https://doi.org/10.51153/kjcis.v5i2.96

Comparative Analysis of Machine Learning techniques to Improve Software Defect Prediction

Muhammad Azam[1] Muhammad Nouman[2] Ahsan Rehman Gill[3]

Abstract

One of the most active areas of research in the software engineering community is defect prediction. The gap between data mining and software engineering must be bridged to increase the rate of software success. Before the testing phase, software defect prediction predicts where these flaws will occur in the source code. Methods for predicting software defects are widely used to investigate the impact area in software using various techniques (clustering, statistical methods, neural networks, and machine learning models). The goal of this research is to examine various machine learning algorithms for predicting software defects. There have been many fault prediction techniques introduced, but no single technique or approach can be used for all types of datasets. To achieve maximum accuracy, different machine learning algorithms such as Bayesian Net, Logistic Regression, Multilayer Perceptron, Ruler Zero-R, J48, Lazy IBK, Support Vector Machine, Neural Networks, Random Forest, and Decision stump were used to uncover the largest subset of defects that could be predicted. This research concern is to find out defects using five NASA data sets JM1, CM1, KC1, KC2, and PC1. Logistic Regression has been shown to have the best output compared to others (93%).

Keyword: Software Defects, Machine Learning, Defect Prediction, SVM, Techniques.

1. Introduction

D effects designate unexpected performance of software system in return of user's given requirements. This abnormal behavior is usually found by the software tester in the phase of software testing marked as a defect. A software defect is also known as "Imperfection in the process of software development that usually causes software failure that could not meet the user's desired expectation." A defect is some deficiency or flaw in a software process or product. Because of an error, fault, or failure. The paradigm defines "error" as a human activity that promotes improper outcomes, and "defect" as a wrong choice that results in erroneous outcomes for a solution to the problem[1].

A software defect is about a condition where a software system or product could not meet software requirements or user's requirements, software having defects will be a failure.

[1]*University of Agriculture Faisalabad, Pakistan | writetoazamkhalid@gmail.com*
[2]*University of Agriculture Faisalabad, Pakistan | m.nouman909@gmail.com*
[3]*University of Agriculture Faisalabad, Pakistan | ahsanrehman41@gmail.com*

Defects occur because of unusual or abnormal behavior of software or system. Unusual or abnormal behavior of software is directly proportional to software and as well it also affects customer requirements[2]. Since the major goal of testing the software in SDLC (software development life cycle) is to detect defects as soon as feasible, the team of software testing generally put the load on the testing phase to make sure that all defects are highlighted or found successfully, after the identification of defects and fix these defects in software testing phase by developers of software. Software defect existence influenced software reliability, software quality, and as well software maintenance price. It is impossible to achieve defect-free software, even the software testing process is put strictly just because most of the time defects are unseen. Stakeholders would ask the software testing team for forecasting software defects, so stakeholders could easily determine that the software is feasible and ready to deploy.

If the software is part defects this will be associated with some reason discussed below. Software Specification could also be wrong or not meet with user requirements either that could because of conflict requirements in the result software features be missing. It could be more complex to decide on the missing requirements that are not well explained or documented or might be poorly styled. It could not be considered that all requirements reflect wrongly[5]. Software developers could not be more competent for software projects just because of incomplete requirements. It could be the drawback of a project manager that the software development life cycle process will not follow by the project manager as needed in table 1.

Table 1: Defect Percentage

Software Development Phases	The defect may occur in a percentage
Requirement phase	20%
Designing phase	25%
Coding phase	35%
User Manual	12%
Bad Fixes	8%

- Software efforts are inclined to pay more observation on the following three fundamental issues in the software development life cycle:
- Software defect prediction from the huge amount of data[6].
- Software time estimation to ensure software reliability.
- Test software design, and test software process as well will affect the number of defects and density of software[7].

Defect prediction is an important activity to develop quality software. Because defect prediction precedes software deployment to reach user satisfaction and improves the

overall performance of the software. Identification of errors or defects earlier leads to sufficient allocation of resources that will because of reduction of time and cost as well to get a quality product. Therefore, the software defect prediction model participates in active responsibility for understanding the evaluation and improving the quality of software [8]. Different approaches have been planned to manage software fault prediction issues or problems. However, there are many techniques introduced in the literature review for fault prediction but there is no single approach for all datasets. Because it depends upon the nature of the dataset. Deciding which method should be used for fault prediction is a challenging activity. There is the most reliable method for defect prediction is Machine Learning[9]. To support a strategic distance from such disappointments in a software product, Defect prediction techniques (DPT) are performed in every stage of software development.

A. Software Quality Assurance:

According to past IT sectors and software firms, software quality is a prime object to focus on. Software defect prediction can straightforwardly impact software quality and achieved significant fame in a recent couple of years. Defective modules have a greater impact on software quality leading to cost, delivery time, and a lot higher maintenance costs [9]. Not being simply prepared to measure up to the assumptions on time and possibly speedy time is required, yet moreover, the ability to convey great quality programming things or infinitely better quality all the while is of most outrageous importance[10].

Hence, a lot of exploration is happening about how to further develop the item quality inside the compelled days open of the entire programming progression life cycle. Various ways exist for working on the overall idea of the item thusly made, for instance, better testing techniques, complete programmed testing works out, and early deformity expectation [11]. Thus, predicting software module whether a software entity contains defects can be helpful to improve the software quality. Therefore, quality is a key point that decides whether the software is according to a customer's need or process in which the software dataset under study is taken, and then pre-processing techniques are applied to data e.g., R, multilayer perceptron, k-neighbor nearest (KNN) and many more applied to retrieve information of defected data. In short, customer satisfaction is a key point for a successful project, for such a purpose we are going to research to find out defects earlier [12].

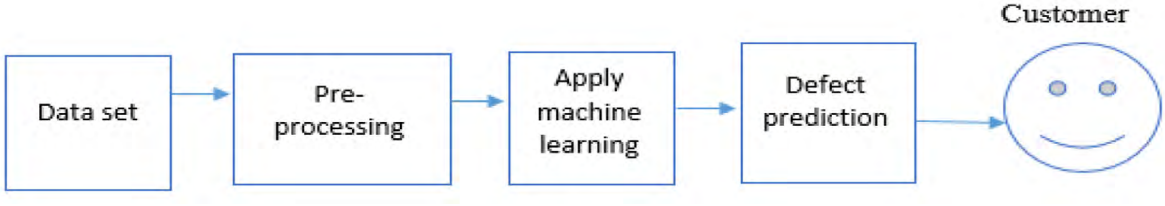

Figure 1: Generic Process of SDP

Software defect prediction achieved notable popularity in the last small number of years. Software Defect prediction directly affects software quality. Bad software modules have a solid impact on the software quality leading to cost overruns, delayed software completion timelines, and higher maintenance costs [13]. There are two basic approaches to Software Quality Assurance first one is defect detection and the second one is defect prevention. Defect prevention means avoiding upcoming defective activities as soon as earlier [12]. Defect prediction deals with existing defects. The approach to defect prevention is the process of improving software quality [14], and our research is concerned to improve software quality by predicting defects. Defect prevention activities are to find an error in software requirement planning, design the algorithm, and review the algorithm implementation [13]. The main object of defect prediction is to predict defects, bugs, or faults from software products and estimate the delivery quality and maintenance effort before the deployment process of the software product. The approach to defect prevention is the process of improving software quality [15].

The main research object idea is to explore the different machine learning algorithms to get maximum accuracy using feature selection for SDP. The prime aim of this research is to predict performance defects without overrunning the estimated cost and to find & analyzed which part of the software is more likely to have defects and deliver quality software.

The rest of this paper is as follows: Section 2 stands for related work. A machine learning algorithm is explained in section 3. The results of the experiments are discussed in section 4. In the end, section 5 concludes the paper and presents some future directions.

2. Literature Review

Most desired research zone of predicting defects using machine learning techniques, data metrics, and other techniques recently several research grounds started new projects. Various models and conclusions have been presented by scholars in different approaches. The investigation of more software defect prediction research papers published since the year 1990 to 2019 [16],[17],.

Discussed machine learning algorithm for decision support which will achieve high precision and accuracy of decision support or decision-maker what was recommended. A deep understanding of the decisions making process was discussed. Two Methods of learning used implicit and explicit[18]. The researcher described that the non-symbolic knowledge provided better predictive accuracy in implicit learning and as well explicit produced symbolic knowledge which was a more comprehensible model[19]. In this paper researcher compared the comprehensibility and predictive accuracy of different machine

learning models (like explicit, implicit, and hybrid). These machine learning models are applied to several standard medical diagnostics, different electronic commerce, financial decision-making problems, and e-marketing. The best methods for every benchmark problem were different, but hybrid methods outperformed standard comprehensible methods, and as well ensemble methods often outperformed all other methods. Hoff ding trees and their variants performed suitable for mobile computing, data streams, or big data and explained less accurate outcomes for these batch problems which have not a vast number of learning examples [20] discussed that quality, performance, and effectiveness of software attributes depended on the software defect prediction model. Eight NASA data sets were used to approach the right DP model. Characteristics of several software attributes are used to forecast whether software modules were defective? or non-defected? If proper attributes are not selected for the defect prediction model, the performance of the model will be decreased.

Therefore, [21] for The viability and execution of the imperfection forecast model, it is critical to choose suitable characteristics which could be utilized to construct a useful indicator model. The specialist proposed a quality determination cycle to distinguish damaged programming or to approach a legitimate SDP model. The trial result showed that the characterized approach gives a similarly productive arrangement of characteristics which expanded the presentation result, quality, and viability of the SDP model[21]. One ML classifier was used to improve the result of software defect prediction. In the future, the researchers look forward to integrating the performance of different ML classifiers to furtherly make an improved proposed approach. [26] proposed techniques that were intended to address particularly the blemished attributes of programming datasets, specifically, the absence of class-imbalanced information and marked examples, for imperfection expectation semi-directed approach of AI, were utilized. The scientist handled two basic issues of programming, comparably for programming deformity expectation, and proposed a semi-managed task-driven word reference AI strategy to foster both marked and unlabeled data completely. The exploratory outcome was performed on nine NASA datasets having subjective and quantitative information. This paper enhanced include extraction and the related classifier boundary, while different misclassification costs were investigated to further develop the classification exactness[22].

The experimental results demonstrate that our method outperforms several representative state-of-the-art defect prediction methods. [23]Used NASA's five data sets for software defect prediction. Different classifiers from the machine learning field: Naïve Bayes, neural networks, logistic regression, k-nearest neighbor, and support vector machine-implemented and evaluated on data sets, each classifier was evaluated on datasets from NASA's metrics. Personal implementations rather than WEKA were used for applying machine learning classifiers. Results of this research determined that all models could detect software defects using static features best model to results was:[24] Naïve Bayes

was overmatched in three datasets, SVM was overmatched by a k-nearest neighbor, and logistic regression for only one of the datasets. The information obtained from these machine learning classifiers was extremely effective for the continuous improvement of the software. In future work, research is defined as the choice of model is a difficult problem and requires more research, by increasing the number of the classifiers as well as a variety of datasets can uncover the underlying structure of software. [25] discussed how the data mining techniques used for defect prediction. Defective modules can be the cause of software failures, decrease customer satisfaction, increase development and maintenance costs and. The focus of their research was to find out the remaining defects from the software data set. Some data mining techniques.

Regression, clustering, classification, and association mining were used to predict flaws in the software and define how data mining techniques enrich the quality of software. [26] Conduct research to help software developers with issues of the undertaking, an attempt to detect defects from software has been made to make quality software. In this paper, the researcher formulates the software defect prediction problem as a classification task of machine learning[27]. It further assesses the impact of different outfit techniques to tackle the imperfection forecast issue. The course on programming imperfection forecast issues has been enunciated as an errand of classification and afterward, it reviews the impact of an assortment of group techniques on the AI strategy classification effectiveness. Specifically[22], researchers have derived a hybrid method of ensemble classification based on an over-sampled approach for software defect prediction in various imbalanced NASA datasets. The high unnecessary allocation of classes in datasets degrades the execution of classification approaches.

The proposed method has been derived based on the Synthetic Minority Over-Sampling Technique (SMOTE)[28][29]. In potential research work, the writer wants to include the verification process of the proposed method on various datasets. [30] proposed a model to forecast defects by the usage of software project metrics that were composed of a review of software design, product, review of source code per as per product deployed, and after defect validation was performed[30]. Linear regression (machine learning algorithm) was performed to selected metrics via software metrics only, software project metrics, and both. As a result, researchers declared that linear regression supplies the right correlation relationship between SD and predictors using both software and software metrics. One more thing proven in this research is to check out the suitability of the proposed analysis of regression to assemble an effective SDP model.

In a future direction, the researcher gives direction to the proposed model to fully automate, and standards rules could be weighted. So, the measurement could be depending not on the weight as well depend on the number of satisfied standard rules or violated rules. [32] Introduced the resample technique with three types of ensemble learners:

Boosting, Bagging, and Rotation Forest. These ensemble learners evaluated seven types of benchmark datasets of NASA. The researcher discussed Single Machine Linear Classifiers (Artificial Neural Network, Support Vector Machine, Locally Weighted Learning, Naïve Bayes, Decision Tree, Random Forest, J48 Decision Tree, Logistic Regression, and PART Algorithm) and Ensemble Machine Learning Classifiers (Bagging Techniques, Boosting algorithm and Rotation Forest). The researcher analyzed the accuracy and performance of three learners Boosting, Bagging, and Rotation Forest for the Software defect prediction dataset. The efficiency of performance, as a result, was loosed by using a Support vector machine algorithm with three homogenous ensemble methods and the researchers recommended that do not use Support Vector machine for defect prediction.

As mentioned above different researchers supply different solutions to overcome the error in the software. Some researcher uses software metric to overcome software error some use a different model to find the buggy module [37]. But no one can use early phase Software Defect Prediction. So, this is a unique research, and this research will provide great benefit to software engineering. During the literature review, it is found that there are some major techniques of machine learning like support vector machine, Bayesian networks, confusion metrics, clustering, Regression, Association rule, Bayesian Belief Network, Bayesian Belief network K-mean clustering, association rule, hybrid selection approach [38], genetic programming [38], k-mean clustering, genetic algorithm [38], static code matrix, automatic static analysis, association rule, and artificial neural networks are discussed to predict the faults.

3. Materials and Methods

Different approaches have been planned to manage software fault prediction issues or problems. However, there are many techniques introduced in the literature review for fault prediction but there is no single approach for all datasets. Because it depends upon the nature of the dataset. Deciding which method should be used for fault prediction is a challenging activity. There is the most reliable method for defect prediction is Machine Learning [31]. Feature selection involves evaluating better accuracy between input and desired variable using Minitab. WEKA machine learning tool is used for feature selection. For Statistical analysis mini tab was used to evaluate two-tail t-testing.

3.1 Feature Selection:

In the two study fields, ML and AI data analysis is the hottest topic for researchers. The feature selection technique determined an effective way to sort out various problems by extracting irrelevant or redundant data [32]. The major goal of feature selection is to improve the prediction performance based on accuracy precision and many more, provide

faster and cost-effective prediction within a time slot and provide a better understanding fundamental process of data generating [33].

The procedure of cutting the input variable that is not affected by the results is known as feature selection. In feature selection, researchers only select these features that key features of the dataset. This was desirable for researchers to reduce input variables to eliminate extra computational effort or cost of modeling, in some scenarios to improvise the performance of the system or model. Feature selection involves evaluating the relationship between two or more variables. Selecting featured variables to have the strongest relationship between input variables and target variables Feature selection methods could be efficient, fastest, and effective.

As it is more challenging to use machine learning to select correct statistical measures for several types of datasets while it is performing filter-based feature selection. Data features used to train ML models have a huge influence on the accuracy or performance that could be better achieved. Irrelevant features or partially relevant features can negatively affect prediction model performance. Data set cleaning and feature selection should be the priority to design an effective model.

3.2 The algorithm is performed using machine learning:

Here are some algorithms discussed below which have been used for research results.

3.2.1 Logistic Regression

A logistic regression algorithm is used to predict the probability. It is used to prove the probability of a specific class for example pass/fall flat, win/lose, alive/dead, or sound/wiped out. This can be stretched out to display a few classes of occasions, for example, deciding if a picture has a feline, hound, lion, and so on. Each article found in the picture would be distributed a probability somewhere in the range of 0 and 1 and the aggregate added to one. It is a statistical technique and utilized for logistics purposes to display a binary (0 and 1 form) dependent variable albeit a lot of progressively complex augmentations exist [34].

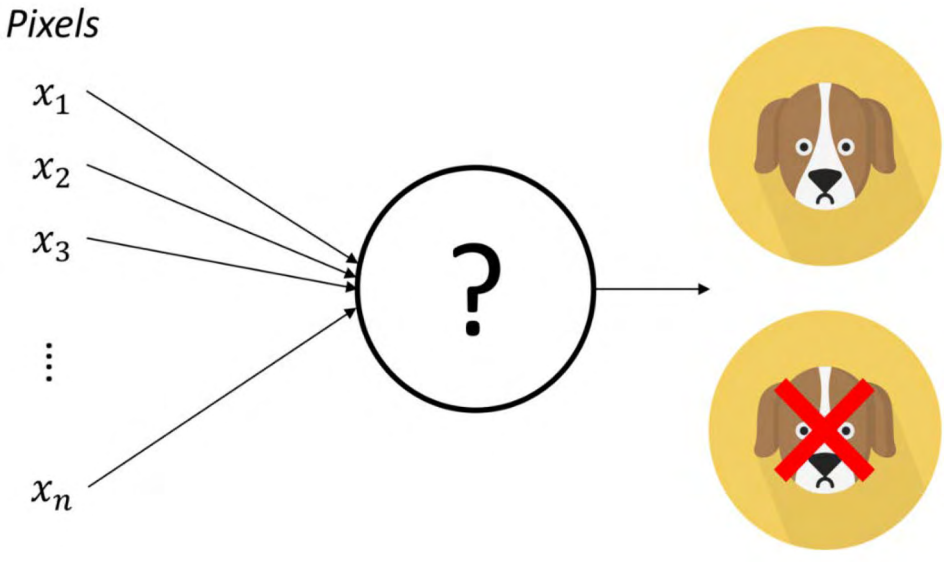

Figure 2: Logistic Regression

3.2.2 Bayes Net

Bayes Net uses to evaluate the probabilistic graphical model used by Bayesian inference for probability computations. Bayes Network determined model condition dependency by presenting conditions dependency of edges defined in the direct graph. Using Bayes Net, the researcher can perfect compact, variable factorization to join probability distribution to get advantages of conditional independence. Bayes network is used for prediction, anomaly detection, decision making in uncertainty situations, time series prediction, and many more [35].

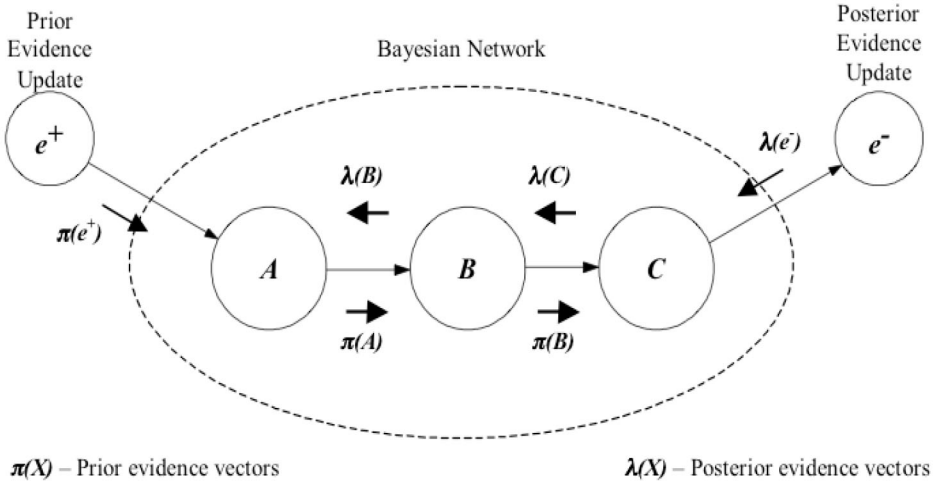

Figure 3. Bayes Net

3.2.3 Multilayer Perceptron (MLP)

It is a chain of perceptrons, systems of linear classifiers and it is a class of ANN. The term MLP is used abstrusely, here and there freely to allude to any feedforward ANN, now and again carefully to allude to a system made from different layers of perceptions. MLP are some of the times informally alluded to as "vanilla" neural systems, particularly when they have a single secreted layer. It consists of three layers of the node the 1st one is the input layer and the 2nd one is a hidden layer or unseen layer and the 3rd one is the output layer.

These three la

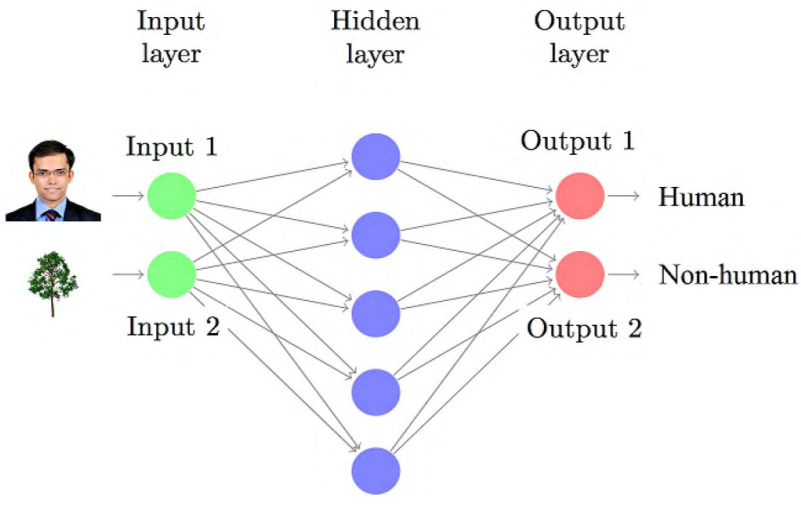

Figure 4: Multilayer Perceptron Layer Example

3.2.4. J48

J48 was used to build a decision tree derived by Ross Quinlan mentioned. J48 is the implementation of iterative Dichotomise 4 derived by the project team. J48 is a data mining tool for, the execution of C4.5 algorithms. It is an open-source Java implementation for deciding. Based on the input value, it generates tree-constructed data output. This theorem was developed by Ross Quinlan [36]. It holds continuous and discrete features. It is proper for bug prediction in all sizes of data sets. this rule is based on a learning classifier as a tree structure where every hub is a leaf.

3.3 Decision Stump

The decision stump ML algorithm has a one-level decision tree. DT is an exact technique for prediction. A decision stump build model to predict based on using a single input variable or feature. It is a well-ordered arrangement of if-Then instructions that can be more

minimal and, in this manner, more reasonable as compared to the decision tree. The choice to discover DT since it is less complex, and less computationally serious calculation than the decision tree method. It is the most straightforward method in ML. It outlines the data set with a DT which contains a similar number of qualities to the original data set [37].

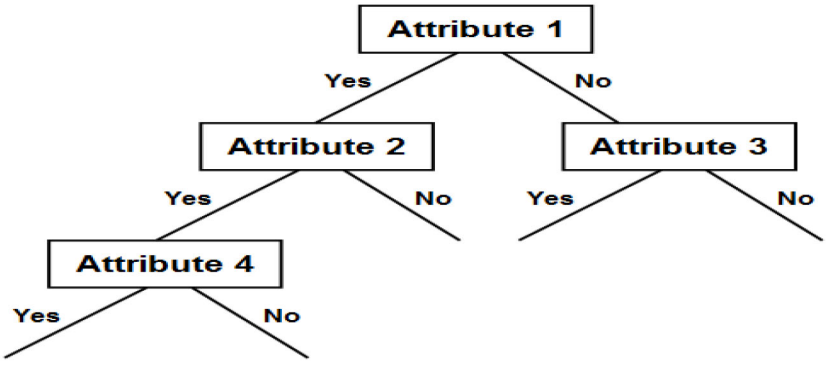

Figure 5: Decision Stump

3.3.1 Support Vector Machine:

SVM is supervised learning which could be used for regression, classification, and many more challenges. It arranges with individual perceptions or factors. In ML SVM are controlled learning models with related learning estimations that separate data utilized for relapse examination and grouping examination. Given a great deal of preparing models, each put aside as having a spot with both of two characterizations, an SVM preparing calculation fabricates a model that circulates new advisers for one class or the other, making it a non-probabilistic twofold straight classifier. An SVM model is a depiction of the models as spotlights on space, planned with the objective that the cases of the various classes are confined by a be normal. New models are then planned into that comparable space and expected to have a put with a class subject to the side of the opening on which they fall [38].

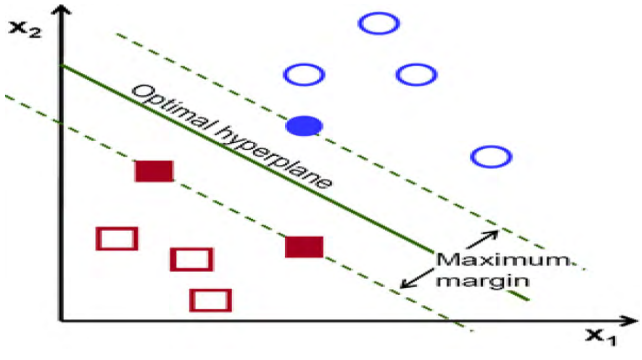

Figure 6: Support Vector Machine

3.3.2 Random forest

Data science presents a classification random forest algorithm. Random forest is the combination of DT, presented self-sufficiently with certain controlled change. It incorporates many tresses, and the outcome is dependent on most of the precise yield (output) selected in the class. It is the greatest classifier for the huge data set. Each root node of the tree has a bootstrap sample data or information which is equivalent to the real data and each tree has different sample bootstrap. Utilizing the best split technique for factors or variables is arbitrarily chosen from input factors or variables.
Every tree is then developed to the most extreme degree conceivable without pruning. At the point when all trees are worked in the forest technique, new occurrences are connected to every one of the trees at that point voting process happens to choose the arrangement with the greatest votes as the original instance expectation [39].

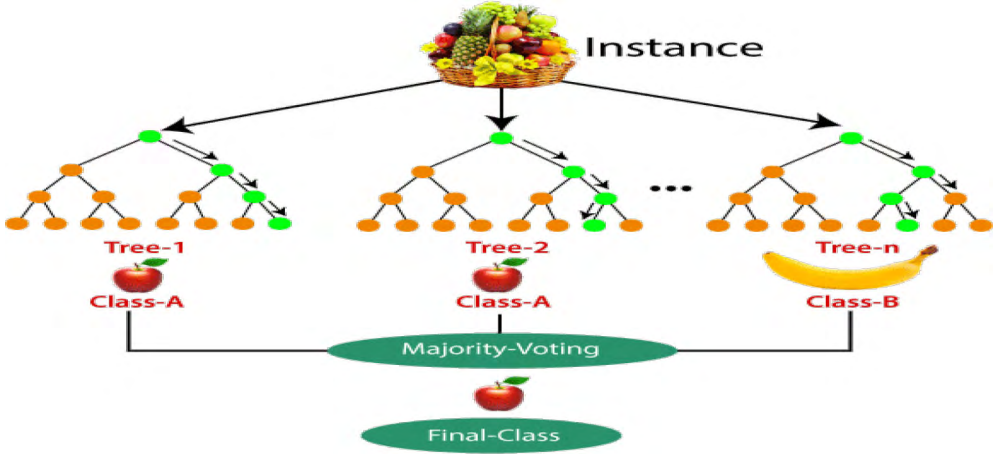

Figure 7: Random Forest

3.3.3 Lazy IBK

IBK is a fundamental algorithm family subset of classification. In k-nearest neighbor algorithm (KNN) is known as Lazy IBK (instance Based Learner). IBK is not useful to generate a model, instead, it is useful to build predictions for test instances within time. Distance is measured to find the k "closest" instance to make a prediction. It is an instance-based (IB) classifier. It varies from other IB learners in that it utilizes an entropy-based separation function [40]. It categorizes an instance by contrasting it with a database of pre-grouped models.

The key supposition that will be comparative instance will have comparative characterizations. The investigation lies in what way to characterize "comparative

instance" and "comparative classification." The comparing segments of an IB are the separation work which decides how comparative two instances are, and the arrangement work which shows how case likenesses yield a last grouping for the advance or new instance. This strategy little bit slow to assess yet useful for expectation [41].

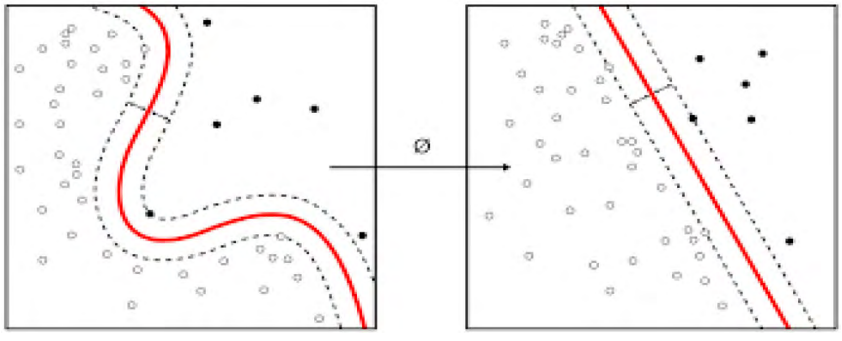

Figure 8: Lazy Inks

3.3.4 Ruler Zero-R

Zero algorithms defeat One if the targeted distribution of data is limited and skewed available for predicting majority class, correct results basing a rule depend on a single attribute. Rule zero algorithm performed on nominal data type [42]. Rule Zero-R always exceeds baseline when it assesses the training data. In evaluating process training data may not be reflected by performance on independent test data.

3.4 Datasets

To promote the replication and verification for this research experiment. Publicly available benchmark datasets from the PROMISE Repository were used to get experimental results. Several datasets are available open-source and available on the internet. For this research, five datasets were obtained from NASA promise dataset repository CM1, JM1, KC1, KC2, and PC1 [43]. Table 1 supplies detail about each data set information like which language was used in the project, faulty instance, on-faulty instance, percentage of description Buggy, no of the attribute, and missing attribute et

Table 2 : Characteristic Of D Use

Project	Languages	# Of instance	Non-Faulty instance	% Of Des Buggy
CM1	C	498	499	9.83%
JM1	C	10885	8779	19.35%
KC1	C++	2109	1783	24.85%
KC2	C++	522	415 105	20.49%
PC1	NA	1109	1032	6.94%

Zero algorithms defeat One if the targeted distribution of data is limited and skewed available for predicting majority class, correct results basing a rule depend on a single attribute. Rule zero algorithm performed on nominal data type [10]. Rule Zero-R always exceeds baseline when it assesses the training data. In evaluating process training data may not be reflected by performance on independent test data.

For this research, six datasets were obtained from NASA promise dataset repository CM1, JM1, KC1, KC2, and PC1 [49]. Table 1 supplies detail about each data set information like which language was used in the project, faulty instance, on-faulty instance, percentage of description Buggy, no of the attribute, and missing attribute. Here is the data set shown below in a graphical form with several instances.

Here is the data set shown below in a graphical form with several instances.

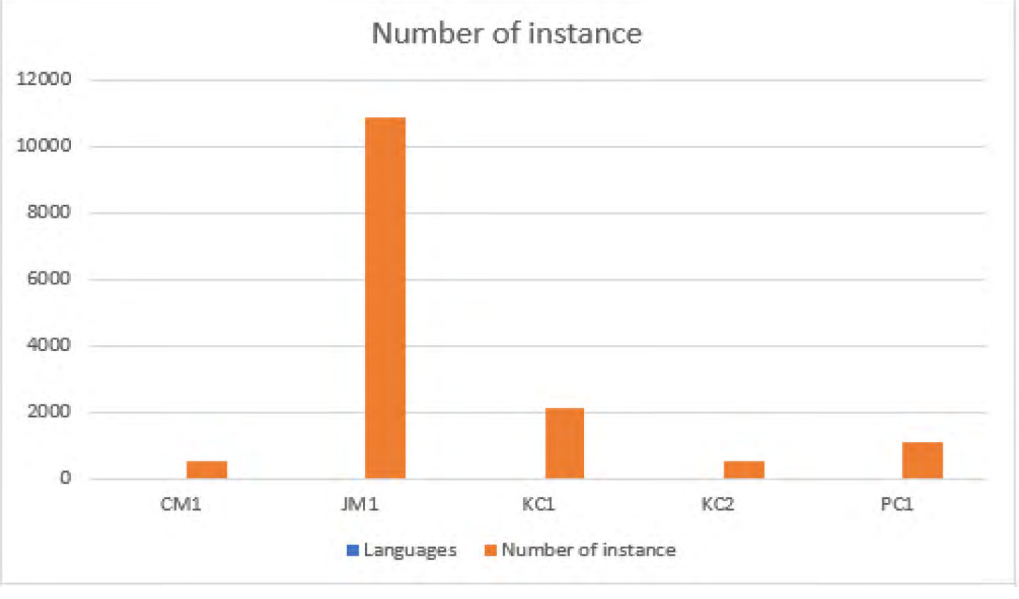

Figure 9: Number Of Instances

Here is the data set shown below in graphical form with the percentage of the buggy module.

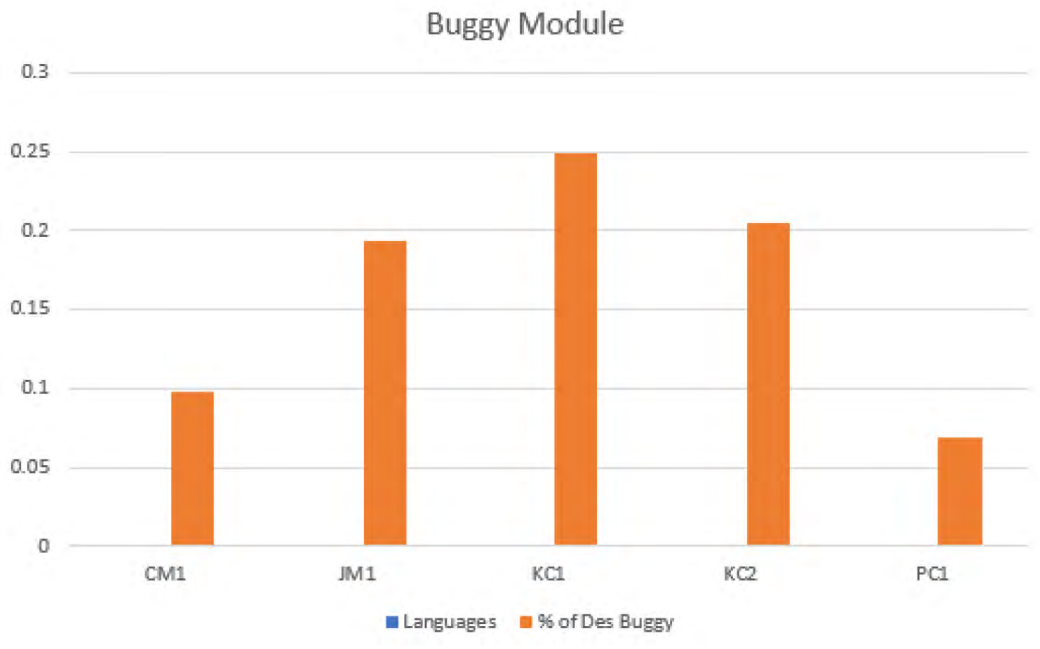

Figure 10: Buggies

4. Result And Discussion

4.1. NASA Repositor JM

JM1 accuracy with feature selection and without feature selection is in the following graph. It is clearly shown that feature selection accuracy is high as compared to without feature selection in figure 11.

Figure 11: JM1 Data Set Accuracy Graph

4.1.2. NASA Repositor CM

CM1 accuracy with feature selection and without feature selection is in the following graph. It is clearly shown that feature selection accuracy is high as compared to feature selection figure 12.

Figure 12: CM1 Data Set Accuracy Graph

4.1.3. NASA Repositor KC1

KC1 accuracy with feature selection and without feature selection is in the following graph 13. It is clearly shown that feature selection accuracy is high as compared to without feature selection.

Figure 13: KC1 Data Set Accuracy Graph

4.1.4. NASA Repositor KC2

KC2 accuracy with feature selection and without feature selection is in the following graph 14. It is clearly shown that feature selection accuracy is high as compared to without feature selection.

Figure 14: KC2 Data Set Accuracy Graph

4.1.5. NASA Repositor PC1

PC1 accuracy with feature selection and without feature selection is in the following graph 15. It is clearly shown that feature selection accuracy is high as compared to without feature selection.

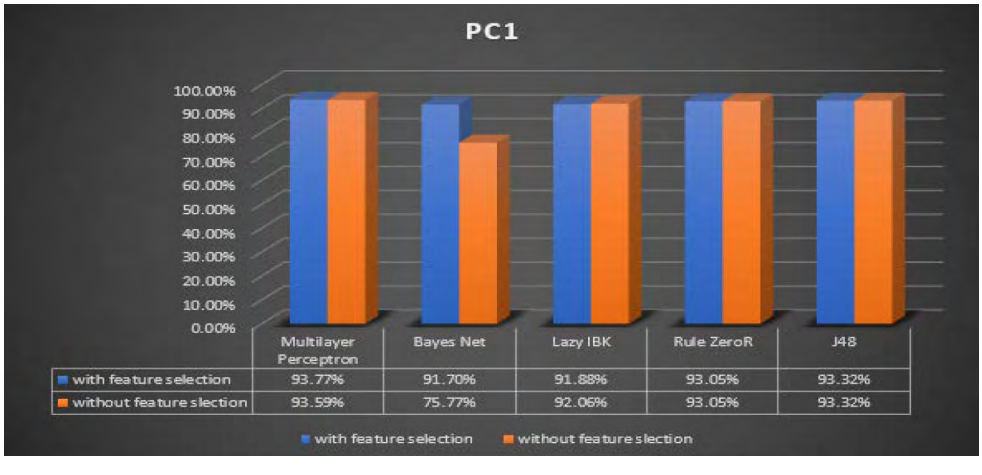

Figure 15: PC1 Data Set Accuracy Graph

To promote the replication and verification for this research experiment. Publicly available benchmark datasets from the PROMISE Repository were used to get an experimental result. Several datasets are available open-source and available on the internet. For this

research, Six datasets were obtained from NASA promise dataset repository CM1, JM1, KC1, KC2, and PC1 [43]. CM1 dataset used for prediction, JM1 dataset used for defect prediction KC1 used for prediction [44]–[46] KC2 dataset used in [43], [47], and PC1 dataset used in [48].

4.2. Results with Out Feature Selection

Accuracy performance without feature selection of 5 NASA datasets CM1, JM1, KC2, and PC1 is shown below tables 4.

Table 4.: Accuracy Table Feature Selection

Classifiers	CM1	JM1	KC2	PC1
Logistic Regression	73%	70%	78%	81%
Random Forest	83%	77%	82%	91%
Decision Stump	78%	71%	78%	87%
Support Vector Machine	75%	69%	79%	79%

All NASA dataset accuracy without feature selection is in the following graph. Here, in CM1, JM1, KC2, and PC1 datasets Random Forest is having the highest accuracy in figure 16. [49] used10 cross-validation folds in which the dataset is divided into ten parts equally.

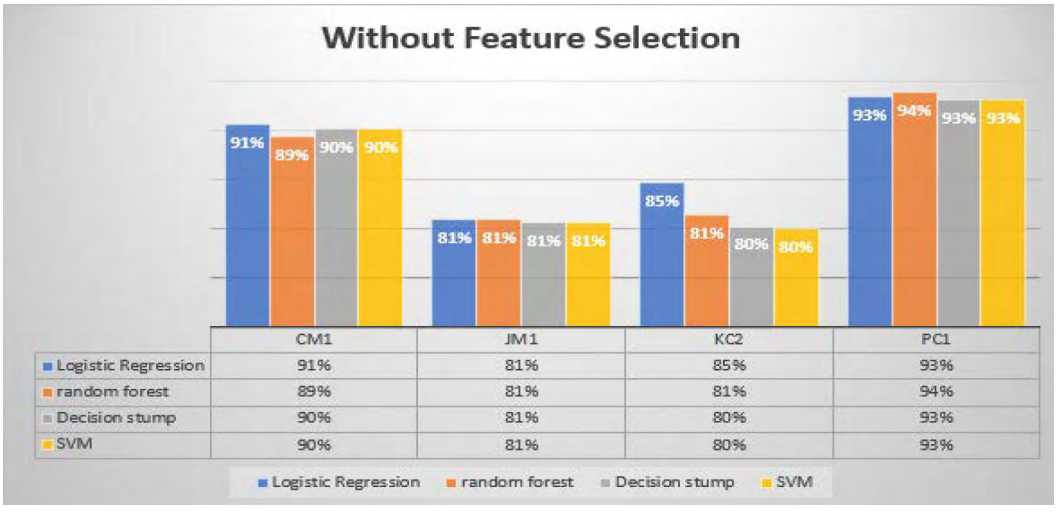

Figure 16: Accuracy Graph Feature Selection

4.1. Results with Feature Selection

Accuracy performance with feature selection of 5 NASA datasets CM1, JM1, KC2, and PC1 shown in below tables.

Table 5 : Accuracy Table With Feature

Classifiers	CM1	JM1	KC2	PC1
Logistic Regression	90.56%	80.94%	84.67%	93.32%
Random Forest	89.35%	80.92%	81.41%	93.86%
Decision Stump	90.16%	80.65%	80.26%	93.05%
Support Vector Machine	90.16%	80.66%	80.07%	93.05%

All NASA dataset accuracy feature selection is in the following graph. Here, in CM1, JM1 and KC2 datasets logistic regression are having the highest accuracy, as in the KC1 and PC1 data set Random Forest has the highest accuracy. In this research, thirty cross-validation folds in which the dataset is divided into 30 parts equally and test the dataset very closely and give a more accurate result. Overall PC1 accuracy is high by using all algorithms.

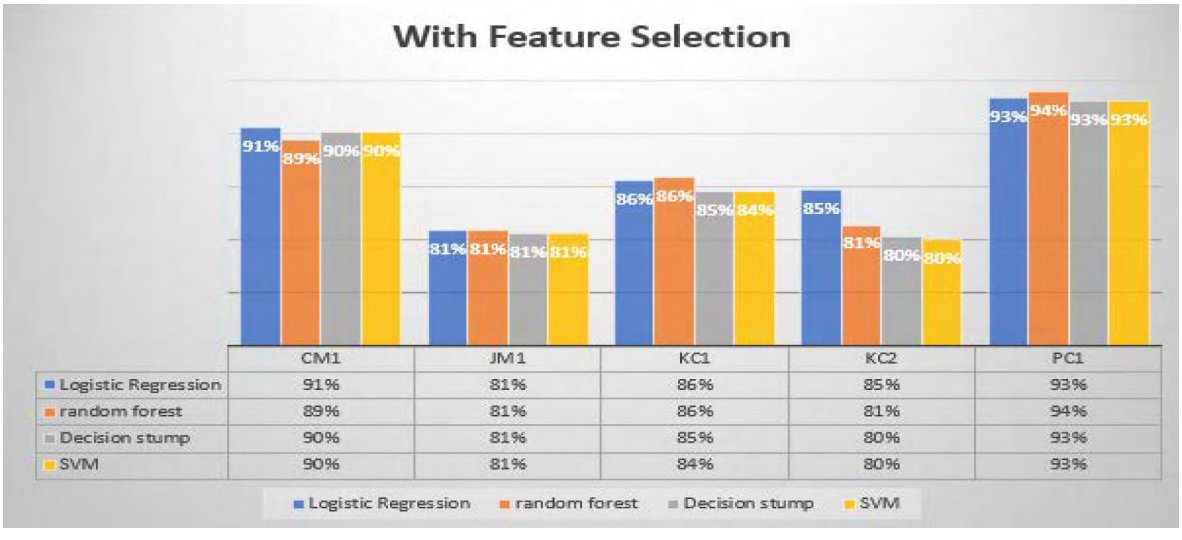

Figure 17: Accuracy Graph With Feature

4.2. Accuracy comparison of WFS and WOFS

Accuracy performance with feature selection and without feature selection on five datasets CM1, JM1, KC1, KC2, and PC1py applying logistic regression, random forest, and support vector machine shown below tables. Accuracy with feature selection is high as compared to accuracy without feature selection. In this research, thirty cross-validation folds in which the dataset is divided into 30 parts equally and test the dataset very closely and give a more accurate result. Here accuracy without feature selection is taken from.

Table 6: Accuracy Comparison WFS and WOFS

Data set	Classifiers	Accuracy WFS	Accuracy WOFS
JM1	Logistic Regression	80.94%	70%
	Random Forest	80.92%	77%
	Decision Stump	80.65%	71%
	Support vector Machine	80.66%	69%
CM1	Logistic Regression	90.56%	73%
	Random Forest	89.35%	83%
	Decision Stump	90.16%	78%
	Support Vector Machine	90.16%	75%
KC2	Logistic Regression	84.67%	78%
	Random Forest	81.41%	82%
	Decision Stump	80.26%	78%
	Support Vector Machine	80.07%	79%
PC1	Logistic Regression	93.32%	81%
	Random Forest	93.86%	91%
	Decision Stump	93.05%	87%
	Support Vector Machine	93.05%	79%

Here in the following table WFS= with feature selection WOFS=without feature selectionHere, in CM1, JM1, and KC2 datasets logistic regression is having highest accuracy, as in KC1 and PC1 data set random Forest has the highest accuracy. Overall PC1 accuracy is high by using all algorithms figure 18.

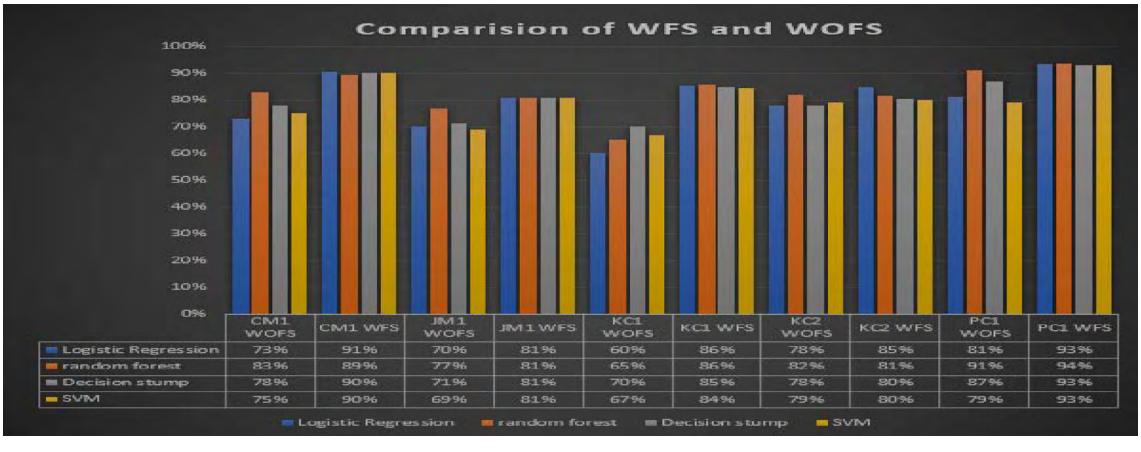

Figure 18: Accuracy Comparison WFS and WOFS

4.3. Results proved using statistics:

Two tail T-tests were applied using a mini tab to prove accuracy statically. The resulting

screenshot is shown below. For two-tail testing, two variables were used for testing accuracy with feature selection and accuracy without feature selection. This condition of p-value is shown below in the figure. Then H_0 will be accepted. But in this case, the p-value is 0.000 so H_0 is rejected. According to statistical decision accuracy with feature selection accuracy increased as compared to without feature selection. Using paired T-test statistical approach, it is proven that accuracy with feature selection is high.

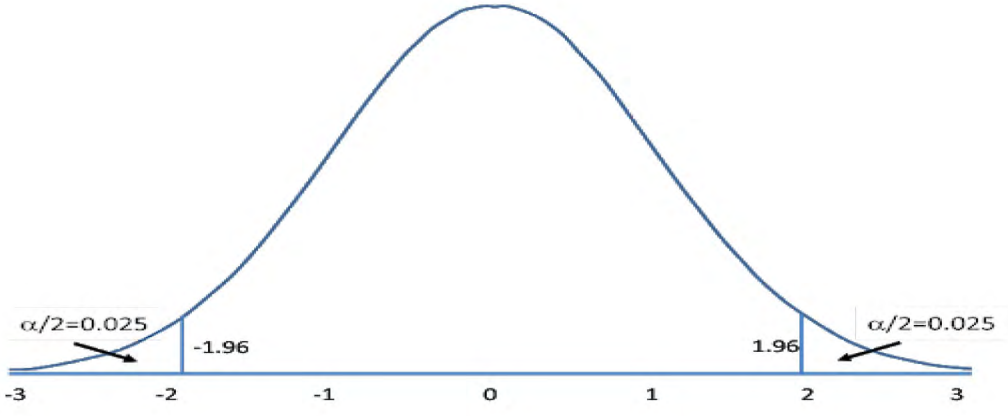

Figure 19: T-Test Result Graph

5. Conclusion

Software Defect perdition models aid to deal with these types of problems. Our research concerned was defected prediction by feature selection technique to get improvise accuracy results. This research result uncovers the largest subset of defects that could be predicted using above mentioned machine learning algorithm.

This paper's concern was to find out defects using Five NASA data sets JM1, CM1, KC1, KC2, and PC1. In this research machine learning algorithms Bayesian Net, Logistic regression, Multilayer perceptron, Ruler Zero, J48, Lazy IBK, Support Vector Machine, Neural Networks, Random Forest, Decision stump were used to perform feature selection to get maximum accuracy. Logistic Regression's highest accuracy founded at ninety-three% and the Bayesian Net averagely increase by an 8% accuracy rate using feature selection.

References

[1] Al-Nusrat, Alaa, Fears Hamadeh, Mohammad Khorramshahr, Mahmoud Al-Ayoub, and Nahla Al-Dhahiri. [2019]. "Dynamic Detection of Software Defects Using Supervised Learning Techniques." *International Journal of Communication Networks and Information Security* 11(1):185–91.

[2] Asaeda, Abdullah, and Mohammad Zubair Khan. [2019]. "Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study." *Journal of Software Engineering and Applications* 12(05):85–100. DOI: 10.4236/jsea.2019.125007.

[3] Bernardo, Marco, Paolo Calcarine, and Lorenzo Donatello. [2019]. "Architecting Families of Software Systems with Process Algebras." *ACM Transactions on Software Engineering and Methodology* 11(4):386–426. DOI: 10.1145/606612.606614.

[4] Boehm, Barry. [2019]. "A View of 20th and 21st Century Software Engineering." Pp. 12–29 in *Proceedings of the 28th international conference on Software engineering*. Shanghai China: ACM.

[5] Bruckert, Remco R. [2018]. "Bruckert - Bayesian Nets in Weka." 23.

[6] Bierman, Leo. 2017. "ST4_Method_Random_Forest." *Machine Learning* 45(1):5–32. DOI: 10.1017/CBO9781107415324.004.

[7] Cai, Jia, Jiawei Luo, Shulgin Wang, and Sheng Yang. [2018]. "Feature Selection in Machine Learning: A New Perspective." *Neurocomputing* 300:70–79. DOI: 10.1016/j.neucom.2017.11.077.

[8] Chen, Xiang, Yinzhou Mu, Key Liu, Zhan Qi Cui, and Chao Ni. 2021. "Revisiting Heterogeneous Defect Prediction Methods: How Far Are We?" *Information and Software Technology* 130:106441. DOI: 10.1016/j.infsof.2020.106441.

[9] Dam, Hao Khan, Trang Pham, Shien Wee Ng, Tuyen Tran, John Grundy, Aditya Ghose, Takes Kim, and Chloe Kim. [2018]. "A Deep Tree-Based Model for Software Defect Prediction." *ArXiv:1802.00921 [Cs]*.

[10] Devas Ena, Lakshmi, I. B. S. Hyderabad, and Lakshmi Devas Ena. [2018]. "Effectiveness Analysis of Xero, RIDOR and PART Classifiers for Credit Risk Appraisal Effectiveness Analysis of Xero, RIDOR and PART Classifiers for Credit Risk Appraisal." *International Journal of Advances in Computer Science and Technology (IJACST)* 3(11):6–11.

[11] Esteves, Granderson, Eduardo Figueredo, Adriano Veloso, Markos Vigias, and Nivea Zaviana. [2020]. "Understanding Machine Learning Software Defect Predictions." *Automated Software Engineering* 27(3–4):369–92. DOI: 10.1007/s10515-020-00277-4.

[12] Esteves, Granderson, Eduardo Figueredo, Adriano Veloso, Markos Vigias, and Nivea Zaviana. 2020. "Understanding Machine Learning Software Defect Predictions." *Automated Software Engineering* 27(3–4):369–92. DOI: 10.1007/s10515-020-00277-4.

[13] Felix, Aquebogue Amara Chukwu, and Sai Peck Lee. [2017]. "Integrated Approach to Software Defect Prediction." *IEEE Access* 5:21524–47. DOI: 10.1109/ACCESS.2017.2759180.

[14] Fu, Wei, and Tim Menzies. [2017]. "Revisiting Unsupervised Learning for Defect Prediction." *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* 72–83. DOI: 10.1145/3106237.3106257.

[15] Gruinard, Dominique, and Vlad Tarifa. [2017]. "Towards the Web of Things: Web Mashups for Embedded Devices." 8.

[16] Gruinard, Dominique, Vlad Tarifa, and Erik Wilde. [2017]. "A Resource-Oriented Architecture for the Web of Things." Pp. 1–8 in *2010 Internet of Things (IoT)*. Tokyo, Japan: IEEE.

[17] Humour, Wani, Mustafa Hammad, Mohammad Elnathan, and Fatima LaSharah. 2018. "Software Bug Prediction Using Machine Learning Approach." *International Journal of Advanced Computer Science and Applications* 9(2):78–83. DOI: 10.14569/ijacsa.2018.090212.

[18] He, Peng, Bing Li, Xiao Liu, Jun Chen, and Yuta Ma. [2018]. "An Empirical Study on Software Defect Prediction with a Simplified Metric Set." *Information and Software Technology* 59:170–90. DOI: 10.1016/j.infsof.2014.11.006.

[19] Herbold, Steffen, Alexander Tatsch, and Jens Grabowski. [2018]. "Global vs. Local Models for Cross-Project Defect Prediction: A Replication Study." *Empirical Software Engineering* 22(4):1866–1902. DOI: 10.1007/s10664-016-9468-y.

[20] Herbold, Steffen, Alexander Tatsch, and Jens Grabowski. [2019]. "Correction of 'A Comparative Study to Benchmark Cross-Project Defect Prediction Approaches.'" *IEEE Transactions on Software Engineering* 45(6):632–36. DOI: 10.1109/TSE.2018.2790413.

[21] Hutchison, David. 2014. "Future Data And." (2018). DOI: 10.1007/978-3-319-12778-1.

[22] Jayanthi, R., and Lilly Florence. [2019]. "Software Defect Prediction Techniques Using Metrics Based on Neural Network Classifier." *Cluster Computing* 22(s1):77–88. DOI: 10.1007/s10586-018-1730-1.

[23] Tiapride, Jaray's, Chakri Klan Tantithamthavorn, Hao Khan Dam, and John Grundy. [2022]. "An Empirical Study of Model-Agnostic Techniques for Defect Prediction Models." *IEEE Transactions on Software Engineering* 48(1):166–85. DOI: 10.1109/TSE.2020.2982385.

[24] Kamila is, Andreas. [2019]. "A Lightweight Resource-Oriented Application Framework for Wireless Sensor Networks." Doi: 10.3929/ETHZ-A-005816888.

[25] Kaur, Aras deep, Parminder S. Sandhu, and Manpreet Singh Brar. [2019]. "Early Software Fault Prediction Using Real-Time Defect Da" *2nd International Conference on Machine Vision, ICMV 2009* 242–45. DOI: 10.1109/ICMV.2009.54.

[26] Kaur, Gaganjeet, and Amit Chhabra. [2014]. "Improved J48 Classification Algorithm for the Prediction of Diabetes." *International Journal of Computer Applications* 98(22):13–17. DOI: 10.5120/17314-7433.

[27] Kondo, Masanari, Cor-Paul Beemer, Yasutaka Kamei, Ahmed E. Hassan, and Osamu Mizuno. [2019]. "The Impact of Feature Reduction Techniques on Defect Prediction Models." *Empirical Software Engineering* 24(4):1925–63. DOI: 10.1007/s10664-018-9679-5.

[28] Lanza, Michele, Andrea Mocci, and Luca Panzanella. [2016]. "The Tragedy of Defect Prediction, Prince of Empirical Software Engineering Research." *IEEE Software* 33(6):102–5. DOI: 10.1109/MS.2016.156.

[29] Taraji, Essam H., Mohammad Alshaya, and Lamoure Ghoul. [2015]. "Software Defect Prediction Using Ensemble Learning on Selected Features." *Information and Software Technology* 58:388–402. DOI: 10.1016/j.infsof.2014.07.005.

[30] Li, Jian, Panji He, Jiaming Zhu, and Michael R. Liu. [2017]. "Software Defect Prediction via Convolutional Neural Network." Pp. 318–28 in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. Prague, Czech Republic: IEEE.

[31] Li, Ning, Martin Shepperd, and Yuchen Guo. [2020]. "A Systematic Review of Unsupervised Learning Techniques for Software Defect Prediction." *ArXiv:1907.12027 [Cs]*.

[32] Lyons, R. [2015]. "Approved for Public Release; Distribution Unlimited." 148.

[33] Manjula, C., and Lilly Florence. [2019]. "Deep Neural Network Based Hybrid Approach for Software Defect Prediction Using Software Metrics." *Cluster Computing* 22:9847–63. DOI: 10.1007/s10586-018-1696-z.

[34] MwanjeleMwagha, Solomon, Masinde Muthoni, and Peter Ochieng. [2014]. "Comparison of Nearest Neighbor (Ink), Regression by Discretization and Isotonic Regression Classification Algorithms for Precipitation Classes Prediction." *International Journal of Computer Applications* 96(21):44–48. Doi: 10.5120/16919-6729.

[35] Naresh, E., Vijaya Kumar B. P, and Sahana P. Shankar. [2017]. "Comparative Analysis of the Various Data Mining Techniques for Defect Prediction Using the NASA MDP Datasets for Better Quality of the Software Product." 10(7):2005–17.

[36] Neu, Lian. [2018]. "A Review of the Application of Logistic Regression in Educational Research: Common Issues, Implications, and Suggestions." *Educational Review* 00(00):1–27. DOI: 10.1080/00131911.2018.1483892.

[37] Novakovic, Jasmine Đ., Alepine Valjavec, and Sinisa S. Ilic. [2016]. "EXPERIMENTAL STUDY OF USING THE K-NEAREST NEIGHBOUR CLASSIFIER EXPERIMENTAL STUDY OF USING THE K-NEAREST NEIGHBOUR CLASSIFIER WITH FILTER METHODS." (May 2018).

[38] Pan, Cong, Minyan Lu, Biao Xu, and Hauling Gao. [2019]. "An Improved CNN Model for Within-Project Software Defect Prediction." *Applied Sciences* 9(10):2138. DOI: 10.3390/app9102138.

[39] Parsons, Shaun, Rami Bassoon, Peter R. Lewis, and Xin Yao. [2019]. "Towards a Better Understanding of Self-Awareness and Self-Expression within Software Systems." 8.

[40] Petrik, Jean, David Bowes, Tracy Hall, Bruce Christianson, and Nathan Badoo. [2016]. "The Jinx on the NASA Software Defect Data Sets." *ACM International Conference Proceeding Series* 01-03-June. DOI: 10.1145/2915970.2916007.

[41] Singh, Kunwar P., Nikita Basant, and Shikha Gupta. [2016]. "Support Vector Machines in Water Quality Management." *Analytica Chemical Acta* 703(2):152–62. DOI: 10.1016/j.aca.2011.07.027.

[42] Son, Le, Nakul Pritam, Manju Khari, Raghavendra Kumar, Pham Phuong, and Pham Thong. [2019]. "Empirical Study of Software Defect Prediction: A Systematic Mapping." *Symmetry* 11(2):212. DOI: 10.3390/sym11020212.

[43] Son, Le, Nakul Pritam, Manju Khari, Raghavendra Kumar, Pham Phuong, and Pham Thong. [2019]. "Empirical Study of Software Defect Prediction: A Systematic Mapping." *Symmetry* 11(2):212. DOI: 10.3390/sym11020212.

[44] Thota, Mahesh Kumar, Francis H. Sajin, and Gaultheria Rajesh. [2019]. "Survey on Software Defect Prediction Techniques." *International Journal of Applied Science and Engineering* 14.

[45] Thota, Mahesh Kumar, Francis H. Sajin, and Gaultheria Rajesh. 2019. "Survey on Software Defect Prediction Techniques." *International Journal of Applied Science and Engineering* 14.

[46] Wahoo, Rome Satria. [2015]. "A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks." *Journal of Software Engineering* 1(1):16.

[47] Wang, Po Wei, and Chi Jen Lin. [2014]. *Support Vector Machines*.

[48] Wilde, Erik. [2007]. "Putting Things to REST." 14.

[49] Xu, Zhou, Shuai Pang, Tao Zhang, Xia-Pu Luo, Jinn Liu, Yu-Tian Tang, Xiao Yu, and Lei Xu. [2019]. "Cross Project Defect Prediction via Balanced Distribution Adaptation Based Transfer Learning." *Journal of Computer Science and Technology* 34(5):1039–62. Doi: 10.1007/s11390-019-1959-z.