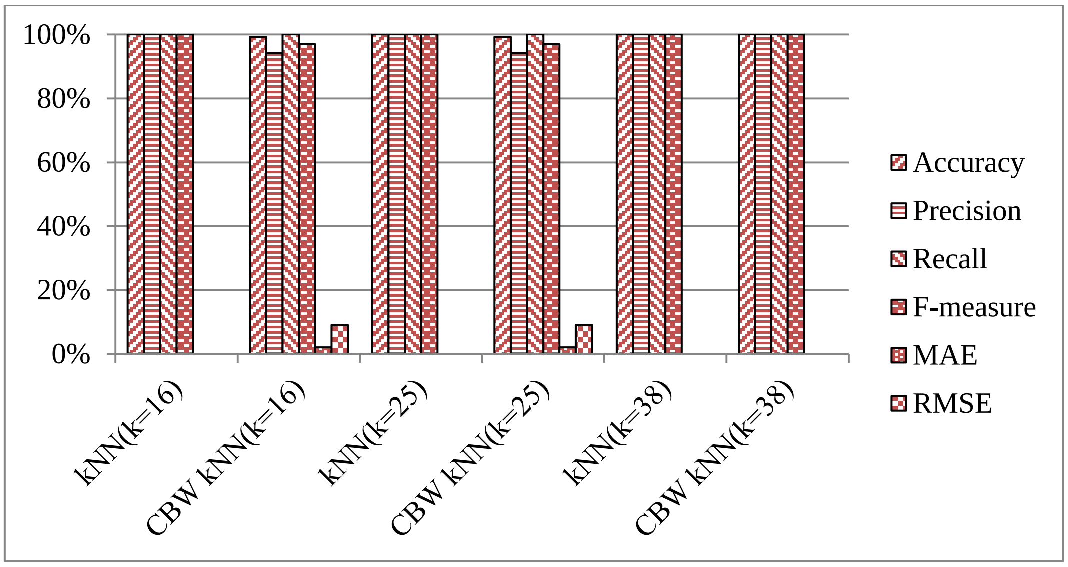


Figure 4.27 Performance Evaluation Result for CM1

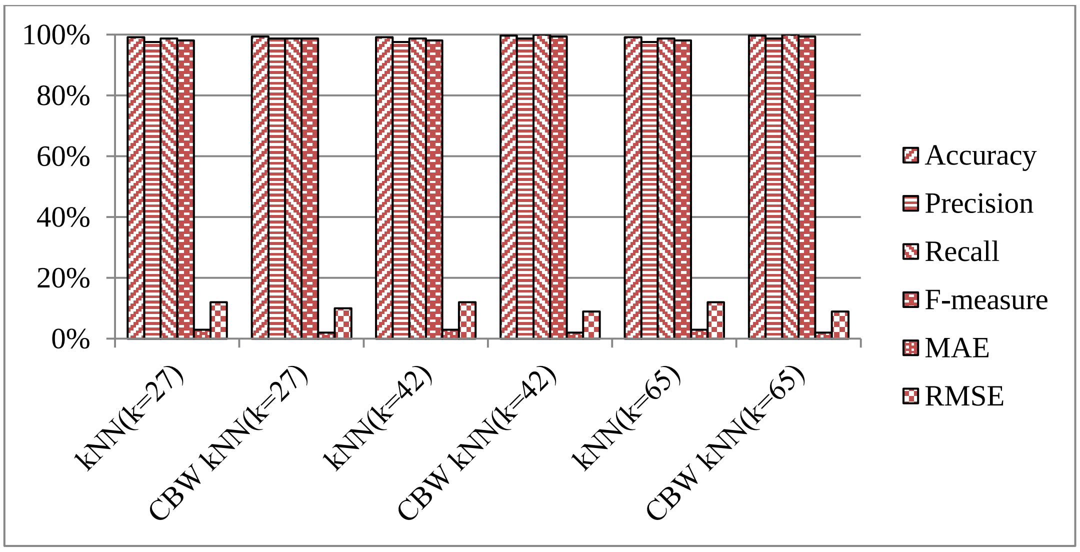


Figure 4.28 Performance Evaluation Result for JM1

CHAPTER 5
CONCLUSION AND FURTHER EXTENSION

The aim of the system is to apply the algorithm of Nearest Neighbors especially k Nearest Neighbor and Class Based Weighted k-NN in software defect detection. To analyze the performance of these algorithms, it is needed to apply some software defect datasets for this detection. After the system has been developed, the more accurate and effective algorithm has appeared according to the evaluation of their performance.

In this implemented system, there are three software defect datasets such as PC1, CM1 and JM1 from NASA MDP datasets. Classification is the most researched topic of machine learning that is applied in the system for the purpose of software defect detection. For selecting an appropriate classifier to test software defective on imbalance data set, this system describes the classification algorithm based on k-nearest neighbors. This system also make available classification of software defective and non-defective using weighted k-nearest neighbor algorithm with class-based weighted factor which is focused on the range of nearest data set using BINER function. One of the algorithms is used to measure the performance result such as accuracy of the test result. It calculates the accuracy, reliability, error-rate (MAE, RMSE) to compare two classification methods. As the accuracy results, the k Nearest Neighbor (k-NN) gives the accurate classification when compared to Class Based Weighted k Nearest Neighbor (CBW k-NN) algorithm. During training of the system, the more the number of threshold value (k) is increased, the more accurate two approaches can be. This system has shown that data mining techniques can be used efficiently to model and classify the class value such as defect or non-defect. The outcome of this system can be used as an assistant tool to provide comparison result of classification methods on imbalanced software defect datasets. The higher threshold value (*k*) is increased, the more accurate in Class Based Weighted k Nearest Neighbor with BINER algorithm.

5.1 Advantages of the System

This system demonstrates effective use of classification methods on software defect dataset. It describes the nature of the software defect datasets and applies k-NN method, CBW k-NN method, and BINER algorithm. The user can test k-NN and CBW k-NN with BINER algorithm for desired software defect dataset. This system calculates performance evaluation results based on multiple threshold (k) values of k-NN and CBW k-NN and compares different performance results according to multiple threshold values. This system supports software engineering groups and software industries to calculate defect rate for existing program.

5.2 Limitation of the System

The implemented system is processed in the only two algorithms such as k-NN and CBW k-NN. It has used only software defect datasets from NASA MDP datasets that are detected C program and had same attributes as PC1, CM1 and JM1. The accuracy result of these algorithms can be varied any other software defect datasets by these may be slightly changed. This system classifies existing testing data only, not classifies new record data.

5.3 Further Extension

The implemented system can test two classification methods such as k-NN and CBW k-NN methods. To improve the classification accuracy of the models, further studies should be conducted using different classification algorithms. And this system can be extended to test any program as defect or non-defect in that program. And, this system can also be extended to generate results for comparison of classification methods on any other datasets. Furthermore, this system should be extended for the classification of the software defect datasets which includes unknown class values for defect/ non-defect by using other classification methods.

REFERENCES

[1] Aleem S, Capretz L F and Ahmed F, “Benchmarking Machine Learning Techniques for Software Defect Detection”, International Journal of Software Engineering & Applications (IJSEA), Vol.6, No.3, May 2015.

[2] Anu K P and BinuRajan, “A Novel Approach for Improving Software Quality Prediction”, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-4 Issue-6, August 2015.

[3] Azeem N and Usmani S, “Analysis of Data Mining Based Software Defect Prediction Techniques”, Global Journal of Computer Science and Technology, Volume 11 Issue 16 Version 1.0 September 2011.

[4] Bhatia N, “Survey of Nearest Neighbor Techniques”, International Journal of Computer Science and Information Security (IJCSIS), Vol. 8, No. 2, 2010.

[5] Carr J, “An Introduction to Genetic Algorithms”, International Journal of Computer Science and Information Security (IJCSIS), May, 2014.

[6] Chadha P and Singh G N, “Classification Rules and Genetic Algorithm in Data Mining”, Global Journal of Computer Science and Technology Software & Data Engineering, Volume 12 Issue 15 Version 1.0 Year 2012.

[7] Dubey H, “Efficient and Accurate kNN based Classification and Regression” CENTER FOR DATA ENGINEERING, International Institute of Information Technology, Hyderabad - 500 032, INDIA, March 2013.

[8] Dubey H and Pudi V, “BINER: Binary Search Based Efficient Regression”, Machine Learning and Data Mining in Pattern Recognition (MLDM): 76-85, Report No: IIIT/TR/2012/-1, July 2012.

[9] Elish K O and Elish M O, “Predicting defect-prone software modules using support vector machines”, The Journal of Systems and Software 81 (2008) 649–660.

[10] Entezari-Maleki R, Minaei B and Rezaei A, “Comparison of Classification methods based on the Type of Attributes and Sample Size”, Journal of Convergence Information Technology · September 2009.

[11] Falangis K, “Mathematical Programming models for classification problems with applications to credit scoring”, Thesis Presented for the degree of Doctor of Philosophy, The University of Edinburgh, 2013.

[12] Gou J, Du L, Zhang Y and Xiong T, “A New Distance-weighted k-nearest Neighbor Classifier”, Journal of Information & Computational Science 9: 6 (2012) 1429–1436.

[13] Gray D, Bowes D, Davey N, Sun Y and Christianson B, “The Misuse of the NASA Metrics Data Program Data Sets for Automated Software Defect Prediction”.

[14] Hamid B, ur-Rehman I, Rauf A, Khan T A, “Using Smote for Convalescing Software Defect Prediction”, Journal of Applied Environmental and Biological Sciences, ISSN: 2090-4274, J. Appl. Environ. Biol. Sci., 5(6)53-59, 2015.

[15] Hechenbichler K and Schliep K, “Weighted k-Nearest-Neighbor Techniques and Ordinal Classification”, Sonderforschungsbereich 386, Paper 399, October 2004.

[16] Kiang M Y, “A Comparative Assessment of Classification Methods”, Decision Support Systems 35 (2003) 441–454, 1 May 2002.

[17] Kilany and Rania M., "Efficient Classification and Prediction Algorithms for Biomedical Information" (2013). Doctoral Dissertations. 105. http://digitalcommons.uconn.edu/dissertations/105.

[18] Miclea L, Nedevschi S, Petrescu M, Cretu V and Potolea R, “Strategies for Dealing with Real World Classification Problems”, Faculty of Computer Science and Automation, Eng. Camelia Lemnaru (Vidrighin Bratu).

[19] Robu R and Holban S, “A Genetic Algorithm for Classification”, Recent Researches in Computers and Computing, ISBN: 978-1-61804-000-8.

[20] Singh A K, Goel R and Kumar P, “Comparative Analysis of Accuracy Prediction using Fuzzy C-Means and KNN Clasiffier”, IJDACR, International Journal of Digital Application & Contemporary research, ISSN: 2319-4863, Volume 2, Issue 7, February 2014.

[21] Singh Y and Chauhan A S, “Neural Networks in Data Mining”, Journal of Theoretical and Applied Information Technology, 2005-2009.

[22] Sudhakar M and Reddy C V K, “Application Areas of Data Mining in Indian Retail Banking Sector”, Global Journal of Computer Science and Technology: C Software & Data Engineering, Volume 14 Issue 5 Version 1.0 Year 2014.

[23] Wahono R S and Herman N S, “Genetic Feature Selection for Software Defect Prediction”, American Scientific Publishers, Advanced Science Letters, Vol. 20, 239–244, 2014.

[24] Zhao M and Chen J, “Improvement and Comparison of Weighted k Nearest Neighbors Classifiers for Model Selection”, Journal of Software Engineering, 2016, ISSN 1819-4311 / DOI: 10.3923/jse.2016.

[25] Witten I H and Frank E, “Practical Machine Learning Tools and Techniques”, Second Edition

AUTHOR'S PUBLICATION

[1] Hnin Yi San, Dr. Khine Khine Oo, "*The Comparison of Classification Methods on Software Defect Data Sets*", the Proceedings of the 9th Conference on Parallel and Soft Computing (PSC 2018), Yangon, Myanmar, 2018.