ISSN 2278-3091
Volume 8, No.4, July – August 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse38842019.pdf
https://doi.org/10.30534/ijatcse/2019/38842019

A Software Defect Learning and Analysis Utilizing Regression Method for Quality Software Development

Rohini B. Jadhav[1] , Shashank D. Joshi [2], Umesh G. Thorat[3] , Aditi S. Joshi[4]
[1]Research Scholar, Bharati Vidyapeeth (Deemed to be University) College of Engineering Pune, India, rohini.jadhav@outlook.com
[2]Professor, Bharati Vidyapeeth (Deemed to be University) College of Engineering Pune, India,sdj@live.in
[3]Solution Architect, Tata Consultancy Services Pune, India,umesh.thorat@tcs.com
[4]Research Scholar, Pune Institute of Computer Technology Pune, India,aditijoshi14@gmail.com

ABSTRACT
The program is a complex object consisting of different units with variable degrees of defects. By predicting the effectiveness and frequency of program defects, program managers can make better use of manpower, cost, and time to obtain better quality assurance. It is always possible to have a set of defects that affect designed and predictable units in order to have close association with the subsidiaries. Most of the current defect prediction rating mechanism is derived from learning the previous project data, but it is not sufficient to predict the defect of the new project because the new design may contain a different type of parameter. This paper proposes a **Software Defect Learning and Analysis utilizing Regression Method (SDLA-RM)** to detect defects and plan a better maintenance strategy, which can support the prediction of a defective or non-defective software unit prior to deployment in any project programs. The SDL-RM mechanism extends Regression Analysis (RAM) to create an effective rule-based model for accurately classifying program faults. This approach improves the predictability of software defects, allowing software development to spend more time testing components that are expected to contain errors. The experimental evaluation is carried out across the NASA-PROMISE repository data sets, that outcome of the results in comparison with existing classifiers suggest the effectiveness and practical perspective in the software development.

Key words: Defect Analysis, Regression Method, Software Defect Learning, Software Development.

1. INTRODUCTION

In software development it is always challenging to prevent defects in the applications because the application is too complex and not able identify any errors or defects. These defects of the program represent as errors or defects in program or program operations and focus mainly on predicting defects that affect the performance of the project or product. Predicting software failure helps detect, track, and resolve anomalies in programs, especially important security systems that can affect safe user retention and age. Predictability reduces program costs and improves customer satisfaction [1]. However, there is no technique for predicting software defects that can solve all defect problems. Therefore, by predicting effectively and appropriately the occurrence of program defects, program project managers can make better use of costs and time to obtain enhanced quality assurance [2-4]. A diversity of methods, tools and tools have been proposed to prevent defects, but they seem insufficient to accurately predict. More work is still needed to prevent defects in terms of Defect Prediction (DEF-PR) technology and schemes utilization. Classifications and predictions [5, 6] can be used to extract models that describe critical defect data categories or predict trends in future defects. The classification predicts specific or distinctive labels and labels in disorder, while the predictive model predicts functions of continuous value. This analysis helps us better understand software defect data. Preventing defects is critical to the quality of the organization. The main purpose of quality costs is not to reduce costs but to save costs in appropriate investments. The joy should not be a waste of time while providing for deep participation. Instead, it should consider saving the time, money and resources are need. It can provide many of the rework it needs when defects appear in the final or post-delivery period. At each stage of the program life cycle, fault prevention should be provide to prevent early failure, corrective action to be taken to eliminate it and avoid recurrence.

The "Regression analysis methods (RAM)" [7, 8] is a commonly used statistical technique for studying the linear relationship between variables. This analytical technique can be useful for the identification the relation among the various types of defects, as these are distributed to a wide range of features. The "Multiple-RAM" are an extension of linear to RAM that contains one or more predictor variables [9]. It assigns the interaction variable to be modelled as a linear function for variables or predictive feature properties as, "F_1, F_2, . . . , F_n", and describing a defect as D, which consists of the attributes as, "$D = (x_1, x_2, \ . \ . \ . \ , x_n)$". The training data set, T, contains data in the form "(D_1, C_1), (D_2, C_2), . . . ,

($D_{|T|}$, $C_{|D|}$)", where, the "D_i → are the "*n*-dimensional training data", and "C_i → is the related class labels to the defect".

This paper provides an understanding of the software defect and its analysis using the regression method (SDL-RM) for defect prediction proficient in software applications. The main principle in SDL-RM is to build a model of the actual rule for categorizing software defects accurately utilizing the repository of "NASA-PROMISE" [10] for the experimental assessment. Because each organization tries to maintain the privacy of its data, and it cannot publish data sets that can be used in its product development for the experiment evaluation. It is the most common data sets being provided by the NASA project modules. SDLA-RM applies RAM to learn an defect-related data sets to create defects knowledge for utilize in the developing a RAM-based defect-prediction classifiers to improve defect prediction approaches.

The following paper is organized as, Section 2 discusses the background works, Section 3 discusses the proposed SDL-RM approach, Section 4 provides an assessment of the experiment evaluation using data sets, and Section 5 discusses the paper conclusion.

2. BACKGROUND WORK

Suggestions for preventing software defects are usually based on tools, technologies, methods, and standards. In the field of software engineering, it is one of the active research focus [11]. Since DEF-PR models include a group of "defect-prone software artefacts" [12], "quality assurance teams" which efficiently assign inadequate resources to test and examine software products [13, 14].

Several DEF-PR studies [15-17] have been carried out and all are based on an ML approach or a statistical approach. ML algorithms are used in software failure prediction models for classification and regression [18]. Many studies have recently used ML techniques to improve the predictability of defects. Pre-processing techniques are also important in software DEF-PR. In order to progress the performance of the ML technique before constructing the DEF-PR model, the following techniques such as "feature selection" and "data normalization and noise reduction" able to applied [19]. Several feature selection techniques are used to extract important functions for the DEF-PR model. However, several studies [20] have shown that predictive performance able to improve by processing techniques, and several studies have not been applied because they have considered that traffic techniques may be optional and able to ignore.

2.1 Regression Support for Prediction

Regression techniques being utilized to advance software quality by utilizing software metrics to predict defect counts in software modules [6-8]. It can assist developers to distribute inadequate resources to modules that contain further defects. The regressive analysis is a method associated with the outcome of the variable and the interaction of one or extra threat features or mystifying variables. The results variable is also identified as reacting or reliant variables and threat features and confounders are identified "predictors", or "explanatory" or "self-sufficient variables". In RA, the relented variable is indicated as "y" and the self-sufficient variables are indicated as "x".

2.1.1 Correlation Analysis

The process of correlation or association analysis approximates the "correlation coefficient (CC)" for a sample data denoted as *R*. It association value "*R* ranges between -1 and +1", which enumerate the direction and potency of the linear relationship among the two variables. Correlation among two variables is "positive" if the variable is associated with higher or higher if the variable is correlated with the shorter level of the erstwhile level.

The value of the CC indicates the direction of the association. The CC indicates the potential of the association. For instance, the "correlation of R = 0.9" recommends that there is a well-built positive connection among the two variables, and the "correlation of R = -0.2" suggests a weak or negative relation. Correlation of "zero" proximity does not allow the existence of a linear connection among two consistent variables.

It is significant to the reminder that there perhaps two continuous variables of nonlinear communication, but it will not be revealed when calculating the CC. Therefore, it is constantly significant to assess the data cautiously prior to the CC.

2.1.2 Multiple Regression Analysis

The regressive analysis is widely utilized in techniques that are constructive for estimating "multiple self-sufficient variables" [9]. As a consequence, it is especially valuable for the assessment and adjustment. It can as well utilized to measure the existence of amendment. In many applications, there is more than one factor that affects the reaction. So many regression models describe how one response variable is $\mathcal{P}$ depending on a number of predictive variables. Multiple Linear RA is a derivation of the simple linear RA, which is utilized to evaluate among more than two self-sufficient variables and one constantly relented variable. The multiple linear RA equation is defined as follows:

$$`\mathcal{P} = \alpha + \beta 1X1 + \beta 2X2 + \ . \ . \ . + \beta_v X_v \qquad (1)$$

where $\mathcal{P}$ - "is the predicted or expected value of the relented variable", α - "is the constant or intercept value of $\mathcal{P}$", β_1 to β_v - "are the estimation regression coefficients", and X_1 to X_v - "are *v* distinct self-sufficient or predictor variables".

Each regressive coefficient is a change in relation to the change in one of the individual self-sufficient variables. In the case of numerous regressions, β_1, for example, $\mathcal{P}$ is a uniform transform in X_1, relative to the entire other self-sufficient variables, where the outstanding self-sufficient variables are supposed in the same sense or are observed. Once again, arithmetical tests able to executed to evaluate whether every regressive coefficient is substantially dissimilar from zero.

Multiple regressive analysis is also utilized to determine if there are conflicts. After multiple linear RA permits to assess the relations of the self-sufficient variable and the results of all the other inconsistent constant, it presents a direction to adjust the potentially confusing variables that are comprised in the model.

For instance, let's suppose we have a defect factor variable, which is denoted as X_1 and the predicted outcome will be presented as $\mathcal{P}$. Utilizing a simple linear regression estimation we can predict as to the relented defect variable as, $\mathcal{P} = \alpha + \beta_1 X_1$, where the β_1 is the approximated regression coefficient that calculates the relationship among the defect factor and the predicted outcome. Similarly, for more number of depended defect factor, we denote it from X_1 to X_n and its estimated regression coefficient as β_1 to β_n, and it is estimated in the form of multiple linear regression as $\mathcal{P} = \alpha + \beta_1 X_1 + \ . \ . \ . + \beta_n X_n$ .The estimation of regression coefficient, β of a defect quantifies the association between the defect and the prediction outcome.

Since the variable is recognized as a co-initiator, we utilize a multiple linear RA to evaluate the relationship defect factor and the results are adjusted to the confounder. The value of the defect's coefficient is associated with the defect factor in determining whether the assessment of the factor among the factor is statistically important on the explanation of one or more confusing variables.

Multiple Linear RA is widely utilized in various predictions of techniques [18]. The popular application is to evaluate the relationship among numerous predicting variables at the same time and one, the uninterrupted result. For instance, it perhaps interesting to see which of the forecasts determines a relatively large number of candidates, the most important or the most closely associated results. This is constantly essential in "statistical analysis", especially in varied fields that statistical modelling leads to associations.

2.2 Software Defect Prediction Techniques

Defects analysis in early stages [21, 22] reduce time, cost and resources. Knowledge of the injecting methods and processes of the defect allows the defect to be avoided. After this knowledge in practice, the quality has improved. Defects can be prevented based on the underlying causes of defects. The analysis can take two appearances, namely "logical analysis" and "statistical analysis". Logical Analysis is an exhaustive analysis of human consumption that requires knowledge, process, development and environmental expertise. It inspects logical relations among faults and bugs. Statistical analysis is derived from similar projects or empirical studies of local written projects.

There are numerous approaches to recognize defects such as "inspections", "prototypes", "testing" and "validation of evidence". The "Formal inspection" is the efficient and cost-effective quality [23, 24] early detection of defects identification technique. Several demands are clearly understood through prototype, which helps to overcome defects. Testing is one of the most effective methods. These defects [22], which have flown through early identification, can be detected during testing. Corrective evidence is also a good way to get out, especially in the coding stage and to construction is the most efficient and economical method for creating software.

The most DEF-PR models are founded on ML approaches. In relenting on what to need to predict, the model-based models are divided into two categories as, "classification" and "regression". After the introduction of new ML techniques, the methods of active or "semi-supervised learning" have been utilized to better DEF-PR models [15, 25]. Besides ML models few non-statistical models are also proposed such as "BugCache" [12].

Y. Liu et al. [24] discuss the problem of modelling of software quality has been studied, which uses the metric database history from a single software project. Classification modelling is not only an adequate, strong and accurate model from a single database. To solve this problem, the quality of the software classification was implemented utilizing different databases from different programs. Previous studies have shown that utilizing multiple data sets for validation can yield a robust genetic programming-based model. It shows that the proposed approach is more efficient and precise for utilizing multiple data sets.

S. Lessmann et al. [5] it has examined the classification algorithm. For comparison of software defect forecasts, it has tested experiments utilizing 10 public domain data utilizing the 22 classifiers from the NASA metric database storage. Predictable accuracy Metric-based classification is generally useful. The results also indicate that the value given for the specific classification algorithm is not as important as it is likely. The results showed no significant difference in the top 17 classification criteria.

J. C. Riquelme et al. [19] utilized the promise repository to acquire the software metric program was utilized and suggested searching the "Genetic algorithm (GA)" for searching the rules of the subdivision, which is due to high probability. The GA implements the difficulty of unbalanced data effectively, particularly when the unstable set consists of more unwanted samples than defective illustrations.

B. Turhan et al. [26] utilized to improve the prediction of the cross-company's defect utilizing the nearest neighbour's filter (NN filter). The main idea of the "NN filter" is to assemble related sources of instances in target cases in order to prepare the forecasting model. In erstwhile, if we are able to create a model of forecasting, the cases of selected sources that have similar data to the target, the model can be improved predict the target case than the model prepared from all sources. The NN filter selects 10 sources as near neighbours for each target-occurrence. Utilizing the NN filter to estimate the performance of the cross-section defect.

The complexity of the software designing and development need an efficient plan for DEF-PR. Therefore, to better plan your maintenance strategy, it is important to predict which software modules are defective before you deploy your software project. The initial knowledge of defected software modules is able to help to plan an effective procedure for enhancement at a realistic time and cost. This ability to lead to quality software in addition to superior customer fulfillment [27]. Software modules are characterized into two grouping, either "defective" or "non-defective", which mostly are prediction utilizing a "binary classification model". We acquire the improvement of these two classes prediction for suggestions on how to categorize and estimate the datasets in the next section.

3. PROPOSED SDLA-RM APPROACH

Classification and prediction of faulty methods are designed to perform accurate fault prediction, which is an essential problem in all software due to indirect measurements and is dependent on several metrics. This rule-based classification method

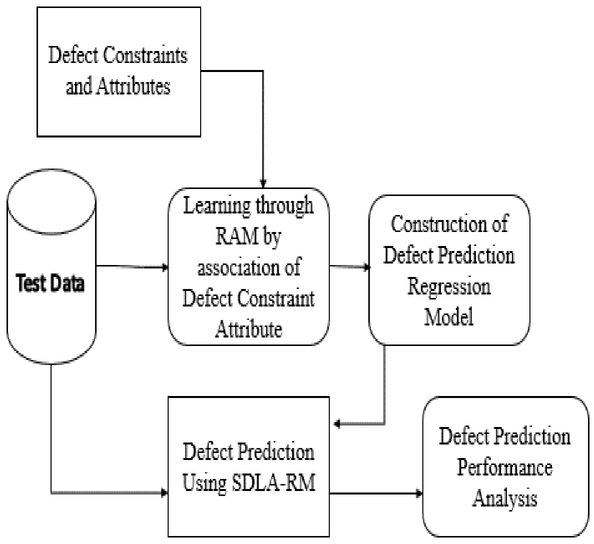
Defect Constraints and Attributes
Test Data
Learning through RAM by association of Defect Constraint Attribute
Construction of Defect Prediction Regression Model
Defect Prediction Using SDLA-RM
Defect Prediction Performance Analysis

Figure 1: SDLA-RM system framework

improves efficiency by inheriting methodology multiple-RAM to improve results and reduce the number and accuracy of rules. In our learning, we utilized a fault predictor utilizing the "static code properties" defined by "McCabe" and "Halstead" [28]. These are "module-based metrics" and it is the least unit of functionality in a complete system.
The proposed SDLA-RM reveals rules from past software defects based on learned multiple RAM associative defects to generate relevant and irrelevant rules for defect prediction in software development. The proposed SDLA-RM system framework is illustrated in Figure 1. The designed system architecture provides two main modules of learning methods through RAM and fault prediction utilizing RAM-rules. The following sections discuss learning and defect prediction mechanisms.

3.1 Learning through Regression Analysis

Association rules Mining and classification are often utilized in fault prediction to analyse the relationship between various attributes by defect type. To establish the rules needed for fault prediction, RA is utilized along with defect types and attributes to determine the relationship between different type defects.
In our framework, we use the ratio partitioning, which is utilized to estimate the performance of each predictive model. That is, every data set is primarily divided into two parts, the predictor is learned from the 60% instance, and the remaining 40% are tested. Regression-based rules contain "data structures" and "knowledge acquisition scenarios" derived from knowledge of human experts. The derived knowledge is coded into a group of rules. The process of learning from defect attributes and learning sets through the regression process is presented in Algorithm-1.
The regression-based learning process consists of two nested loops. The outer loop chooses the class value and the inner loop creates the rule until the class is applied. The function "*best_RPattern"* returns a combination of terms covering only the instances of the present class. The learning process makes use of an effortless term assortment through an empirical method depend on the probability that an instance will have a certain classification specified for a few "attribute-value" pairs.

```
Method:DEF_PR
Rules(Attrs,Training_set):DEF_PR_Rule_Set
Var RULE: DEF_PR_RULE
RAM_Rules: DEF_PR_Rule_Set
START:

For(“Defect_Class set_of_defect_class_values”) do
{
            While(“t: t. Training_set” and “t.class-
            Defect_Class”);
{
            Rule.Class:=Defect_Class;
            Rule.RAPattern :=best_RAPattern
            (“Defect_Class”,”Attrs,
            Training_Set”,”Constraints”);
            Remove(“Training_Set”,”Defect_Class”);
            RAM_Rules:=RAMRules.Rule;}
return(RAM_Rules);
}
end
```

“DEF_PR_Rule_Set” is a set of “DEF_PR_RULE”
“DEF_PR_RULE” is a structure with 2 components:

- Association_Pattern:”DEF_PR_Regression Patterns”
- Defect_Class: Class value predicted for an instance that matches regression patterns.

DEF_PR_Regression Patterns is a combination of “DEF_PR_terms”
DEF_PR_terms are the form of attribute

3.2 Regression Model Based Defect Prediction

Knowledge-based systems that have a large structure of concepts and rules are now being used in many applications. Acquiring knowledge of these systems when modern environments occur is a constant requirement that interaction between rules increases the complexity of the system. The regression rule mechanism creates a two-way dependency between the rules so that the rule creation is checked only in the context of creating another rule. If the introduction to the original rule is "correct" for a particular individual, the individual conclusion will be provided if there is no dependency. However, if they are "correct", the rule will be studied and claimed and the original result will be claimed only if the premises of the institution are valid for the legal entity. On the contrary, if the premise of the parental rule is wrong for a specific individual, this can not only call for conclusions, but if it has a "false false" subordination, it will also be tested and approved.

This regression rule forms a "binary decision tree" dissimilar from the "standard decision tree" in that it uses a compound clause to determine the branch, and this clause does not have to deal with all cases thoroughly to make decisions at the internal node. This contrasts with the standard tree where all decisions are made at the root node. However, the functionality of the "standard decision tree" remains that only one decision node is active for each case. Maintenance is simple because you need to consider the node only and the previous cases that were under it if there is a defect in reaching the decision. Extensions to regression rules include an extremely simple "statistical decision-making" method that generates a rule that is recursively called on the remaining data set to generate "if-true" and "if-false" rules as well as simple and simple rules It is very natural in point.

Defects and predictive analysis are performed utilizing regression rules generated in the following steps.

- First, the most frequent defect is the diagnosis of the portion of the data set that takes into account the selected target defects.
- Second, an assertion is initialized to associate the defect patterns with the *DEF_PR_ Regression Patterns.*
- Third, iteratively, each possible attribute value of the *DEF_PR_term* permutation is tested with a likely regression pattern and selects the finest according to the relevant *DEF_PR_term.*
- Finally, based on the similarity index of the defect pattern and the regression pattern, it is determined whether to determine the predicted defect according to the rule. If it is not predicted, the process repeats to the third stage and ends with a defect output prediction otherwise.

The data structure of the *DEF_PR_Rule* is in a decision tree structure where each node has rules for the fault class. This structure is interpreted in the form of conditional rules for each fault class as,

IF $cond_1$ AND $cond_2$ AND … AND $cond_N$ *are* TRUE **THEN** the defect conclusion. (2)

Each "*Cond" is* an attribute condition for the Boolean evaluation. For instance, a defect, D=1, if it's depended attribute "*Cond"* is also true. Each individual defect node has accurately two "successor nodes", these "successor nodes" are associated with its "predecessor node" by an "ELSE" or "EXCEPT" condition.

3.3 Defect Prediction Analysis

Classification is a process that is utilized to identify models that define and categorize an unclassified data class or concept of predicting the wrong object class whose model is unknown. Definition of proposed defect methodology aims to define a class of prediction class according to the selected attributes and restrictions imposed by the learning process.

The difficulty of deriving empirical DEF_PR is utilized by specifying the set of possible test conditions in the form of "S" for the entity universe of the entity "E" whose target predicate is "Q" in the materialized entity "E". The intention predicate can be conditional the rule set specifies the evaluation of the test predicate. With the intent of statistical regression, the emergence of "S" and "Q" is not important. It should consider "S" as an identifier to select "e" in the various separations of "E" that require "Q (e)", measure the assortment of rules by indiscriminate identification.

For example, if a computer program has identified with a set of faults as *E*, and the RA based algorithm predicted *Q* instances as having faults, and others are as *S*. But, form the predicted *Q* only "*x*" instances are having faults, and from the *S* instance only "*y*" has faults. So, the C can be as $(x+y)$ or "Q ∩ S" instances. According to these defined DEF_PR confusion matrix, a probability of the defect class will be predicted using RA rules.

4. EXPERIMENTAL EVALUATION

Experiments were performed utilizing the algorithm implemented in the "WEKA environment" [14] utilizing the "NASA - Metric Data Program (MDP) Repository". Below we discuss data sets, evaluation measurements, and analysis of results.

4.1 Dataset

The data set was taken from the PROMISE repository for a NASA project [10] consisting of 12 data sets. The data store shows software metrics, which are attributes of the data set, and whether a particular data set is "Defective" or "non-defective". Each data set consists of a number of software modules (cases), each containing an equivalent number of defects and diverse "software static code attributes". After pre-processing, one or more faulty modules are labelled as faulty. A more thorough explanation of the properties of the code or the origin of the data set able to found in [29].These four data sets are "CM1, JM1, KC1 and PC1", contain static code measurements such as, "Halstead, McCabe, and LOC", with defines the fault of defected codes. Table 1 presents the description for each of these data sets.

Table 1: Database Project Description

Project	Source Code	Description
CM1	C	NASA spacecraft instrument
KC1	C++	Storage management for receiving/ processing ground data
KC2	C++	Science data processing. No software overlap with KC1
JM1	C	Real-time predictive ground system
PC1	C	Flight software for earth orbiting satellite

Every one data set contains 21 software product metrics, depending on the "size", "complexity", and "vocabulary" of the product. The class attribute of each data set is "TRUE", meaning that the component has one or more defects and "FALSE" is defective.

4.2 Performance Measures

Performance is measured according to the confusion matrix given in Table 2, which is utilized by many researchers as in [29-30]. It shows confusing matrices for two class problems with positive and negative class values.

The software DEF_PR performance of a proposed plan based on "accuracy, sensitivity, and specificity" is defined as:

- Accuracy measures the percentage of DEF_PR that are correctly classified.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{3}$$

- Sensitivity measures the percentage of positive classified instances that predicted as positive.

$$Sensitivity = \frac{TP}{TP+FN} \tag{4}$$

- Specificity measures the percentage of positive classified instances that predicted as negative.

$$Specificity = \frac{TN}{FP+TN} \tag{5}$$

Actual Class	**Predicted Class**	
	Defective	**Not Defective**
Defective	True Negative (TN)	False Positive (FP)
Not Defective	False Negative (FN)	True Positive (TP)

Table 2: Confusion Matrix

4.3 Result Analysis

To compare the results with the best classifier for the prediction, such as "Naive Bayes", "OneR", "J48", and "RIDOR" To analyse the improvement of the proposal by comparing the results of the classification program execution of the WEKA Tool on the collected datasets. The comparative performance of the accuracy, specificity and sensitivity results of the proposed SDLA-RM is shown Figure 2, 3 and 4, respectively.
The proposed SDLA-RM shows an improvisation in the Accuracy value in comparison to existing classifier expect with the CM1 datasets. An average of **10% enhancement in the accuracy being achieved**. In the case of both sensitivity and specificity measure also it shows an improvisation The measure of sensitivity and specificity show the efficiency of the probability of detection of classifiers. The detection of the SDLA-RM utilizing the regression rules makes to predict defect accurately and enhance the sensitivity and specificity of the proposal.

- **Accuracy Analysis**

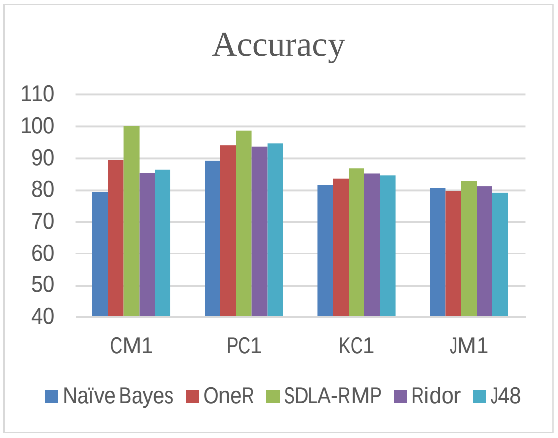

Figure 2: Accuracy Comparison

- **Sensitivity Analysis**

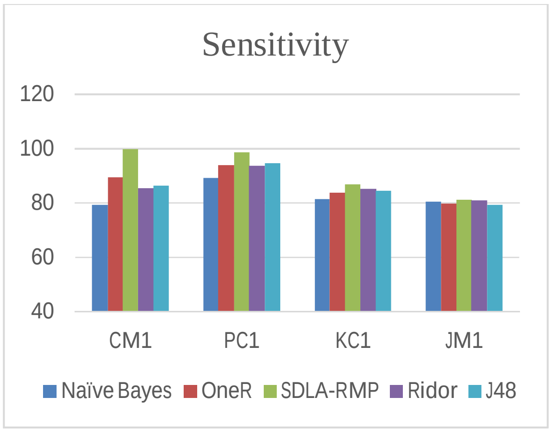

Figure 3: Sensitivity Comparison

- **Specificity**

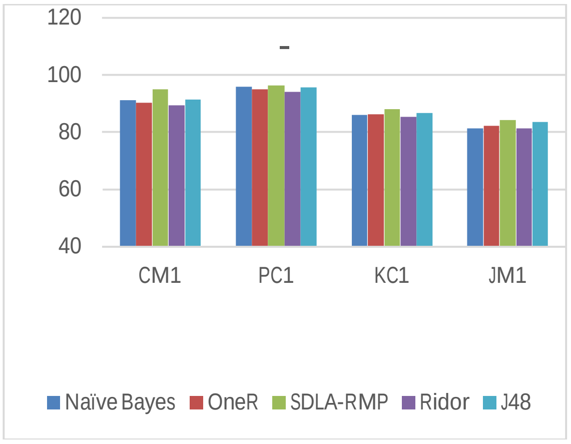

Figure 4: Specificity Comparison

5. CONCLUSION

Finding and correcting defects makes it easy for developers to understand the program. In order to improve the efficiency and excellence of software development, it is able to take advantage of data mining techniques to analyse and predict a large number of flawed data in software aggregation. This paper presents a Software Defect Learning and Analysis utilizing Regression Method (SDLA-RM). The approach designed to predict the defects using the learning method and RAM rule through regression analysis. The learning method creates rules with two types of exceptions that are easy to understand and find search rules automatically, so the designer does not have to actually do so. The rule is a hierarchy of features that have been enhanced to fit known design flaws. The empirical analysis shows the improved performance in the prediction of defects assessed according to the current classification methods. In future improvement, SDLA-RM can be considered for the alternative model of software development to improve defects in non-functional software. To support the need in real time, it can be used to predict the run-time defect or as a tool to expand software quality development.

REFERENCES

[1]. Q.Song, M.Shepperd, M.Cartwright, and C.Mair, "**Software Defect Association Mining and Defect Correction Effort Prediction**", *IEEE Transactions on software engineering*, Vol. 32, no. 2, February 2016.

[2]. S.Tantithamthavorn, S. McIntosh, A. Hassan, K. Matsumoto "**An Empirical Comparison of Model Validation Techniques for Defect Prediction Models**", *IEEE Transactions on Software Engineering*, Volume: 43, Issue: 1, 2017. https://doi.org/10.1109/TSE.2016.2584050

[3]. X.-Yuan Jing, Fei Wu, X. Dong, Baowen Xu "**An Improved SDA Based Defect Prediction Framework for Both Within-Project and Cross-Project Class-Imbalance Problems**", *IEEE Transactions on Software Engineering*, Volume: 43, Issue: 4, 2017. https://doi.org/10.1109/TSE.2016.2597849

[4]. R.Jadhav, S.Joshi, U.Thorat, A.Joshi, "**A survey on Software Defect Prediction in Cross Project**" In: *Proc of 13th INDIACom-2019,IEEE Conf. 6th International Conf. on "Computing for Substanable Global Development"*.

[5]. S.Lessmann, B.Baesens, C.Mues, and S. Pietsch. "**Benchmarking classification models for software defect prediction: A proposed framework and novel findings**". Software Engineering, IEEE Transactions, pp. 485-496, 2008. https://doi.org/10.1109/TSE.2008.35

[6]. Z.Abraham, P.Tan, "**A Semi-supervised Framework for Simultaneous Classification and Regression of Zero-Inflated Time Series Data with Application to Precipitation Prediction**", *In: Proc of IEEE International Conf. on Data Mining Workshops*, pp. 644 - 649, 2009. https://doi.org/10.1109/ICDMW.2009.80

[7]. G.K.Rajbahadur, S.Wang, Y.Kamei, A.E. Hassan, "**The Impact of Using Regression Models to Build Defect Classifiers**", In: *Proc IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pp. 135 - 145, 2017. https://doi.org/10.1109/MSR.2017.4

[8]. M.Dhiauddin, M.Suffian, S.Ibrahim, "**A Prediction Model for System Testing Defects using Regression Analysis**", *International Journal of Soft Computing And Software Engineering (JSCSE)*, pp. 2251-7545, Vol.2(7), 2012. https://doi.org/10.7321/jscse.v2.n7.6

[9]. S. Bibi, G. Tsoumakas, I. Stamelos, I. Vlahvas, "**Software Defect Prediction Using Regression via Classification**", In: *Proc of IEEE International Conf. on Computer Systems and Applications*, pp. 330 - 336, 2006. https://doi.org/10.1109/AICCSA.2006.205110

[10]. Software Defect Dataset, PROMISE Repository for NASA Projects, http://promise.site.uottawa.ca/SERepository .

[11]. E.A.Felix, S.P.Lee, "**Integrated Approach to Software Defect Prediction**", *IEEE Access*, Vol. 5, pp. 21524 - 21547, 2017. https://doi.org/10.1109/ACCESS.2017.2759180

[12]. F.Rahman, D.Posnett, A.Hindle, E.Barr, P.Devanbu, "**BugCache for inspections: hit or miss?**", In: *Proc. for 19th ACM SIGSOFT symposium of software engineering*, pp. 322-331, 2011. https://doi.org/10.1145/2025113.2025157

[13]. X.Yu, J.Liu, Z.Yang, X.Jia, Q.Ling, S.Ye "**Learning from Imbalanced Data for Predicting the Number of Software Defects**", *IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 78 - 89, 2017. https://doi.org/10.1109/ISSRE.2017.18

[14]. P.Ambardekar, A.Jamthe, M.Chincholkar "**Predicting defect resolution time using cosine similarity**", In: *Proc International Conf. on Data and Software Engineering (ICoDSE)*, pp. 1 - 6, 2017. https://doi.org/10.1109/ICODSE.2017.8285884

[15]. O.Okutan, T. Yildiz, "**Software defect prediction using Bayesian networks**", *Empirical Software Engineering*, pp. 1-28, 2012. https://doi.org/10.1007/s10664-012-9218-8

[16]. H. Can, X. Jianchun, Z. R. L. Juelong, Y. Quiliang and X. Liqiang, "**A new model for software defect prediction using Particle Swarm Optimization and support vector machine**", *In: Proc. Of 25th Chinese Control and Decision Conference (CCDC)*, pp. 4106 - 4110, 2013. https://doi.org/10.1109/CCDC.2013.6561670

[17]. F.Zhang, Q.Zheng, Y.Zou, A.E. Hassan "**Cross-Project Defect Prediction Using a Connectivity-Based Unsupervised Classifier",** In: *Proc of IEEE/ACM 38th International Conf. on Software Engineering (ICSE)*, pp. 309 - 320, 2016 https://doi.org/10.1145/2884781.2884839

[18]. Z. Yan, X. Chen, P. Guo, "**Software Defect Prediction Using Fuzzy Support Vector**

Regression", *Springer Advances in Neural Networks*, Vol. 6064, pp 17-24, 2010.
https://doi.org/10.1007/978-3-642-13318-3_3

[19]. J.C.Riquelme, R.Ruiz, D.Rodriguez, J.S.A. Ruiz, "**Finding Defective Software Modules by Means of Data Mining Techniques**", *IEEE Latin America Transactions*, Vol. 7(3), pp. 377 - 382, 2009.
https://doi.org/10.1109/TLA.2009.5336637

[20]. J. Wang, B. Shen, and Y. Chen, "**Compressed C4.5 Models for Software Defect Prediction**", In: *Proc of IEEE 12th International Conf. on Quality Software (QSIC)*, August, pp. 13-16, 2012.
https://doi.org/10.1109/QSIC.2012.19

[21]. R. Chillarege, I.S. Bhandari, J. Chaar, M.J. Halliday, D.S. Moebus, B.K. Ray, and M.Y. Wong, "**Orthogonal Defect Classification-A Concept for In-Process Measurements**," *IEEE Trans. Software Eng*., vol. 18, no. 11, pp. 943-956, Nov. 1992.
https://doi.org/10.1109/32.177364

[22]. Jon.T "**A Comparison of IBM's Orthogonal Defect Classification to Hewlett Packard's Defect Origins, Types, and Modes**", pg 13-16, *Hewlett Packard company Metrics*, 1999.

[23]. R.G.Dromey, "**Software Control Quality - Prevention Verses Cure?**" Vol. 11 (3), pp. 197 - 212, 2003.
https://doi.org/10.1023/A:1025162610079

[24]. Y.Liu, T.M.Khoshgoftaar, N.Seliya "**Evolutionary Optimization of Software Quality Modelling with Multiple Repositories**", *IEEE Transactions on Software Engineering* Vol. 36(6), pp. 852 - 864, 2010.
https://doi.org/10.1109/TSE.2010.51

[25]. A. Koru and H. Liu, "**Building effective defect-prediction models in practice**", *IEEE Software*, pp. 23-29, 2005.
https://doi.org/10.1109/MS.2005.149

[26]. B.Turhan, AT.Msrl and A.Bener. "**Empirical evaluation of the effects of mixed project data on learning defect predictors**". *Inform. Software Tech.* 2013; 55, 1101-18.
https://doi.org/10.1016/j.infsof.2012.10.003

[27]. F.Rahman, S.Khatri, ET.Barr, and P.Devanbu, "**Comparing static bug finders and statistical prediction**", In: *Proc. of the 36th ACM International Conf. on Software Engineering*, pp. 424-34. 2014.
https://doi.org/10.1145/2568225.2568269

[28]. M. H. Halstead, "Elements of Software Science", *Elsevier*, *New York*, 1977.

[29]. H.Zhang, X.Zhang, and M.Gu. "**Predicting defective software components from code complexity measures**", *In IEEE 13th Pacific Rim International Symposium on Dependable Computing*, pp. 93-96, 2007.
https://doi.org/10.1109/PRDC.2007.28

[30]. N. E. Fenton and S. L. Pfleeger, "**Software metrics: a rigorous and practical approach**", PWS Publishing Co., (1998).