Received: August 25, 2017


International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

# Software Fault Prediction and Classification using Cost based Random Forest in Spiral Life Cycle Model

**Hosahalli Mahalingappa Premalatha[1]*** **Chimanahalli Venkateshavittalachar Srikrishna[2]**

[1]*People's Education Society University, India*
[2]*People's Education Society Institute of Technology, India*
* Corresponding author's Email: premalathaphd2017@gmail.com

**Abstract:** In the domain of software engineering many new techniques are deployed for identifying the fault in software modules. This part of software design plays a fundamental role cause of its assurance towards higher reliability and stability. Many existing techniques like Bayesian approach have been employed to minimize the software faults but they can't able to predict efficiently within limited resources. In this paper, a new classification and prediction methodology is put forth to progress the accuracy of defect forecast based on Cost Random Forest algorithm (CRF) which reduces the effects of faults in irrelevant software modules. The proposed algorithm predicts the quantity of faults present in the modules of software in less time and classify based on measures of similarity obtained from Robust Similarity clustering technique. The overall results inferred from this methodology proven that this CRF can be capable to rank the module's faults in order to enhance the software development quality.

**Keywords:** Random forest, Robust similarity clustering, Spiral life cycle, Software module, Software defect forecast.

## 1. Introduction

Software had become a crucial part of day to day activities and is very significant in technological in addition to economic development. A Software Life Cycle (SLC) consist of several major steps namely Planning and requirement analysis, defining, Designing, Developing product, Testing, and Maintenance. Among all that, testing stage is prioritized to be an important task. The software industry grows the size of software to be larger and fault prediction has become a significant task due to its necessity for lots of human effort as well as time in development of software [1]. A Software testing process is an important and indispensable stage to build high quality software and ensure software reliability in Software Development Process (SDP) [2]. For reduction of cost and the effort of testing process, methods for fault prediction are employed to evaluate the faulty software modules with the help of software testing metrics [3]. Software Defect Forecast (SDF) is said to be a prediction and classification process which recognizes a software module has defect or not by monitoring its characteristics. The major aim of SFP is to identify the fault prone software modules by using some underlying software metrics before the beginning of testing phase [4]. Since, humans are dependent on software in day to day life a fault in a software module may cause severe effects in human lives [5]. In order to avoid the software damages a reliable software fault monitoring technique should be adopted. Nowadays a lot of software modules are sold to the clients with unsolicited defects so, different software metrics are used to find the software defects (SD).

Several methods are used for finding the software faults such as machine learning, statistical method and so on [6]. Several consistent approaches are solicited by the software developers to diminish the error in classification during the progress of forecast models to categorize the faulty modules, Neural network (NN) based techniques, which functions with better arithmetic approaches are introduced. Probabilistic NN is highly robust in nature which is reducing the misclassification rate in prediction

system [7]. To improve the software flaw prediction Artificial Neural Network is adopted, that helps to reduce the cost as well as time of the prediction system. This will achieve the high efficiency in training datasets in multiple domains [8]. Another existing method is Bayesian classification with spiral model-based for proficient prediction and classification of software fault. This method identifies dependability of software modules and testing the life cycles of each module. The Bayesian classifier exactly classifies the defective and good software modules [9]. A different fault prediction model is built with the help of Least Square Support Vector Machine (LSSVM) training method related to the kernel function, linear function and radial basis. This method identified the Object Oriented metrics of source code is able to forecast defective and good classes with more accuracy and reduced the value of misclassified errors [10]. In the earlier system software developers struggled in some issues such as spend more time for identifying the permanent defects in the software, high budget for recognizing the SD based on misclassified faults. In this paper, cost based Random Forest (CRF) algorithm is utilized to minimize the Error Rates(ER) and improves accuracy for SFP by means user specified feature budget. Using CRF user can able to include their affordable amount of features in their budget like for time, memory etc. With help of CRF conditions on pruning the trees of random forest the classification is constrained into affordable budgets and the risk forecast is done within user's affordable criteria while existing forecast processes give out some risk values to a software for which user might not be able to afford some necessary features.

The rest of the article is composed in the following manner. Section 2 comprises the analysis and survey of the recent existing SDF models. Section 3 proposes the CRF based approach for an effective SDF system and its evaluation and results are discussed over at section 4. Section 5 concludes the CRF based SDF system along with the possibility to enhance at future.

## 2. Literature review

G. Abaei, A. Selamat, and H. Fujita [11] presented a SD recognition system using semi-automatic Hybrid Self-Organizing Map (HySOM). The HySOM method is a semi-automatic model derived from SOM and ANN. The Benefit of this model is the capability to forecast the label of modules in a semi-automatic manner by means of software dimension threshold standards in the lack of eminent data. In semi-automatic HySOM, the responsibility of specialist for recognizing defective modules became less crucial and more helpful. This model efficiently recognizes the modules defects, enhanced the excellence of software building and did software testing in less time as well as budget. HySOM does not try to improve the precise forecast of software fault only focussed on resources.

R. Moussa, and D. Azar [12] presented Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) algorithm for classify the fault-prone software modules. The search direction in a particular region of search space is guided by the PSO then GA constructs a classifier recombination. This algorithm classifies the software model as defective or not. This hybrid algorithm balances the classification accuracy and reduces the misclassification rates over the particular class. This technique lags in the case of utilization of resources it wastes the available data and other resources to train for producing better results.

P.S. Bishnu, and V. Bhattacherjee [13] predicted the faults in software modules using a Quad Tree-based K-Means approach. Initial cluster centres were found by the Quad Tree which was the input to K-Means algorithm. The amount of clusters for K-Means Quad Tree-based algorithm can give K initial cluster centresare taken as input to the simple K-Means algorithm. This is facilitated by varying the value of the threshold parameter which is input to the Quad Tree algorithm. The Overall Error Rate (OER) was decreased in SDF technique by QDK algorithm. This technique has a drawback of necessity for more data to train classifier.

K. Dejaeger, T. Verbraken, and B. Baesens [14] presented the Bayesian network algorithm for SDF model. With the help of Markov Blanket (MB) Feature Selection technique selects the count of attributes in datasets. After that, MB effectively reduced the count of selected features. The outcome indicates that MB is able to reduce the count of variables while not negatively impacting performance. However, other feature assortment approaches are possibly able to select an even smaller set of well prognostic features. Bayesian approach was not able to limit its utilization of resources even though it is capable to perform precise prediction.

C. Catal, U. Sevim, and B. Diri [15] presented Eclipse based SDF with the help of Naïve Bayes algorithm. Due to deficient in software tools to computerize this forecast process no one of these prediction systems have achieved applicability in the industry. Data regarding SD and a robust SDF tool, can allow eminent managers to target on defective modules. This tool limits the data utilization with the trade off in accuracy.

To resolve the above mentioned SDF issues the proposed CRF approach had been coined out for accurate forecast with limited resources and its method of functioning will be described in later section.

## 3. Proposed methodology

Software products are rising rapidly; Software SDF has become a vital task in software design stage. SDF is very essential for reducing fault effects in software modules and enhancing the efficiency of the SDF. In this paper, CRF algorithm is proposed to minimize the ER in SDF. In this process the software is initially identified based on their dependability in each stage of the design process. CRF is used to predict the amount of fault present in the software modules and classify regarding the similarity value among them and the proposed architecture described in Figure.1.

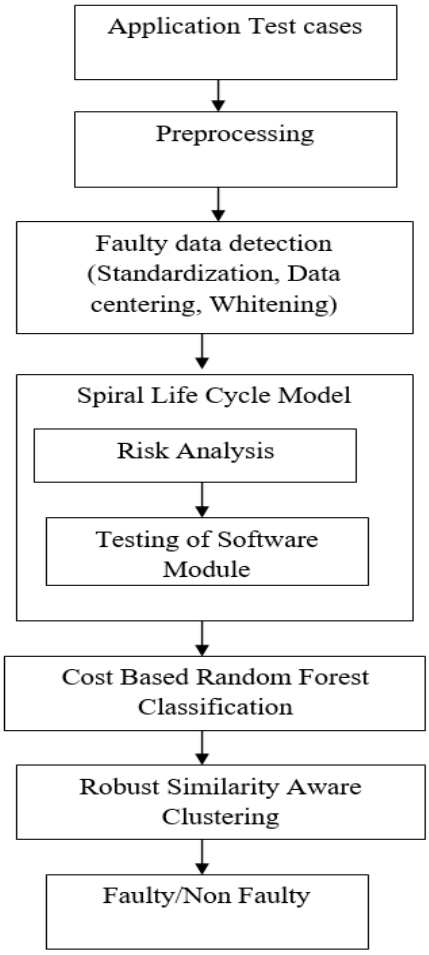


Figure.1 Proposed Architecture

### 3.1 CM1 dataset

The CM1 dataset is used for the performance diagnosis of the proposed system.CM1 is written in C language which is a NASA spacecraft instrument and composed of about 498 modules among them 10% are defective modules. Halstead and McCabe features are used as extractors of source code to obtain data. These features are already defined for object oriented characterization of the code features associated with software quality. In this process, initially identify the dependability of software modules. After that, the standardization, data centering, whitening process is performed for detecting faulty data. The software modules are tested using the spiral cycle model during each cycle of the SDP. In the earlier stage of SDP by spiral cycle model the defective module is recognized.

### 3.2 Spiral model approach

The spiral cycle model of the SDP includes four phases such as the functions of the software, identify and resolve the risk, development and testing phase and planning for the next iteration. The first stage of the spiral cycle model starts with the recognition of the functions of the software, its performance, alternative steps of developing the particular section of product and limitations given during performance of those alternates. Second stage works with diagnosis of alternates related to the constraints and objectives. Evolution of the specification of the overall nature of the product or plan for the next level of the prototype is the next step, if the program development risks are strongly dominated by the user interface risks. In the third stage, progressive growth is completed further if performance associated risks are previously resolved by the preceding prototypes. Planning for the next stage originates after the end of this incremental approach. The risks and requirements are analysed before the commencement of the SDP. Through the comprehensive analysis of the risks and requirements, it is highly guaranteed that the system contains only feasible and possible requirements. Furthermore, the developing phase of the spiral cycle model does sequential investigation of product. For identifying and evaluating software project risks the spiral cycle model is implemented as a risk based software testing model. By mitigating these risks of the software project the overall cost is reduced. Earlier detection of the risks is definite in the spiral cycle model. Among other training models the unambiguous risk administration is terminated in spiral cycle model.

### 3.3 Cost based random forest prediction model (CRF)

A cost based random forest (CRF) learning algorithm to minimize prediction error for a user-

specified average feature acquisition budget. Tree structure which is grown by independent data sampling & feature splitting, results in a set of identical trees is called as Random Forest. Difference between the count of lines in the correct and incorrect code corresponds to the defect forecast of a software module. Using this technique, the defective modules are recognized accurately.

Suppose sample pairs $(x, y)$ are distributed as$(x, y)\ dH$. The major aim is for training a classifier $f$ from a family of functions $F$ that minimizes expected loss subject to a budget constraint,

$$f^{min} \in FE_{xy}\big[L\big(y, f(x)\big)\big], E_x[C(f, x)] \leq B \quad (1)$$

Where $L(y, y\hat{})$is a loss function, $C\ (f, x)$ is the cost of evaluating the function of f on example x and B is a user specified budget constraint. In this equation (1), the feature acquisition cost $C\ (f, x)$ is a modular function of the support of the features used by function $f$ on example $x$, that is acquiring each feature has a fixed constant cost. Then minimize the empirical loss subject to a budget constraint:

$$f^{min} \in F - \frac{1}{n}L\big(y_i, f(x_i)\big), \frac{1}{n}\sum_{i=1}^{n} C(f, x_i) \leq B \quad (2)$$

In our context the classifier f is a random forest, $T$, consisting of $K$ random trees, $D_1, D_2, D_3, \ldots\ldots, D_K$, which are learnt while training data. Consequently, the expected cost for an instance $x$ during prediction-time can depicted as follows,

$$E_f[E_x[C(f, x)]] \leq \sum_{j=1}^{K} E_{D_j}[E_x[C(D_j, x)]] \quad (3)$$

where, $E_{D_j}[E_x[C(D_j, x)]]$expected cost values in the RHS are averages with respect to each of the random trees. Since the trees of a random forest are distributed identically, the RHS increases with the count of trees. This upper limit restricts the complex function of a random forest because of low feature correlation between trees. With the help of CRF approach, developers spend less time for detecting the fixed fault in the software and reduce the cost of SFP.

### 3.4 Robust similarity aware clustering (RSC)

RSC focus on categorizing the software modules according to the measure of similarity between the features. Thus the classification of the software modules as defective and correct modules is accomplished. RSC performs much faster than other regular clustering algorithms and also it is robust towards the outliers. RSC necessitates a measure of similarity among the different collection of features. According to the smallest amount of distance measures, RSC creates several amounts of clusters. Each cluster is represented by a cluster centroid. The level of similarity among the sample module and centroid is fixed, and distance within the features is computed. The clusters are created according to the distance between features. The smallest amount of distance within the features should be considered, so that they are located close to one another within each cluster centroid. The samples present in the training part of the dataset are assigned to the cluster according to the similarity measurement.

The variance value $\beta$ is calculated as

$$\beta = \frac{\sum_{j=1}^{n}\|x_j - \bar{x}\|^2}{n} \quad (4)$$

where,

$$\bar{x} = \frac{\sum_{j=1}^{n} x_j}{\beta} \quad (5)$$

Temporary Variable is computed as,

$$Temp = \frac{\|x_j - x_k\|^2}{\beta} \quad (6)$$

The similarity coefficient between the attributes is calculated by using the following equation

$$P_{sim}(x_a)_{\rho m} = \sum_{j=1}^{n}(exp - (Temp))^{\rho m} \quad (7)$$

where, $k$ =1…, n and $\rho m$ =5m.

For classification threshold coefficient is compared against similarity coefficient. If the similarity coefficient is above the threshold value, then the clustering factors are determined. The cluster centroid is calculated as,

$$z_i^k = \frac{\sum_{j=1}^{c} s_{ij}^{\rho} x_j}{\sum_{j=1}^{c} s_{ij}^{\rho}} \quad (8)$$

The level of similarity within the sample data and cluster centroid is computed.

$$S_{ij}^k = S^k\big(x_j, z_i\big) = (\exp(-Temp))^k \quad (9)$$

The maximum cluster centroid is given as

$$Z_i = max\big\|z_i^k - z_i^{k-1}\big\| \quad (10)$$

where, $Z_i = Z_1, Z_2, Z_3 \dots \dots, Z_n$

The maximum cluster centroid is compared against the classification threshold. The distance within the features is computed as

$$d_i = Z_i - Z_{i-1} \quad (11)$$

The clusters are formed with the least amount of distance among the features. The distance between the clusters are depicted as

$$Cd_i = C_i - C_{i+1} \quad (12)$$

The minimum cluster distance is removed from the cluster (C) and the minimum distance between the clusters is computed as,

$$\min(mCd_i, mCd_{i+1}) = minCd_i \; for \; all \; c_i \; in \; C \quad (13)$$

The proposed technique is reducing the misclassification rate software modules and improves the SDP quality. With the help of RSC easily classify the defective and correct software modules separately.

## 4. Experimental Result

The proposed algorithm has been written as code and evaluated in Java NetBeans 8.2 version and 32 bit operating system, 8GB RAM. The main aim of the research work is to predict the amount of fault present in the software modules and classify the faults according to the measure of similarity among them. The CM1 dataset is used for the performance analysis. In order to compute efficiency of the proposed system an evaluation metric is employed. It contains a set of measures that pursue a general underlying evaluation methodology. Some of the metrics are selected for evaluation purpose, namely F-Measure, Recall, Accuracy, Misclassification Rate, and OER.

Accuracy: For the effective forecast of the software modules rapidly the classification technique is employed. Accuracy is defined as the ratio of forecasted defective modules $(TP + TN)$ that are found among all software modules $(TP + TN + FP + FN)$. Accuracy is the ratio of the correct forecast to the total forecast done by the SDF model and formulated as:

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \times 100 \quad (14)$$

where, $TP$ = True Positive $TN$ =True Negative
$FP$ =False Positive and $FN$ -False Negative

F-Measure: Harmonic mean of precision and recall values is said to be F-Measure. It is defined as,

$$F - Measure = \frac{2 \times Recall \times Precision}{(Recall + Precision)} \times 100 \quad (15)$$

Recall: Recall is referred as the count of modules that are correctly forecasted as defective $(TP)$ to the total count of software modules $(TP + FN)$.

$$Recall = \frac{TP}{TP+FN} \quad (16)$$

False Positive Rate: FPR is called as the proportion of correct modules that are forecasted as defective module. The FPR is the relative amount of the $FP$ value to the summation of the $FP$ and $TN$ values.

$$FPR = \frac{FP}{FP+TN} \quad (17)$$

False Negative Rate: FNR is said to be the proportion of defective modules that are forecasted as correct module. FNR is the contribution of the $FN$ value to the total of the $TP$ and $FN$ values.

$$FNR = \frac{FN}{TP++FN} \quad (18)$$

Precision: Precision is described as the relative amount of the count of modules that are correctly forecasted as defective $(TP)$ to the total count of modules that are forecasted as defective $(TP + +FN)$. If the precision value is high, the time and effort required for testing and inspecting the modules is reduced.

$$Precision = \frac{TP}{FN+TP} \quad (19)$$

Overall Error Rate(OER): In the SDF process, the OER is referred as the proportion of the defect forecast $(FP + FN)$ to the total count of predictions $(TP + TN + FP + FN)$ . ER of the defective modules generally incurs much higher rate than the ER of the correct modules. The overall rate is the relative amount of the total of $FN$ and $FP$ values to the sum of $TP$, $TN$, $FP$ and $FN$ values. The OER is used to diagnose the variation in the errors in the SDF process.

$$OER = \frac{(FP+FN)}{(TP+TN+FP+FN)} \times 100 \quad (20)$$

The below Table 1 represent the performance evaluation of FNR, FPR and OER analysis report of CRF technique and existing algorithm. The proposed CRF approach is compared against the existing techniques such as semi supervised ANN, Hybrid SOM, and Semi-NB of the work done by C.G. Dhanajayan, and S.A. Pillai [9].

The above table 1 indicates the OER, FPR and FNR value is lower compare to the existing techniques for the CM1dataset. Hence, the proposed CRF technique can predict and classify the faulty modules effectively than various existing techniques.

Table 1. Performance evaluation of FNR, FPR and OER of proposed and existing technique

| Techniques | FNR | FPR | OER |
|---|---|---|---|
| HySOM [9] | 0.625 | 0.500 | 0.181 |
| Semi-ANN [9] | 1.000 | 0.340 | 0.198 |
| Semi-NB [9] | 0.729 | 0.271 | 0.135 |
| Proposed CRF | 0.520 | 0.05 | 0.102 |

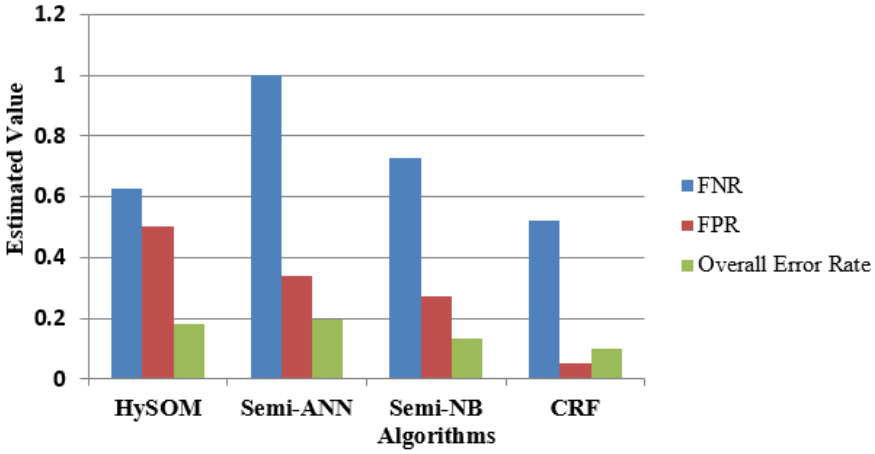


Figure.2 FNR, FPR and OER evaluation of the proposed CRF approach and some of the existing techniques

The Fig.2 depicts the performance evaluation of FNR, FPR and OER of prospered and existing. C.G. Dhanajayan, and S.A. Pillai [9] presented methods such as HySOM, Semi-ANN (Semi-Artificial Neural Network), and Semi-NB (Semi- Naïve Bayes). The proposed CRF technique represents the less ER compare to the existing technique and reduces the ER of software modules. So, overall performance of proposed technique is relatively higher than the existing approaches.

The Table 2 depicts the precision recall, F-measure and accuracy of CRF and existing technique. The existing HySOM and Semi-ANN precision is approximately same and better accuracy. As, the Semi-NB consists of precision is very low, so the accuracy is decreased. Finally, our proposed CRF technique represents the better results compared to the existing approaches like Bayesian technique etc.

The Fig.3 depicts results of the recall and precision of the CRF and existing technique. The comparison graph of the precision and recall of the proposed CRF technique and existing approaches performance are shown in the above figure. Better fault prediction performance than the existing techniques is achieved with higher precision and recall. The Fig.4 indicates the comparative graph of the F-Measure and accuracy of the CRF and existing approaches from C.G. Dhanajayan, and S.A. Pillai study [9]. From the graph, it is clearly inferred that the accuracy of the CRF approach is higher than other existing methodologies. F-measure value is also inferred to be higher than the existing systems such that it indicates that the exact prediction of software modules by CRF methodology.

Table 2. Precision, recall, accuracy and F-measure of proposed and existing technique

| Techniques | Precision | Recall | Accuracy (%) | F-Measure (%) |
|---|---|---|---|---|
| HySOM [9] | 0.890 | 0.820 | 88 | 80 |
| Semi-ANN [9] | 0.895 | 0.854 | 89.3 | 85 |
| Semi-NB [9] | 0.765 | 0.894 | 87.3 | 90.3 |
| ProposedCRF | 0.973 | 0.926 | 92 | 92.7 |

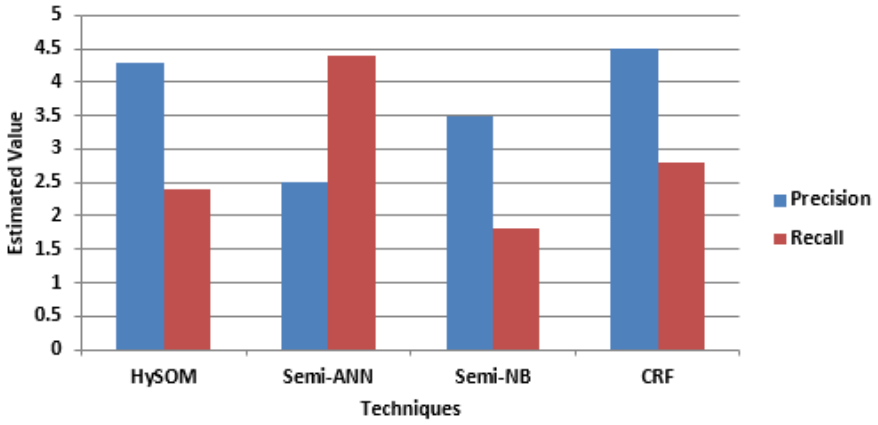


Figure.3 Precision and Recall performance of proposed and existing techniques

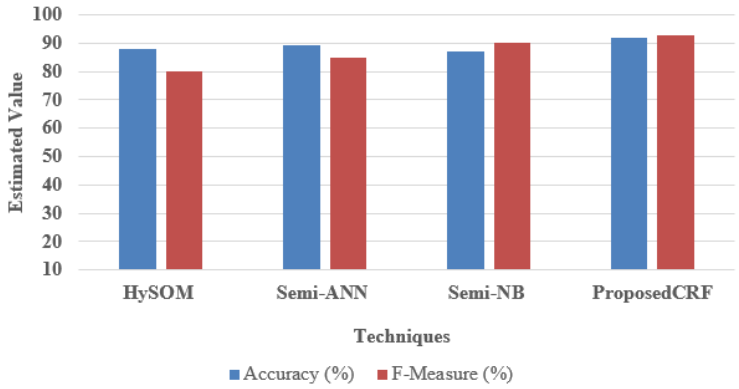


Figure.4 Accuracy and F-measure performance of proposed and existing techniques

From the performance analysis results, it is evident that the majority of the software does not cause faults in software systems, and only less of all modules is found to be the faulty modules. The majority of the modules belong to the correct software group and the rest belong to the faulty group. It is also observed that the classification and prediction performance of our proposed technique achieves better accuracy and low ER with the help of getting the prior knowledge what risk the user can take via a user-defined feature budget. CRF has the capability of pruning itself into the fixed range of features and classify the risk value of those features with higher concentration for the firm budget fixed by user. On the other hand, existing SDP systems they try to concentrate on all the possibilities without any knowledge about user requirements and gives out a risk value without any concern that whether that quoted features for affording that predicted risk can be possessed or not. So, they fell in a drawback of less accuracy and higher ER. CRF tackles this limitation effectively with the concept of user-defined feature budget and produce efficient SDP with better accuracy and less ER.

## 5. Conclusion

Software fault forecast is important in improving the quality of a software system. The spiral cycle model and cost based random forest classification models effectively predict the software faults and classify the SD. In this process initially, the dependability of software modules is recognized by spiral model. The spiral model is used for testing the software in each cycle of the SDP. Based on the measure of the similarity of features in the dataset, the RSC algorithm performs categorization of the defective and correct modules. The proposed technique accurately predicts and classifies the faulty modules. The experimental analysis demonstrated that the proposed algorithm is better than various existing approaches with respect to Accuracy, Precision, Recall, and F-measure. In future, the CRF approach can be enhanced in a further way that requires only lesser training data and short span of time to perform SDF.

## Acknowledgments

The authors would like to thank PES university for allowing to do research and would like to thank Dr. K N B Murthy, vice chancellor, PES university for his encouragement to do research.

## References

[1] C. Jin, and S.W. Jin, "Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization", *Applied Soft Computing*, Vol.40, pp.283-291, 2016.

[2] L. Chen, B. Fang, and Z. Shang, "Software Defect Forecast based on one-class SVM", In: *Proc. of International Conf. On Machine Learning and Cybernetics (ICMLC)*, pp.1003-1008, 2016.

[3] Y. Abdi, S. Parsa, and Y. Seyfari, "A hybrid one-class rule learning approach based on swarm intelligence for Software Defect Forecast", *Innovations in Systems and Software Engineering*, Vol.11, No.4, pp.289-301, 2015.

[4] S.S Rathore, and S. Kumar, "Towards an ensemble based system for predicting the count of software faults", *Expert Systems with Applications*, Vol.82, pp. 357-382, 2017.

[5] Z.A. Rana, M.A. Mian, and S. Shamail, "Improving Recall of software defect prediction models using association mining", *Knowledge-Based Systems*, Vol.90, pp.1-13, 2015.

[6] R. Mahajan, S.K. Gupta, and R.K. Bedi, "Design of Software Defect Forecast model using BR technique", *Procedia Computer Science*, Vol.46, pp.849-858, 2015.

[7] S. Kanmani, V.R. Uthariaraj, V. Sankaranarayanan, andP. Thambidurai, "Object-oriented Software Defect Forecast using neural networks", *Information and software technology*, Vol.49, No.5, pp.483-492, 2007.

[8] T. Sethi, "Improved approach for software defect prediction using artificial neural networks", In: *Proc. of International Conf. On Reliability, Infocom Technologies and Optimization*, pp. 480-485, 2016.

[9] R.C.G. Dhanajayan, and S.A. Pillai, "SLMBC: spiral life cycle model-based Bayesian classification technique for efficient Software Defect Forecast and classification", *Soft Computing*, Vol.21, No.2, pp.403-415, 2017.

[10] L. Kumar, S.K. Sripada, A. Sureka, andS.K. Rath, "Effective fault prediction model developed using Least Square Support Vector Machine (LSSVM)", *Journal of Systems and Software*, pp.1-28, 2017.

[11] G. Abaei, A. Selamat, and H. Fujita, "An empirical study based on semi-supervised hybrid self-organizing map for Software Defect Forecast", *Knowledge-Based Systems*, Vol.74, pp. 28-39, 2015.

[12] R. Moussa, and D. Azar, "A PSO-GA approach targeting fault-prone software modules", *Journal of Systems and Software*, Vol.132, pp. 41-49, 2017.

[13] P.S. Bishnu, and V. Bhattacherjee, "Software Defect Forecast using quad tree-based k-means clustering algorithm", *IEEE Transactions on knowledge and data engineering*, Vol.24, No.6, pp.1146-1150, 2012.

[14] K. Dejaeger, T. Verbraken, and B. Baesens, "Toward comprehensible Software Defect Forecast models using bayesian network classifiers", *IEEE Transactions on Software Engineering*, Vol.39, No.2, pp.237-257, 2013.

[15] C. Catal, U. Sevim, and B. Diri, "Practical development of an Eclipse-based Software Defect Forecast tool using Naive Bayes algorithm", *Expert Systems with Applications*, Vol.38, No.3, pp.2347-2353, 2013.