International Journal of Early Childhood Special Education (INT-JECSE)
DOI:10.9756/INTJECSE/V14I5.866 ISSN: 1308-5581 Vol 14, Issue 05 2022

Software Defect Prediction Based on Machine learning

1. Dr. Molli Srinivasa Rao,
Professor,
Dadi Institute of Engineering and Technology,
Vizag, Andhra Pradesh, India
Email: drmollisrinivasarao@gmail.com

2. M Manju Bhargavi,
M.Tech (CSE)
Dadi Institute of Engineering and Technology,
Vizag, Andhra Pradesh, India

Abstract

There was rapid boom of software program development. Due to numerous reasons, the software program comes with many defects. In latest years, defect prediction, one of the principal software program engineering problems, has been inside the consciousness of researchers because it has a pivotal function in estimating software program errors and defective modules. Researchers with the intention of enhancing prediction accuracy have advanced many models for software program defect prediction. But, there are a number of crucial conditions and theoretical issues a good way to reap higher consequences. In this paper, we are able to be discussing SVM classifier, Naïve bayes classifier ,logistic regression, decision tree and KNN with cross validation is used to locate the accuracy. The effects show that consistency in high accuracy prediction turned into done the use of this strategies.

Keywords— defect prediction; SVM classifier; Naïve bayes; Logistic Regression ,Decision Tree ,KNN.

1 .INTRODUCTIONS

Software quality can be measured by means of fault proneness information. a number of the most up-to-date strategies attempted to investigate that whether or not to be had metrics in requirement and code could be used to pick out fault prone modules. It ought to be stated that, those metrics and requirement statistics have been collected throughout software program development cycle and extensive efforts have been deployed to build extra accurate defect prediction modules by way of those information to estimate the satisfactory of centered application modules. On this regard, unique techniques were proposed to expect defective modules in latest years, like statistical approaches, data mining and deep learning procedures. but, defect prediction modules could be applied in extraordinary phases in the following lessons: the first magnificence that is in testing phase consist of the subsequent models: seize-recapture models [1], neural network models [2], measure technique primarily based on scalable approach based totally on supply code complexity [3]. Subsequent class, which changed into employed to expect wide variety of defects inside the software improvement system, is earlier than the real developing segment of the centered software. The following models are blanketed on this category: phase primarily based approach this is recommended in [4], An Ada-primarily based defect prediction technique is proposed in [5], and to expect defections at first degrees of programming, a model has been proposed with the aid of Smits [6].

The two primary issues, which frequently bring about defected facts, are excessive dimensionality and imbalanced training. In [7], a single classifier technique is supplied by means of Kehan et al. That is based totally on facts sampling and function choice to deal with the aforementioned problems. They don't forget 3 scenarios, such that function selection is based totally on two distinctive varieties of information, i.e., original information or sampled information. They concluded that the state of affairs that's characteristic choice have executed on sampled statistics and have modeled on unique statistics have significantly better performance than the other situations.

Modules/lessons in software defect Prediction (SDP) can be categorized into : fault-susceptible and not fault-susceptible. SDP models can be constructed using the fault records and the software program metrics obtained from previous software releases or similar software projects [8,9]. After constructing the model, it can be incorporated into present day initiatives and assist classify all the modules/instructions as being fault-prone or now not fault-prone [10]. The usage of these consequences, the software practitioners can now make an informed decision to work on all of the fault prone regions in the course of the early ranges of development.

For instance, if only 30% of testing assets were assigned to a certain software program, having understanding of all of the fault-susceptible regions will make sure that each one the available sources are allotted toward the correction of the modules/instructions in these regions [11]. Thereby, resulting in a excessive first-rate and maintainable that is of high quality and produced with the given time body and budget [12]. A great a part of SDP studies interest is targeted in the detection of whether software components are defect inclined or now not by means of relying on the usage of software metrics drawn from the code [13]. At the same time as specific machine studying algorithms have been used in supporting with the type of software additives as being defect-inclined or now not with the aid of trying to nice rules or patterns inside information, none of them has proved be correct on a consistent basis. A number of these strategies used consist of combined algorithms, parametric models, machine learning techniques and statistical strategies. But, before concluding on whether this hassle is largely unsolvable, there may be need for the identity of the satisfactory prediction technique to help with predicting a trouble primarily based on the context.

This examine will rely on open source software program repositories to research key software program defect prediction models including SVM classifier, Naïve Bayes classifier, Decision tree, Logistic Regression , KNN. By means of giving clues approximately these models, and the way they react with unique datasets, we do hope that consequences acquired on this have a look at will assist growth confidence in them. Key findings of this study display that the usage of stacking more than one classifiers can be of use to defect prediction.

II. LITERATURE REVIEW

They four varieties of machine learning task which include reinforcement, semi-supervised, unsupervised and supervised learning. Although supervised and un-supervised learning remain the most popular task group.

Supervised learning is machine learning method that involves the usage of labelled training data, which houses various training examples to infer a characteristic. The training instance includes an input object and the favored output price and consists of the regression and classification of supervised learning responsibilities [32]. The regression category mission focuses on non-stop variety version constructing even as the class studying project focuses on constructing predictive version that functions within a discreet range. Instance of supervised device learning techniques include aid vector system, neural network, linear regression, Bayesian learning, instance primarily based learning, rule learning and learning classification [33].

The speedy increase of studies in system studying has resulted in the creation of various learning algorithms that can be used throughout distinct programs [34]. Additionally, the potential of machine learning algorithms to clear up-real global troubles will often decide its remaining value making the duplicate and application of algorithms in new tasks essential to the sphere’s progress. However, the contemporary research landscape functions numerous guides concerning software defect prediction model improvement. These may be located into categories based on and classification techniques.

Design [42] and code metrics [43] are used to evaluate the accuracy of fault prediction models which can be to be had earlier than and after the device is carried out. Code metrics and layout metrics are to be had most effective after the machine is applied and earlier than the coding has began [44]. Models are primarily based on the data from one launch of a big telecommunication device evolved with the aid of Ericsson the use of linear regression. Of their have a look at, prediction made after the device is 34% more correct than earlier than the machine. The range of metrics to be had earlier than the implementation is 43% and after the implementation is 58% [44] however the performance of the machine is identical whilst metrics are not used. Professionals use Statistical techniques and machine getting to know techniques to anticipate the inability inclination of the code of their software. Of their examine, execution of lines of code (LOC) metric is nicely and accuracy of loss of cohesion on methods (LCOM) metric is remarkable but its end result pleasant is low .

Aleem et al [40], after suing various machine learning strategies to behavior a have a look at on 15 datasets (KC3, KC1, CM1, AR6, and AR1 and many others) determined out that bagging, multilayer perceptron (MLP) and support vector system (SVM) carried out high degrees of overall performance and accuracy.

Bibi et al. follow a device learning technique to the trouble of estimating the range of defects called Regression via category (RvC) [41]. RvC to start with robotically discretizes the variety of defects into some of fault classes, then learns a version that predicts the fault magnificence of a software gadget. eventually, RvC transforms the class output of the model back right into a numeric prediction.

Challagulla et al. examine unique predictor models on four distinct actual- time software program defect datasets [45]. Their effects display that a combination of 1R and example-based totally gaining knowledge of along with the Consistency based Subset evaluation method gives a particularly better consistency in accuracy prediction as compared to different methods. They also declare that length and complexity metrics are not enough for appropriately predicting actual-time software program defects.

Ratzinger et al. examine the influence of evolution sports along with refactoring on software defects [46]. In a case look at of 5 open source tasks they used attributes of software program evolution to are expecting defects in time intervals of six months. They use versioning and trouble tracking structures to extract 110 statistics mining features, which are separated into refactoring and non-refactoring associated functions. Those features are used as input into type algorithms that create prediction models for software defects.

An investigation performed by [48] relied on a unique benchmark framework in comparing and predicting software defect. The activities concerned comparing and comparing distinctive getting to know schemes to the selected one and the usage of it to build a predictor that has all of the historical records [47]. This predictor is now equipped to predict any defect in any new data.

III. RELATED WORKS

In latest years, wide kind of machine learning techniques have been proposed and implemented to extraordinary domain names by researchers. However, in defect prediction context, to the pleasant of our understanding few works were performed which we are able to evaluate in this phase.

Y. Chen and et al [14] reviewed the preceding work in subject of defect management and software prediction. They introduce a singular method for defect prediction by way of the use of data mining strategies and claim that their proposed model is able to lead the developmental levels of a brand new software program. In the beginning, defect database is generated that is made from all of the statistics about the defect facts within the software life cycle. After that, through mining techniques, especially Bayesian network, the defect prediction version is built for the going.

An stronger multilayer perceptron neural network is explored via [15], and also fault-proneness prediction modeling is completed through comparative evaluation for software program systems and then tested by means of NASA's Metrics data program (MDP). Gabriela Czibula et al. present in [16] a singular classification model regarding relational association rules mining.

Figuring out faulty modules is not always a clear-cut challenge. To reap excessive overall performance, diverse issue have to be considered in defect prediction models. Ishani and Arora and et al. in [17] introduce a number of them in element. Their studies display that those issues are caused by the following issues:

- Relationship between Attributes and fault.
- No benchmark to evaluate overall performance correctly.
- Problems with defect prediction in cross-mission.
- No to be had widespread framework.
- Economic boundaries of defect prediction in software program.
- Class imbalance trouble.

Besides the above techniques, association based type approach is taken into consideration on this context by Baojun Ma and et al [33]. They use CBA2 algorithm and evaluate it with the opposite rule primarily based type strategies. Their experimental effects indicates CBA2 acts higher than C4.five and RIPPER algorithms.

Commonly, defect prediction manner is designed through supervised machine learning (classification), that's called within-project defect prediction (WPDP) due to the fact all processes are conducted 'within' a single software program mission. Some preprocessing techniques together with characteristic selection and normalization are widely implemented in these studies [18], [19], [20]. But, WPDP has some intrinsic boundaries since training models without records of defect statistics generate the categorized dataset. Researchers have also proposed strategies to improve cross project defect prediction (CPDP) [19]–[23], [25], [26] this is defect prediction for unlabeled datasets [27], [28]. CPDP normally has low performance.

Overall performance. But, most CPDP strategies have a few limitations that have some giant consequences on the performance as an instance; they need to use identical metrics if source data set and goal statistics set had heterogeneous metrics. If you want to clear up this problem Jae chang Nam and Sunghun Kim [24] supplied a new algorithm. They proposed heterogeneous defect prediction (HDP) technique for predicting defects across challenge sets (whether or not heterogeneous metrics exist in dataset). Indeed, source project and destination project may be exceptional from each other. We classified most crucial latest studies based on 3 classes as they're taken into consideration in table 1. Six parameters are mentioned in details for techniques in table 1. In general, the following consequences are located from table 1 (Taxonomy table) by means of scrutiny of the models:

- The studying strategies are deployed for pattern classifications in most techniques.
- The class imbalanced problem changed into now not taken in to account in most current studies and they try to strong their strategies.
- A great range of methods did now not do not forget preprocessing step, at the same time as some of others have complicated approach for doing it.
- This paper offers a manner to improve defect prediction via leveraging the strength of logistic regression,SVM and KNN in machine learning models. The steps are discussed in detail after introducing machine learning techniques .

Approach	**PAPER**	**Method (Mining, Learning, Optimization)**	**Methodology**	**processing step**	**ss imbalance problem**	**Supervised Semi-supervised**	**Datasets**
The Proposed Method	-	**Machine Learning**	**Logistic regression,SVM,**	**Normalization**	**Not Considered**	-	**),ev(g),iv(G),n,v,i,d,l,e,b,t,**
WPDP	[17]	inining & Learning	neR, J48, and naïve Bayes	removing the module identifier attribute	Not Considered	Supervised	CM1,KC4,MW1,PC1, PC2,PC3,PC4
	[18]	inining & Learning	Extended transfer component analysis +logistic regression	min-max and z-score normalization methods	Not Considered	Supervised	ReLink,AEEEM
	[19]	Minining & Learning	WC and CC-data models	NN-filtering	Not Considered	Supervised	PC1,KC1,KC2,CM1, KC3, AR3,AR4,AR5 ,MW1,MC2
CPDP	[20]	Learning	nsfer Naive Bayes	NN-filtering	Not Considered	Supervised	kc3,Pc1,kc1,kc2, cm1, ar3,ar4,ar5, mw1,mc2
	[21]	Mining & Learning	context-aware rank transformations	Clean Data (Understand)	Not Considered	Semi-supervised	Generate a dataset
	[22]	Learning	ensemble approaches	Minimizing collinearity	Considered	Supervised	Bugzilla ,Columba ,Gimp ,Eclipse JDT ,Maven-2 ,Mozilla

HDP	[23]	Learning	canonical correlation analysis (CCA) nearest neighbor (NN)	z-score normalization	Considered	Supervised	SOFTLAB,ReLink AEEEM
	[24]	Learning	Logistic regression	Feature selection (gain ratio, chi-square, relief-F)	Considered	Supervised	EEM,ReLink,MORPH .SA,SOFTLAB

TABLE 1. A taxonomy of the related works along with the proposed method

IV. MACHINE LEARNING TECHNIQUES

The proposed scheme is shown in the Fig.1.

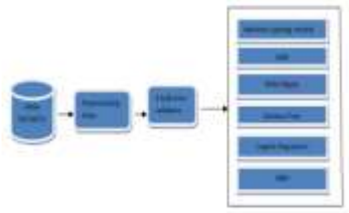


Fig.1. the general view of the proposed model

Following that, every step is discussed in detail. The scheme is designed based totally on our test, that are include four steps as it is shown in Fig.1. in this we are going to use two algorithms on the way to conduct this research. The first set of rules is SVM algorithm,we plot raw data as factors in an N-dimensional area(n= no of functions you have got).The fee of every function is then tied to a particular coordinate, making it clean to categorise the data. The second set of rules we use naïve bayes set of rules to assume that the presence of a selected characteristic in a category is unrelated to the presence of every other function. When the getting to know manner is accomplished, it's far first spiltted and educated to get tested to conduct classification. This algorithms pleasant suits for selection boundary, speedy and quick classification can be completed with the aid of this algorithms.

A. Support vector machine

Support Vector system (SVM) is introduced in COLT-ninety two through Boser, Guyon & Vapnik. it's far theoretically a completely well motivated set of rules. Vapnik & Chervonenkis (1960s) evolved SVM from Statistical mastering theory. In SVM, records is being differentiated into two units; training set and testing set. Each file within the training set consists of one target fee or elegance call and carries some properties referred to as watched variables. SVM discovers a direct dividing hyper- plane. The equation for partition is ax+by way of=c. SVM is applied as a part of numerous fields. SVM is applied as a part of two fold association errands. SVMs are any other promising non-direct, non-parametric order approach. SVM is utilized as part of the restorative diagnostics, optical individual reputation, electric powered load anticipating and different several fields.[35]

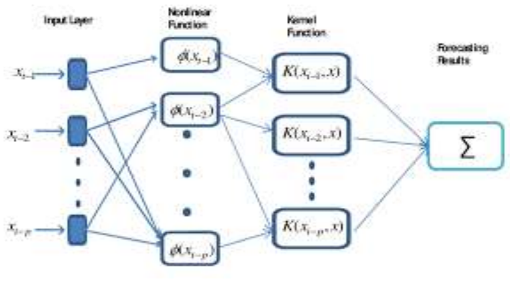


Fig.2. SVM Scheme

SVM may be of two types:

1.Linear SVM: Linear SVM is used for linearly separable statistics, which means if a dataset may be categorized into two instructions by using using a single straight line, then such data is termed as linearly separable information, and classifier is used known as as Linear SVM classifier.

2. Non-linear SVM: Non-Linear SVM is used for non-linearly separated statistics, which means if a dataset can't be categorized by using a straight line, then such data is named as non-linear facts and classifier used is known as as non linear SVM classifier.

B. Naïve Bayes algorithm

This classification algorithm that works with both multi-magnificence and binary (magnificence) classification troubles and can be pretty simple to understand when described the use of specific or binary enter values [29]. Naïve Bayes permits extension to actual fee attributes, which is likewise called the Gaussian Naïve Bayes. Working with the ordinary distribution (Gaussian) is quite simple, all one has to do is find the training records to estimate the standard deviation and mean.

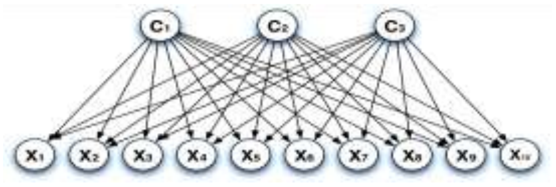


Fig.3. NAÏVE BAYES Scheme

Naïve Bayes algorithm is a supervised getting to know set of rules, that is based totally on Bayes theorem and used for fixing class problems. It's far specially used in textual content category that consists of a high-dimensional education dataset. Naïve Bayes Classifier is one of the easy and handiest classification algorithms which allows in constructing the quick machine learning models which could make quick predictions. It's far a probabilistic classifier, because of this it predicts on the premise of the chance of an item. Some popular examples of Naïve Bayes set of rules are unsolicited mail filtration,Sentimental analysis, and classifying articles[36].

C. Decision tree classification algorithm

A extreme trouble that most auto-encoders need to deal selection Tree is a Supervised getting to know technique that may be used for both category and Regression problems, however mostly it is desired for fixing category problems. It's far a tree-established classifier, wherein inner nodes constitute the capabilities of a dataset, branches represent the selection policies and every leaf node represents the final results. In a decision tree, there are nodes, which are the decision Node and Leaf Node. Selection nodes are used to make any choice and feature multiple branches, whereas Leaf nodes are the output of these decisions and do no longer incorporate any in addition branches. The selections or the check are completed on the idea of functions of the given dataset. It's far a graphical illustration for buying all the possible answers to a trouble/decision primarily based on given situations. It's miles referred to as a choice tree due to the fact, just like a tree, it begins with the foundation node, which expands on further branches and constructs a tree-like shape. In an effort to build a tree, we use the CART set of rules, which stands for class and Regression Tree set of rules. A decision tree sincerely asks a query, and based on the answer (sure/No), it further cut up the tree into subtrees . Below diagram explains the general structure of a decision tree:[37]

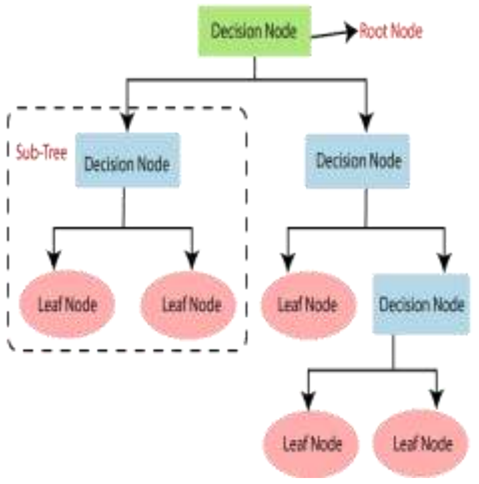


Fig.4. Decision Tree Scheme

D. Logistic Regression

Logistic regression is one of the most popular Logistic regression is one of the most famous machine learning algorithms, which comes below the Supervised machine learning method. It's far used for predicting the categorical established variable the use of a given set of independent variables. Logistic regression predicts the output of a specific established variable. Therefore the outcome should be a categorical or discrete fee. It is able to be both yes or No, zero or 1, true or false, and so on. But in preference to giving the exact cost as 0 and 1, it gives the probabilistic values which lie among 0 and 1. [38]

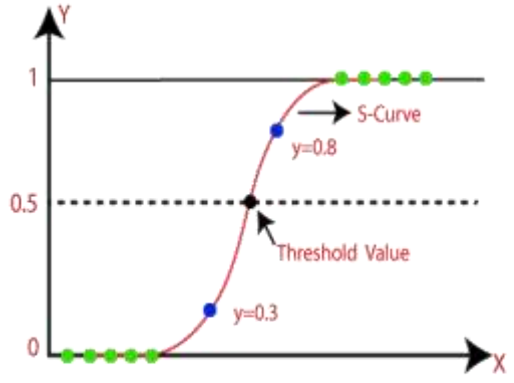


Fig.5. Logistic Regression Scheme

Logistic Regression is a lot similar to the Linear Regression except that how they may be used. Linear Regression is used for solving Regression problems, while Logistic regression is used for solving the classification troubles. In Logistic regression, instead of fitting a regression line, we healthy an "S" formed logistic characteristic, which predicts two maximum values (0 or 1).The curve from the logistic function suggests the likelihood of something consisting of whether or not the cells are cancerous or not, a mouse is obese or not based on its weight, and many others. Logistic Regression is a huge machine learning set of rules because it has the ability to provide chances and classify new information the usage of continuous and discrete datasets. Logistic Regression may be used to categorise the observations the use of exclusive kinds of records and may without problems determine the only variables used for the category.

E. KNN Algorithm

k-Nearest Neighbour is one of the only system getting to know algorithms primarily based on Supervised gaining knowledge of method. K-NN set of rules assumes the similarity between the new case/data and available cases and put the brand new case into the category that is maximum just like the available categories. K-NN set of rules stores all of the available facts and classifies a brand new facts factor primarily based at the similarity. This means when new facts appears then it may be effortlessly categorized into a nicely suite category by means of using K- NN set of rules. K-NN set of rules may be used for Regression as well as for class but in general it's miles used for the category troubles. K-NN is a non-parametric set of rules, which means it does not make any assumption on underlying records. It's also called a lazy learner algorithm as it does not analyze from the education set at once alternatively it stores the dataset and on the time of type, it plays an action on the dataset.KNN algorithm on the learning phase simply stores the dataset and while it receives new records, then it classifies that records into a class that is tons just like the new facts.[39]

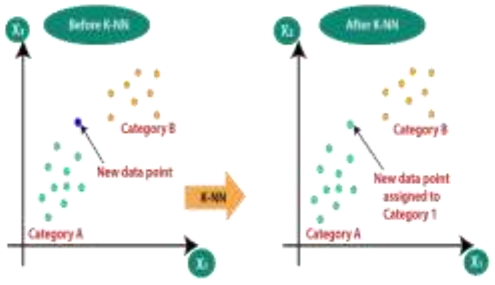


Fig 6: KNN Algorithm

V. METHODOLOGY

This section presents the methodological gear, steps and tactics utilized in reaching the observe goals.
A. Data Preparation

The use of Machine-learning techniques is essential in attaining software program reusability, maintainability and exceptional because it enables with finding the bas scent, ambiguity, fault and defect in software program. Carrying out this requires software program default prediction techniques, which rely on statistical strategies to any software defects [24]. But, software program detection can also be performed through machine learning strategies.

B. Datasets

A set of NASA information were accumulated and research is being made for this reason. In current years, those datasets have drawn a notable quantity of attentions from researchers. The information of the datasets are proven in the table. Pre-processing allows shape the information into a shape that the category engine can use [30,31]. Key benefits of pre-processing consist of normalizing numeric facts and helping fill in missing information.
The experiments depended on datasets drawn from the PROMISE facts repository amassed from actual NASA software tasks and diverse software modules. The benchmarking involved the use of public area datasets. This benchmarking manner allows other researchers evaluate their studies. A number of the code metrics used inside the datasets include McCabe's cyclomatic complexity, code length and Halstead's complexity among others. The outline of Datasets is summarized inside the table 1. The goal variables inside the NASA MDP facts units are binary in nature, 1: real, 0: false. Table 2 suggests the overall performance matrices that are used in this have a look at. Python programming and Scikit-analyze (machine learning framework) is used in data examination.

Variables	Description	Metrics Type
Loc	Line count of Code	McCabe
v(g)	Cyclomatic Complexity	McCabe
ev(g)	Essential Complexity	McCabe
iv(g)	Design Complexity	Halstead

N	Total operators and Operands	Halstead
V	Volume	Halstead
L	Program Length	Halstead
D	Difficulty	Halstead
I	Intelligence	Halstead
E	Effort	Halstead
B	Number of Bugs	Halstead
T	Time estimator	Halstead
lOCode	Line Count	Halstead
lOComment	Line count of Comments	Halstead
lOBlank	Count of Blank Lines	Halstead
lOCodeAndComment	Lines of Comment and Code	N/A
Uniq_Op	Unique Operators	Halstead
Uniq_Opnd	Unique Operands	Halstead
Total_Op	Total Operators	Halstead
Total_Opnd	Total Operands	Halstead
branchCount	Flow Graph's Branch Count	Halstead
Problems	Reported Defects	N/A

Table 1: NASA MDP Dataset

C. Preprocessing Step (normalization)

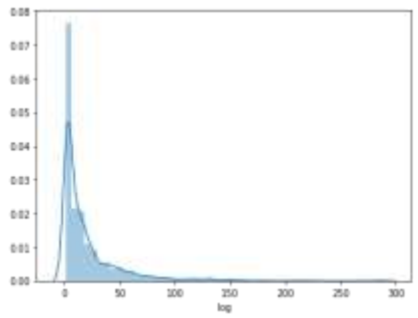
On account that each sample carries unique values which could vary substantially, function scaling, which is considered one of popular techniques in normalization, is performed to normalize capabilities (impartial variables). To achieve this aim, standardization technique is selected and used for this segment. Standardization is broadly used for normalization in lots of system getting to know algorithms. Feature normalization is performed according to the formula below,

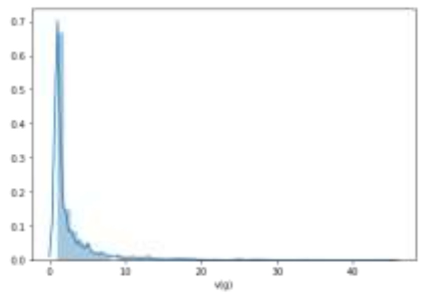
$$z = \frac{x_i - \mu}{\sigma}$$

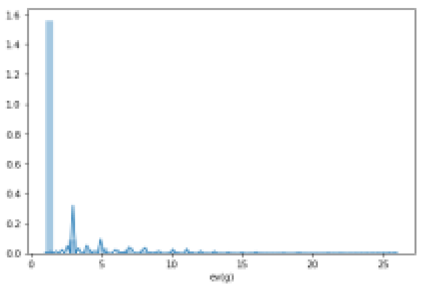
Where x_i defines the ith data dimension, μ is average, and defines standard deviation of that dimension.

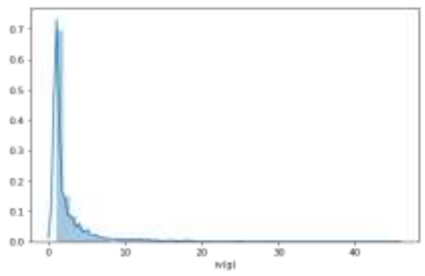
Datatypes are printed,shape of the data is taken, no missing values and outliers are removed.After removing we got Highest mean =109.635 and lowest value =-68.89. The below table gives the result of dataset (log) after removing outliers.

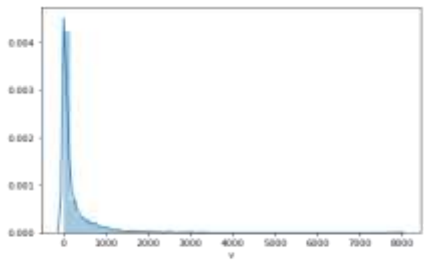
Count	168
Mean	6.75
Std	0.807
Min	6.0
25%	6.0
50%	7.0
max	8.0

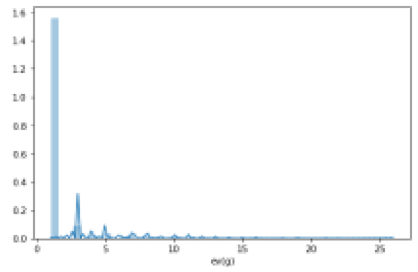

loc

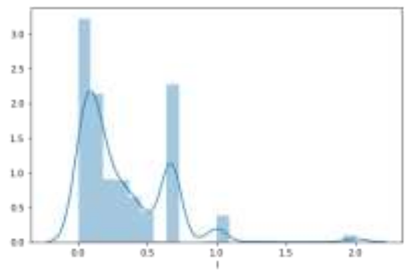

v(g)

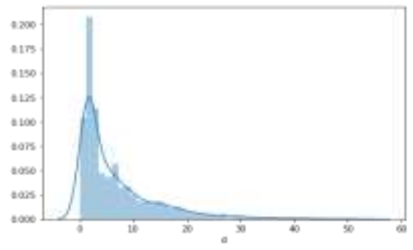

ev(g)

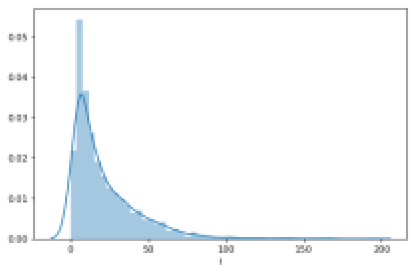

iv(g)

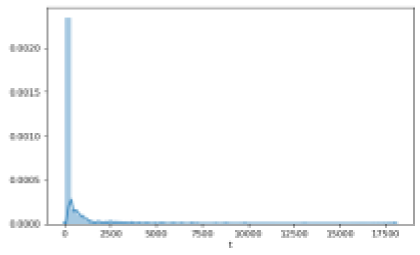

v

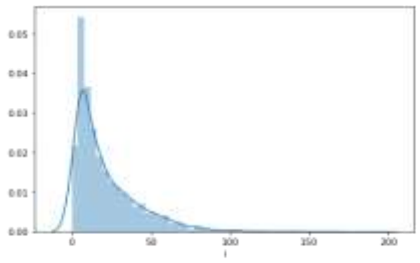

ev(g)

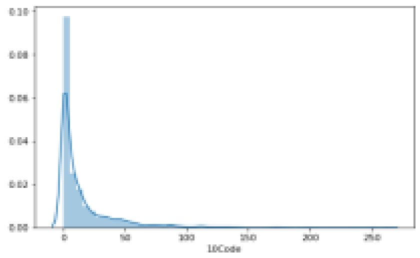

l

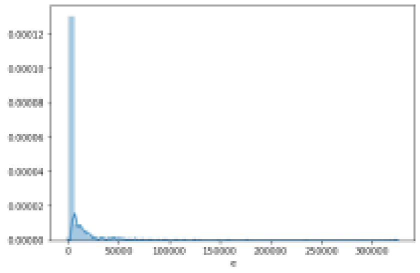

d

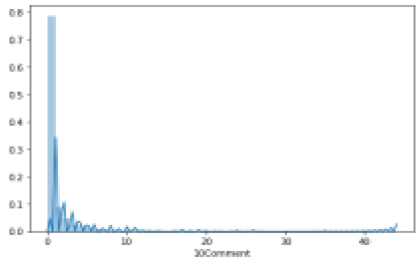

i

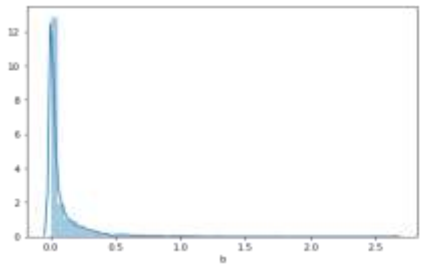

t

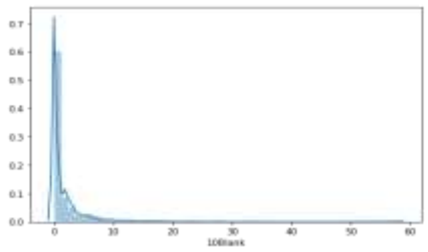


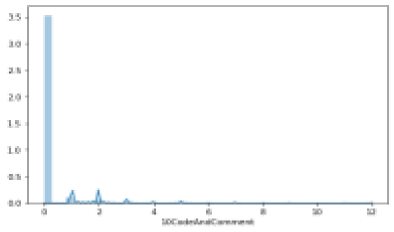


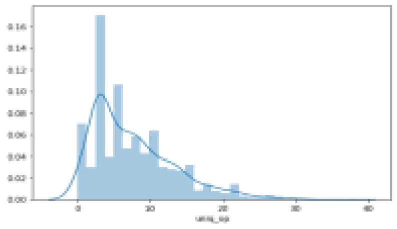


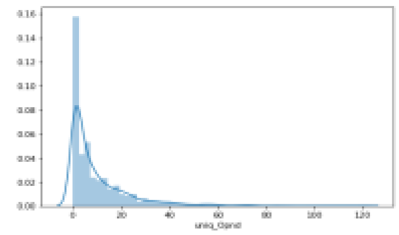


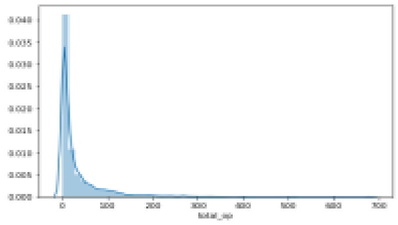


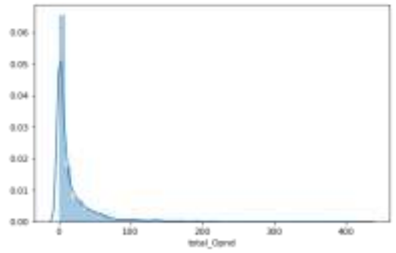


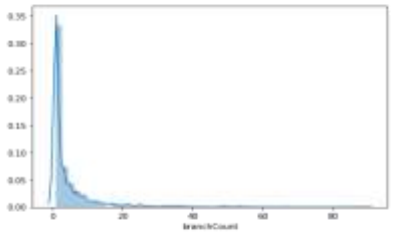












*The above graphs represents distribution plots .

D. Classification

Solving the translation problem allowed the creation of numerous classification algorithms, which can be customizedin line to flows to defect, fragments or machining source codetokens. Each classifier comes with different strength and weaknesses aimed to fit specific needs. Finally, the performance of the mentioned algorithms is measured based on the performance metrics in Table 2.

Table 2: Performance Matrices

Performance Matrices	Formula
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
F1	$\frac{2 * Recall * Precision}{Recall + Precision}$
MAE	ǀTrue values-Predicted values ǀ

E. K-Fold Cross Validation

After normalization step, a 10-fold cross-validation strategyis applied to compute the parameters of the test set. Each dataset is randomly partitioned into K subsets, each of which is equal to others in terms of its size and one of which is considered as test data every time while the other k-1 subsets are considered as training data. This action should be reiteratedten times for running same algorithm on data. Finally, the mean of these k runs is computed.

VI. RESULTS

The results of the different ML techniques for defect prediction using various datasets are shown in Table 3, 4,5,6,7 and 8. The training was performedbased on 10-fold cross validation.

Table 3: Performance of Supervised Learning Algorithms

Datasets	**Svm**	**Naïve Bayes**	**Logistic regression**	**Decision tree**
CM1	89.1	84.4	89.1	85.4
KC1	81.6	78.4	79.1	76.6
KC2	80.89	81.9	81.2	81.5
KC3	73.1	78.89	75.59	71.3
KC4	80.2	78.8	81.19	78.6
MC1	80.4	60.3	78.78	62.5
MC2	85.9	84.9	87.75	84.4
PC1	96.5	97.18	70.82	97.0
PC2	62.61	60.31	97.63	85.8
PC3	96.54	97.18	70.82	95.4
PC4	89.71	60.31	97.63	79.9
PC5	96.2	97.18	92.1	97.7
JM1	80.77	60.31	96.8	79.9
MW1	72.9	97.1	84.42	79.9
Mean	84.6	82.6	86	82.5

Table 4: Performance of classifiers with cross validation

Datasets	Svm	Naïve Bayes	KNN
CM1	91.3	83.43	82.1
KC1	71.8	74.6	85.9
KC2	80.86	80.54	96.5
KC3	73.8	70.3	62.61
KC4	80.2	75.6	96.54
MC1	72.4	61.5	89.71
MC2	90.73	82.48	96.2
PC1	93.5	94.18	96.5
PC2	75	57.31	62.61
PC3	91	96.18	96.54
PC4	89.32	52.31	89.71
PC5	84.9	92.18	96.2
JM1	90.15	58.31	80.77
MW1	94.07	95.18	72.9
Mean	85.6	81.6	85.87

Table 5: F-measure Performance of Supervised Learning Algorithms

Datasets	Svm	Naïve Bayes	Logistic regression	Decision tree
CM1	86.1	81.4	85.1	83.43
KC1	78.6	74.4	78.1	74.6
KC2	76.89	77.9	78.2	80.54
KC3	71.1	75.89	71.59	70.3
KC4	74.2	74.8	79.19	75.6
MC1	78.4	59.3	76.78	61.5
MC2	81.9	83.9	84.75	82.48
PC1	88.5	94.18	68.82	96.02
PC2	66.61	57.31	94.63	83.89
PC3	89.54	96.18	67.82	92.42
PC4	85.71	52.31	95.63	78.9
PC5	92.2	92.18	89.1	96.77
JM1	77.77	58.31	95.8	78.9
MW1	69.9	95.18	82.42	77.94
Mean	80.6	78.6	83.87	77.51

Table 6: F-measure Performance of classifiers with cross validation.

Datasets	Svm	Naïve Bayes	KNN
CM1	80.2	75.31	76.78

KC1	80.4	96.18	84.75
KC2	85.9	81.4	68.82
KC3	96.5	74.4	94.63
KC4	62.61	77.9	67.82
MC1	96.54	75.89	95.63
MC2	89.71	74.8	89.1
PC1	96.2	59.3	95.8
PC2	84.9	83.9	80.54
PC3	90.15	94.18	70.3
PC4	94.07	57.31	75.6
PC5	78.1	96.18	61.5
JM1	76.02	52.31	80.14
MW1	80.01	92.18	79.12
Mean	83.5	78.16	83.87

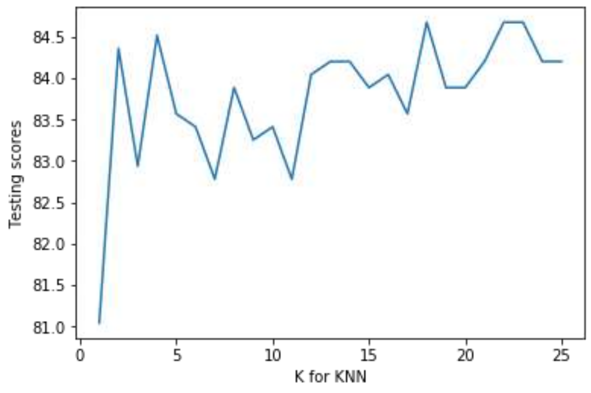
Table 7: MAE Performance of Supervised Learning Algorithms

Datasets	Svm	Naïve Bayes	Logistic regression	Decision tree
CM1	0.08	0.11	0.08	0.07
KC1	0.12	0.27	0.12	0.25
KC2	0.19	0.23	0.15	0.43
KC3	0.20	0.29	0.20	0.42
KC4	0.18	0.24	0.19	0.31
MC1	0.17	0.38	0.18	0.54
MC2	0.14	0.35	0.14	0.32
PC1	0.03	0.07	0.03	0.02
PC2	0.27	0.33	0.10	0.32
PC3	0.07	0.07	0.02	0.37
PC4	0.11	0.22	0.53	0.21
PC5	0.09	0.07	0.49	0.40
JM1	0.80	0.22	0.04	0.51
MW1	0.14	0.47	0.25	0.54
Mean	0.18	0.22	0.17	0.28

Table 8: MAE Performance of Ensemble LearningAlgorithms

Datasets	Svm	Naïve Bayes	KNN
CM1	0.09	0.07	0.49
KC1	0.80	0.33	0.04
KC2	0.14	0.07	0.25
KC3	0.20	0.22	0.15
KC4	0.04	0.58	0.20
MC1	0.12	0.87	0.19
MC2	0.10	0.35	0.18
PC1	0.14	0.48	0.14
PC2	0.04	0.15	0.03

PC3	0.12	0.02	0.10
PC4	0.14	0.09	0.02
PC5	0.25	0.31	0.15
JM1	0.52	0.23	0.14
MW1	0.19	0.12	0.18
Mean	0.17	0.24	0.17



graph of KNN for each K's value how testing score change

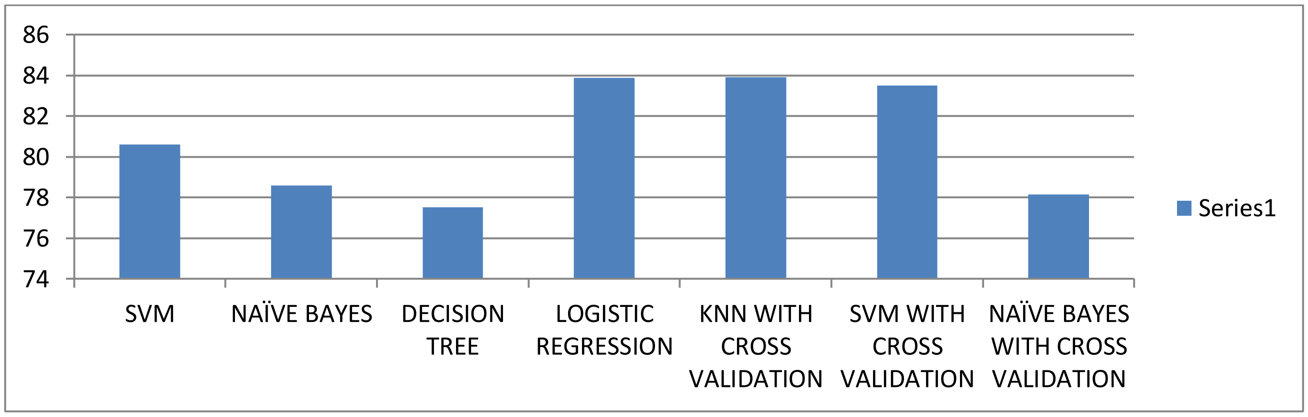


Figure 1: Accuracy Chart of Different algorithms

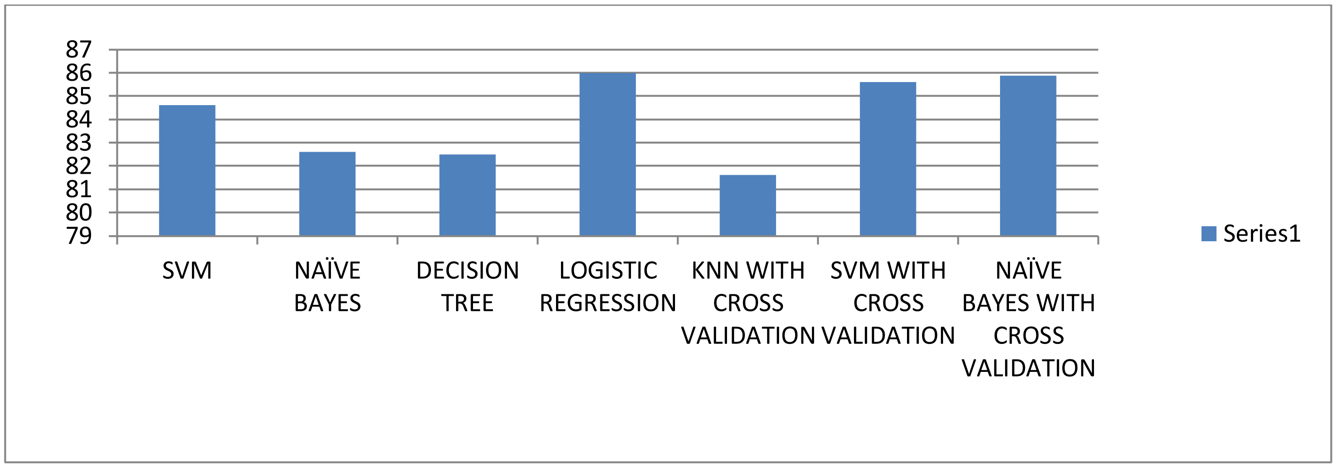


Figure 2: F-Measure of Different Algorithms

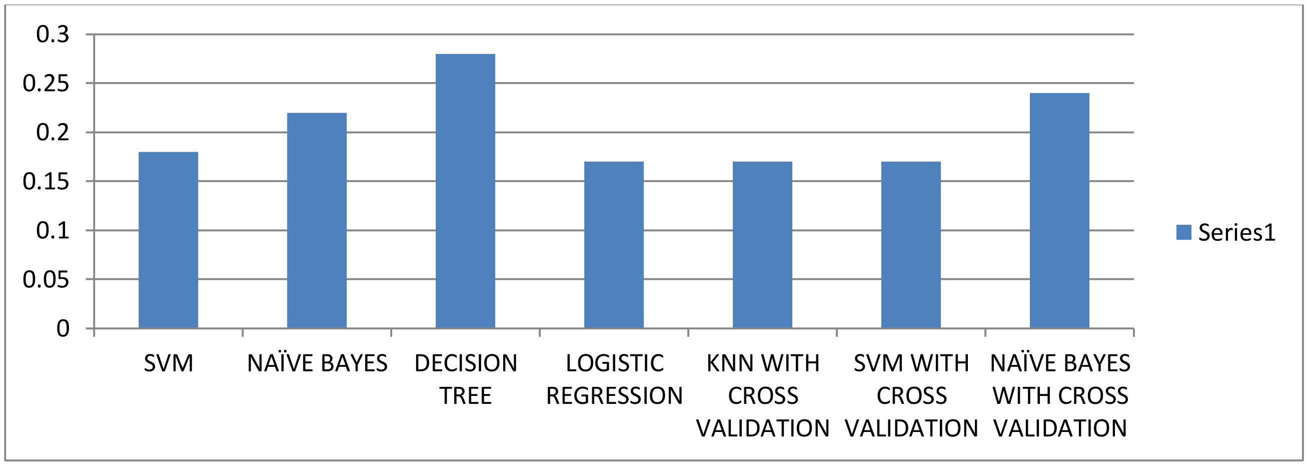


Figure 3: MAE Performance of Different Algorithms

Based on MAE score chart (Figure 3), all classifiers have lowest MAE score where Decision tree is on top. KNN with cross validation, logistic and SVM with cross validation acquired the lowest score among all the learning algorithms. Performance of both algorithms are slightly different.

Overall, Logistic regression performed well in all 3 performance measures and out performed all other algorithms. Algorithms with cross validation performed relatively well except naïve bayes. In supervised learning, SVM and logistic regression showed promising performance. In cross validation, SVM and KNN both performed well in all. Decision tree and Naïve Bayes performance is low compared to other classifiers. The proposed method has given the accuracy of 0.8515.

VII. CONCLUSION

Recent years have seen a growth in the development of software-based systems even though the quality of the system has to be best before delivery to the end-users. Software quality can be enhanced through several quality metrics such as ISO standards, CMM, and software testing. The need for software testing grows with each day, and its efficiency can be improved by using software defect prediction. The objective of this study was to investigate different software defect prediction models. We have used generative machine learning models in defect prediction process. As our experimental results shows, these models achieve higher accuracy. Although for some datasets the obtained results are impressive, some other datasets, which do not have sufficient samples and suffer from data unbalancing engendered poor results. However, Logistic Regression presents the best generalization ability with accuracy numerical mode. In future, we will investigate how deep learning models can affect the results.. The findings of this study show that how Logistic Regression can be used to defect prediction. It is our hope that these results will help increase the confidence in these models. In the future, we have to spent on time and resources when dealing with error-prone modules.

REFERENCES

1. L. C. Briand, K. El Emam, B. G. Freimut, and O. Laitenberger, "A comprehensive evaluation of capture-recapture models for estimating software defect content," IEEE Transactions on Software Engineering, vol. 26, no. 6, pp. 518–540, 2000.
2. S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," IEEE Transactions on Software Engineering, vol. 34, no. 4, pp. 485–496, 2008.
3. Y. K. Malaiya and J. Denton, "Estimating the number of residual defects [in software]," in High-assurance systemsengineering symposium, 1998. Proceedings. Third IEEE international, 1998, pp. 98–105.
4. J. E. Gaffney and C. F. Davis, "An approach to estimating software errors and availability," in Eleventh Minnowbrook Workshop on Software Reliability, 1988.
5. W. W. Agresti and W. M. Evanco, "Projecting software defects from analyzing Ada designs," IEEE Transactions on SoftwareEngineering, vol. 18, no. 11, pp. 988–997, 1992.
6. H. Cao, Z. Qin, and T. Feng, "A Novel PCA-BP Fuzzy Neural Network Model for Software Defect Prediction," Advanced Science Letters, vol. 9, no. 1, pp. 423–428, 2012.
7. K. Gao and T. M. Khoshgoftaar, "Software Defect Prediction for High-Dimensional and Class-Imbalanced Data.," in SEKE, 2011, pp. 89–94.
8. Erturk, and E.A. Sezer. **A comparison of some soft computing methods for software fault prediction**. Expert systems with applications, 42(4), 1872-1879, 2015. https://doi.org/10.1016/j.eswa.2014.10.025.
9. M.K. Albzeirat, M.I. Hussain, R. Ahmad, F.M. Al- Saraireh, A. Salahuddin, and N. Bin-Abdun. **Applications of Nano-Fluid in Nuclear Power Plants within a Future Vision**. International Journal of Applied Engineering Research, 13(7), 5528-5533, 2018 .
10. S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. **Benchmarking classification models for software defect prediction: A proposed framework and novel findings.** IEEE Transactions on Software Engineering, 34(4), 485- 496, 2008.
11. M. Singh, and D.S. Salaria. **Software defect prediction tool based on neural network**. International Journal ofComputer Applications, 70(22), 2013.
12. X. Tan, X. Peng, S. Pan, and W. Zhao. **Assessing software quality by program clustering and defect prediction**. 18th working conference on Reverse Engineering, pp. 244-248, 2011.
13. N. Li, M. Shepperd, and Y. Guo. **A systematic review of unsupervised learning techniques for software defect prediction**, Information and Software Technology, Vol. 122, 106287, 2020 https://doi.org/10.1016/j.infsof.2020.106287
14. Y. Chen, X. Shen, P. Du, and B. Ge, "Research on software defect prediction based on data mining," in Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on, 2010, vol. 1, pp. 563–567.
15. M. Gayathri and A. Sudha, "Software defect predictionsystem using multilayer perceptron neural network with data mining," International Journal of Recent Technology and Engineering (IJRTE), vol. 3, no. 2, pp. 54–59, 2014.
16. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," Information Sciences, vol. 264, pp. 260–278, 2014.
17. Arora, V. Tetarwal, and A. Saha, "Open issues in software defect prediction," Procedia Computer Science, vol. 46, pp. 906–912, 2015.
18. T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," IEEE transactions onsoftware engineering, vol. 33, no. 1, 2007.
19. Nam, S. J. Pan, and S. Kim, "Transfer defect learning,"in Proceedings of the 2013 International Conference on Software Engineering, 2013, pp. 382–391.
20. B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "Onthe relative value of cross-company and within-

company data for defect prediction," Empirical Software Engineering, vol. 14, no. 5, pp. 540–578, 2009.

21. Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," Information and Software Technology, vol. 54, no. 3, pp. 248–256, 2012.
22. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model," in Proceedings of the 11th Working Conference on Mining Software Repositories, 2014, pp. 182–191.
23. T. Fukushima, Y. Kamei, S. McIntosh, K. Yamashita, and N. Ubayashi, "An empirical study of just-in-time defect prediction using cross-project models," in Proceedings of the 11th Working Conference on Mining Software Repositories, 2014, pp. 172–181.
24. Nam and S. Kim, "Heterogeneous defect prediction," in Proceedings of the 2015 10th joint meeting on foundations of software engineering, 2015, pp. 508–519.
25. S. Watanabe, H. Kaiya, and K. Kaijiri, "Adapting a fault prediction model to allow inter languagereuse," in Proceedings of the4th international workshop on Predictor models in software engineering, 2008, pp. 19–24.
26. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A.Panichella, and S. Panichella, "Multi-objective cross-project defect prediction," in Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on, 2013, pp. 252–261.
27. C. Catal, U. Sevim, and B. Diri, "Clustering and metrics thresholds based software fault prediction of unlabeled program modules," in Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on, 2009, pp. 199–204.
28. S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Unsupervised Learning for Expert-Based Software Quality Estimation.," in HASE, 2004, pp. 149–155.
29. M.S. Naidu, and N. Geethanjali. **Classification of defects in software using decision tree algorithm**. International Journal of Engineering Science and Technology, 5(6), 1332, 2013.
30. W.H.W. Ishak, K.R.K. Mahamud, and N.M. Norwawi. **Modelling of Human Expert Decision Making in Reservoir Operation**, Journal Teknologi, 77(22), 1-5, 2015.
31. W.H.W. Ishak, K.R.K. Mahamud, and N.M. Norwawi.**Intelligent Decision Support Model Based on Neural Network to Support Reservoir Water Release Decision,** In J.M. Zain et al. (Eds.): ICSECS 2011, Part I, Communications in Computer and Information Science (CCIS) 179, pp. 365-379, 2011. https://doi.org/10.1007/978-3-642-22170-5_32
32. M.A. Hall. **Correlation-based feature selection of discrete and numeric class machine learning**, Proceedings of the Seventeenth International Conference on Machine Learning, pp. 359-366, 2000
33. S. Karim, H. L. H. S. Warnars, F.L. Gaol, E. Abdurachman, and B. Soewito. **Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset**. IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), pp. 19-23, 2017.
34. Kaur, and R. Malhotra, **Application of Random Forest in Predicting Fault-Prone Classes**, International Conference on Advanced Computer Theory and Engineering, pp. 37-43, 2008, doi: 10.1109/ICACTE.2008.204.
35. https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm.
36. https://www.javatpoint.com/machine-learning-naive-bayes-classifier.
37. https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
38. https://www.javatpoint.com/logistic-regression-in-machine-learning
39. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning.
40. S. Aleem, L.F. Capretz, and F. Ahmed. **Benchmarking machine learning technologies for software defect detection**. International Journal of Software Engineering & Applications (IJSEA), 6(3), pp. 11-23, 2015.
41. Bibi, S., Tsoumakas, G., Stamelos, I., &Vlahavas, I. P. (2006, March). Software Defect Prediction Using Regression via Classification.In AICCSA (pp. 330-336.
42. El Emam, K., Melo, W., & Machado, J. C. (2001).The prediction of faulty classes using object-oriented design metrics. Journal of Systems and Software,56(1), 63-75.
43. Zhao, M., Wohlin, C., Ohlsson, N., &Xie, M. (1998). A comparison between software design and code metrics for the prediction of software fault content. Information and Software Technology, 40(14), 801-809.
44. Tomaszewski, P., Lundberg, L., &Grahn, H. (2005). The accuracy of early fault prediction in modified code.In Proceedings of the Fifth Conference on Software Engineering Research and Practice in Sweden (SERPS) (pp. 57-63).
45. Challagulla, V. U. B., Bastani, F. B., Yen, I. L., & Paul, R. A. (2008).Empirical assessment of machine learning based software defect prediction techniques. International Journal on Artificial Intelligence Tools, 17(02), 389-

400.

46. Ratzinger, J., Sigmund, T., & Gall, H. C. (2008, May). On the relation of refactorings and software defect prediction. In Proceedings of the 2008 international working conference on Mining software repositories (pp. 35-38). ACM.
47. J. Tian, and M.V. Zelkowitz. **Complexity measure evaluation and selection,** IEEE Transactions on Software Engineering, 21(8), 641-650, 1995. https://doi.org/10.1109/32.403788.
48. M. Shepperd, D. Bowes, and T. Hall. **Researcher bias: The use of machine learning in software defect prediction**. IEEE Transactions on Software Engineering, 40(6), 603-616, 2014.