Software Defect Prediction Analysis Using Machine Learning Algorithms

Praman Deep Singh
University School of Information and Communication Technology,
GGS IP University, Dwarka, New Delhi-110078
praman.deep@outlook.com

Anuradha Chug
University School of Information and Communication Technology,
GGS IP University, Dwarka, New Delhi-110078
anuradha@ipu.ac.in

***Abstract*—Software Quality is the most important aspect of a software. Software Defect Prediction can directly affect quality and has achieved significant popularity in last few years. Defective software modules have a massive impact over software's quality leading to cost overruns, delayed timelines and much higher maintenance costs. In this paper we have analyzed the most popular and widely used Machine Learning algorithms – ANN (Artificial Neural Network), PSO (Particle Swarm Optimization), DT (Decision Trees), NB (Naïve Bayes) and LC (Linear classifier). The five algorithms were analyzed using KEEL tool and validated using k-fold cross validation technique. Datasets used in this research were obtained from open source NASA Promise dataset repository. Seven datasets were selected for defect prediction analysis. Classification was performed on these 7 datasets and validated using 10 fold cross validation. The results demonstrated the dominance of Linear Classifier over other algorithms in terms of defect prediction accuracy.**

Keywords—Software Quality, Defect Prediction, Machine Learning, KEEL, NASA Promise dataset

I. INTRODUCTION

Software Quality is the most important aspect of a software. It is the degree of accordance to both explicitly defined and implied requirements or expectations of a customer in a software program. It has a huge impact over business as it can glorify or ruin the brand image of a company. If not handled properly, it can lead to cost overruns, delayed timelines and much higher maintenance costs [1]. If fault occurs after the software goes "live" i.e., after the product is released to market, entire faulty module has to be re-examined for bug(s) and the code is altered to fix them as part of maintenance. Then the ripple effect is taken care in the form of Regression Testing.

As per International Software Testing Qualifications Board (ISTQB), Quality is "The degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations". It defines Software Quality as "The totality of functionality and features of a software product that bear on its ability to satisfy stated or implied needs". [Standard glossary of terms used in Software Testing, ISTQB V 2.2].

Software Quality is inversely proportional to the density of defects.As per ISTQB, Defect is "A flaw in a component or system that can cause the component or system to fail to perform its required function".If it is encountered during execution of the program, it may cause failure of the component or even thewhole system.Defects are nightmares for deemed organizations.They dent the reputation of the organization leading to customer dissatisfaction which further effects the market hold of the company. Neither any software can be 100% defect-free nor 100% testing can be achieved but a good quality software has much less number of defects and is more reliable [12].

Many real-world problems, often highly complex in nature, need to be handled with "customized" algorithms according to their need.Inventing specialized algorithms to handle such problems every time is unrealistic, if not impossible.Machine Learning Algorithms make it possible to handle such problems in a customized manner.As per the definition of Machine Learning given by Tom Mitchell of Carnegie Mellon University, "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E".There are two types of machine learning algorithms, Supervised machine learning algorithms and Unsupervised machine learning algorithms.

Supervised machine learning algorithms are fed with pre-defined set of training data. The algorithms then gains "experience" from training dataset and generates rules to predict the class label for new set of data."Learning" phase consists of using mathematical algorithms to produce and improve the predictor function.The training data used in this phase have an input value of an attribute and its known output value.The predicted value of ML algorithm is compared with already known output.It is then "modified" to make it accurate with respect to the prediction output.This is repeated in several "generations" of training data until a threshold prediction accuracy is obtained or maximum number of cycles are performed.

In case of unsupervised machine learning algorithms, the output value of the class label in data is not known.Instead, the program is loaded with a cluster of data and the algorithm finds patterns and relationships within it. The main focus is on relationships amongst the attributes within data.An example would be identifying a circle of friends in a social networking website.

978-1-5090-3519-9/17/$31.00 ©2017 IEEE

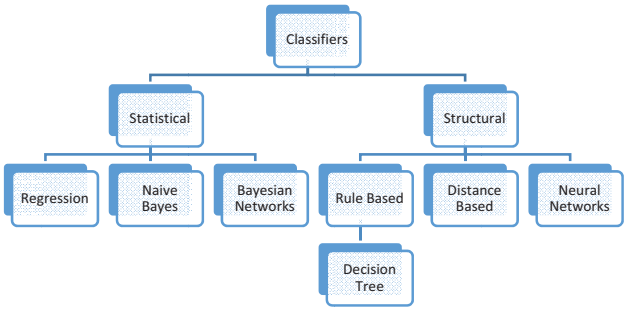

Fig 1: Classification of Machine Learning algorithms[19]

Software Quality can be improved by forecasting the faults modules. Software Defect Prediction is the process of developing models which are used in the early phases of SDLC for detecting faulty units such as modules or classes.It can be done by classifying the modules as defect prone or not [18].The classification of the modules is done by various methods, most popular ones are Decision Trees (DT), Naïve Bayesian (NB), Neural Networks (ANN), Support Vector Machines (SVM) and Evolutionary Algorithms (EA).The identified defect prone modules are prioritized higher in testing phase of SDLC and the non-defect prone modules are tested as the time and cost allows.The classification function, known as the classifier, analyzes the relationship between the attributes and the class label of the training dataset and forms classification rules. These rules are then used in identifying the class labels of future datasets. Thus, we can classify the unknown datasets with the help of classification rules and a classifier.

Research questions which we aim to answer in this paper:

RQ1: Which techniques and algorithms have been used in Software Defect Prediction (SDP)?

RQ2: Which common metrics are used in SDP studies?

RQ3: Which data sets are widely used in SDP studies?

RQ4: Which method gives highest accuracy amongst the selected methods?

II. Related Works

Song, Zia, Shepperd, Ying and Liu [3] discussed about a general defect prediction framework which will use learning algorithms together with historical data. They classified defect proneness of software components in two classes, defect-prone and not defect-prone. It takes into account learning algorithms, which learn from past data and form classification rules by themselves. The proposed framework comprises of two components, scheme evaluation and defect prediction. During scheme evaluation stage, different learning schemes are assessed with historical data and their performance is evaluated. Historical data is broken down into parts, training dataset and testing dataset. Training data is used to build learning algorithm's classification rules and testing data is used to assess the prediction accuracy of learning algorithms. During defect prediction stage, a learning scheme is selected on the basis of prediction accuracy report obtained in first stage. The selected scheme is then used to build a prediction model and predict defect proneness in future datasets. The authors have used publicly available datasets from NASA PROMISE repository (http://promise.site.uottawa.ca/SERepository). 17 datasets were used in total, 13 from MDP and 4 from PROMISE repository. Authors concluded that no learning scheme is dominant, i.e., always better than other schemes in all datasets which means different schemes should be chosen for different datasets depending upon the problem under consideration.

Jureczko [5] examined the importance of different product metrics used in the prediction model. In this paper, the author has performed correlation analysis and inspected the relation between different metrics and the defect count. 36 releases of 6 proprietary projects were used to collect these metrics. Ckjm tool was used to calculate the product metrics while Buginfo tool was used in process metric collection. Three types of analysis were done on the data namely correlation analysis, discriminant analysis and metric occurrence in defect prediction model. All the process metrics were found to be correlated with defect count. NR, NDC, NML, NDPV, LOC, RFC, CBO and AMC metrics were concluded as the most significant in defect prediction but other metrics were not neglected. Hence, adding more software metrics may contribute to improvement in defect prediction efficiency.

Bavisi, Mehta and Lopes [10] presented various data mining algorithms which are most widely used these days. They have compared four algorithms, namely k-Nearest Neighbors, Naïve Bayes classifier, C-4.5 and Decision trees. Authors have compared advantages and disadvantages of each algorithm along with their applications. The authors opposed the Conservation law, according to which any single learning algorithm cannot perform better than a different learning algorithm when both are compared according to the performance measure.They instead concluded that the accuracy of an algorithm depends on various factors such as the nature of problem, the dataset used in experiment and its performance matrix. They also stated that not all concepts are equiprobable for a given domain and the performance of any algorithm differ from domain to domain.

Prasad, Florence and Arya [14] have focused on software quality metrics which help in identifying software defects using data mining techniques to improve the overall quality of software. Authors have reviewed various classification techniques used in software defect prediction. Various product metrics were studied such as Chidamber and Kemerer metrics, cohesion and coupling, QMOOD metrics, Class level metrics by McCabe, lines of code (LOC) etc. This study also included various software defect prediction techniques such as Regression, Association Rule Mining, Clustering and Classification. Different algorithms which were analyzed included Neural Networks, Decision Trees, Naive Bayes, Support Vector Machines and Case Based Reasoning. Four different classification techniques were studied, compared and analyzed which were Supervised learning, Semi-supervised learning, Unsupervised learning and Machine learning.

Rathore and Gupta [4] investigated the relationship between object-oriented metrics and defect proneness in an

object-oriented software system. Four publicly available PROMISE datasets and four machine learning algorithms were used in this study. They have used eighteen different metrics and divided them under cohesion, coupling, complexity, inheritance and size categories of an object oriented system to be used for defect prediction of a given class. They also combined the design attributes to achieve higher levels of correctness. Univariate logistic regression (ULR), Spearman's correlation and AUC (Area under ROC curve) were applied solitary in this analysis. In the next step, these techniques were applied in combination to build multivariate defect prediction model. In their conclusion, authors recommended to use coupling and complexity metrics in defect prediction to build the model which will lead to lower levels of defects.

Ratzinger, Sigmund and Gall [7] analyzed the impact of evolution activities like refactoring on defects present in the software. In this paper, authors have used versioning and issue tracking systems, thereby extracting and labelling Data Mining features as refactoring or non-refactoring related. They concluded that the more the refactoring done during software development, the less would be the chances of software defects. Hence, refactoring is an essential part of bug fixing as well as other evolutionary activities to reduce the software defects to minimum.

Fenton and Neil [8] recommended universal models for predicting defect which used Bayesian Belief Networks. Authors used Bayesian networks as an alternative approach to single-issue models which are being used quite moderately nowadays. They further reasoned the area of research in software decomposition for testing the hypotheses related to defect introduction.

Sandhu, Brar and Goel [9] suggested that if the fault is predicted early in software development life cycle (SDLC), it is possible to get an improved software process control along with high software reliability. In this paper, authors have used Decision tree based model with amalgamation of K-means clustering approach to predict faults in SDLC. Both early SDLC metrics (requirement metrics) and later SDLC metrics (code metrics) were used in this approach. They used CM1 defect dataset from NASA repository to test the model. The results exhibited high accuracy in predicting the fault tendency of software modules early in SDLC.

Adebowale, Idowu and Amarachi A [15] have focused their research on intrusion detection. In this paper, they have selected five data mining algorithms, namely Decision trees, Naïve Bayes, Artificial neural network, K-nearest neighbor and Support vector machines. Their advantages, disadvantages and applications have been discussed thoroughly with respect to intrusion detection. The impact of data mining techniques over intrusion detection was explored. NSL-KDD dataset was used in experiment performed on Weka (Waikato Environment for Knowledge Analysis) tool. Each algorithm was trained on KDD dataset using Weka tool to assess its effectiveness. Testing of algorithms was done using 10 fold cross validation technique in which dataset is divided into 10 subsets. Decision tree algorithm provided the best fault detection rate but fastest build time was achieved by K-nearest neighbor algorithm. In the end, it was concluded to have a combined approach in which multiple algorithms will overcome the disadvantages of each other.

Malhotra and Singh [16] have presented the relationship of fault proneness and object oriented metrics of a class. To predict the fault LR techniques, seven machine learning algorithms ANN, RF, LB, AR, NB, Kstar, Bagging and one Logistic regression method was used. This model helps the testers to focus on the faulty parts of software. It will support the testers and researchers in forecasting the faulty classes of software in early stages. The developers can reexamine the software design and therefore take required corrective actions. Among all the above LB is the best technique in terms of area under cover. Object oriented matrices helps the practitioners and testers in predicting the faulty free software a reasonable cost. The results depends upon open source software. The results shows the projecting accuracy of machine learning technique logisticboost is maximum.

Goel and Dewan [11] have performed a comprehensive review of various machine learning approaches of software defect prediction. Experts focus on the areas that are prone to deficiency and thus, there motive is to forecast the fault in such a manner that they are characterized into fault modules and non- fault modules as soon as possible in software development to avoid the risk, cost and time in later stages. In case of large scale projects with high complexity and multiple releases, it plays an important role. Software systems have a vital role in mission critical applications. There is much reliance on their fault-free functioning in such cases. In software development, the application of software quality modules in early stages give an efficient defect removal technique and makes the results more reliable. In this paper, authors built a tool which can be used to predict fault prone modules in a software system automatically.

Kumari and Rajnish [17] presented a metric to capture software complexity aspects through grouping of characteristics of a class. The relationship among fault-proneness and object oriented software metrics were explored at the class level. The proposed metric, CLCM (Class Level Complexity Metric) is based on a class abstraction. Effectiveness of software metric was used to find error categories of classes using well known software Eclipse on its three versions. Logistic regression models were used to calculate fault finding ability of both binary and multinomial object oriented software metrics. Authors concluded that the proposed CLCM metric showed most efficient results in case of multinomial object oriented metrics and was the best in predicting fault proneness.

III. RESEARCH METHODOLOGY

Various algorithms were studied for doing the comparative analysis research. The selection was made such that it covers most popular and widely used Machine Learning algorithms. Following are the selected algorithms along with their brief description:

A. Neural Network

A neural network, also termed as "Artificial Neural Network" (ANN) is a mathematical and computational model with its base on biological neural networks. It comprises of an

artificial group of neurons interconnected and processing information as a unit. It is an adaptive system which changes its structure according to the information flow, whether internal or external, during the learning phase. It is an imitation of the biological neural system.

The two main components of an ANN are Nodes (processing units) and Links (connection between nodes). They are conferred as the system with interconnection between the neurons which can exchange messages to communicate with each other. The working of an ANN is explained as follows. Firstly, the Neural Network takes the variable values as input at the input layer nodes. Weights are assigned on the links which connect the nodes.These numeric weights are tuned based on experience which makes these neural network capable to learn and adapt. Nodes are traversed and the variable values are calculated going through the network. The weight of each link effects the outcome of variable value. At the output node, the variable value is compared with a threshold value and the outcome is predicted.

B. Particle Swarm Optimization

Particle Swarm Optimization (PSO) algorithm [6] is based on the behavior of a flock of birds. Dr. Eberhart and Dr. Kennedy developed this algorithm in 1995. In this algorithm, an optimized solution to the issue is found by improving the quality of candidate solution which is an initial solution. Each and every candidate in the algorithm is known as particle. These particles traverse search space according to a formula which is based on the displacement and speed of the particle. Another parameter on which traversal depends is best positions in search space. PSO is capable of searching solution (close to optimal) in huge search spaces. There is no assurance that the solution could be founded using this algorithm.

This algorithm have various applications in different fields like artificial neural network, fuzzy controllers and optimization problems. To have better understanding about PSO lets take an example. Suppose a group of friends are searching to get out of a forest and there is only one way out. They have no idea about the route to exit but they know only one parameter that is relative distance. Every individual in this problem can be considered as a particle and therefore these particles would be assigned a particular position and speed. According to PSO the speed of every particle depends on the relative position towards the point of exit.

There are some parameters in PSO which can be adjusted which makes it even more attractive. Slight variation in one version works well with variety of applications. Particle Swarm Optimization is used in different fields where applications are very wide and also specific applications which focus on specific requirement.

C. Decision Trees

Decision Trees [20], in comparison to neural networks, represent only rules. The rules formed in this approach are easy to comprehend for the human brain. They can also be directly utilized in database languages like SQL to retrieve records falling into a specific class. Decision tree permits calculations to be done in forward and reverse manner which enhance decision correctness. Decision tree has a tree structure where every node is either a Leaf node or a Decision node. Leaf nodes display the value of the attribute assigned to them whereas decision nodes indicate branching, where a specific test has to be done on attribute with one branch and sub-tree as a result of the test.

This learning technique uses decision tree as a predictive model in which item's target value conclusion is mapped with observation about the item. This is a predictive modelling technique which is used in data mining, statistics and machine learning. A decision tree can be utilized to visually classify the decisionmaking and the decisions.

The goal of this algorithm is to build a structure that is able to predict value of target variable relying on various input variables. Every interior node represents one of the input variables.For each possible value of these variables, there are edges to their children. Leaf in the tree represents target variable value in which the given values of the input variables can be traversed by path from root to the leaf.

D. Naïve Bayes classifier

Naïve Bayes classifier [7] is an administered learning algorithm which is utilized for information grouping utilizing statistical strategy. This is a probabilistic classifier that characterizes particular input over an arrangement of classes utilizing mathematical probability distribution. As the name suggest Naïve this method innocently assumes that attributes of a given class are independent. Grouping is then finished by applying Bayes theorem to calculate the probability of the right class given the specific attributes of a situation.

These come from a simple probabilistic classifier family which is based on the theorem of Bayes. Features have an assumption of strong or naïve independence. It is a simple technique to build classifiers. It acts like a model which is assigned to problem objects as a class labels, assigned as a vector which is used to draw the class labels from finite sets. It is not just an algorithm but algorithm family is worked on principles: naive classifiers assume that feature value is not dependent on any value which has class variables. For instance, apple is a fruit which has red color, round in shape and approximate 10 cm diameter. A naive Bayes considers probability of each feature. For example an apple, regardless of its features of color, correlation, roundness or diameter.

For other probability models, it is trained in a supervised learning background in an efficient manner. In other applications, maximum likelihood method is used as a parameter estimation of the naive Bayes, in other words, anybody can work with it without the acceptance of Bayesian Probability and methods. Naive Bayes are classified very well in the complex situation of real-world problems in spite of their quite simplified design and assumptions.

E. Linear classifier

In the field of machine learning, the objective of measurable order is to utilize an item's qualities to distinguish which class (or group) it belongs to. A Linear classifier accomplishes this by settling on an order choice in view of the estimation of a direct combination of the attributes. An item's attributes are otherwise called highlight values and are normally introduced to the machine in a vector called a feature

vector. These classifiers are used efficiently in the problems like document classification and the problems of variables ; need to reach the level of accuracy compare to non-linear classifier during the time of training which is less.

Linear classifiers belong to a particular class of support vector machines, which are supervised learning models. They contain learning algorithms which help in analyzing the data used in classification and regression analysis. An SVM model is represented as sample points in space which are mapped such that they are separated by a gap as far as possible according to the categories they belong to. The category of new samples is predicted according to the side of the gap they fall on, after being mapped into that same space.

Technically a support vector machine builds a set of hyperplanes in a multi-dimensional space, which is used in regression, classification etc.A hyperplane having the greatest distance to the data point of a specific class is called functional margin. In general, functional margin is inversely proportional to the generalization error. Higher the functional margin, lower is the generalization error.

IV. DATA MINING TOOL

In last few years Evolutionary Algorithm (EA) [19, 20], specifically Genetic Algorithm (GA) [13], has proven to be a vital technique for information extraction and machine learning which makes it a promising tool in the field of Data Mining (DM). The idea of discovering knowledge automatically from the databases makes the task even more appealing and challenging. Consequently, there has been a developing enthusiasm for DM in a few AI-related fields, including EAs. The principle inspiration for applying EAs to data miningactivities is that the search methods are adaptive and robust due to which global search can be performed. The utilization of EAs in solving problem is practiced broadly. Different tasks such as e-learning system improvement, learning of controllers in the field of robotics, image retrieval demonstrate their appropriateness as problem solvers in an extensive variety of scientific fields.

In spite of the fact that EAs are capable for understanding an extensive range of scientific issues, their utilization requires a specific programming skill alongside significant effort and time to compose a PC program for actualizing the regularly complex calculation as per client needs. This work can be monotonous and should be done before clients can begin centering their consideration on the issues that they ought to be truly chipping away at.

In the most recent times, there has been great development in Data Mining tools. Open source tools play an essential part. Examples are ADaM, D2K (with E2K), KNIME, MiningMart, Orange, Tanagra, Weka and RapidMiner. These tools had no or only basic support for Evolutionary Algorithms. Also, they had a representative set of algorithms which they integrated for each type of preprocessing or learning task. Sometimes running EAs is not practical for some situations since it involves high computation times. In some situations, researchers need to compare and analyze the behavior of EAs and non-EAs. We discuss KEEL tool next, which will cater all these requirements of researchers and analysts.

KEEL(Knowledge Extraction based on Evolutionary Learning) is an open source java software tool which stands for knowledge extraction based on evolutionary learning. Using this tool user can access different methods of the techniques likes Soft Computing, evolutionary learning. It is said to be useful in various Data Mining problems such as pattern mining, regression, clustering and many more.

It contains various library files for evolutionary learning algorithms for different prototypes and makes it easier to collaborate different evolutionary learning algorithms with pre-processing techniques. These all reduces the programming work and makes their implementation easier for the programmers so that they can focus on the comparison of the new learning models and the existing models.

Since lots of algorithms are available so it makes it an easy-to-use software as it reduces the knowledge and experience level that researchers needs to possess. This tool can be used to apply successfully into their problems which extends the range of users applying these learning algorithms.

It implements purely object oriented techniques for both software's and the library files so they could be used along with java in all the machines. So all the characteristics of java are present in this tool hence any user can use this tool in their respective machine irrespective of their operating systems.

KEELfacilities the analysis of various properties and behaviors of the learning for different which in turn makes the management techniques easier for all the users. This uses well-implemented models for all the behaviors provided such as instance selection, learning classifier systems and many more.

The functional blocks present in the current version of KEEL (http://www.keel.es/) areData Management, Design of Experiments (offline module) and Educational Experiments (online module).Due to the presence of all these functional blocks KEEL is proved to be useful for different types of users where everyone expects different features in the Data Mining software.

V. DATASET

There are various open source datasets available online. The datasets were obtained from NASA Promise dataset repository. Seven datasets namely CM1, JM1, KC1, KC2, PC1, Datatrieve (abbreviated as AT in this report) and kc1-class-level-numericdefect (abbreviated as KC1 LC) were used.

Below table provides detailed information about each dataset such as its attributes, instances, faulty and non-faulty instances, missing attributes etc.

Dataset	Language used	no of attributes	no of instance	Non-faulty instance	Faulty instance	defective instance	Missing attribute
CM1	C	22	498	449	49	9.83 %	None
JM1	C	22	10885	8779	2106	19.35 %	None
KC1	C++	22	2109	1783	326	15.45 %	None
KC2	C++	22	522	415 105		20.50 %	None
PC1	NA	22	1109	1032	77	6.94 %	None
AT	NA	9	130	119	11	8.46 %	None

KC1 CL	NA	95	145	85	60	44.82 %	NA

Table 1: Characteristics of the dataset used

Below chart depicts the datasets with respect to number of each type of instances i.e., faulty, non-faulty, total and the attributes.

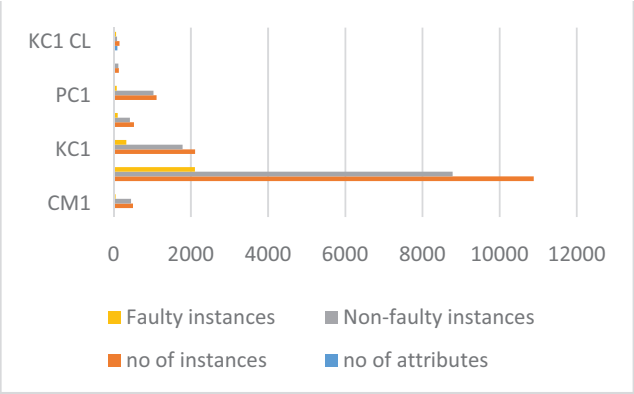

Fig 2: Dataset Characteristics

VI. Results and Analysis

After research and analysis, Keel tool was selected to perform the experiment. The datasets were obtained from NASA Promise dataset repository. Seven datasets namely CM1, JM1, KC1, KC2, PC1, AT and KC1 LC were used. The algorithms selected for the analysis are Particle Swarm (CPSO-C), Naïve Bayesian (NB-C), Decision Tree (TARGET-C), Linear Classifier (MPLCS-C) and Neural Network (ENSEMBLE-C).

The datasets were obtained from NASA repository in the arff format, supported by WEKA tool. Then the Data Management section of KEEL was run to make the datasets compatible with it.Summary of Test results is shown in the following table. It depicts the accuracy rate of each algorithm (percentage-wise). The highest algorithm in a dataset is marked bold to indicate it amongst others.

Algorithms	Datasets						
	CM1	JM1	KC1	KC2	PC1	AT	KC1 CL
Particle Swarm	*88.16 %*	*80.20 %*	*81.27 %*	*73.72 %*	*92.78 %*	*82.30 %*	*59.23 %*
Naïve Bayesian	*82.95 %*	*77.03 %*	*80.55 %*	*81.79 %*	*90.08 %*	*90.76 %*	***73.76 %***
Decision Tree	*90.16 %*	*80.58 %*	***84.96 %***	*80.45 %*	*93.05 %*	*90.00 %*	*69.66 %*
Linear Classifier	*87.95 %*	***80.84 %***	*84.49 %*	***82.73 %***	***93.59 %***	***90.76 %***	*66.80 %*
Neural Network	***90.16 %***	*80.64 %*	*84.54 %*	*79.50 %*	*91.25 %*	*63.84 %*	*65.38 %*

Table 2: Defect prediction accuracy of each algorithm

As it is clearly visible from the results that Linear Classifier algorithm has highest defect prediction accuracy in four of the seven selected datasets, it is the most reliable technique due to its greater accuracy. The rest three algorithms having highest accuracy in one dataset each were Naïve Bayesian, Decision Tree and Neural Network.

The next table shows the summary of standard deviation error in each of the algorithms. It depicts the error rate of each algorithm (percentage-wise). The highest algorithm in a dataset is marked bold to indicate it amongst others.

Algorithms	Datasets						
	CM1	JM1	KC1	KC2	PC1	AT	KC1 CL
Particle Swarm	*0.0240*	*0.0097*	*0.0267*	*0.1033*	*0.0113*	*0.0976*	***0.0281***
Naïve Bayesian	*0.0637*	*0.0112*	*0.0148*	*0.0339*	*0.0208*	*0.0670*	*0.0692*
Decision Tree	*0.0056*	*0.0015*	*0.0079*	*0.0294*	***0.0040***	***0.0492***	*0.0623*
Linear Classifier	*0.0252*	*0.0041*	*0.0134*	*0.0476*	*0.0130*	*0.0670*	*0.0991*
Neural Network	***0.0056***	***0.0003***	***0.0022***	***0.0084***	*0.0550*	*0.1242*	*0.1624*

Table 3: Standard deviation error of each algorithm

This table represents the standard deviation of results from predicted defect. The lowest error rate is of Neural Network algorithm. In case of a tie in two algorithms in terms of defect prediction accuracy, the error rate will help breaking the tie. The lower the error, higher the accuracy.

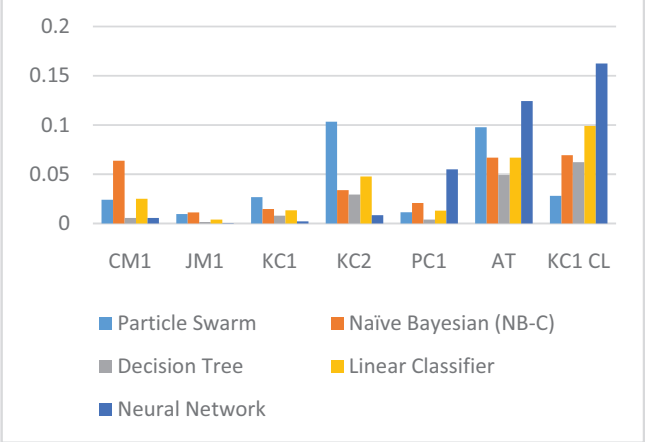

Fig 3: Standard deviation error in algorithms

It is clear from the results that Neural Networks have the lowest error rate in the experiment followed by Decision Trees. However the highest accuracy rate lies with Linear Classifiers. In case of a tie in accuracy prediction, we can use error rate metric to determine which algorithm gives best results.

VII. CONCLUSION AND FUTURE SCOPE

The research questions laid in the beginning of this paper are answered as follows:

RQ1: Which techniques and algorithms have been used in Software Defect Prediction (SDP)?

Most widely used methods in Software Defect Prediction models [2] are Decision trees (DT), Bayesian learners (BL),

Artificial Neural networks (ANN), Support vector machines (SVM), Rule based learning (RBL), Evolutionary algorithms (EA).

RQ2: Which common metrics are used in SDP studies?

Most widely used metrics in Software Defect Prediction [2] are Traditional static code metrics defined by Halst ead and McCabe, Size metrics such as LOC (lines of code) metric, Object-oriented metrics like co hesion, coupling and inheritance, Hybrid metrics used both Object Oriented as well as procedural metrics for defect proneness prediction.

RQ3: Which datasets are widely used in SDP studies?

Most widely used metrics in Software Defect Prediction are:NASA datasets which are publically available in the NASA repository by NASA Metrics Data Programme and are the most commonly used data sets for S DP, PROMISE repository datasets and NASA datasets along with other datasets donated by individuals from research backgrounds,Inbuilt software datasets which are publically available defect datasets inbuilt in the software tools like Eclipse, Weka, and Keel etc.,Open source project datasets w hich involves studies that use other open source projects such as UCI machine learning repository, Ant, Apache etc. amongst others and lastlyEducational institutional datasets which are academic datasets developed and maintained by educational institutions for research purposes such as The University of Edinburgh etc.

RQ4: Which method gives highest accuracy amongst the selected methods?

As per the resul ts obtained in VI, it is evident that the Linear Classifier algorithm has highest defect pre diction accuracy amongst four of the seven selected datasets, hence proving to be the most reliable technique amongst supervised learning algorithms in D ata Mining. The datasets in which it had maximum prediction accuracy were JM1, KC2, PC1 and AT.As per the analysis performed in this paper, Linear Classifiers are the m ost accurate and rel iable amongst defect prediction algorithms. Seven datasets were selected for defect prediction analysis. Classification was performed on these seven datasets and were validated using 10 fold cross validation run on KEELsoftware tool. The results demonstrated the supremacy of Linear Classifier over other algorithms in terms of defect prediction accuracy.

These results can be further refined by using more number of datasets. Increased number of da tasets will strengthen the results. Also, comparison can be done amongst more number of algorithms. Most popular and widespread used alg orithms were taken into account in this research, hopefully new techniques will arise i n future and could be included in the comparative analysis, providing new and improved results.

It is clear from the results that Neural Networks have the lowest error rate in the experiment followed by Decision Trees. However the highest accuracy rate lies with Linear Classifiers. In case of a tie in accuracy prediction, we can use error rate metric to determine which algorithm gives best results.

REFERENCES

[1] A. Chug and S. Dhall, "Software defect prediction using su pervised learning algorithm and unsupervised learning algorithm", Confluence 2013: The Next Generation Information Technology Summit (4th International Conference), pp. 173 – 179, 2013.

[2] R. Malhotra, "A sy stematic review of machine learning techniques for software fault prediction", Applied Soft C omputing, Elsevier Science Publishers B. V. Amsterdam, pp. 504-518,2015.

[3] Q. Song, J. Zia, M. Shepperd S. Ying and J. Liu, "A General Software Defect-Proneness Prediction Framework", IEEE Transactions on Software Engineering, pp. 356 – 370,2010.

[4] S.S. Rathore and A. Gupta, "Investigating Object Oriented Design Metrics to Predict Fault Proneness of Software Modules", Software Engineering (CONSEG), pp. 1 – 10,2012.

[5] M. Jureczko, "Significance of different software metrics in de fect prediction", Software Engineering: An International Journal, pp. 86-95, 2011.

[6] Sfenrianto, I. Purnamasari and R.B. Bahaweres, "Naive Bayes classifier algorithm and Particle Swarm Optimization for classification of cross selling", 4th I nternational Conference on Cyber and IT Service Management, pp. 1-4, 2016.

[7] J.Ratzinger, T.Sigmund and H. C. Gall, "On the relation of re factoring and software defect prediction", In Proceedings of the 2008 international working conference on M ining software repositories ACM, pp. 35-38, 2008.

[8] N. Fenton and M.Neil, "A critique of softw are defect prediction models", Software Engineering IEEE Transactions, Vol. 25, No.5, pp. 675-689, 1999.

[9] P. Sandhu, A. Brar and R. Goel, "A model for early prediction of faults in software systems", In (ICCAE) IEEE, pp. 281-285, 2010.

[10] S. Bavisi, J. Mehta and L. Lopes, "A Comparative Study of D ifferent Data Mining Algorithms", International Journal of Current Engineering and Technology, Vol. 4, pp. 3248-3252, 2014.

[11] S. Goel and K. Dewan, "Comprehensive Review on Fault Prediction in Software Modules by Machine Learning Approaches", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 5, pp. 1686-1691, 2015.

[12] R. Malhotra and Y. Singh, "On the Applicability of Machine Learning Techniques for O bject Oriented Software Fault Prediction", Software Engineering: An International Journal (SEIJ), Vol. 1, pp. 24-37, 2011.

[13] R. Malhotra and A. Gupta, "Study & Analysis For Software Test-Data Generation using G enetic Algorithm for a Use Case", NNGT Int.J. on Software Engineering, Vol. 2, pp. 1-5, 2015.

[14] M. CM Prasad, L. Florence and A. Arya, "A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques", International Journal of Database Theory and Application, Vol. 8, pp.179-190, 2015.

[15] Adebowale, I. S. A and A. Amarachi A, "Comparative Study of Selected Data Mining Algorithms Used For Intrusion Detection", International Journal of S oft Computing and Engineering (IJSCE), Vol. 3, pp. 237-241, 2013.

[16] R. Malhotra and Y. Singh, "On the Applicability of Machine Learning Techniques for O bject Oriented Software Fault Prediction", Software Engineering: An International Journal (SEIJ), Vol. 1, pp. 24-37, 2011.

[17] D. Kumari and K. Rajnish, "Investigating the Effect of Object-oriented Metrics on Fa ult Proneness Using Empirical Analysis", International Journal of So ftware Engineering and Its Applications, Vol. 9, pp. 171-188, 2015.

[18] E.K. Chainani and R.R. Shelke, "Evolutionary Clustering Algorithm in Data Mining", International Journal of Advance Research in Computer Science and Management Studies,vol 3, issue 3, p.p. 295-301, 2015.

[19] X. Wu,V. Kumar,J.R. Quinlan,J. Ghosh,Q. Yang,H. Motoda,G.J. McLachlan,A. Ng,B. Liu,P.S. Yu,Z. Zhou,M. Steinbach,D.J. Hand,D. Steinberg, "Top 10 algorithms in da ta mining", Knowledge and Information Systems, pp. 14:1-37, 2008.

[20] A. Balasundaram, P. T. V. Bhuvaneswari,"Comparative study on decision tree based data mining algorithm to assess risk of e pidemic", IET Chennai Fourth International Conference (SEISCON), pp. 390-396, 2013.