Mehran University Research Journal of Engineering and Technology
Vol.40, No. 3, 617- 629, July 2021
p-ISSN: 0254-7821, e-ISSN: 2413-7219
DOI: https://doi.org/10.22581/muet1982.2103.15

A Regression Analysis Based Model for Defect Learning and Prediction in Software Development

Mashooque Ahmed Memon[1], Mujeeb-ur-Rehman Maree Baloch[2], Muniba Memon[3], Syed Hyder Abbas Musavi[4]

RECEIVED ON 19.03.2019, ACCEPTED ON 07.10.2019

ABSTRACT

The development of software undergoes multiple regression phases to deliver quality software. Therefore, to minimize the development effort, time and cost it is very important to understand the probable defects associated with the designed modules. It is possible that occurrence of a range of defects may impact the designed modules which need to be predicted in advance to have a close inter-association with the depended modules. Most of the existing defect prediction classifier mechanisms are derived from the past project data learning, but it is not sufficient for new project defect predicting as the new design may have a different kind of parameters and constraints. This paper recommends Regression Analysis (RA) based defect learning and prediction Defect Prediction (RA-DP) mechanism to support the defective or non-defective prediction for quality software development. The RA-DP approach provides two methods to perform this prediction analysis. It initially presents an association learning through RA to construct the regression rules from the learned knowledge required for the defect prediction. The constructed regression rules are used for defect prediction and analysis. To measure the performance of the RA-DP a regression experimental evaluation is performed over the defect-prone PROMISE dataset from NASA project. The outcome of the results is analyzed through measuring the prediction Accuracy, Sensitivity and Specificity to demonstrate the improvisation and effectiveness of the proposal in comparison to a few existing classifiers.

Keywords: Software Defect Prediction, Association Learning, Regression Analysis

1. INTRODUCTION

Software is a complicated object composed of several modules with different degrees of defect incidence. Software defects represent errors or defects in software artifacts or software processes and are primarily focused on predicting defects that affect the project or product performance. Software failure prediction helps detect, track, and resolve software anomalies, particularly critical security systems that may affect the user's safekeeping and lifespan.

Defect Detection (DD) techniques work out hand in hand to limit deficit attempts [5] and enhance the probability of meeting the investigation team with its defined investigation targets and goals. The occurrence of defect avoidance schemes does not merely imitate a high stage of investigation-discipline development, but it as well characterizes the significant cost-effective overhead related to the on the

[1] Department of Computer Science, Benazir Bhutto Shaheed University, Layari, Karachi, Sindh, Pakistan. Email: pashamorai786@gmail.com
[2] Institute of Mathematics and Computer Science, University of Sindh, Jamshoro, Sindh, Pakistan. Email: mujeeb@usindh.edu.pk
[3] Department of Information Technology, Quaid-e-Awam University of Engineering, Science and Technology, Nawabshah, Sindh, Pakistan. Email: muniba@quest.edu.pk
[4] Department of Electronic Engineering, Mehran University of Engineering and Technology, Shaheed Zulfiqar Ali Bhutto Campus, Khairpur Mir's, Sindh, Pakistan. Email: drhyderabbas@muetkhp.edu.pk (Corresponding Author)

whole testing effort. There are numerous mechanisms, tools, techniques, and practices for defect prevention, but the entire seems to be inadequate. Much work remains to be done to avoid defects in the techniques to agree, the tools to use, and the guidelines to be drafted for the prevention.

Classifications and predictions [6-7] able to be utilized to extract models that describe critical defect data classes or to predict possible defect tendencies. The classification predicts definite or distinct labels and labels without order, while the predictive model predicts functions of continuous value. This analysis helps us to better understand software defect data. In case of defects are detected in the "development lifecycle" or "requirements specifications" is able to prevent defects from being migrated from design and code development. Defect prevention is extremely important for improving the eminence of the institutions [8].

The foremost goal of high-quality costs is not to reduce costs, however, to invest costs in the right investments. Instead, it has to be considered to save time, costs and resources. It provides a lot of reprocessing, which is essential while the defects occur in the concluding stage or after the delivery. Defects have to be implemented at each phase of the software lifecycle to obstruct defects as quickly as possible, take corrective action to eliminate it and prevent it from being repeated. Even existing metrics and defect prediction models may not guarantee good overall predictive performance.

Software Defect Prediction (DP) framework [3] refers to a system that specifies that the specific software modules are defective or not. In general, this model is trained to utilize software metrics and defect data, collected by beforehand developed software releases or alike projects. DP model [9] are functional to program modules with unidentified defective data. The characteristics or features of the set of prediction data for software defects influence the efficiency and effectiveness of the DP model.

The existing DP methods [4] are not always able to fix all defects from accessing applications under testing since the application is extremely difficult and it is impractical to grab the entire defects. DP reduces software costs and improves customer satisfaction [1]. However, there is no software DP technology that able to solve all defect problems. Therefore, with efficiently and appropriately predicting the occurrence of software flaws, software project supervisors able to utilize in the improved manner of the costs and time to gain enhanced quality assurance [2-3]. Most of the experiments associated with DP are executed in the Machine Learning (ML) tool, such as "WEKA" [10]. In ML approaches the construction of a predictable model and classified software modules related to defect, one of the significant characteristics of the software, and also analyses the defects. Different data mining approaches such as, DT (Decision Tree), Bayesian Belief Network (BBN), Artificial Neural Network (ANN), Support Vector Machine (SVM) and Clustering are various techniques which are usually utilized to predict defects in software [11-12]. But these approaches and prediction mechanism are mostly based on past project defects learned, which is insufficient to handle the new project development conditions and designs. So, its required a regressive analysis of the defect in a continuous manner to learn from the past and current defect to suggest an effective DP mechanism to cater the today's agile software development needs.

This paper presents a regression-based defect-association prediction (RA-DP) mechanism for proficient DP in software applications. The main principle of RA-DP is to build an efficient rule model for precisely classifying software defects utilizing "NASA" repository datasets [20] for experimental evaluation. As each institution attempt to maintain their data private, it cannot publish data sets that able to utilize in testing. The most frequently existing datasets are "MDP" and "PROMISE" repositories presented by NASA.

The RA-DP implements an RA method for fault-related learning of data sets to generate defect knowledge to be used to develop RA-based fault prediction classifiers to improve DP strategies. It will be supporting the classification of a defective or non-defective software project in an efficient manner for the deployment. The RA-DP mechanism extends the methodology of multiple regression-based classifications to construct an effective regression

analyzer that accurately classifies software defects. This method improves software DP, allowing software testers to utilize additional time for testing defective modules. The subsequent paper is organized as the 2nd Section discuss the background study, the 3rd Section discuss the recommended based defect prediction, the 4th Section discussed the experiment assessment implementing the datasets and finally in the 5th section it summarizes the conclusion of the paper.

2. BACKGROUND STUDY

Suggestions for preventing software defects are usually based on tools, technologies, methods, and standards [16, 17]. In the field of software engineering, it is one of the active research focus [18]. Since DP models include a group of "defect-prone software artifacts" [18], "quality assurance teams" which resourcefully allocated to the inadequate resources to analysis and examine software products [20, 21].

Several DP studies [23-25] have been carried out and all are based on ML approach or a statistical approach. ML algorithms are employed in software failure prediction models for classification and regression [26]. Several learning methods have recently used ML techniques to improve the predictability of defects. Pre-processing techniques are also important in software DP. In order to progress the performance of the ML technique before constructing the DP model, the following techniques such as "feature selection" and "data normalization and noise reduction" able to applied [27, 28]. A quite number of "feature selection techniques" are utilized to mine essential functions for the DP model. However, several studies [29] have shown that predictive performance able to improved through 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 eminence by utilizing software metrics to predict defect counts in software modules [7, 13, 14]. It can assist developers in distributing adequate 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".

RA [13, 14] is a statistical technique commonly used for studying the linear relationship between variables. This analysis technique is useful when defects are distributed over a wide range of classes. Multiple RA is an extension of linear RA that includes one or more predictor variables [15]. It allocates a related variable to be modeled as a linear function of, say, n predictor variables or features, "A_1, A_2, . . . ,A_n", describing a defect, D, that is, "D= (x_1, x_2, . . . , x_n)". The training data set, T, includes 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" with related to the defect class labels, C_i.

2.2 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 assist to assess

the data cautiously prior to the CC.

2.3 Multiple Regression Analysis:

The regressive analysis is widely utilized in techniques that are constructive for estimating "multiple self-sufficient variables" [15]. Therefore, it is especially valuable for 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. Therefore, many regression models describe how one response variable is ′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:

$$\mathcal{P} = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \cdots + B_v X_v$$

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 - is 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, ′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 it has a defect factor variable, which is denoted as X_1 and the predicted outcome will be presented as ′P. Utilizing a simple linear regression estimation it able to predict as to the relented defect variable as, $'P = \alpha + \beta_1 X_1$, 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, it denotes 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 $'P = \alpha + \beta_1 X_1 + \ldots + \beta_n X_n$. 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, it utilizes 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.

ML RA is widely utilized in various predictions of techniques [26]. 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 modeling leads to associations.

2.4 Software Defect Prediction Techniques

Defects analysis in early stages [30-31] reduce time, cost and resources. The understanding of the injecting methods and practices 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 investigation is able to acquire two appearances, specifically in "logical analysis" and "statistical analysis". Logical Analysis is an exhaustive investigation 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 an efficient and cost-effective quality [4, 8] early detection of defects identification technique.

Several demands are clearly understood through prototype, which helps to overcome defects. Testing is one of the most successful methods. These defects [31], 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 production is the most efficient and cost-effective method for creating software. The most DP 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 dynamic or "semi-supervised learning" have been utilized to improve the DP models [9, 23]. Besides ML models few non-statistical models are also proposed such as "BugCache" [19].

Liu *et al.* [8] discuss the difficulties of modeling of software quality has been studied, which uses the metric database history from a single software project. Classification modeling 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 illustrates that the recommended approach is additional efficient and precise for utilizing multiple data sets.

Lessmann *et al.* [6] 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. The predictable accuracy utilizing the metrics depended classification is generally constructive. The outcomes also signify 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.

Riquelme *et al.* [27] 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.

Turhan *et al.* [32] utilized to improve the prediction of the cross-company's defect utilizing the Nearest Neighbor's (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 it able to create a model of forecasting, the cases of selected sources that includes related data to the goal, the model can be improved predict the target case than the model prepared from all sources. The "NN filter" selects 10 sources as near neighbors for each target-occurrence. Utilizing the NN filter to estimate the performance of the cross-section defect. The complexity of software designing and development need an efficient plan for DP. Therefore, to better plan it will be a maintenance strategy, it is important to predict which software modules are defective before it deploys 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 satisfaction [33].

Software modules are characterized into two groups such as, "defective" or "non-defective", which mostly are predicted through exploiting a "binary classification model". It acquires the improvement of these two classes prediction for implications on how to categorize and estimate the datasets in the next section.

3. PROPOSED REGRESSION ANALYSIS BASED DEFECT PREDICTION

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 improves efficiency by inheriting methodology multiple RA to improve results and reduce the number and accuracy of rules. In the learning method, it will utilize a fault predictor to make use of the "static code properties" described by "McCabe" and "Halstead" [34]. These are "module-based metrics" and it is the least unit of functionality in a complete system.

The proposed RA-DP mechanism reveals rules from past software defects based on learned multiple RA associative defects to generate relevant and irrelevant rules for defect prediction in software development. The proposed RA-DP system architecture is illustrated in Fig. 1. The designed system architecture provides two main modules of learning methods through RA and fault prediction utilizing RA-rules. The functionality of the system works on two types of datasets and the fault constraints and attributes. The "Training Data" consists of classified fault data which is used to learn the knowledge for defect prediction utilized the defined fault constraints and attributes. The "Test Data" is being used the evaluation of the prediction mechanism utilizing the generated RA-Rules for defect prediction. The "Defect Prediction Analysis" block provides the measures to analyze the accuracy of the prediction in comparison to other classifiers. In the following sections it discusses these two learning and fault prediction mechanisms.

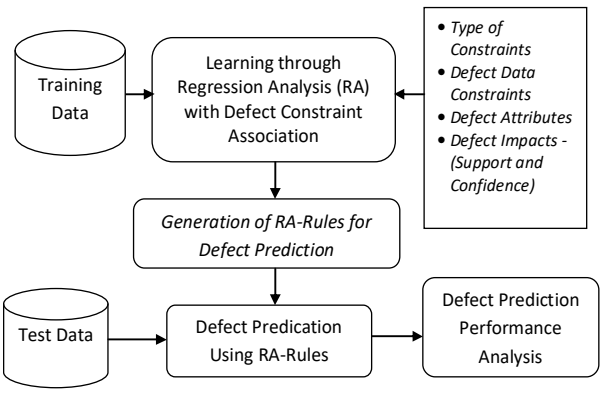

Fig. 1: System Architecture

3.1 Learning through Regression Analysis

Association rules mining and classification are often utilized in fault prediction to analyze 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 this framework, it utilizes the ratio partitioning, which is utilized to estimate the development of every predictive model. Specifically, every dataset which is primarily separated into two elements, the predictor is learned from the 60% instance, and the remaining 40% are tested.

Fig. 2 illustrates the data flow mechanism for learning through RA. It takes 60% of training data as input and reads the attributes of the defects in association with defect types. The obtained attributes of a fault are processed to learn the close association through the RA based on the defect constraints and attributes. The outcome of these leaning generates different prediction rules to predict a fault in a data script.

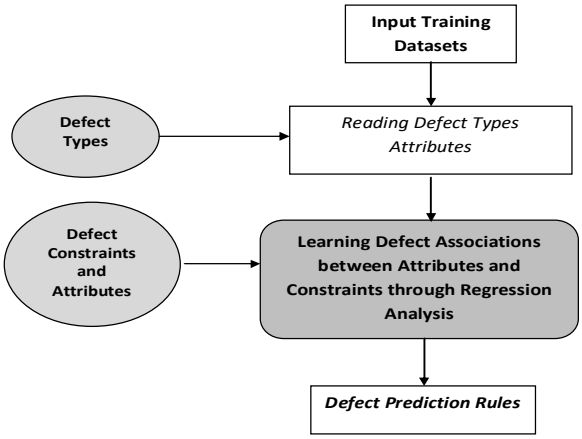

Fig.2: Data Flow Diagram for the learning process utilizing RA

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 by the regression process is presented in Algorithm-1.

Algorithm-1: Regression-Based Learning Process

Method: Generate_DP_Rules (Attrs, Training_set): DP_Rule_Set
var Rule: DP_Rule
Rules: DP_Rule_Set
Begin
forDefect_Classset_of_defect_class_values do
while "t :t.Training_set" and "t.class = Defect_Class" do
Rule.Class := Defect_Class;

```
                    Rule.RPattern := best_RPattern
(Defect_Class, Attrs, Training_set, Constraints);
                    remove from Training_set instances of
Defect_Class;
                    Rules := Rules. Rule;
end while

                return instances of Defect_Class to Training_set;
done
return Rules
```

End Generate_DP_Rules

DP_Rule_Set is a set of DP_Rules.
DP_Rule is a structure with two components:
Pattern: DP_ Regression Patterns.
Defect_Class: Class value predicted for an instance that matches Regression Patterns.
DP_ Regression Patterns is a combination of DP_terms.
DP_termare the form attribute–value.

The regression-based learning process has two nested iterations. The outer iteration chooses the class-value and the inner iteration generates the rule until the class is applied. The function "best_RPattern" returns a grouping of terms covering merely 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.

The subsequent terms that are supplementary to a rule are terms for choosing the most "positive instance" and the "least negative instance". This is specified by the relation in the form of "z/s", where s - "is the various instances selected by the term", and z - "is the number for which they are positive". The condition is supplement until the rule choices only positive instances. Ensures a group of rules containing all instances covered by one or more rules and it is consistent for every instance belongs to single fault class.

3.2 Regression Rules-Based Defect Prediction

The system depends on the understanding of the large constitution of concepts and rules are now consistently utilized in numerous purposes. Acquisition of knowledge of these systems when modern environments take place is an ongoing requirement that the interaction between the rules increases the complexity of the system.
The regression rule mechanism creates a bidirectional dependency between rules, in order that rule establishment is inspected only in the context of another rule establishment. If the premise of the parent rule is "true" for a particular individual, then the conclusion regarding the quality is presented if there is no relented. On the other hand, if it is "true", the rule and its relented will be examined and the unique conclusion will merely be asserted if the premises of the establishment are true to the authorized entity. On the contrary, if the premise of the parent rule is false for a particular individual, this not only cannot claim conclusions but if it has a 'false-false' dependency, it and its relented will also be tested.

This regression rule forms a binary DT dissimilar from the standard DT 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 inside node. This contrasts with the standard DT where all decisions are made at the root node. Nevertheless, the utility of the standard DT remains that simply one decision node is lively for everyone condition. Maintenance is simple because it needs 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 specifically recursively called on the outstanding dataset 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 DP_ Regression Patterns.
- Third, iteratively, each possible attribute value of the DP_term permutation is tested with a likely regression pattern and selects the finest according to the relevant DP_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 DP_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.

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

An illustration of RA-DP tree which described the recursive mechanism to reach the defect class is shown below:

IF (DEP_ON_CHILD = "1") THEN
Child-1:
IF ($cond_1$ AND $cond_2$ AND ... AND $cond_N$) THEN
DEFECTIVENESS="0" or "1" ;
Child-2:
IF ($cond_1$ AND $cond_2$ AND ... AND $cond_N$) THEN
DEFECTIVENESS="0" or "1" ;
ELSE
IF ($cond_1$ AND $cond_2$ AND ... AND $cond_N$) THEN
DEFECTIVENESS="0" or "1" ;

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 DP 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 ruleset specifies the evaluation of the test predicate. With the intent of statistical regression, the emergence of "S" and "Q" is not important. It has to consider "S" as an identifier to opt for "e" in the various partitions of "E" that require "Q (e)", measure the assortment of rules by indiscriminate identification, and the inquiries to locate "what is the prospect of the random classification of the identical degree of simplification would accomplish the similar or better precision" is been illustrated in Fig.3.

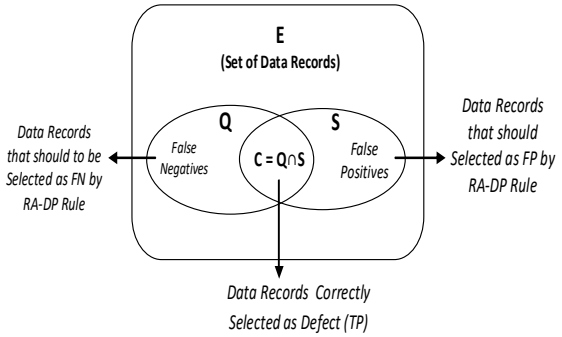

Fig.3: DP Confusion Matrix illustration

The illustration of DP confusion matrix in Fig. 3, describes the possible predictive instance as *False Negative (FN)* as "*Q*", or *False Positive (FP)* as "*S*" from a complete collection of records instances as "*E*". The fraction of relevant instances that have predicted as *True Positive (TP)* is represented as "*C*" is an intersection of instance between the classified "*Q*" and "*S*".

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 DP confusion matrix, a probability of the defect class will be predicted using RA rules.

4. EXPERIMENT EVALUATION

To evaluate the proposed prediction mechanism it implements the method in a WEKA Tool environment [21]. It utilizes the data repository of "NASA -Metric

Data Program" to perform the comparative analysis for the enhancement. The description of the datasets and its measures are discussed below.

4.1 Datasets

The data set was taken from the PROMISE repository for a NASA project [22] consisting of 12 data sets. The data store shows software metrics, which are attributes of the dataset, and whether an exact dataset is "Defective" or "non-defective". Every dataset consists of various software modules (conditions), each one includes an equivalent count of defects and diverse software attributes code statistic. In subsequent to pre-processing, one or more faulty modules are labeled as faulty. Additional clarification of the belongings of the code or the origin of the data set able to found in [35].The collection of the datasets consists of 5 project data scripts of the "CM1, JM1, KC1, KC2, and PC1". It encloses the measurement of the static codes such as, "Halstead, McCabe, and LOC", which defines the fault of defected codes. The description of these datasets is presented in Table1.

Table 1: Datasets 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	A real-time predictive ground system
PC1	C	Flight software for earth-orbiting satellite.

Every dataset has 21 software product parameters, based on the various measurements such as, "size", "complexity", "vocabulary", etc., of the product. The class attribute of each data set is considered as "TRUE", if the module has various defects, and "FALSE" if it's non-defective.

4.2 Performance Measures

Performance is measured according to the confusion matrix given in Table2, which is utilized by many researchers as in [35-36]. It shows confusing matrices for two-class problems with positive and negative class values.

The software DP performance of the proposed RA-DP along with existing classifiers are measured for the "Accuracy", "Sensitivity", and "Specificity" for the evaluation.

Table 2: Confusion Matrix

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

- Accuracy determines the percentage of DP that are "correctly classified".

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

- Sensitivity determines the percentage for the "positive classified" instances that predicted as "positive".

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP+FN}}$$

- Specificity determines the percentage for the "positive classified" instances that predicted as "negative".

$$\text{Specificity} = \frac{\text{TN}}{\text{FP+TN}}$$

4.3 Result Analysis

To compare the results with the best classifier for the prediction, such as "Naive Bayes", "OnerR", "J48", and "RIDOR" To analyze 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 RA-DP is shown in Tables 3-5 and Figs. 4-6, respectively.

4.4 Accuracy Analysis

The proposed RA-DP shows an improvement in the Accuracy value in comparison to existing classifier expect with the CM1 datasets. An average of 10%

enhancement in the accuracy being achieved as shown in Fig.5.

Table 3: Accuracy Results					
Methods	Prop. RA-DP	OnerR	RIDOR	Naive Bayes	J48
CM1	94.17	91.56	91.36	84.93	91.56
KC1	80.65	79.77	80.65	80.42	80.65
KC2	85.49	84.4	84.54	84.63	84.54
JM1	83.33	79.5	79.5	83.52	79.5
PC1	94.86	90.89	93.41	89.54	93.41

Table 4: Sensitivity Results					
Methods	Prop. RA-DP	OnerR	RIDOR	Naive Bayes	J48
CM1	0.922	0.92	0.91	0.85	0.92
KC1	0.807	0.8	0.81	0.80	0.81
KC2	0.850	0.84	0.85	0.85	0.85
JM1	0.814	0.8	0.8	0.84	0.8
PC1	0.940	0.91	0.93	0.90	0.93

Table 5: Specificity Results					
Methods	Prop. RA-DP	OnerR	RIDOR	Naive Bayes	J48
CM1	0.851	0.831	0.827	0.70	0.831
KC1	0.652	0.596	0.613	0.61	0.613
KC2	0.723	0.688	0.691	0.69	0.691
JM1	0.666	0.59	0.59	0.67	0.59
PC1	0.883	0.818	0.868	0.79	0.868

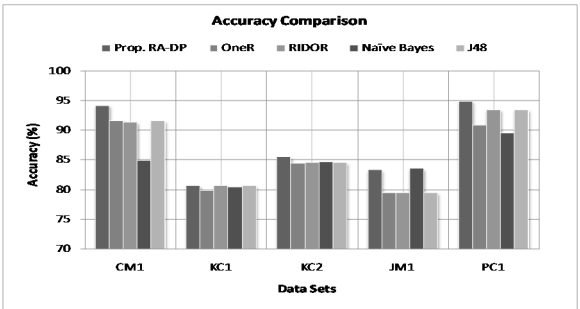

Fig.4: Accuracy Comparison

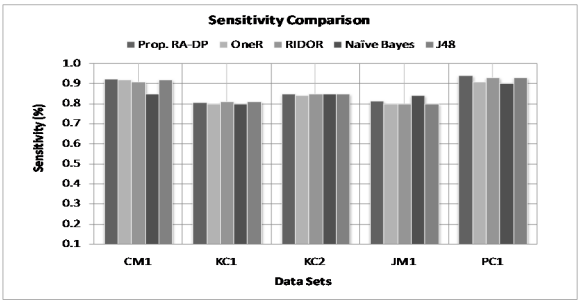

Fig. 5: Sensitivity Comparison

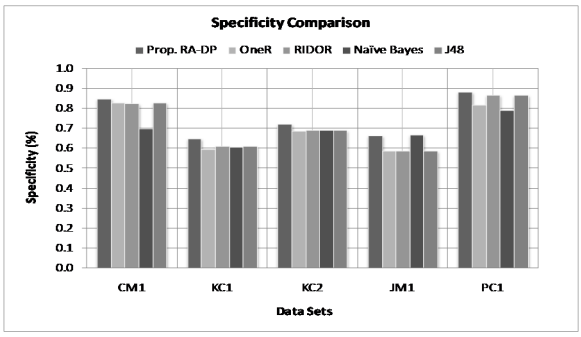

Fig. 6: Specificity Comparison

In the case of both sensitivity and specificity measure also it shows an improvement. The measure of sensitivity and specificity show the efficiency of the probability of detection of classifiers. The detection of the RA-DP utilizing the regression rules makes to predict defect accurately and enhance the sensitivity and specificity of the proposal.

As it was known that most software development needs the best quality and reliability development. But, in case of real-time scenario, a defect-prone module highly affects the cost, time and resources. So, an early prediction of defect quite useful in reducing the wastage of the resources. The practices in the effective defect prediction classifier are successfully applied. The proposed RA-DP clearly demonstrate the usage of a classifier in defect prediction through RA. The obtained results in terms of accuracy, sensitivity, and specificity show the impact of the proposal. The statistical analysis among the classifiers justifies the significance of metrics evaluation in the software defect prediction to improvise the quality design and coding development.

The variety of learning algorithm has a different impact on prediction since defect prediction is very important in finding a noble balance in designing. So, the RA-DP provides an RA based learning with a close correlation analysis among the attributes to provide a better impact on the defect prediction in future software development. As in Table3 shows, RA-DP has better accuracy improvisation in comparison among the existing classifiers. The role of classifiers in defect prediction is being learned in the past. Since every classifier has a different approach for learning and prediction makes the variation in result.

4.5 Sensitivity and Specificity Analysis:

The better accuracy with an average sensitivity comparison as in Table-4 and better Specificity as in Table 5 make it more beneficial in current software defect prediction. The sensitivity results in comparison to the different prediction classifiers as shown in Fig. 5 suggest an improvisation of ~ 2% in the positive data prediction. It shows an average 1% better in comparison to CM1 and PC1, and almost similar results in comparison to KC1, KC2, and JM1. Whereas, Fig. 6 shows the specificity comparison results. It shows an improvisation of ~3% in terms specificity measures. In shows an average of 2% better specificity with CM1, 3% with KC1, KC2, 5% with JM1 and PC1.

5. CONCLUSIONS

Finding and fixing defects makes it simpler for developers to comprehend the program. In order to get better, the competence and excellence of software development are able to obtain the benefit of data mining techniques for analysis and predict a big numeral of defect data assemble in software development. This paper presents a regression-based defect prediction (RA-DP). The designed system implements defect prediction utilizing a learning method and RA-rule by RA. The learning method creates rules with two exception types that are easy to understand and automatically find search rules, so the designer does not have to do that physically. A rule is characterized as an arrangement of attributes that enhanced to fits in known cases of design flaws. The experimental analysis illustrates improved performance in defect prediction evaluated to the existing classification methods.

6. FUTURE WORK

In future improvement, the RA-DP can be considered for the variant model in the software development for improvising non-functional software defects. In support of the real-time need, it can be used for runtime defect prediction or as a tool to extend the software quality development process.

ACKNOWLEDGMENT

Authors acknowledge to the Department of Computing, Faculty of Engineering, Science & Technology, Indus University, Karachi Pakistan, for continuous support to carry out this research.

REFERENCES

1. Song Q., Shepperd M., Cartwright M., Mair C., "Software Defect Association Mining and Defect Correction Effort Prediction", *IEEE Transactions on Software Engineering*, Vol. 32, No. 2, pp. 69 - 82, February 2006.
2. Tantithamthavorn C., McIntosh S., Hassan, A. E., Matsumoto, K., "An Empirical Comparison of Model Validation Techniques for Defect Prediction Models", *IEEE Transactions on Software Engineering*, Vol. 43, No. 1, pp. 1 - 18, 2017.
3. Jing X.-Y., Wu F., Dong X., Xu B., "An Improved SDA Based Defect Prediction Framework for Both Within-Project and Cross-Project Class-Imbalance Problems", *IEEE Transactions on Software Engineering*, Vol. 43, No. 4, pp. 321 - 339, 2017.
4. Dromey R. G., "Software Control Quality - Prevention Versus Cure?", Vol. 11, No. 3, pp. 197 - 21, 2003.
5. Ge J., Liu J., Liu W., "Comparative Study on Defect Prediction Algorithms of Supervised Learning Software Based on Imbalanced Classification Data Sets", *Proceedings of the 19th IEEE/ACIS International Conference on Software Engineering*, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 399 - 406, Busan, South Korea, 27 – 29 June 2018.
6. Lessmann S., Baesens B., Mues C., Pietsch S., "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.
7. Abraham Z., Tan P.-N., "A Semi-supervised Framework for Simultaneous Classification and Regression of Zero-Inflated Time Series Data with Application to Precipitation Prediction", *Proceedings of the IEEE International Conference on Data Mining Workshops*, pp. 644 - 649, Miami, F.L., U.S.A., December 2009.

8. Liu, Y., Khoshgoftaar, T. M., Seliya, N., "Evolutionary Optimization of Software Quality Modeling with Multiple Repositories", *IEEE Transaction on Software Engineering*, Vol. 36, No. 6, pp. 852 - 864, 2010.
9. Koru A., Liu H., "Building effective defect-prediction models in practice", *IEEE Software*, Vol. 22, No. 6 pp. 23 - 29, 2005.
10. Witten I., Frank E., "*Data Mining: Practical Machine Learning Tools and Techniques*", Morgan Kaufmann, San Francisco, 2 editions, pp 1-33, 2005.
11. Marian Z., Mircea I.-G., Czibula I.-G., Czibula G., "A Novel Approach for Software Defect Prediction Using Fuzzy Decision Trees", *Proceedings of the 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Comp. (SYNASC)*, pp. 240 - 247, Timisoara, Romania, 24-27 September 2016.
12. Shan C., Zhu H., Hu C., Cui J., Xue J., "Software defect prediction model based on improved LLE-SVM", *Proceedings of the 4th International Conference on Computer Science and Network Technology (ICCSNT)*, Vol.1, pp. 530 - 535, Harbin, China, 2015.
13. Rajbahadur G. K., Wang S., Kamei Y., Hassan, A. E., "The Impact of Using Regression Models to Build Defect Classifiers", *Proceedings of the 14th IEEE/ACM International Conference on Mining Software Repositories (MSR)*, pp. 135 - 145, Buenos Aires, Argentina, 20-21 May 2017.
14. Suffian M. D. M., Ibrahim S., "A Prediction Model for System Testing Defects using Regression Analysis", *International Journal of Soft Computing and Software Engineering,* Vol. 2, No. 7, pp. 2251-7545, 2012.
15. Bibi S., Tsoumakas G., Stamelos I., Vlahvas, I., "Software Defect Prediction Using Regression via Classification", *IEEE International Conference on Computer Systems and Applications*, pp. 330 - 336, Dubai, U.A.E., March 2006.
16. Ali S., Iqbal N., Hafeez, Y., "Towards Requirement Change Management for Global Software Development using Case Base Reasoning", *Mehran University Research Journal of Engineering and Technology*, Vol. 37, No 3, pp. 639-652, 2018.
17. Hani S. U., Alam A. T., Shaikh A. B., "Tuning COCOMO-II for software process improvement: A tool based approach", *Mehran University Research Journal of Engineering and Technology*, Vol. 35, No 4, pp 505-522, 2016.
18. Felix E. A., Lee S. P., "Integrated Approach to Software Defect Prediction", *IEEE Access*, Vol 5, pp. 21524 - 21547, 2017.
19. Rahman F., Posnett D., Hindle A., Barr E., Devanbu P., "BugCache for inspections: hit or miss?", *Proceedings for 19th ACM SIGSOFT Symposium of Software Engineering*, pp. 322-331, Szeged, Hungary, September 2011.
20. Yu X., Liu J., Yang Z., Jia X., Ling Q., Ye S., "Learning from Imbalanced Data for Predicting the Number of Software Defects", *Proceedings of the 28th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pp. 78 - 89, Toulouse, France, 23-26 October, 2017.
21. Ambardekar P., Jamthe A., Chincholkar M., "Predicting defect resolution time using cosine similarity", *Proceedings of the International Conference on Data and Software Engineering (ICoDSE)*, pp. 1 - 6, Palembang, Indonesia, 1-2 November 2017.
22. Software Defect Dataset, PROMISE Repository for NASA Projects, available at: http://promise.site.uottawa.ca/SERepository
23. Yildiz O. O. T., "Software defect prediction using Bayesian networks", *Empirical Software Engineering*, Vol. 19, No. 1, pp 154–181, 2014.
24. Can H., Jianchun X., Juelong Z. R. L., Quiliang Y., Liqiang X., "A new model for software defect prediction using Particle Swarm Optimization and support vector machine", *Proceedings of the 25th Chinese Control and Decision Conference (CCDC)*, pp. 4106 - 4110, Guiyang, China, 25- 27 May 2013.
25. Zhang F., Zheng Q., Zou Y., Hassan A. E., "Cross-Project Defect Prediction Using a Connectivity-Based Unsupervised Classifier", *Proceedings of the 38th IEEE/ACM International Conference on Software Engineering (ICSE)*, pp. 309 - 320, Austin, T.X., U.S.A., 14-22 May 2016
26. Yan Z., Chen X., Guo P., "Software Defect Prediction Using Fuzzy Support Vector

Regression", *Advances in Neural Networks*, Vol. 60, No 64, pp 17-24, 2010.

27. Riquelme J. C., Ruiz, Rodriguez, R., D., Aguilar Ruiz, J. J. S., "Finding Defective Software Modules by Means of Data Mining Techniques", *IEEE Latin America Transactions*, Vol. 7, No. 3, pp. 377 - 382, 2009.
28. Hafeez A., Musavi S. H. A., "Ontology-based verification of UML class/OCL model", *Mehran University Research Journal of Engineering and Technology*, Vol. 37, No 4, pp 521-534, 2018.
29. Wang J., Shen B., Chen Y., "Compressed C4.5 Models for Software Defect Prediction", *Proceedings of the 12th IEEE International Conference on Quality Software (QSIC)*, pp. 13-16, Xi'an, China, 27-29 August 2012.
30. Chillarege R., Bhandari I. S., Chaar J., Halliday M. J., Moebus D. S., Ray B. K., Wong M. Y., "Orthogonal Defect Classification-A Concept for In-Process Measurements", *IEEE Transaction on Software Engineering*, Vol. 18, No. 11, pp. 943-956, 1992.
31. Huber J.T., "*A Comparison of IBM's Orthogonal Defect Classification to Hewlett Packard's Defect Origins, Types, and Modes*", Hewlett Packard Company Metrics, pp. 13-16, 1999.
32. Turhan B., Msrl A. T., Bener A., "Empirical evaluation of the effects of mixed project data on learning defect predictors", *Information and Software Technology*, Vol. 55, No. 6, pp. 1101-1118, 2013.
33. Rahman F., Khatri S., Barr E. T., Devanbu P., "Comparing static bug finders and statistical prediction", *Proceedings of the 36th ACM International Conference on Software Engineering*, pp. 424-34, Hyderabad, India, May 2014.
34. Halstead M. H., "*Elements of Software Science*", Elsevier, New York, pp 124-127, 1977.
35. Zhang H., Zhang X., Gu M., "Predicting defective software components from code complexity measures", *Proceedings of the 13th IEEE Pacific Rim International Symposium on Dependable Computing*, pp. 93-96, Melbourne, V.I.C., Australia, 2007.
36. Fenton N. E., Pfleeger S. L., "*Software Metrics: a Rigorous and Practical Approach*", PWS Publishing Co, pp 1-77, 1998.