ISSN 2278-3091
Volume 8, No.3, May - June 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse78832019.pdf
https://doi.org/10.30534/ijatcse/2019/78832019

Hybrid Model for Cross Project Fault Prediction Using Random Forests and Multi-Objective Ant Lion Optimization

Yogomaya Mohapatra[1*], Dr. Mitrabinda Ray[2]
[1*]Assistant Professor, Computer Science & Engineering, Orissa Engineering College, Bhubaneswar, Odisha, India
[*] E-mail: yogomaya0201@gmail.com
[2] Associate Professor, Computer Science & Engineering, S'O'A University, Bhubaneswar, India

ABSTRACT

In software development, the software fault is dealing an important task. The faults presence is not reduce the software quality and increase the cost development. The software system has the prediction in the cross project for the software system, the large number of models is presented. The task of fault prediction is difficult because most of them provide the information as inadequate. The proposed method of this paper is hybrid model for predicting the cross project faults using the random forest (RF) technique and multi-objective Ant Lion optimization (MO-ALO) approach in the given software system. By using the eight software projects, the data in PROMISE data repository is used for the experimental results have been done. The performance evaluation of the method is evaluated with the existing techniques which are Support Vector Machine (SVM), RF and the K-nearest Neighbor (KNN). The results for the number of cross project faults prediction shown in the RF and MOALO based model.

Key words: Faults prediction, multi-objective ant lion optimization, prediction, PROMISE

1. INTRODUCTION

While programming the fault prediction is the most important and most effective in the cost. The framework is hard to build in fault free software [1-6]. Hence, most software system advancement affiliations are trying to acknowledge and proper the fault. In any case, to boost the strategy and venture execution before they unharnessed their software system product [7, 8]. Distinctive forms of fault prediction instruments (or models) are created to predict fault modules or elements in software system real frameworks. These procedures unremarkably use varied options, e.g., method metrics, past defect metrics, source text file metrics, etc, to characterize a class/document/module and use a characterization calculation to predict if a class/file/module is insufficient or not [7]. The varied need methodologies are: cross-version prediction, cross-validation prediction, cross-project fault prediction [8].

Cross-Project fault Prediction has beginning late complete up being necessary stream within the field of software package fault prediction. It had been overall viewed as a binary classification issue or a regression issue in most of the past examinations [9, 10]. Here, the prediction models are worked by exceptive knowledge from completely different ventures because the coaching data and a take a look at set got from the close enterprise because the objective venture data [11, 12]. To wear down this, instances of supply knowledge like target knowledge are picked to manufacture classifiers. Programming datasets have category awkwardness issue that means the extent of disadvantage category to shortcoming class is much lower. It typically cuts down the execution of classifiers [13, 14]. Cross-project fault prediction is grounded on participating (i) it licenses anticipating deserts in ventures that the provision of knowledge is confined, and (ii) it grants creating generalizable want models [13, 15-17]. Regardless, existing strategies prescribes that cross-project prediction is very making an attempt and prediction precision is not unremarkably extraordinary thanks to non-uniformity of activities [18, 19]. Completely different techniques to predict cross project fault are multi-target cross-project fault prediction, Multi-Objective (MO) Learning techniques, ROCPDP (Ranking oriented CPDP) technique, HISNN (Hybrid Instance selection using Nearest-Neighbor), HYDRA (Hybrid Model Reconstruction Approach) and etc.

Although, the cross-project fault prediction is completely addicted to multi-objective logistic regression. Rather than equipping the product engineer with one predictive model, the multi-target approach licenses software engineers to choose predictors achieving a exchange between varied possible defect-prone artifacts (adequacy) and LOC to be analyzed/attempted (which may be thought of as a delegate of the expense of code examination) [1, 19]. A selective learning dependent on the nearby information of CP information is performed by the HISNN method. K-closest neighbor is received for predictor defects on test information when the nearby learning is solid. Something else, Naive Bayes using worldwide learning is embraced. Precedents having strong close-by data are recognized by methods for closest neighbors

with a similar class mark. In any case it's low Pd (probability of detection) or high PF (probability of false alarm) that is unreasonable to use. [4]. Powerful Multi-Objective Improved Teaching–Learning based optimization technique (MO-ITLBO) calculation uses a grid based methodology with a particular true objective to keep not too bad assortment in the external record. This count is beneficial and has centered execution over the MO issues. Regardless, these figuring have needed in improving a bit of the multi-target issues [1, 18].Genetic rule (GA) [20] section and Ensemble Learning (EL) phase are the two periods of Hybrid model reconstruction approach (HYDRA) for cross-project fault prediction. These 2 phases create a massive composition of classifiers [19].

This paper introduces a novel hybrid approach for predicting faults in cross project to reduce the above mentioned drawbacks. This hybrid approach is based on Random Forest technique and Multi-Objective Ant Lion Optimization approach. At first the issue prediction issue is formalized as a multi target streamlining issue. We detail our multi-target deficiency prediction issues with the accompanying differentiating destinations (1) Maximize the likelihood of recognition (2) Minimize the likelihood of false caution (3) Minimize misclassification cost. To locate the best number of trees and leaves per tree in the woodland, the ALO calculation is utilized for upgrade the RF procedure [21]. From the Promise vault the aftereffects of an exact assessment on ten datasets, the prevalence and the convenience of the multi-target approach as for single-target indicators are demonstrated by ten datasets. Our results demonstrate that RF and MOALO based fault prediction models are huge to anticipate the deficiencies in the product framework. The evaluation measures of coming about insights affirmed the prescient precision and consistency of the build prediction models. Rest of the manuscript is organized as follows: Section 2 presents a detail discussion on related works on previous methods. Section 3 presents the hybrid model for predicting cross project faults using RF technique and MO-ALO approach. The results and discussion are illustrated in section 4. Section 5 finalizes the paper with conclusions.

2. RELATED WORKS

The fundamental software engineering activity is software testing for confirmation of quality that is customarily very expensive. To tackle this problem improvement and research in mining repositories can be utilized. For fault prediction this information can be modelled and applied to future activities. An extensive variety of statistical models have been created based on this concept and for fault prediction software is applied. Some of the related research findings are presented in this section.

Jayanthi and Florence [22] introduce a combined approach for software system fault prediction and prediction of software system bugs was exhibited. Their approach conveys a feature thought decrease and AI wherever feature diminishment was applied by well-known principle element analysis (PCA) theme that was to boot increased by incorporating estimation of maximum-likelihood for error reduction in PCA knowledge reconstruction. At last, neural network based mostly classification technique was applied that demonstrates the results of prediction.

A Feature Dependent Naive Bayes (FDNB) identification technique was introduced by Arar and Ayan [23]. Features are incorporated for estimation as pairs to form reliance between each other. This methodology was connected to the software fault prediction issue and examinations were finished exploitation normally perceived NASA PROMISE information sets. The obtained results show this new methodology was additional fruitful than the quality Naive Bayes approach which it's a competitive performance with alternative part weight systems. An extra aim of this examination was to exhibit that to be dependable; a learning model should be worked by utilizing and just preparing information, as for the most part deceptive results rise up out of the use of the entire datasets.

Chen et al. [24] have present a multi-target optimization based administered procedure MULTI to create JIT-SDP models. Specifically, they need formalized JIT-SDP as a multi-target optimization issue. One goal was planned to spice up the number of perceived carriage changes and another article was planned to constrain the undertakings in computer code quality assertion exercises. There exists associate degree positive conflict between these 2 objectives. MULTI uses strategic relapse to manufacture the models and uses NSGA-II to form a great deal of non-commanded arrangements, wherever each arrangement indicates the constant vector for the calculated relapse.

Hosseini et al. [25] have used the Nearest Neighbor (NN)-Filter, implanted in genetic algorithm to convey approval sets for creating developing preparing datasets to handle CPDP while speaking to for potential clamor in shortcoming marks. They in like manner have investigated the impact of using diverse capabilities. They have expanded their methodology, Genetic Instance Selection (GIS), by consolidating feature choice in its setting. They have used 41 arrivals of 11 multi-variant project to assess the exhibition GIS in correlation with benchmark CPDP (NN-filter and Naive-CPDP) and within project (Cross Validation (CV) and Previous Releases (PR)). To review the impact of capabilities, they have used two arrangements of highlights, SCM+OO+LOC (all) and CK+LOC just as iterative data addition sub setting (IG) for feature selection.

For cross-project fault prediction, Canfora et al. [26] have used a multi-objective provision regression model factory-made utilizing a genetic calculation. As opposition furnishing the software engineer with one predictive model, the multi-target approach was allowed programming architects to choose indicators accomplishing a trade-off between varied possible prediction inclined antiquities (effectiveness) and LOC to be tested (which are often thought-about as an proxy of the code- code analysis).

A Hybrid Model Reconstruction Approach (HYDRA) for cross-project fault prediction was presented by Xia et al. [27], genetic algorithm (GA) phase and ensemble learning (EL) phase are the two phases they are incorporated from the HYDRA. The classifiers are composite as two phases. The advantage of HYDRA was analyzed, 29 datasets taken from the PROMISE repository that contains a total of 11,196 instances (i.e., Java classes) named as fault or clean these experiments were performed by the authors. The results of HYDRA have a normal FI-measure as 0.544. Overall sage of 29 datasets the results are compared with the FI-score of 26.22%, 34.99%, 47.83%, 28.61%, and 30.14% with the methods of TCA+, GP, MO, CODEP and Peters Filter respectively.

Choi et al. [28] mentioned concerning the package fault prediction was a standout amongst the foremost essential tasks for package quality modification. The benefits of CPDP square measure inspecting the educational in imbalance. In their approach, the uneven misclassification price and therefore the similarity weights got from spatial arrangement qualities square measure nearly associated with management the correct re-sampling system. A-statistics take a look at is to access the modification for the impact for enjoying the estimate. Wilcoxon rank-sum take a look at square measure used for the take a look at in applied mathematics important. The explorative results exhibited their approach may offer higher prediction execution than each the present CPDP procedure and therefore the current category imbalance methodology.

2.1 Background of Research Work

From the literature, it can be observed that majority of the studies have utilized industrial datasets for cross project fault prediction. The ten open source datasets are used in the proposed method. The main drawback of the prior studies can be condensed as: difficult to comprehend the delivered models, most strategies can't manage with the unbalanced data, some of them are not require the preprocessing step, fault dataset have common characteristics are focused on the restrict of aspects in evaluation and the fault prediction problem is single in objective when they address. To tolerate the disadvantages we presented in this paper the RF and MOALO model, portrayed and assessed in the accompanying sections.

3. RF AND MOALO BASED FAULT PREDICTION MODEL

The proposed approach constructs RF-MOALO based fault prediction model. In every segment the predictor set is first registered in the system software. To decrease the impact of data heterogeneity, by preprocessing the computed data. When performing the cross project fault prediction the preprocessing step is especially useful, as data from various projects and in the same project have different properties at some case [29]. For the most part the prediction model does not explicitly consider the nearby distinction between different software projects; its exhibitions can be unsteady when it endeavors to predict fault across projects. After preprocessing, a machine learning approach (RF) is utilized to build a predictive model. The novel and ensemble machine learning procedure is RF. However, when contrasted the RF shows a great deal of focal points and that of other modeling approach inside the classification. The RF can deal with both discrete and continuous variables which is the fundamental favorable circumstances. The tree leaves in number (at each node the number of splits in the subset) and the trees in number in the forest are the two hyper-parameters of RF. Optimal number of leaves per tree and number of trees are selected for guarantee precise cross project fault prediction. The trees and leaves in number of forest are find for optimize the RF, the ALO algorithm can be used. Thus, the optimization is used to enhance the RF execution that implies less error rate for fault prediction.

3.1 Data Preprocessing

In this paper we perform a standardization data to diminish the heterogeneity impact in between of various software objects. That is a z distribution is converted from metrics. The value of metric is given to compute M_i on software component (C_j) of project C is characterized as $M_z(i, j, C)$ and changed it into:

$$M_z(i, j, C) = \frac{M(i, j, C) - \mu(i, C)}{\sigma(i, C)} \quad (1)$$

Also, the value of metric (M_i) is subtracted, all components of the system S the mean value $\mu(i, C)$ is acquired, and the standard deviation $\sigma(i, C)$ is used to separate it. Gyimothy et al. [30] has applied the comparable approach; however such preprocessing is utilized to decrease metrics to same interim for inside the prediction of project before joining them. In our research, the preprocessing technique is utilized to lessen the project heterogeneity impact in cross project prediction.

3.2 Multi-Objective Predictive Modeling

The multivariate logistic regression is the broadly utilized machine learning procedure. For fault prediction such a technique is especially appropriate, where the two conceivable results are find the presence of fault prone or not in the software component. In this manuscript we formulate the fault prediction model definition in problem of multi-objective. With the following three objectives: the probability of detection is maximized ($PD_{\max}$), the probability of false alarm is ($PF_{\min}$) minimized, minimize misclassification cost ($Cost_{\min}$) we formulate our multi-objective fault prediction problems.

The class unbalance context is considered by the multi-objective optimization technique. In the specific circumstance, high accuracy is the basic level of training dataset for the student needs, in the discussion of the non-fault class. Thus the performance of probability of detection and false alarm probability are taken into consideration. $PD = T_p/(T_p + F_n)$ is formulated from the probability of detection means the ratio of relevant instances among all the retrieved instances. The false alarm probability is defined as the portion of non-fault class instances predicted which is formulated as $PF = F_p/(F_p + T_n)$. Where, for true class, the T_p (true positive) is denoted as the number of fault class instances predicted as defective, the number of non-fault class instances predicted as non defective is T_n (true negative), and F_p (false positive) is denoted as the number of fault class is non- defective, F_n (false negative) is denoted as the number of non fault class is in defective.

$$PD_{\max}(X) \geq PD_{\max}(Y) \text{ or } PD_{\max}(X) > PD_{\max}(Y) \quad (2)$$

The condition is the probability of false alarm is minimized

$$PF_{\min}(X) < PF_{\min}(Y) \quad \text{or} \quad PF_{\min}(X) \leq PF_{\min}(Y) \quad (3)$$

The objective function to minimize misclassification of cost ($Cost_{\min}$) is given by

$$Cost_{\min} = \sum_{i=1}^{M} S(Y_i).L_c(Y_i) \quad (4)$$

Here, the Boolean (0 or 1) evaluated fault-proneness of Y_i is represented as $S(Y_i)$, i.e., the Boolean value indicate the actual fault proneness of Y_i and the number of code lines is measured by $L_c(Y_i)$.

To acquire the high overall performance as conceivable i.e., obtain high PD, low PF and misclassification cost. However, there has balance between three objectives. While acquiring the acceptable performance the highest PD can be the solution. The software quality prioritizes this case than the effort of testing. On the other hand, the other solution have the performance is high when the acceptable performance of PD is acquired. This case organizes the capabilities are adjusted between the testing effort and software quality. Then software system is predicted faults in the random forests technique based on ALO are utilized, discussed in the accompanying segment.

3.3 Hybrid Random Forests Technique Based on Ant Lion Optimizer

The Hybrid RF technique is based on ALO is described in this section. Breiman in 2001 [31] proposed an ensemble learning method of trees in regression used for RFs are a bagging based strategy. The combination of the decision tree is RF and it used to create the bootstrapping technique. CART is respect to (Regression and Classification Trees) model the principle of RFs is done [32]. The two contrasts can be noted by meanwhile. At every node in split in RFs initially, randomly select the learning dataset and only within this step the best split is calculated. In the forest all the maximal trees are supposed tree there is no clipping step is achieved. By a variable significance measure the RFs rank the input variable, Based on the prediction accuracy output, the input variables impact is reflected. The flowchart of fault prediction model is appeared in figure 1. The following stages depict the fault prediction model.

Stage 1: Random Forests Technique: The RFs algorithm evaluates the variables significance by contrasting prediction fault with data term OOB (Out-Of-Bag). It gets an unbiased estimate running of fault prediction as RFs is constructed for the training phase also used to determine the importance of variable. The combination of training stage and testing stages are the RFs algorithm. The structure of RFs algorithm is appeared in Fig. 2.

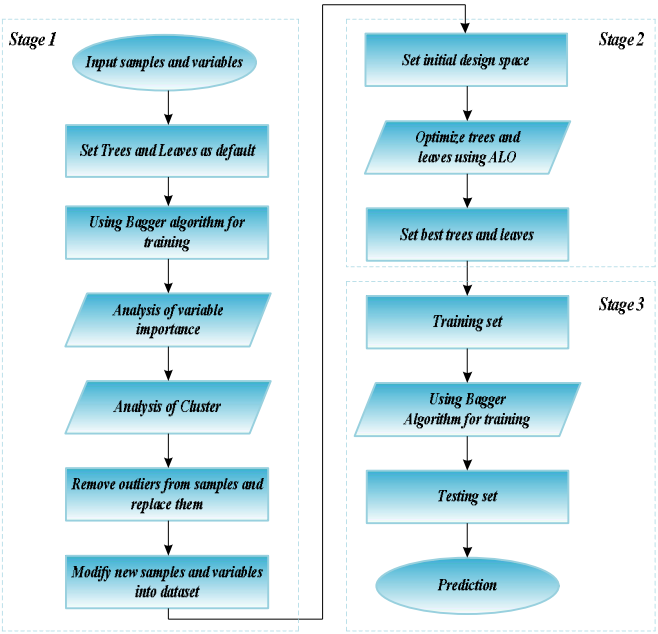

Figure 1: Flowchart of RF-ALO prediction model

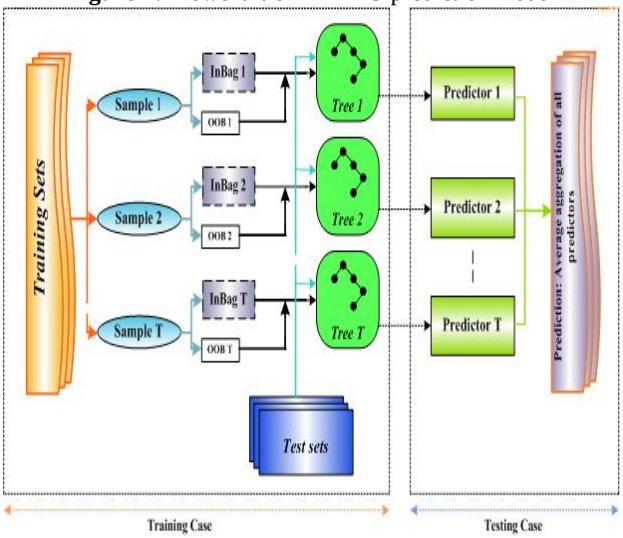

Figure 2: Structure of the Random Forests algorithm

The RFs method can be depicted as takes after:

- Initially from the learning datasets with replacement is randomly form the bootstrap samples. The number of samples equal to trees. By taking the best one, the creator [33] recommends attempting the fault (500) trees and twice the fault of the trees in number and to set the optimal tress in number of the forest. Diverse patterns of dataset are utilized to create in each sample the algorithm model which is developed.
- In a regression tree at every hub various split is chosen randomly to make binary rule. Be that as it may, the MSE (mean squared error) is assessed in each split and the trees in number for the best tree is selected by contrasted with the OOB acquired data. Amid forest growing the leaves in number (5 leaves) of tree is held consistent.
- During the training stage, based on OOB data the algorithm give the huge measure to important variable and on the measure of modification significance. Ascertain the measure of variable importance as the average in sum of distinction between the accuracy of prediction after and before changing the variable every one of the predictors.

$$V_i^{tn}(Z) = \frac{\sum_T \left(\frac{\sum_{X_{a\in\alpha^{C(tn)}}} I(N_b = C_a^{tn})}{\left|\alpha^{C(tn)}\right|} - \frac{\sum_{X_{a\in\alpha^{C(tn)}}} I(N_b = C_{a,3.14(z)}^{tn})}{\left|\alpha^{C(tn)}\right|} \right)}{T} \quad (5)$$

where the particular tree OOB samples relates to $\left|\alpha^{C(tn)}\right|$, the tree number is represents as tn $(1, 2, \ldots T)$ the total number of trees is signified as T , for each sample the predicted classes are C_a^{tn} and $C_{a,3.14(z)}^{tn}$ in the before and after adjusting the variable separately. In the training stage, X_a is represent as the sample value, N_b represents the genuine label, I is the importance function that got based on the value N_b, the tree in number for leaves is sample and is denoted as a and the tree in number samples in the forest are denoted as b.

- After that the clustering analysis is used to recognize the outliers in the training dataset. Here, the thickness model cluster analysis type is utilized. This model can without quite a bit of a stretch defect points of cluster and noise that doesn't have a place with any of these clusters. In machine learning procedures the learning information is removed from the data collection will inside and out extends the results in the accuracy. Here the training dataset in the outliers are distinguished at that point supplanted and removed.
- At last to predict values the testing stage is finished. Then, to locate the last predicted values the predictors average of all regression trees are ascertained.

Stage 2: Ant Lion Optimizer: The proposed hybrid method utilized the ALO algorithm for the optimization procedure. ALO is a nature motivated algorithm that copies ant lions foraging behavior. The stochastic population based optimization algorithm is ALO which is considered as meta-heuristics by Mirjalili in 2015 [34]. The trees and leaves in number is best of tree in the forest are finding by using the ALO to optimize the RF. Here, with the optimal number the variables and samples are distinguished by taking at the trees

and leaves in number of the forest. The primary objective of the ALO is utilized for choosing best number of trees and the leaves in number of the forest. The best trees and leaves in number is chose based on the error. For the prediction function aims to modify the optimization techniques RFs model is developed for execution with the less error rate. For finding the fault prediction, the distance between the trees is required. The ALO structure is shown in figure 3.

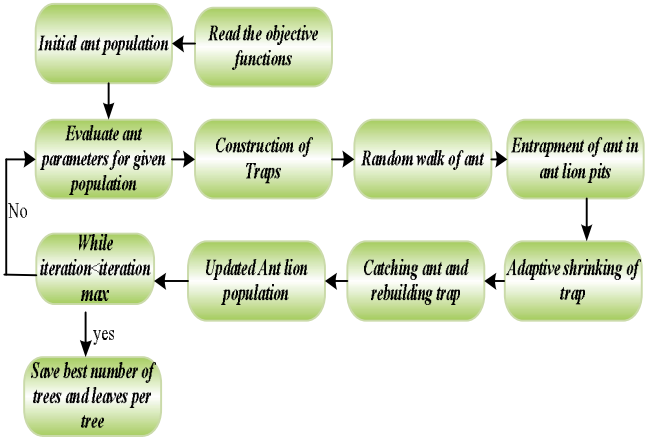

Figure 3: Structure of ALO algorithm

For optimization problems the ALO approximates the optimal solutions with utilizing a set of random solutions similarly to other algorithms. Based on the principles inspired this set is enhanced from the interaction between ant lions and ants. In the ALO there are two populations: ants and ant lions sets. The two sets are changed and estimating the optimization problem in the global optimum in the ALO general advances are in the part of the following:

- Around the hunt space ant move randomly, these move are influenced by ant lions traps
- Highest fitness ant lions build a larger pit.
- Ant lions are utilized for catching as ant, proportional to the ant lion fitness.
- At iteration, ant lions can get an ant.
- Sliding ants are recreated towards the ant lions; the random walk range is diminished adaptively.
- If ant lions becomes filter over ant, then ant lions is caught and pulled to the soil.
- To the latest caught prey the ant lions repositions itself and constructs the pit after each hunt to enhance its chances of catching.

Step 1: Random walks: Utilizing random walks around the hunt space (position updating) ant move at each iteration based on condition (6).

$$X(m) = [0, cumsum(2R_n(m_1) - 1), cumsum(2R_n(m_1) - 1), \ldots, cumsum(2R_n(m_t) - 1)] \quad (6)$$

where, the aggregate entirety is denoted as $cumsum$, the iteration is m, m_t is the most number of iteration and the stochastic function $(2R_n(m)$ takes value 1, if a arbitrary value is less (less than) and generally zero. To guarantee that inside the limits of hunt space the ants move, utilizing the eq. (7) the random walks are standardized.

$$X_j^m = \left[\frac{(X_j^m - a1_j).(d1_j - c1_j^m)}{d1_j^m - a1_j} \right] + c1_j \quad (7)$$

where, the maximum and minimum of random walk of jth variable are $a1_j$ and $b1_j$, $c1_j^m$ is the minimum and $d1_j^m$ is the maximum jth variable in jth iteration.

Step 2: Ant lions pits trapping: By the traps of ant lions the random walks of ants are influenced, which is joined by accompanying equations:

$$c1_j^m = AL_i^m + c1^m \quad (8)$$

$$d1_j^m = AL_i^m + d1^m \quad (9)$$

Here, the ant lions is denoted as AL at ith position, the vectors that represent minimum and maximum value at m^{th} iteration of all variables are $c1^m$ and $d1^m$.

Step 3: Trap building: To display the ant lions chasing capability the determination component ought to be utilized. With the high fitness the ant lion has a higher opportunity to get an ant. Here, RWS (Roulette Wheel Selection) is utilized for choosing the ant lions based on the fitness value applied.

Step 4: Sliding ants toward ant lion: The ant endeavors to escape when it slips into the pit. On the off chance that there is a prey in the pit the ant lion acknowledges and shoot the sand towards the pit focus. To display this behavior the range of random walk of ants is diminished which is scientifically expressed in underneath eq. (10) and (11).

$$c1^m = \frac{c1^m}{r} \quad (10)$$

$$d1^m = \frac{d1^m}{r} \quad (11)$$

where, the ration r is characterized as,

$$r = 10^{\kappa} \frac{m}{t_{\max}} \quad (12)$$

where the present iteration is m, $t_{\max}$ the maximum iteration, the constant (κ) can change the accuracy level of exploitation, the constant is characterized based on the present iteration.

Step 5: Prey catching and rebuilding the pit: In the chasing final stage, at the base of the pit the prey reaches and caught the jaw of ant lions. After that inside the sand the ant pulls by the ant lions and the body is expends. Here, the ant lion updates its position to the chased ant position to raise its chasing ability of new ant by eq. (13).

$$AL_i^m = \begin{cases} ant_j^m & ; \ if \ \ f(ant_j^m) < f(AL_i^m) \\ AL_i^m & ; \ otherwise \end{cases} \tag{13}$$

Step 6: Elitism: In each iteration, the ant lion with the higher fitness is considered as elite. The selected and elite ant lion utilized the selection mechanism to direct the ant random walk and thus the given ant repositioning follows the following eq. (14).

$$ant_j^m = \frac{r_A^m + r_E^m}{2} \tag{14}$$

Where r_A^m is represents the random walk around the selected ant lion for utilizing the selection mechanism and r_E^m represents the elite ant lion random walk. Finally the ALO algorithm delivers best tree and leaves of forest in tree.

Stage 3: Fault prediction: Then the produced best tree and leaves per tree in the forest is given to the Bagger algorithm to train the RF samples and variables. At that point the trained samples in forest are given to the classifier for fault prediction.

4. EXPERIMENTAL METHODOLOGY

4.1 Dataset Description and Performance Measurements

The experimental study is done in this section for the software defect prediction using the MOALO methods. The experiment is done by using the PC (personal computer) with memory of 2GB, 64-bit windows, 8 operating system and with the Intel ® core (TM) IN THE Java software tools. In this work, we have considered eight datasets from PROMISE repository which are named as CM1, KC1, KC2, KC3, MC2, PC1, PC3 and PC4. These datasets were developed in C/C++ language and it is a NASA MDP software projects form flight control of satellite, instrumentation spacecraft, storage management of ground data and scientific data processing.

Table1 gives the description of the dataset, where the dataset are in language, total number of available modules, defective modules, non-defective modules, imbalanced ratio and percentage defect are depicted. The ratio of imbalance is represented in number of ratio for the defective and non-defective software modules. The dataset of the software defect prediction is imbalance in the nature of the defective software modules where comparing with the non-defective modules.

Table 1: Description of Dataset

Dataset	Language	Modules	Defective	Non-defective	Imbalanced ratio	% defect
CM1	C	498	49	449	9.16	9.83
KC1	C++	2109	326	1783	5.46	15.45
KC2	C++	522	107	415	3.87	20.49
KC3	Java	458	43	415	9.65	9.38
MC2	C/C++	161	52	109	2.09	32.29
PC1	C	1109	77	1032	13.40	6.94
PC3	C	1563	160	1403	8.76	10.23
PC4	C	1458	178	1280	7.19	12.2

Table 2: Performance of predicted class and real class

Predicted Class	Real Class	
	Defective	Non- defective
Fault	TP	FP
Non- fault	FN	TN

Table 2 represents the performance of the predicted class and the real class. The software modules are predicted correctly in the defective and non-defective modules in the table is denoted as true prediction (TP or TN), and are predicted incorrectly that is denoted as false prediction (FP or FN). The proposed defect predictor model is evaluated by using the geometric mean (G-mean). The classifier performance is evaluated by data distribution scenario. The balanced performance is calculate by using the G-mean by considering the geometric mean is given as,

$$G - Mean = (sensitivity \times specificity)^{1/2} \tag{15}$$

Specificity and sensitivity are defined as

$$specificity = \frac{T_N}{T_N + F_N} \tag{16}$$

$$sensitivity = \frac{T_P}{T_P + F_P} \tag{17}$$

Using precision and *F*-measure we have also compared the proposed method performance which is defined as follows:

$$F-measure = \frac{2(precision \times sensitivity)}{precision + sensitivity} \quad (18)$$

$$precision = \frac{T_P}{T_P + F_N} \quad (19)$$

5. RESULT COMPARISON AND DISCUSSION

The results and discussion are discussed for the software defect predictor with the previous methods, including K-NN, SVM and random forest. All these approaches are implemented in Java. The experiment is perform using the ten-fold cross validation method and the dataset is divided randomly into a size of equal in ten subsets. For learning every time nine subsets using the training and for testing datasets remaining one will be used in the evaluation of the software defect prediction system. Every ten subsets are using the training and testing datasets while repeating this process to ten times. The performance of defect prediction system is estimate by averaging the ten-fold results. The comparison is for the precision, sensitivity, specificity, geometric mean and the F-measure metrics for the proposed and existing techniques with the 8 datasets of software defect prediction. Table 3 shows the comparison of sensitivity values on eight datasets. From the table 4, the comparative results obtained for the proposed method with the existing for the case of sensitivity values for CM1, MC2, KC1, KC2, KC3, PC1, PC3 and PC4 datasets. From the table 4, the proposed method obtains better specificity for CM1, KC1, MC2 and PC1 datasets when compared with SVM. Table 5 and 6 shows the comparison of precision and F-measure values. The precision value of proposed method is 0.99 for KC2 dataset, 0.83 for KC2 dataset, 0.928 for PC3 dataset and the remaining datasets have the precision value as 1. From the table 6, it is clear that the F-measure of the proposed method is higher for all datasets when compared with existing algorithm. The geometric mean value is better in our proposed method of defect predictor for the CM1, MC2, PC1, PC3, KC1, KC2, KC3, and PC4 datasets. From table 7, K-NN and SVM geometric gains the highest mean value for KC2 dataset.

Table 3: Comparative results for Sensitivity for eight datasets

Dataset	K-NN	SVM	Random Forest	Proposed
CM1	0.54166	0.46153	0.619047	0.65
KC1	0.72727	0.5833	0.6666	0.8
KC2	0.970588	0.94339	0.941747	0.961538
KC3	0.5	0.45454	0.6	0.625
MC2	0.90909	0.75	0.8333	0.9166
PC1	0.3333	0.35	0.378378	0.46875
PC3	0.38028	0.4	0.472727	0.52
PC4	0.48387	0.473118	0.56626	0.6486

Table 4: Comparative results for Specificity for eight datasets

Dataset	K-NN	SVM	Random Forest	Proposed
CM1	1	0.98214	1	1
KC1	1	0.96	1	1
KC2	0.931034	0.96	0.85714	0.96296
KC3	0.97435	0.9736	1	0.9756
MC2	0.95238	0.9	0.95	1
PC1	0.99259	0.9927	0.992857	1
PC3	0.99497	1	0.990697	0.9909
PC4	0.98897	0.985294	0.996454	1

Table 5: Comparative results for Precision for eight datasets

Dataset	K-NN	SVM	Random Forest	Proposed
CM1	1	0.9230769	1	1
KC1	1	0.875	1	1
KC2	0.980198	0.99009	0.96039	0.99009
KC3	0.833333	0.83333	1	0.8333
MC2	0.90909	0.818181	0.90909	1
PC1	0.93333	0.93333	0.93333	1
PC3	0.964285	1	0.92857	0.92857
PC4	0.9375	0.916666	0.979166	1

Table 6: Comparative results for F-measure for eight datasets

Dataset	K-NN	SVM	Random Forest	Proposed
CM1	0.7027	0.615385	0.7647	0.7878
KC1	0.8421	0.7	0.8	0.8889
KC2	0.975369	0.96618	0.95098	0.9756
KC3	0.625	0.588235	0.7499	0.714285
MC2	0.90909	0.7826	0.86956	0.95652
PC1	0.491228	0.50909	0.538461	0.638297
PC3	0.54545	0.57142	0.626506	0.666666
PC4	0.63829	0.624113	0.717557	0.786885

Table 7: Comparative results for geometric mean for eight datasets

Dataset	K-NN	SVM	Random Forest	Proposed
CM1	0.73598	0.6732	0.78679	0.80622
KC1	0.8528	0.74833	0.816496	0.8944
KC2	0.9506056	0.9516	0.898449	0.96225
KC3	0.697982	0.66526	0.77459	0.7808
MC2	0.93048	0.82158	0.88975	0.95742
PC1	0.5752	0.58944	0.61292	0.6846
PC3	0.6151	0.63245	0.68434	0.7178
PC4	0.73598	0.6732	0.78679	0.80622

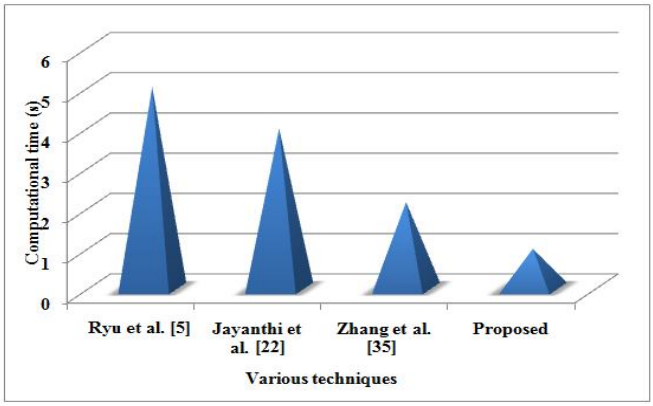

Figure 4: Comparison analysis of computational time with various research works

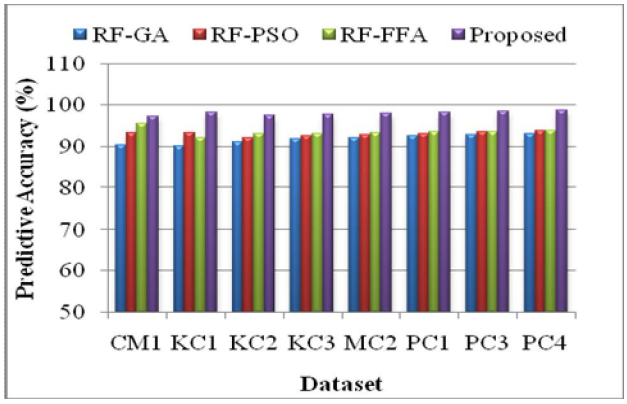

Figure 5: Performance comparison of predictive accuracy with various methodologies

The computational time analysis of various cross project fault prediction works is shown in figure 4. It is observed that from the figure, the computational time of the proposed method is vey less of 80.3%, 75.2% and 54.09% when compared with Ryu et al. [5], Jayanthi et al. [22] and Zhang et al. [35].

Figure 5 shows the performance comparison of predictive accuracy with various methodologies. It is clearly observed that, the accuracy of proposed method is better of 7.02%, 3.9% and 1.7% for CM1 dataset, 8.1%, 4.8% and 6.1% for KC1dataset, 6.6%, 5.5% and 4.7% for KC2 dataset, 5.9%, 5.2% and 4.6% for KC3 dataset, 6.05%, 5.2% and 4.82% for MC2 dataset, 5.79%, 5.33% and 4.89% for PC1 dataset, 5.63%, 5.02% and 4.81% for PC3 dataset, 5.68%, 4.78% and 4.73% for PC4 dataset when compared with RF-GA, RF-PSO, and RF-FFA. Therefore for the datasets on 6 to 8 in every case the performance is better in other three methods in an effective software prediction of our method

6. CONCLUSION

In this paper, a hybrid model using the Random Forest technique and multi-objective Ant Lion Optimization is proposed for predicting cross project faults in software system. Using the proposed model the multi objective problems are solved. The experimental study was applied in the PROMISE repository datasets for eight software fault prediction. The considered datasets are CM1, MC2, PC1, PC3, KC1, KC2, KC3, and PC4.The result have been evaluated in terms of precision, sensitivity, specificity, and geometric mean, F-measure. The proposed approach show higher geometric mean, F-measure, sensitivity, precision, and specificity while comparing with existing techniques like K-NN, SVM and RF. The software system of accuracy is predicting the number of faults by using the model of our proposed fault prediction.

REFERENCES

1. P. K. S. Kumar. **Defect Prediction Model for AOP-based Software Development using Hybrid Fuzzy C-Means with Genetic Algorithm and K-Nearest Neighbors Classifier**, *International Journal of Applied Information Systems*, Vol. 11, pp. 26-30, Jul. 2016. https://doi.org/10.5120/ijais2016451579
2. G. Canfora, A. Lucia, M. D. Penta R. Oliveto and A. Panichella. **Defect prediction as a multiobjective optimization problem. Software Testing**, *Verification and Reliability*, Vol. 25, pp. 426-459, Jun. 2015. https://doi.org/10.1002/stvr.1570
3. T. Khoshgoftaar and Y. Liu. **A Multi-Objective Software Quality Classification Model Using Genetic Programming**, *IEEE Transactions on Reliability*, Vol. 56, pp. 237-245, Jun. 2007. https://doi.org/10.1109/TR.2007.896763
4. D. Ryu, J. Jang and J. Baik. **A Hybrid Instance Selection Using Nearest-Neighbor for Cross-Project Defect Prediction**, *Journal of Computer Science and Technology*, Vol. 30, pp. 969-980, Sep. 2015. https://doi.org/10.1007/s11390-015-1575-5
5. D. Ryu and J. Baik. **Effective multi-objective naïve Bayes learning for cross-project defect prediction**, *Applied Soft Computing*, vol. 49, pp. 1062-1077, Dec. 2016. https://doi.org/10.1016/j.asoc.2016.04.009
6. G. You, F. Wang, and Y. Ma. **An Empirical Study of Ranking-Oriented Cross-Project Software Defect Prediction**, *International Journal of Software Engineering and Knowledge Engineering*, Vol. 26, pp. 1511-1538, Dec. 2016. https://doi.org/10.1142/S0218194016400155
7. X. Jing, F. Wu, X. Dong and B. Xu. **An Improved SDA Based Defect Prediction Framework for Both Within-Project and Cross-Project Class-Imbalance Problems**, *IEEE Transactions on Software Engineering*, Vol. 43, pp. 321-339, Aug. 2017. https://doi.org/10.1109/TSE.2016.2597849
8. Z. Zhu, J. Xiao, S. He, Z. Ji, and Y. Sun. **A multi-objective memetic algorithm based on locality-sensitive hashing for one-to-many-to-one**

dynamic pickup-and-delivery problem *Information Sciences* Vol. 329, pp. 73-89, Feb. 2016. https://doi.org/10.1016/j.ins.2015.09.006

9. C. Hu, X. Xue, L. Huang, H. Lyu, H. Wang, X. Li, H. Liu, M. Sun, and W. Sun. **Decision-Level Defect Prediction Based on Double Focuses**. *Chinese Journal of Electronics*, Vol. 26, pp. 256-62 Mar. 2017 https://doi.org/10.1049/cje.2017.01.005
10. Y. Shi, M. Li, S. Arndt and C. Smidts. **Metric-based software reliability prediction approach and its application**, *Empirical Software Engineering*, Vol. 22, pp. 1579-1633, Aug. 2016. https://doi.org/10.1007/s10664-016-9425-9
11. F. Porto and A. Simao, **Feature Subset Selection and Instance Filtering for Cross-project Defect Prediction - Classification and Ranking**, *CLEI electronic Journal*, Vol. 19, pp. 4:1-4:17, Mar. 2018.
12. X. Yang, D. Lo, X. Xia, and J. Sun, **TLEL: A two-layer ensemble learning approach for just-in-time defect prediction**, *Information and Software Technology*, vol. 87, pp. 206-220, Jul. 2017. https://doi.org/10.1016/j.infsof.2017.03.007
13. T. Lee, J. Nam, D. Han, S. Kim and H. Peter. **In, Developer Micro Interaction Metrics for Software Defect Prediction**, *IEEE Transactions on Software Engineering*, Vol. 42, no. 11, pp. 1015-1035, Nov. 2016. https://doi.org/10.1109/TSE.2016.2550458
14. M. Bisi and N. Goyal. **An ANN-PSO-based model to predict fault-prone modules in software**, *International Journal of Reliability and Safety*, Vol. 10, pp. 243, Mar. 2016. https://doi.org/10.1504/IJRS.2016.081611
15. W. Li, Z. Huang and Q. Li. **Three-way decisions based software defect prediction**, *Knowledge-Based Systems*, Vol. 91, pp. 263-274, Jan. 2016. https://doi.org/10.1016/j.knosys.2015.09.035
16. R. Rana, M. Staron, C. Berger et al. **Analyzing defect inflow distribution and applying Bayesian inference method for software defect prediction in large software projects**, *Journal of Systems and Software*, Vol. 117, pp. 229-244, Jul. 2016. https://doi.org/10.1016/j.jss.2016.02.015
17. F. Zhang, A. Mockus, I. Keivanloo and Y. Zou. **Towards building a universal defect prediction model with rank transformed predictors**, *Empirical Software Engineering*, Vol. 21, pp. 2107-2145, Oct. 2015. https://doi.org/10.1007/s10664-015-9396-2
18. D. Ryu, J. Jang and J. Baik. **A transfer cost-sensitive boosting approach for cross-project defect prediction**, *Software Quality Journal*, Vol. 25, pp. 235-272, Mar. 2015. https://doi.org/10.1007/s11219-015-9287-1
19. M. Cheng, G. Wu, H. Wan, G. You, M. Yuan, M. Jiang. **Exploiting correlation subspace to predict heterogeneous cross-project defects**. *International Journal of Software Engineering and Knowledge Engineering*, Vol. 26, pp. 1571-80, Dec. 2016 . https://doi.org/10.1142/S0218194016710017
20. P. Srikanth, M. SrijaM, S. Chakravarthy S, G.V.Rao, G.A. Raju**. Compare various Circuits Area Reduction using Genetic Algorithm and Hybrid Partitioning Algorithm**, *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 7, no.6, pp.111-114, Dec 2018. https://doi.org/10.30534/ijatcse/2018/08762018
21. D. Hema Latha , P. Premchand. **Estimating Software Reliability Using Ant Colony Optimization Technique with Salesman Problem for Software Process**, *International Journal of Advanced Trends in Computer Science and Engineering*, Vol.7, no. 2, pp. 20-29, March –April 2018. https://doi.org/10.30534/ijatcse/2018/04722018
22. R. Jayanthi and L. Florence**. Software defect prediction techniques using metrics based on neural network classifier**, *Cluster Computing*, 2018. https://doi.org/10.1007/s10586-018-1730-1
23. O. Arar and K. Ayan. **A feature dependent Naive Bayes approach and its application to the software defect prediction problem**, *Applied Soft Computing*, Vol. 59, pp. 197-209, Oct. 2017. https://doi.org/10.1016/j.asoc.2017.05.043
24. X. Chen, Y. Zhao, Q. Wang and Z. Yuan,. **MULTI: Multi-objective effort-aware just-in-time software defect prediction**, *Information and Software Technology*, Vol. 93, pp. 1-13, Jan. 2018. https://doi.org/10.1016/j.infsof.2017.08.004
25. S. Hosseini, B. Turhan and M. Mäntylä**. A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction**, *Information and Software Technology*, Vol. 95, pp. 296-312, Mar. 2018. https://doi.org/10.1016/j.infsof.2017.06.004
26. S. Shukla, T. Radhakrishnan, K. Muthukumaran, and L. Neti, **Multi-objective cross-version defect prediction**, *Soft Computing*, vol. 22, no. 6, pp. 1959-1980, Mar. 2018. https://doi.org/10.1007/s00500-016-2456-8
27. X. Xia, D. Lo, S. J. Pan , N. Nagappan, X. Wang. **HYDRA: Massively Compositional Model for Cross-Project Defect Prediction**, *IEEE Transactions on Software Engineering*, Vol. 42, pp. 977-998, Oct. 2016. https://doi.org/10.1109/TSE.2016.2543218
28. D. Ryu, O. Choi and J. Baik. **Value-cognitive boosting with a support vector machine for cross-project defect prediction**, *Empirical Software Engineering*, Vol. 21, pp. 43-71, Feb. 2014. https://doi.org/10.1007/s10664-014-9346-4
29. T. Menzies, A. Butcher, A. Marcus, T. Zimmermann and D. Cok, **Local vs. global models for effort estimation and defect prediction**", in *Proc. of the 26th IEEE/ACM International Conference on Automated Software Engineering*, New York, USA, pp. 343-351, Nov. 2011. https://doi.org/10.1109/ASE.2011.6100072

30. T. Gyimothy, R. Ferenc and I. Siket. **Empirical validation of object-oriented metrics on open source software for fault prediction**, *IEEE Transactions on Software Engineering*, Vol. 31, pp. 897-910, Oct. 2005. https://doi.org/10.1109/TSE.2005.112
31. L. Breiman. **Random forests**, *Machine learning*, vol. 45, pp. 5-32, Oct. 2001. https://doi.org/10.1023/A:1010933404324
32. G. Tang, A. Rabie, and U. Hägg. **Indian Hedgehog: A Mechanotransduction Mediator in Condylar Cartilage**, *Journal of Dental Research*, vol. 83, pp. 434-438, May 2004. https://doi.org/10.1177/154405910408300516
33. I. Ibrahim and T. Khatib. **A novel hybrid model for hourly global solar radiation prediction using random forests technique and firefly algorithm**, *Energy Conversion and Management*, Vol. 138, pp. 413-425, Apr. 2017. https://doi.org/10.1016/j.enconman.2017.02.006
34. S. Mirjalili, **The Ant Lion Optimizer**, *Advances in Engineering Software*, vol. 83, pp. 80-98, May 2015. https://doi.org/10.1016/j.advengsoft.2015.01.010
35. Y. Zhang, D. Lo, X. Xia and J. Sun. **An Empirical Study of Classifier Combination for Cross-Project Defect Prediction**, in *Proc. of the 39th Annual Computer Software and Applications Conference*, Taichung, Taiwan, vol. 2, pp. 264-269, Jul. 2015. https://doi.org/10.1109/COMPSAC.2015.58