Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 432 – 442

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

Adaptive PSO based Association Rule Mining Technique for Software Defect Classification using ANN

B.Dhanalaxmi[a,*], G.Apparao Naidu[b], K.Anuradha[c]

[a]Department of Information Technology Institute of Aeronautical Engineering, Hyderabad,India
[b]Department of Computer Science and Engineering, JBIET, Moinabad, Hyderabad,India
[c]School of Computing, GRIET Hyderabad, India, Pin: 500090

Abstract

The proposed system categorizes various defects by using association rule mining dependent problem classification approach, which is applied to collect the actual defects using recognition. Association rule mining algorithm at times results in useless policies. To avoid this kind of concerns, the principles prior to classification determined by assistance as well as confidence value has to be optimized. In this exploration, Adaptive Particle Swarm (APSO) optimization algorithm is used. This can discover the best assistance and confidence value to have the best policies. And finally Artificial Neural Network (ANN) can be used to classify the actual defects determined.

Keywords: Association rule mining, Adaptive particle swarm optimization algorithm, Artificial neural network, Software testing, Defect prevention.

1. Introduction

Defect will be damaging in most levels connected with software package development. Any defect is often a drawback, insufficiency or even inaccuracy in the software package item. Problem stays in the experience of living connected with software package; each and every defect that arises in the software package levels is the defect for the reason that software package[1]. Exploration in program code critiques features typically devoted to defect matters

*Corresponding author. Tel.: 09866546280
E-mail address: dhanalaxmib1283@gmail.com

doi:10.1016/j.procs.2015.02.041

rather than defect kinds, which offers an imperfect view connected with program code examine positive aspects[6]. The huge number of study about the dissimilarities connected with defects in addition to their particular character cannot be protected completely in this article although most of us give full attention to a number of notable examples. We could roughly separate these into few classes: (1) defect taxonomies, (2) root cause analysis, in addition to (3) defect group[10]. The price of finding in addition to repairing bugs or even defects is the major solitary expense take into account a brief history connected with software package. Bugs maintenance tasks focus on needs in addition to go on through development[14].

Software defect recognition aspires in order to automatically determine faulty software template modules for productive software test out so that you can increase the caliber of some sort of software system[13]. Computer software high quality has become an important difficulty considering that prolonged. There are several professions linked to the caliber of software yet we now have concentrated towards defect elimination for outsourcing techniques tasks. Recently, high quality is usually accepted as the most important element that has strong have an effect on achievement on the software product[2]. The products some sort of software system might be described using different capabilities for instance dependability, maintainability etc.[15]. Among software high quality improvement actions screening is usually likely the most important. It's, thus, involving distinct fascination to gauge along with appreciate how beneficial is usually a distinct set of evaluations with respect to it is chance to identify by far the most troublesome (post-release) defects[8]. Computer software high quality begins while using the treatment or even considerable decrease involving software defects ahead of other high quality capabilities for instance maintainability, portability, dependability, or even simplicity could be[5]. Defect elimination is usually a high quality confidence alternate for strengthening high quality involving software product. The target involving defect elimination methods would be to generate far better high quality items in available cost range along with time period. Top quality of a product directly relates to the quantity of defects: small defects bring about accomplish far better high quality[12].

Software defect elimination function generally concentrates on specific evaluation along with testing method. ODC is usually a mechanism where computer software problem of which happen in the computer software development life cycle will be exploited[11]. Defect avoidance is one of the most essential although habitually overlooked element of computer software high quality assurance in any challenge. When functional whatsoever levels associated with computer software development, it could reduce time, overheads along with wherewithal required to help manufacture a high quality product[4]. Defect avoidance is usually a process of figuring out most of these defects, their causes along with fixing these and to avoid those through continuing[3]. Defect avoidance may boost both high quality along with productivity. When the quantity of defects treated lowers, then the high quality improves as the quantity of extra defects in the supplied computer software minimizes[7]. Defect avoidance is the many essential although habitually overlooked facet of computer software high quality assurance in any challenge. When functional whatsoever levels associated with computer software development, it could reduce time, overheads along with wherewithal required to help manufacture a high quality product[9].

2. Related Works

A quick determine of your couple of most up-to-date experiments is exists here. A new test request improvement product made to face these kinds of troubles with the PT3800 tester, continuity and also insulation resistance tester have been spelled out through Eric Bean[16]. The actual improvement device seemed to be thought out using prominent built-in drawback avoidance and also locating functions. The outcomes undoubtedly are a diminution in rephrase and a two-fold increase in output. The actual drawback avoidance and also locating functions were spelled out and also his or her applicability to help other test equipment software program.

SakthiKumaresh along with Baskaran Ramachandran[17] have recommended any defect composition that is based on analyzing this problems of which experienced by a variety of development connected with software program growth similar to Demands, Design, Coding, Testing and Timeline (defects caused by deficit of time in the course of development). Their particular function isn't limited to simply for pinpointing the origin connected with problems at a variety of periods connected with software program growth but additionally experienced found the causes regarding these kinds of problems, along with defect precautionary (DP) actions.

Utilizing a scenario-based technique that finds out the actual important characteristics along with probable complications terrifying the system from the stakeholder's perspective, an approach branded SHADD regarding defect detection may be suggested simply by Sayed MehranSharafi[18]. Instances were subsequent utilized as being a basis with the course of action regarding new defects detection.

Arpita Mittal and also Sanjay kumarDubey[19] discovered around the different varieties of defect methods and then they've gone through with the review of COQUALMO cost positive model that is some sort of two-step software program imperfection prediction model for creating the program quality. Upfront they've proposed the imperfection presented from the software program, the enhanced software program will probably effect.

In a situation modify concerning empirical analyze involving tasks from the primary product-based application market has become encouraged by simply T.R. Gopalakrishnan Nair et al.[20]. This kind of pursuit robustly noticed that your require pertaining to alertness in addition to apply involving quality description involving procedure and the ones in realizing productive defect operations. Delivery involving two newly launched quality metrics depth involving evaluation, an operation metric in addition to inspection overall performance metric, any people metric accomplish your bettering crew to provide high-quality application.

Driving basic principle for your defect relaxed evaluation have been offered by Marcos Kalinowski et al.[21]. To understand as well as look at leads to relate with specific defect kinds, Deficiency causal analysis (DCA) was a prepared method offering organizations chances to formulate method resources and to acquire behavior to be able to evade this recurrence of the flaw sort. For that reason, DCA supplies a procedure intended for product-focused software package method development based on product defect details.

T.R. Gopalakrishnan Nair along with R. Selvarani[22] have riveted details powered methodology that is good observed modify with the relationship displayed amid pattern variables along with problem proneness. In initial stage, some sort of mapping with the correlation one of many pattern metrics along with usual happening structure regarding flaws have been produced. That was signified seeing that a collection of non linear multifunctional regression equations which in turn replicates the power regarding specific pattern metrics about problem proneness. By simply weighted linear amalgamation of such multifunctional regression equations, the particular problem proneness analysis type was next produced. By way of GQM paradigm, the particular weighted coefficients were usually considered.

3. Problem Definition

1. The cost of finding the defects represents one of the most expensive software development activities.
2. To generate patterns, Association rule mining is a major technique. On the other hand, as huge numbers of association rules which may be sometimes insignificant.
3. Genetic optimization algorithm is limited to random solutions and convergence.
4. Association rule mining is frequently very expensive.
5. For extracting rules from software using a single mining technique will not be efficient. So an optimization technique is required.
6. The Naive Bayes classifier requires a very large number of records to obtain good results.

4. Proposed Methodology

Defect reduction may be the majority of vivid nevertheless usually dismissed attribute of software package good quality warranty in different project. In the event that helpful by any means steps of software package improvement, it could possibly slow up the time, overheads as well as methods included to manufacture a superior quality merchandise. This proposed system categorizing diverse errors by using association tip exploration structured deficiency category method, that's placed on group this defects following recognition. Association rule mining formula sometimes causes incomprehensible rules. Therefore it is very difficult to classify these defects dependent on these incomprehensible rules. To prevent like problems, we must enhance the principles prior to category dependent on help as well as confidence value. In this investigation, we've centered on seeking the best rules dependent on adaptive Particle Swarm (PSO) optimization strategy. This may obtain the best help as well as confidence value to obtain best rules. Right after getting best rules, we all will probably classify these defects dependent on artificial neural network. Lastly the quality is going to be assured by employing different good quality metrics like deficiency solidity, Level of sensitivity and many others.

The software defect prediction dataset KC1 is used in the defect prediction work. The input attributes used from the dataset are Essential complexity, Design complexity, Total operators + operands, Intelligence and False, True. The features were subjected to frequent item set mining in order to extract the association rules.

4.1. Frequent Item Set Mining

Every product may be the component pair of candidate. This help values are generally computed with the product sets independently. When using the system, the particular support value is pushed

$$Support\,(A \rightarrow B\,) = P\left(A \cup B\right) \tag{1}$$

Where A and B = Frequent item sets.

While using minimum support value, this support values are estimated for your separate product packages are usually when compared. An item packages along with support value below this minimum support value is usually taken away. This left product packages are usually picked. Future this picked product packages had been merged while using exact same product packages. Using the support value, once more this support value is usually computed for your product packages plus they are taken away. Through the removal as well as the pruning move the product set which is intended for providing connection rules are located out. Using the formula, this confidence value can be determined.

$$Confidence\,(A \rightarrow B\,) = \frac{P\left(A \cup B\right)}{P\left(A\right)} \tag{2}$$

Where A and B = Frequent item sets.

To produce association guidelines, your typical item set made finally as well as the lowest confidence value is actually utilized. The complete item sets made have been in use. The product sets with the objects in typical item set is actually known. Applying solution (2), your confidence values for that decided on item sets have been computed. The product sets with confidence value greater than your lowest confidence value allotted usually are decided on. The actual left item sets usually are diminished. The actual decided on item sets will be the association guidelines. The association rules produced is given as input to the adaptive PSO algorithm for optimization.

4.2. Adaptive Particle Swarm Optimization (APSO)

Population based search protocol may become Particle swarm Optimization (PSO). It really is made to help pretend the good manners regarding birds throughout hunt for foodstuff using a cornfield or even fish institution. The technique can effectively come across best or even around best remedies throughout big search rooms. You'll find a pair of different sorts of variants are used as outlined by PSO. The foremost is “individual best” in addition to the second is “global best”.

As a result, Adaptive Particle Swarm Optimization (APSO) approach is required offering far more specific clustering result the spot that the inertia weight can be as effectively regarded as. Right now, your association principles were optimized by way of APSO. The working technique of your APSO is actually chosen in figure 2.

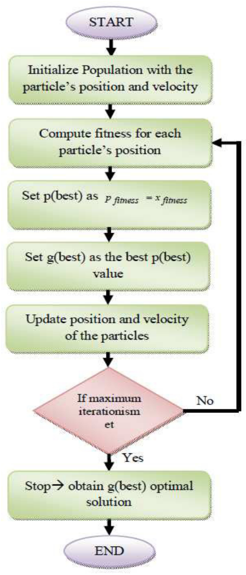
START
Initialize Population with the particle's position and velocity
Compute fitness for each particle's position
Set p(best) as $p_{fitness} = x_{fitness}$
Set g(best) as the best p(best) value
Update position and velocity of the particles
If maximum iterationism et
No
Yes
Stop→ obtain g(best) optimal solution
END

Fig. 1. Working Mechanism of APSO

4.2.1 Association rules optimization using APSO

- **Swarm initialization:** For a population size u, develop the particles randomly.
- **Define the fitness function:**In line with the provided population, the fitness function need to beestablished for the constraints. The following eqn. (3) is the fitness function.

$$fitness = \sum_{i=1}^{n} x_1 + x_2 + ...x_n \tag{3}$$

- gb **and** pb **Initialization:**

In the beginning the fitness value estimated for every particle is set as the Pbest value of each particle. Among the Pbest values, the optimal one is preferred as the gb value.

Velocity Computation: The novel velocity is determined by the following equation. (4),

$$V_i^{d+1} = w^d V_i^d + c_1.r_1.(pb_i^d - x_i^d) + c_2.r_2.(gb^d - x_i^d) \tag{4}$$

$$x_i^d = x_i^d + \delta V_i^d \tag{5}$$

In eqn. (4),

c_1, c_2 - constants with the value of 2.0

w^d - Inertia weight.

r_1, r_2 - independent random numbers generated in the range [0.1]

V_i^d - velocity of ith particle

x_i^d - current position of the particle i

pb_i^d - best fitness value of the particle at the current iteration

gb^d - best fitness value in the swarm.

- **Inertia weight calculation**

$$w^d = (w_{mx} - w_{mn}) \times (m_{it} - d) / m_{it} + w_{mn} \tag{6}$$

w_{mx} and w_{mn} - Maximum and minimum Inertia weight

m_{it} - Maximum number of iteration

- **Swarm Updation:**Determine the fitness function again and revise the pb and gb values. If the novel value is better than the previous one, exchange the old by the current one. And in addition pick the optimal pb as the gb .
- **Criterion to stop:** Prolong till the solution is actually suitable or maximum iteration is achieved.

After the association rules optimization is established the defects were classified using Artificial Neural Network

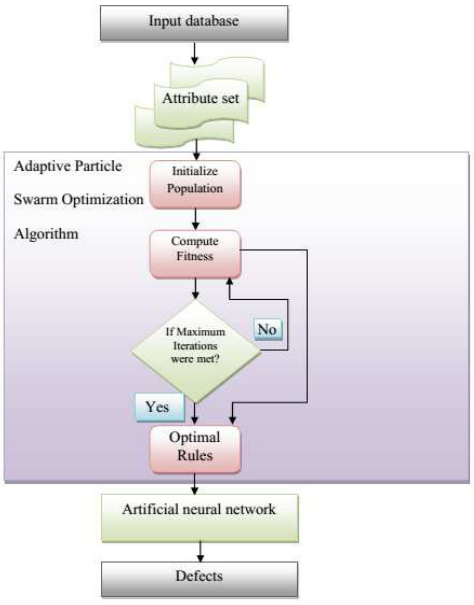

Fig. 2. Architecture of the proposed methodology

4.3 Classification Phase using Artificial Neural Network

Among the category methods making use of Feed Forward Back Propagation Neural Network classifier (FFBNN) is employed regarding classifying software problems. Neural network is often a three-layer regular classifier together with in suggestions nodes, n input nodes and l hidden nodes and k output nodes. It really is looked at that if both the invisible levels utilized the other invisible layer is to affiliate each and every couple in a single significant unit and next is undoubtedly to be the real invisible layer immediately after classifying this input data from the initial invisible layer. For our proposed task, this input levels will be the association guidelines, HU_a Hidden Items the other production unit, f . The particular framework from the FFBNN classifier is usually confirmed from the pursuing fig. 2.

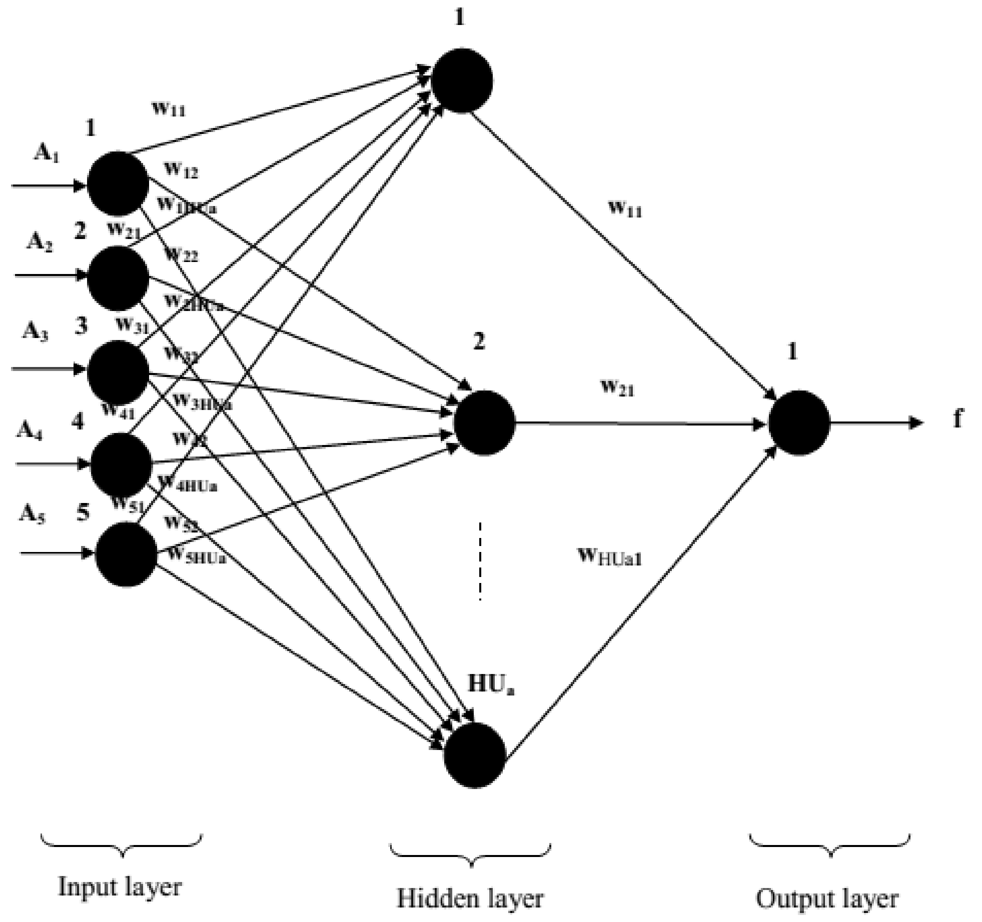

Fig. 3. Structure of FFBNN classifier for the proposed work

4.3.1 NN Function Steps

1) Fix loads for every neuron's except the neurons in the input layer.
2) Develop the neural network with the extracted features $\{A_1, A_2, A_3, A_4, A_5\}$ as the input units, HU_a Hidden units and age f as the output unit.
3) The computation of the suggested Bias function for the input layer is,

$$X = \beta + \sum_{n'=0}^{H_{NH}-1} W_{(n')} A_1(n') + W_{(n')} A_2(n') + W_{(n')} A_3(n') + \ldots + W_{(n')} A_5(n') \tag{7}$$

The activation function for the output layer is estimated as

$$\text{Active}\,(\mathrm{X}) = \frac{1}{1+\mathrm{e}^{-\mathrm{X}}} \tag{8}$$

4) Recognize the learning error as offered beneath.

$$LE = \frac{1}{H_{NH}} \sum_{n'=0}^{N_{NH}-1} Y_{n'} - Z_{n'} \tag{9}$$

where, LE - learning rate of FFBNN.

$Y_{n'}$ - Desired outputs.

$Z_{n'}$ - Actual outputs.

4.3.2 Learning Algorithm – Back Propagation Algorithm used for minimizing the error

In Feed Forward Neural System, Back Propagation Algorithm is employed as the Learning Algorithm. Back Propagation Algorithm is really a monitored Learning strategy and more over it is definitely a breakdown of delta rule. To make working out collection, it needs a dataset of the necessary productivity for numerous inputs. Typically, Back Propagation Algorithm is ideal for Feed-Forward Networks. That Learning algorithm wants that the service purpose utilized by the neurons be differentiable.

Back propagation Algorithm Steps for FFBNN

1) The loads for the neurons of hidden layer and the output layer are developed arbitrarilyselecting the weight. Nevertheless the input layer has the constant weight.
2) The suggested Bias function and the activation function are estimated using Eqn. (7) and (8) for the FFBNN.
3) The Back Propagation Error is determined for each node and then the weights are updated as follows,

$$W_{(n')} = W_{(n')} + \Delta W_{(n')} \tag{10}$$

4) The weight $\Delta W_{(n')}$ is transformed as follows.

$$\Delta W_{(n')} = \delta .\ X_{(n')} .\ E^{(BP)} \tag{11}$$

where, δ - Learning Rate, which normally ranges from 0.2 to 0.5.

$E^{(BP)}$ - BP Error.

5) The process is recurred applying (2) and (3) steps, until the BP error gets minimized. i.e. $E^{(BP)} < 0.1$.
6) If the minimum value is received, then the FFBNN is properly qualified for the screening phase.

Consequently, FFBNN classifier is properly qualified and the association principles are tried utilizing the attributes. The classification of flaws is moved out which categorizes the whole flaws are created from the classifier.

5. Results and Discussion

The proposed Defect classification system with Artificial neural network is implemented in the working platform of NETBEANS version 7.2 (jdk 1.7). Here we have applied some data in PROMISE dataset such as KC1/ software defect predictionhttp://promise.site.uottawa.ca/SERepository/datasets/kc1.arff. PROMISE indicates the Predictor Models in Software Engineering. PROMISE data sets is to handle noise (e.g. with outlier eradication, or characteristic subset collection, or statistical methods that deal with outliers better) in order to complete lacking data with surrogates coming from other features.

The quality of the system is determined using the quality metrics. The quality metrics estimated in our suggested strategy are as follows: Defect Density, Sensitivity, Specificity and Accuracy

1. Defect density

Defect density is defined as the ratio of total number of defects to the lines of code.

$$\text{Defect density} = \frac{\text{Total number of defects}}{\text{Lines of code}} \tag{12}$$

Table 1. Defect Density values

Number of defects	Lines of Code	Defect Density
28	670	.042

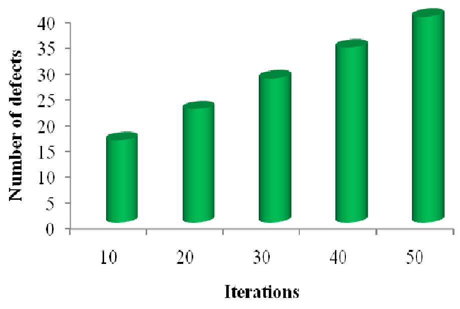

Fig.4. Graph for number of defects based on iterations

Table 2. Fitness values depending on Iterations

Iterations	Fitness Values
10	60
20	70
30	80
40	90
50	100

2. Sensitivity

Sensitivity (also named the real positive rate) determines the ratio of real positives which are appropriately estimated.

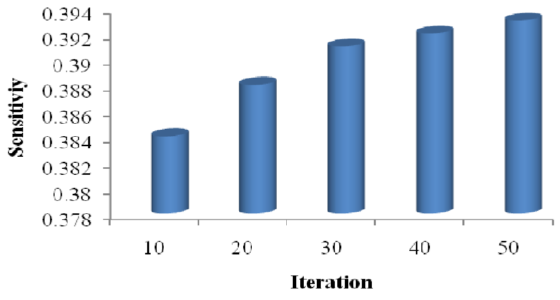

Fig. 5. Graph for the sensitivity values based on iterations

3. Specificity

Specificity determines the proportion of negatives that are appropriately identified.

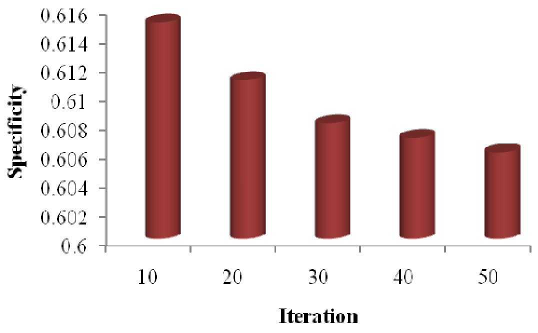

Fig. 6. Graph for the specificity values based on iterations

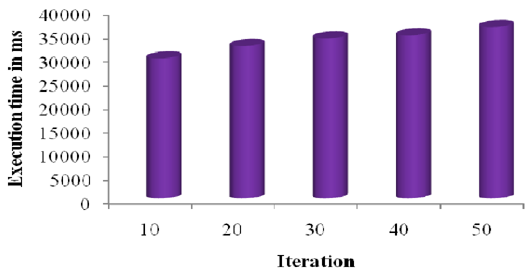

Fig. 7. Graph for Execution time based on iterations

4. Accuracy

Accuracy is estimated by employing both sensitivity and specificity relations. FAR and FRR can be calculated using sensitivity and specificity.

$$\text{Accuracy} = 100 - ((\text{FAR} + \text{FRR}) / 2) \qquad (13)$$

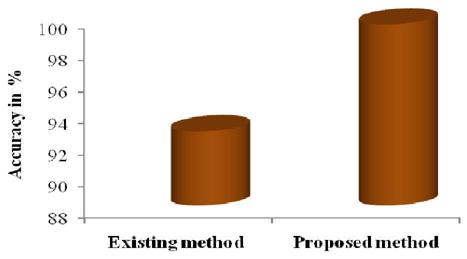

Fig. 8. Graph for comparison of Accuracy values

The Prevailing strategy introduced this can be a software defect classification centered on Association principle mining, ABC algorithm and naïve bayes classifier. Through Figure 8 it's distinct that the precision of our planned strategy efficiency was larger in comparison with the prevailing method. Hence the efficiency methods computation revealed which our planned strategy is successful than the prevailing method.

5. Conclusion

Defect classification was performed by association rule mining and artificial neural network. Association rule mining algorithm occasionally contributes to useless rules. Therefore it is really problematic for flaw classification predicated on these useless rules. To be able to prevent such problems, we've improved the guidelines applying Adaptive PSO algorithm before classification predicated on help and confidence value. In that report we've labeled the problems applying artificial neural network. The principles were removed from the input applying association rule mining. The principles were improved applying adaptive PSO algorithm. Problems were labeled applying artificial neural network. The product quality was sure utilizing the quality metrics such as for instance flaw density, Sensitivity etc. Our planned process has accomplished a precision value of 99.5%. It's larger in comparison with the prevailing method. Hence the efficiency methods formula revealed which our planned process is efficient.

References

1. Hafiz Ansar Khan, Establishing a Defect Management Process Model for Software Quality Improvement, *International Journal of future Computer and Communication*, Vol. 2, No. 6, 2013.
2. Muhammad Faizan, Sami Ulhaq and M.N.A. Khan, Defect Prevention and Process Improvement Methodology for Outsourced Software Projects, *Middle-East Journal of Scientific Research*, Vol. 19, No. 5, pp. 674-682, 2014.
3. Suma V, Defect Prevention Approaches In Medium Scale It Enterprises, *National Conference on Recent Research Trends in Information Technology*, 2008.
4. Suma. V. and T.R. Gopalakrishnan Nair, A Effective Defect Prevention Approach in Software Process for Achieving Better Quality Levels, *World Academy of Science*, Engineering and Technology, 2008.
5. IrajHirmanpour and Joe Schofield, Defect Management through the Personal Software Process, *The Journal of Defense Software Engineering*, 2003.
6. Mika V. Ma¨ntyla¨ and Casper Lassenius, What Types of Defects are really Discovered in Code Reviews?, *IEEE Transactions on Software Engineering*, Vol. 35, 2009.
7. Pankaj Jalote and Naresh Agrawal, Using Defect analysis Feedback for Improving Quality and Productivity in Iterative Software Development, *In proceedings of ITI 3rd International Conference on Information and Communications Technology*, pp. 703-713, 2007.
8. AudrisMockus and NachiappanNagappan, Test Coverage and Post-Verification Defects: A Multiple Case Study, *Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement*, ESEM, pp. 291-301, 2009.
9. Abhiraja Sharma, Naveen Hemrajani, SavitaShiwani and Ruchi Dave, Defect Prevention Technique in Test Case of Software Process for Quality Improvement, *International Journal on Computer Technology Applications*, Vol. 3, No. 1, pp. 56-61, 2012.
10. Stefan Wagner, Defect Classification and Defect Types Revisited, *ACM journal on software testing*, 2008.
11. Prakriti Trivedi and SomPachori, Modelling and Analysing of Software Defect Prevention Using ODC, *International Journal of Advanced Computer Science and Applications*, Vol. 1, No. 3, 2010.
12. Muhammad Faizan, Muhammad Naeem Ahmed Khan and Sami Ulhaq, Contemporary Trends in Defect Prevention: A Survey Report, *International Journal on Modern Education and Computer Science*, Vol. 3, pp. 14-20, 2012.
13. Yuan Jiang, Ming Li and Zhi-Hua Zhou, Software Defect Detection with ROCUS, *Journal of Computer Science and Technology*, 2011.
14. Capers Jones, Software Defect Origins and Removal Methods, 2012.
15. Stefan Wagner, A Model and Sensitivity analysis of the Quality Economics of Defect-Detection Techniques, *ACM journal on software testing*, 2006.
16. Eric Bean, Defect Prevention and Detection in Software for Automated test Equipment, *IEEE Transactions on Instrumentation and Measurement*, Vol. 11, No. 4, pp. 16-23, 2008.
17. SakthiKumaresh and andBaskaran Ramachandran, Defect Prevention based on 5 Dimensions of Defect Origin, *International Journal of Software Engineering & Applications* (IJSEA), Vol.3, No.4, 2012.
18. Sayed MehranSharafi, SHADD: A scenario-based approach to software architectural defects detection, *Elsevier Journal of Advances in Engineering Software,* 2012.
19. Arpita Mittal and Sanjay kumarDubey, Defect Handling in Software Metrics, *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 1, May 2012.
20. T.R. Gopalakrishnan Nair, V. Suma and P. Kumar Tiwari, Significance of Depth of Inspection and Inspection Performance Metrics for Consistent Defect Management in Software Industry, *Journal of IET software*, 2012.
21. Marcos Kalinowski, David N. Card and Guilherme H. Travassos, Evidence-Based Guidelines to Defect Causal Analysis, *IEEE Transactions on Software Engineering*, 2012.
22. T.R. Gopalakrishnan Nair and R. Selvarani, Defect Proneness Estimation and Feedback approach for Software design quality Improvement, *Elsevier Journal of Information and Software technology*, 2012.