Original Research Paper

System Error Estimate using Combination of Classification and Optimization Technique

Khalid Alattas

Department of Computer Science and Artificial Intelligence,
College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia

Article history
Received: 31-12-2020
Revised: 28-02-2021
Accepted: 09-03-2021

Email: kaalattas@uj.edu.sa

Abstract: The representation of software must produce flawless significances without any inadequacies. Software imperfection evaluations scheme determines defective mechanisms in software. The eventual creation would have minor or negligible shortcomings to harvest great eminence software. Software quality metrics are a division of software metrics that spotlight the quality aspects of the product. The software flaw prediction system helps in the early discovery of flaws and contributes to talented removal and producing a quality software system through numerous metrics. The aim of the paper was to show how static model of data mining is used to extract defects and the PSO algorithm. Another aim of the research was to develop an optimized software flaw prophecy system on data mining techniques namely Association Rule mining, Decision Tree, Naive Bayes and Classification integrated with Particle Swarm Optimization technique. The proposed software flaw prediction system is deliberated through Data Mining techniques with Particle Swarm Optimization algorithm has been verified and compared the results. This proposed system is very useful to identify the relationships between the quality metrics and the potential defective modules. The optimized data mining systems have pragmatic perfect prediction of these defective modules. In the future, optimized data mining systems can be improved by the use of different platforms and particularly by improving data mining using PSO algorithms. It is necessary to develop algorithms that can identify faults in advance, which will minimize costs and promote the quality of developed software systems. Future optimized data mining systems will improve the relationship between quality metrics and the potential defective modules, which will lead to improved performances, productivity and lower operation costs.

Keywords: Association Rule Mining, Data Mining, Decision Tree, Naive Bayes, PSO, Software Flaws, Software Quality

Introduction

Data mining is a technique of discovering patterns and sums up them through useful knowledge. Data mining is among the computational tools for data processing and the most common software vulnerabilities that occur in such a wide range of software and application development frameworks. Technology issues that appear in computer project development: Are commonly referred to as software failure. They result in software bug triggers that lead to system failure or produces inaccurate coding performance. In software engineering field, the software flaw prediction system is having more importance. Common flaws are created by Software developers either in the source code of the software or its architecture in platforms (Sharafi, 2012). Operating systems used by such applications and in rare situations slight errors are created by compilers producing incorrect code. Numerosity related trainings and methodologies have been accompanied to appear with the accurate failing projection model. Additionally, software flaw prediction schemes which consist of independent variables such as software metrics gathered and evaluated for the duration of software development life cycle through dependent variable maybe defective. Optimization or mathematical programming refers to the selection of a suitable element with respect to certain constraints from some set of available alternatives. Mostly the population-based evolutionary computation techniques are motivated from the evolution of nature. The test case optimization is exercised for reducing, minimizing and prioritizing the test case so that we can reduce the cost and time (Sharma *et al.*, 2012). The key stages throughout the handling of faults should involve recognition of faults, the classification of defects and analysis of the faults. In addition, detecting that fault and eliminating the fault form a critical part of handling the detection process.

A first stage would be to recognize the incidence of software malfunctions. Defects should be classified, analyzed, predicted and identified after recognition of defects. In this study an optimized software flaw prophecy system is developed based on Data Mining (DM) techniques namely Association Rule mining, Decision Tree, Naive Bayes Classification integrated with Particle Swarm Optimization technique.

Data mining and meta-heuristics have found numerous applications today, including system error estimation, weather prediction, conducting business forecasts, occurrence of natural disasters, social, political and cultural changes and diagnosis among others. Today, software development has become a highly competitive field. However, the process of software development is complex and exposed to errors, especially during the development process. In this context, data mining and meta-heuristics can be applied in his area to ensure that these errors are alleviated and the quality of systems improved. Through data mining, software developers can turn raw data into useful information, which will help them become more vigilant and reduce errors. Problem independent approaches can be applied in system error estimation through the application of algorithms, which will portray relationships between the presence of errors in software development and the competence of software developers. Data mining and meta-heuristics will be applied in the source code and the architecture of the systems. In this study an optimized software flaw prophecy system is developed based on Data Mining (DM) techniques namely Association Rule mining, Decision Tree and Naive Bayes Classification integrated with Particle Swarm Optimization technique.

Research Gap and Contribution

The current research contributes to the existing knowledge on identifying the contributions of Particle Swarm Optimization (PSO) in identification of errors during software development. Software project managers have relied on different techniques to identify faults in the process. Identification of these faults has enabled software project managers make optimal use of people, costs and time to improve quality assurance. Also, these techniques have been used to plan production process improvements. The research will provide newer techniques to software developers and managers, which, in turn will enable them develop quality software systems. This research will add to the current knowledge on how the process of software development can be optimized, ensuring all stages of the production process are in check. Ensuring that the implemented policies and standards are adhered to will improve the quality of developed systems.

Numerous researches have been conducted in the area of system error estimation to identify the relationships between the quality metrics and the potential defective modules. The current research seeks to address the existing gap on the use of novel technologies like Particle Swarm Optimization (PSO) to determine these relationships. The research will explore how Machine Learning Clustering has been developed to avoid the software system's failure. Comparing the existing techniques with PSO will enable the audience design or rather adopt effective approaches to analyze large data volumes, which will help make informed decisions and solve problems experienced during software development.

Literature Review

Whenever a project manager detects a software fault early, the manager is placed in a better position to ensure effective allocation of resources, people and also can improve the quality of the software. The technique that has proved to help in picking the function required is that of F-score. This research study has proposed the application of the LSTSVM model for predicting the defects that software might be having at any given time. The prediction model's efficiency could be improved with reduced metrics after the selection of functions and used for the identification of faulty modules. You may also equate the output of the proposed work with other feature selection techniques (Wang and Li, 2019).

Many new metrics for Object-Oriented systems have been suggested, but only a few have been validated. Each of the three metrics per se was considered helpful for predicting maintenance success in the experiment (Puška *et al.*, 2020). Authors logically know that systems with higher complexity take more time to perform maintenance tasks than systems with lower complexity. Further research to further validate metrics in the OO design complexity literature is needed. Treatments using these principles are not included (Ghaffarinasab *et al.*, 2020).

Additionally, it has been noted that prediction of software defects might increase software reliability while at the same time decreasing the costs incurred in developing it. However, there are still some predictions methods that apply the traditional methods and are considered to be less effective in doing so. The P-SVM models utilize the PSO algorithm in order to measure SVM’s best parameters. Afterwards, the model adopts the optimized SVM model to assist it in predicting programmes failures. This model has proved to be stable as individual particle errors have appeared not to influence the final optimization solution. In addition, the stability shown by this model has indicated a high robustness making it reliable. The programmes failed in the public JM1 data set is expected as an experiment and results show that the P-SVM model is predictable better than other models (Goli *et al*., 2019).

Figure 1 shows the flow of software detection, showing detection of software errors using DM and PSO. This flow begins with presentation of the work for error identification. Errors are predicted using PSO algorithms, associated data mining, decision trees D Naive Bayes. These machine learning algorithms help identify errors and help improve validity or outcomes.

Data mining is the method of identifying and encapsulating trends in useful knowledge. Data mining is an important aspect of software development. It refers to one of the scientific data analysis methods. At times during the functioning of software, it might not function as it should be and this is the point where the software is said to have indicated defects. Once software has indicated defects, it leads to incorrect performance. This is the point where tech business comes in to apply the life cycle of software to provide consumer quality for that given product. Techniques for data mining are used to gather valuable information from data repositories. A software defect is an irregular computer program execution that results in incorrect results. The prediction of software bugs works correctly on large data sets without the right data mining model. The research has many purposes under which the reliability of software is a good practice of software products. Software error monitoring is a fundamental method for error identification and bug detection. More defects can be found in the software development life cycle. There are actively involved bug reports, proper bug compilation and bug prediction (Tirkolaee *et al*., 2019).

The method that has been proposed in this research study groups various defects by utilizing a problem classification style that is founded on the association rule mining. However, the algorithm of the association rule mining might at times lead to excessive policies. Before classification has been conducted, the principles determined by assistance as well as trust value must be optimized. It should also be noted that, during the creation stages of the software, defect would be dangerous in most of these software. The developers ought not to worry much about the defects that might be associated with the software initial development stages. This situation has been eased by the fact it is possible to automatically choose defective modules of the software template to carry out software testing efficiently. One of the most important things in development of the computer software is defect avoidance since it guarantees high quality in any kind of challenge. The quality of this computer software is determined by the number of defects involved in the development process; the lower the defects noted during this process, the higher the quality. In addition to fixing bugs or flaws, the cost of detecting is the greatest solitary cost and takes into account a little background of software packages. Besides, bug maintenance tasks concentrate on needs and need for growth. Whenever the computer software production levels are functional, it will minimize the time and overhead to deliver a high-quality product (Khan, 2013). The avoidance of defects will improve both high quality and productivity if the number of defects treated decreases, the consistency increases (Suma and Gopalakrishnan Nair, 2008). The mining algorithm association rules often lead to useless regulations (Wang, *et al*., 2014). Problems with the use of the artificial neural network were found. Quality of the product was secure using quality measures such as fault density, sensitivity, etc. (Agarwal and Tomar, 2014; Trivedi and Pachori, 2010).

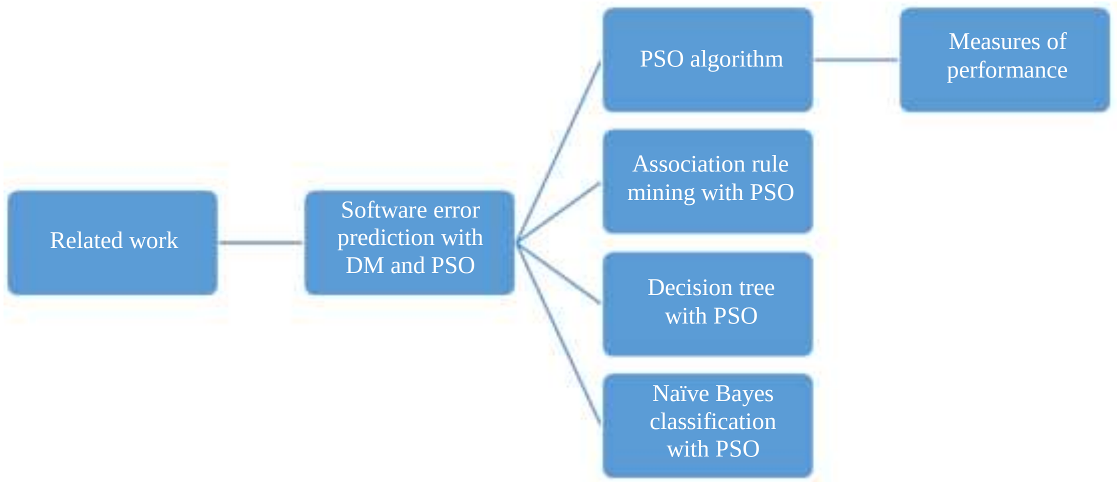
Related work
Software error prediction with DM and PSO
PSO algorithm
Measures of performance
Association rule mining with PSO
Decision tree with PSO
Naïve Bayes classification with PSO

Fig. 1: Flow of software detection

Identifying faulty computing organizations is essential for the quality assurance of information and software security is often improved. In this study, the authors suggested HyGRAR fix the problem of computer failure prediction. We have demonstrated the promise of our suggested solution in studies focused on ten open-source data sets. Detection of program malfunctions is often useful when doing a code analysis, often used as a quality improvement tool for agile software production methodologies (Nair and Selvarani, 2012). Cross-project prediction is one of the main challenges in predicting a flaw, especially according to the study. The authors claim that the training results were too imbalanced, with a relatively limited number of flawed instances compared to the number of non-defective injuries. HyGRAR is a new computer malfunction predictor solution that predicts malfunctioning modules in software systems. It is a hybrid supervised learning approach focused on incremental associations for mining and Artificial Neural Network (ANN) combinations. GRARs extracted from the mining process are then used to differentiate between faulty and non-defective tech companies (Bandi *et al*., 2003).

In this research, the author focuses on the prediction of the NASA repository dataset's programmes defect. Machine Learning Clustering has been developed to avoid the software system's failure. This thesis uses intuitionistic clustering based on K-medoids. With the support of software methods and the software fault dataset collected from software or projects based on the prediction model, software faults are trained and developed. In order to schedule, monitor and control test execution activities, testing teams may use the expected errors. Defect prediction may also be an early predictor of consistency in the testing process for any programmes. The prediction of defects as part of the evaluation process helps the test team to improve test strategies (Suma and Gopalakrishnan Nair, 2008). In this study, the software defect prediction level's performance is discussed for the Intuitionistic Fuzzy K-Medoids-based clustering of applications. The reduced features are used to cluster and the fluid logic-based rules are created to ensure classification accuracy (Can *et al*., 2013).

Studies were performed to estimate the existence of static code metrics in software code defects. This study includes a manual review of the predictions made with NASA Metrics Data Software data through Help Vector Machine Classifiers. The results indicate that the forecasts are usually well-motivated and that the classification was more 'confident' in their accurate predictions on average (Miholca *et al*., 2018). The researchers claim that an effective software defect prediction system will result in better quality, more dependable software that can be manufactured faster than previously possible. The experiment involved evaluating the performance of vector support classifiers against the same data they were trained with. NASA Metrics Data Software (MDP) registry collected data. The Data Visualization Tool was used to display the distribution of classes within the functional region. A manual analysis of forecasts for one data set showed that the ratings were generally well-motivated. SVM classifiers have consistently been more optimistic than incorrect in the incorrect predictions they have made. The findings could be included in a real-world classification system where the predicted modules could be classified according to decreasing decision values. Code inspections would then be granted priority in the form of this order (Jalote and Agrawal, 2005).

The importance of using static code attributes to learn predictors of defects was explored extensively. These predictors could be useful for a resource-based code scan that has not yet been reviewed as a priority (Faizan *et al*., 2014). Artificial intelligence can dynamically use data mines to learn consistent software predictors. Software managers may use such predictors to concentrate on parts of the system that are vulnerable to malfunctions. The meaning as a defect predictor of static code attributes has been extensively discussed. The naive Bayes data miner has outperformed the decision-making approaches used in previous work. According to scientists, Bayesian methods are smooth over fragility by analyzing various Gaussian numerical distributions (Gray *et al*., 2010). The choice of the learning approach is much more critical than the subset of data available for learning. The findings are well confirmed by predictors of construction defects of naive Bayes (with LogNums). The combination of learner philters formed predictors with average results of 1⁄4 71% and 1⁄4 25%. This is an important discovery since the static code properties do not capture the source code very well (Han *et al*., 2011).

Empirical experiments mostly on computer prediction do not achieve convergence on the query, "Predictive algorithm is best?" Such a lack in conjunction is not very well known. In the analysis, the authors used the simulation and compared the machine learning with the regression model. The findings show that the investigation itself is false. A significant number of various prediction models have been proposed in the last 20+ years. Jorgensen: Experiments are not converging on the question of which model is better" Studies differ significantly in the effectiveness of AFAs. Jorgensen: "To put it simply, a poorly performed analysis will come to any conclusion" The lack of convergence is attributable to falsified test results, otherwise well conducted by small sample size, he writes. The technique is inefficient when used in a software model compare. Authors claim that the lack of convergence can, in no small degree, be due to low reliability in the measuring process (Kumar and Sureka, 2018).

Approximately one-third of the overall software cost is due to software bugs (Wagner, 2006). Using data analysis tools, software bugs are classified. This paper has examined different forms of bug classification techniques. Data mining methods may be used to retrieve from the data secret information. The KNN and NBM algorithms reach the maximum accuracy of 89% for gravity. The demand for early software delivery is rising day by day to ensure that software quality is essential. This paper finds a more practical approach (Clerc, 2007).

The latest challenge in RNAi technology is to develop the most successful siRNA based on optimal feature selection. An SVM classification has currently been used on the Huesken data set and has been provided with 77% filtering accuracy. There has recently been considerable interest in using evolutionary and natural computation methods for both the analysis of massive systems with different features. In this study, the SVM-based classification and PSO, ACO and GA were used for Huedken's siRNA data collection and two primary liquor and breast cancer benchmark gene datasets. The findings were presented with considerably high accuracy. Both groups of features are significant in the prediction of siRNA effectiveness (Clerc, 2007).

Software errors or software defects are dangerous and costly (Faizan *et al*., 2012). It is a lack of a software product that leads to its unexpected success. The precise prediction of faulty software modules will help guide test effort, minimize costs increase software quality and achieve a highly reliable system (Capers, 2012). Recent studies have shown that the odds of fault prediction detection models are better than those of software reviews. A panel at IEEE Metrics 2002 found that 60% of errors can be found in manual software reviews. The authors suggest combining methods and bagging to boost the efficiency of the prediction of the software defect. For large-scale software systems, an accurate defect prediction model is required. The outcome can be used as an essential indicator by the software developer and the software process can be managed. Application for software defect prediction was made to different classification algorithms, including logistic regression, decision trees, neural networks and naive bays (Wagner, 2008). However, there may be no substantial difference in output and no specific classifiers that perform best for all datasets (Menzies *et al*., 2006; Bean, 2008).

The management and repair of the information technology network relate primarily to the timely identification, location and treatment of any network faults. The APPSO algorithm is proposed based on a variation of the Apriori (AP) algorithm and the Particle Swarm Optimization (PSO) algorithm with the support and confidence coefficient assessment method. Professional network operation management's central management function would make it very difficult to show a marked rise incomplete report (Hirmanpour and Schofield, 2003). As per the report, the network management system has been built to update the integrated network management system. APPSO algorithm improves both the mining performance as well as the algorithm concept (Myrtveit *et al*., 2005; Nair *et al*., 2012).

Software Error Prediction with DM and PSO

Particle Optimization (PSO) is a population-based stochastic method that embraces optimization difficulties. The particle undertaking is predisposed to its local best-recognized position and the best-recognized exploration-space circumstances that are restructured as special situations arise from other components (Wahono *et al*., 2014). The proposed software fault prediction method is based on data mining techniques and Particle Swarm Optimization (PSO) (Jiang *et al*., 2011). The PSO algorithm chooses powerful program features for the fault prediction process to minimize processing time (Dhanalaxmi *et al*., 2015). The Remarkable machine topographies are classified as contestants for imperfection extrapolation using data mining practices, such as Association Law Mining, Decision Tree and Naïve Bayes Classification.

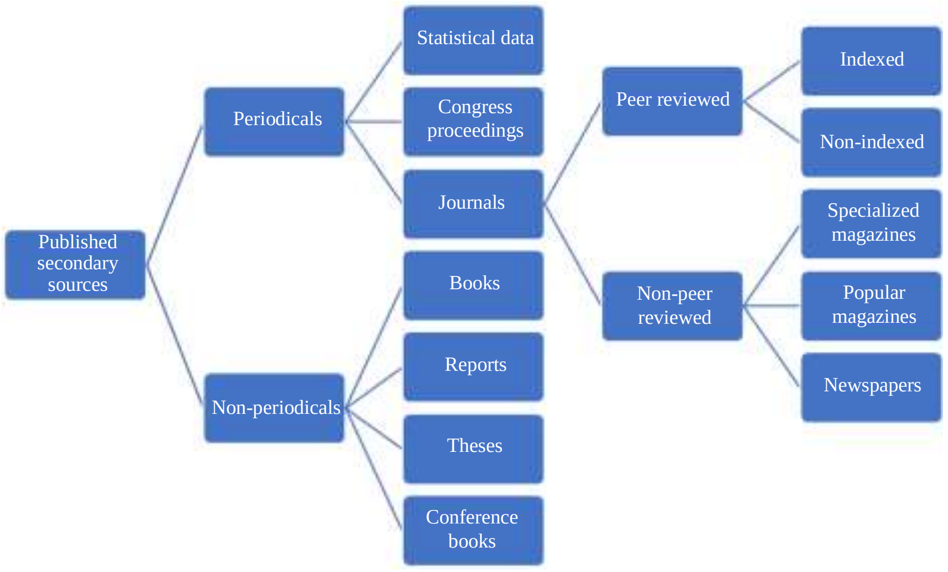

Fig. 2: Flow chart on how information is collected

Figure 2 shows how information was collected during the research. Secondary sources were used, including periodicals and non-periodicals. Periodicals included statistical data, congress proceedings, and journals. Non-periodicsls included books, reports, theses, and conference books. Journals were further categorized as either peer reviewed or non-peer reviewed journals. Peer reviewed journals were categorized into indexed and non-indexed. On the other hand, non-peer reviewed journals were categorized in specialized magazines, popular magazines, and newspapers. Using these sources helped gather information that could guide a comprehensive, valid, and reliable conclusion.

PSO Algorithm

The PSO algorithm is given below:

Step:1 For each particle, the fitness value is evaluated
Step:2 If the fitness value is higher than the best fitness value (pbest) in the record
Step:3 Set the current value to the new pbest
Step:4 End
Step:5 chooses a particle with the highest fitness value of all particles as the best partition.
Step:6 For each of the particles
Step:7 Assess particle velocity via velocity update equation:

$$V_{k+1} = W * V_k + C_1 * r_1 * \left(p_k - x_k\right) + C_2 * r_2 * \left(g_k - X_k\right) \quad (1)$$

Step:8 Update particle position via position update equation:

$$x_{k+1} = x_k + v_{k+1} \quad (2)$$

Step:9 End

While given condition is not satisfied, each particle attempts to adjust its current location and velocity depending on its current position and pbest and the distance between its current position and gbest. A particle has discovered its superior fitness value by choosing a smaller amount of inertial weight and is used in this proposed method to find the estimated position of the ideal solution within a wide range easily. A random number(r) is utilized to locate the suitable inertia weight and also avoid local optimum and early convergence is deferred by way of smaller amount of inertial weight. Considering a maximization problem, the inertial factors of the particles are modified according to Equation 3:

$$w_m = \frac{r}{pr}\left|\frac{f_{cp} - f_{opc}}{f_{opc}}\right|, \ if \ \ w_m > w_0 \ \text{ then } \ w \\ = w_m \,(or)\ if \ \ w_0 > w_m \ \text{ then } \ w = w_0 \quad (3)$$

where, r-random number, pr-Parameter, f_{cp}-fitness of current particle, f_{opc}-optimal particle currently. The parameter values such as number of particles p = 25, random numbers r_1 = 1, r_2 = 1 and number of iterations = 30 is used for fitness function training in PSO schemes. Given the maximization problem, the inertial factors of the particles are modified in conjunction with Equations (1) and (2) as set out in this section. The fitness function f(x) for the PSO training proposed for the program fault prediction method is seen in the Equation (4).

Fitness Function:

$$f(x) = C + \frac{1}{n}\sum_{i=1}^{n} CA_i * PM_i \quad (4)$$

where, C-Constant, CA-Classification Accuracy, PM-Parameter.

Association Rule Mining with PSO

The Association Rule Mining algorithm and PSO is as follows:

Select Significant software features with help of Particle Swarm Optimization Algorithm.
Consider L1 = (frequent items)
Perform Cartesian Product
C_k = (Lk-1 × Lk-1) to generate candidates.
Perform removal method to eliminate a collection of k-1 size objects that are not frequent
Repeat transactions t throughout the Database
Phase 6. Step 6. Increase the list of all C_k candidates included in t Compute L_k = candidates in C_k with Min_sup
If the C_k value not end go to step 4.
If dissolution circumstance is not contented, then go to step 2.
End the process.

Decision Tree with PSO

The construction of the decision tree is arranged in separate phases. In each stage, an attribute is considered for generating new branches in the tree. After developing components, the selected feature and some training data set records are considered at each stage (Tu *et al*., 2007). This progression continues for several steps for the remaining data and attributes until the tree is fully established.

The Algorithm of the Decision Tree and the PSO is as follows:

Select important software features with help of Particle Swarm Optimization Algorithm.
Generate Decision Tree with samples and features.
Create leaf node with node label classification
Create root node with root test condition for choosing best branch for further process.

Add child node as a descent of root and label the edge

If conclusion circumstance is not gratified then go to step 3.

End the process.

Naive Bayes Classification with PSO

Naive Bayes classification system is a probabilistic classifier and it plays vital role in less amount of training data sets used. The Bayes Theorem is a system for making predictions:

$$P(h\mid d)=\left(P(d\mid h)*P(h)\right)/P(d)$$

Where:

$P(h\|d)$	=	The likelihood of hypotheses h based on data d. It is referred to as the posterior likelihood
$P(d\|h)$	=	The likelihood of d data provided that the h hypothesis was valid
$P(h)$	=	The likelihood that the hypothesis h is valid (regardless of the data). This is referred to as the previous likelihood of h
$P(d)$	=	The data likelihood (regardless of the hypothesis)

In fact, a probability is not necessary to predict the most likely class for a new data instance. The numerator is needed and the class that gives the largest response, which will be the predicted output:

$$MAP(h)=\max\left(P(d\mid h)*P(h)\right)$$

Measures of Performance

The following measures are used to evaluate the performance of software defect prediction models. In the occasion of software short coming calculation models, there are four conceivable consequences for an individual afterwards extrapolation is made about whether the thing is imperfect or uncontaminated. The results are as follows:

(i). The Defective individual shall be listed as defective (True Positive, TP)
(ii). A faulty entity is listed as a clean entity (False Negative, FN)
(iii). Clean entity is graded as clean entity (True Negative, TN)
(iv). A clean individual shall be listed as defective (False Positive, FP)

Based on these results, the measures for computing the exactness of a software defect pre-diction model are defined. The most popular measure F-score is used for evaluating the performance of a defect prediction model in this research covering the software defect prediction models. The F-score is definite as the harmonic mean of exactness and remembrance. Higher F-score is an indication of a better model.

Results and Analysis

The proposed software flaw prediction system is devised by means of Data Mining techniques with Particle Swarm Optimization. The Data Mining algorithms such as Association Rule Mining, Decision Tree and Naive Bayes Classifier are incorporated with Particle Swarm Optimization technique individually. The Particle Swarm Optimization algorithm is utilized to select important software metrics for processing (Veysel and Kevin, 2011). The proposed Optimized Software Flaw Prediction System provides better solutions to the software quality issues. The proposed software flaw prediction system is offered in section 3 have been experimented with 2 different NASA software projects namely JM1 and PC1. The experiments with the proposed software flaw prediction system are performed repeatedly on the above-mentioned different datasets and the results are presented in section 5.3. In order to prove the effectiveness of the recommended system two NASA data sets namely JM1 and PC1 are employed and software flaws are detected through the performance measures such as Precision, Recall, F-Measure, Accuracy, Probability of False Alarm, Specificity, G-Measure and Error rate offered in section 4.

Table 1 shows the performance of SFP with DM and PSO. Thr performance of the algorithms are shown in terms of precision, accuracy, recall ability, and error rate. Table 1 represents, JM1, the first dataset used by NASA. Data in Table 2 shows performance of the algorithms using the second dataset, i.e., PC1.

Figures 3 and 4 show the performance of the algorithms using JM1 and PC1 datasets used by NASA. Graphical representation of the data allows a better and easy understanding.

The important measures such as Accuracy, F-measure and Error Rate are of different proposed software flaw prediction systems are considered for comparison to prove the effectiveness of software flaw prediction systems. The comparison of Accuracy, F-measure and Error Rate values on JM1 and PC1 datasets are offered in Table 3.

Table 1: Performance of SFP with DM and PSO on Data set JM1

Algorithm PSO-JM1	True Positive (TP)	False Negative (FN)	True Negative (TN)	False Positive (FP)	Precision (%)	Recall (%)	Accuracy (%)	F-Measure (%)	Probability of False Alarm (%)	Specificity (%)	G-Measure (%)	Error rate (%)
Association	704	873	5218	283	71.37	44.33	83.56	54.69	5.13	94.87	60.43	16.44
Decision Tree	686	900	5163	338	66.99	43.25	82.53	52.56	6.14	93.86	59.21	17.47
Naives Bayes	668	918	5047	454	59.54	42.12	80.64	49.34	8.25	91.75	57.74	19.36

Table 2: Performance of SFP with DM and PSO on data set

Algorithm PSO-PC1	True Positive (TP)	False Negative (FN)	True Negative (TN)	False Positive (FP)	Precision (%)	Recall (%)	Accuracy (%)	F-Measure (%)	Probability of False Alarm (%)	Specificity (%)	G-Measure (%)	Error rate (%)
Association	139	99	816	53	72.4	58.4	86.27	64.65	6.1	93.9	72.01	13.73
Decision Tree	141	97	804	65	68.45	59.24	85.37	63.51	7.48	92.52	72.23	14.63
Naives Bayes	123	115	817	52	70.29	51.68	84.91	59.57	5.98	94.02	66.7	15.09

Table 3: Comparison of accuracy, F-measure and error rate values of SFP with DM and PSO on datasets JM1 and PC1

Proposed software flaw prediction system	Accuracy (%)		F-Measure (%)		Error rate (%)	
	Dataset JM1	Dataset PC1	Dataset JM1	Dataset PC1	Dataset JM1	Dataset PC1
Association rule mining with PSO	83.56	86.27	54.69	64.65	16.44	13.73
Decision tree with PSO	82.53	85.37	52.56	63.51	17.47	14.63
Naive Bayes classification with PSO	80.64	84.91	49.34	59.57	19.36	15.09

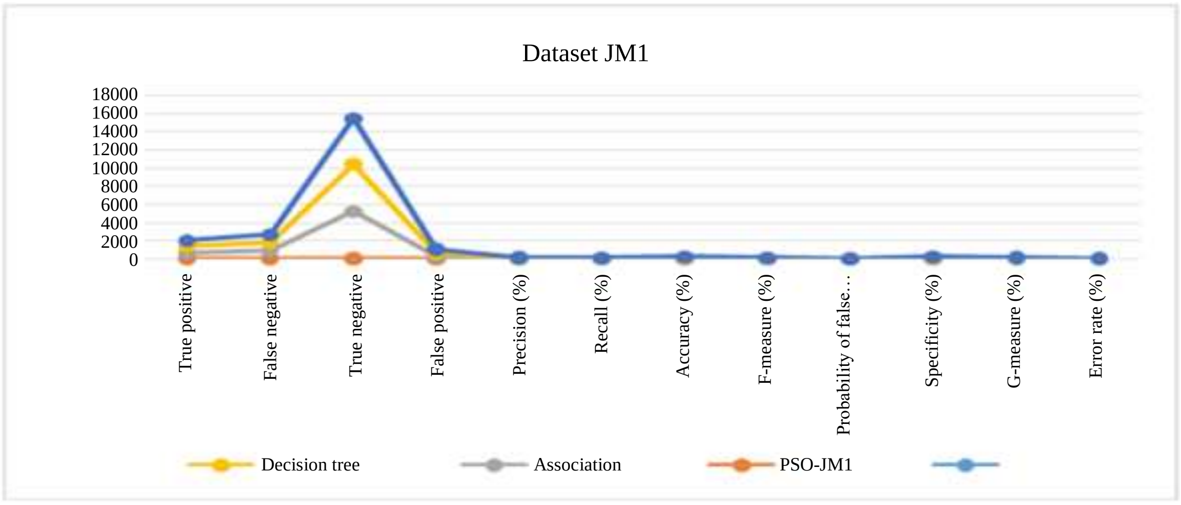

Fig. 3: Performance of SFP with DM and PSO on Data set JM1

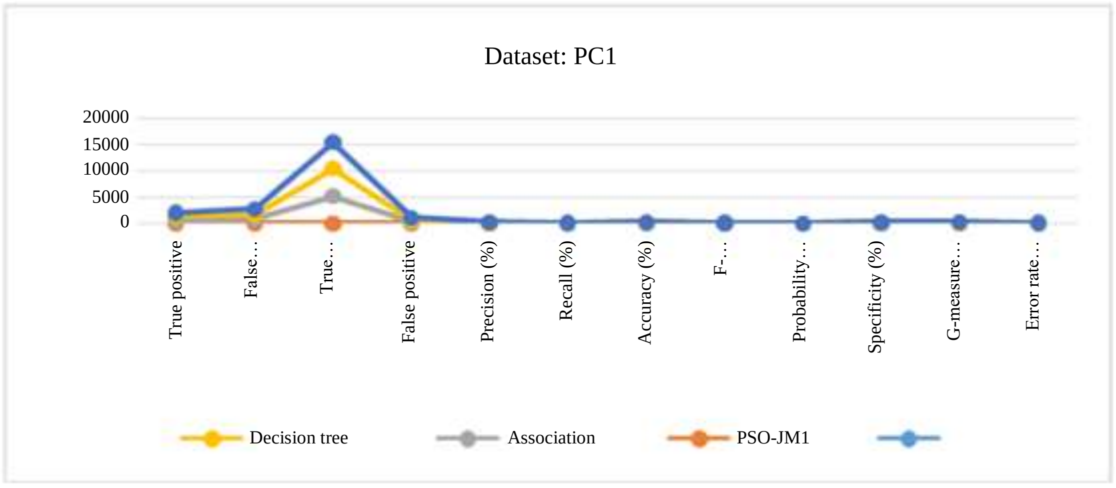

Fig. 4: Plots on performance of SFP with DM and PSO on data set PC1

The association rule mining with PSO for software flaw prediction system yields higher value of accuracy is 86.27% with help of PC1dataset. The lowest value of accuracy is 80.64% achieved by Naive Bayes Classification with PSO on dataset JM1. In case of F-Measure, the association rule mining with PSO is producing 64.65% as the higher value with dataset PC1 and the lower value 49.34% is given by Naïve Bayes classification with PSO on dataset JM1. With respect to error rate, the higher value 19.36% is presented by Naive Bayes Classification with PSO on dataset JM1 and minimum value 13.73% is attained by association rule mining with PSO on dataset PC1.

Limitations and Future Research Areas

Limitations during the research were the use of secondary resources. Using secondary sources for the manuscript were a limitation because the information could be faulty and it was not gathered first-hand. The sample in the research was small, limiting the scope of the research. These results could be bias. Lastly, the paper addressed system error estimation in software development and fails to address error estimation in other industrial and business errors.

Future Directions are as Follows

In the future, optimized data mining systems can be improved by the use of different platforms and particularly by improving data mining using PSO algorithms. It is necessary to develop algorithms that can identify faults in advance, which will minimize costs and promote the quality of developed software systems. Future optimized data mining systems will improve the relationship between quality metrics and the potential defective modules, which will lead to improved performances, productivity and lower operation costs (Azeem and Usmani, 2011).

Conclusion

The proposed software flaw prediction system is deliberated through Data Mining techniques with Particle Swarm Optimization algorithm has been verified and compared the results. This proposed system is very useful to identify the relationships between the quality metrics and the potential defective modules. The optimized data mining systems have pragmatic perfect prediction of these defective modules. This optimized software flaw prediction system is better than normal software error prediction methods.

The aim of the paper was to show how static model of data mining is used to extract defects and the PSO algorithm. Another aim of the research was to develop an optimized software flaw prophecy system on data mining techniques namely Association Rule mining, Decision Tree, Naive Bayes and Classification integrated with Particle Swarm Optimization technique.

The current research contributes to the existing knowledge on identifying the contributions of Particle Swarm Optimization (PSO) in identification of errors during software development. Software project managers have relied on different techniques to identify faults in the process. Identification of these faults has enabled software project managers make optimal use of people, costs and time to improve quality assurance. Also, these techniques have been used to plan production process improvements. The research will provide newer techniques to software developers and managers, which, in turn will enable them develop quality software systems. This research will add to the current knowledge on how the process of software development can be optimized, ensuring all stages of the production process are in check. Ensuring that the implemented policies and standards are adhered to will improve the quality of developed systems.

The proposed software flaw prediction system is deliberated through Data Mining techniques with Particle Swarm Optimization algorithm has been verified and compared the results. This proposed system is very useful to identify the relationships between the quality metrics and the potential defective modules. The optimized data mining systems have pragmatic perfect prediction of these defective modules. This optimized software flaw prediction system is better than normal software error prediction methods.

Limitations during the research were the use of secondary resources. Using secondary sources for the manuscript were a limitation because the information could be faulty and it was not gathered first-hand. The sample in the research was small, limiting the scope of the research. These results could be bias. Lastly, the paper addressed system error estimation in software development and fails to address error estimation in other industrial and business errors.

In the future, optimized data mining systems can be improved by the use of different platforms and particularly by improving data mining using PSO algorithms. It is necessary to develop algorithms that can identify faults in advance, which will minimize costs and promote the quality of developed software systems. Future optimized data mining systems will improve the relationship between quality metrics and the potential defective modules, which will lead to improved performances, productivity and lower operation costs.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Agarwal, S., & Tomar, D. (2014). A feature selection based model for software defect prediction. International Journal of Advanced Science and Technology, 65, 39-58. https://doi.org/10.14257/ijast.2014.65.04

Azeem, N., & Usmani, S. (2011). Analysis of data mining based software defect prediction techniques. Global Journal of Computer Science and Technology, 11, 1-7. http://computerresearch.org/index.php/computer/article/view/806

Bandi, R. K., Vaishnavi, V. K., & Turk, D. E. (2003). Predicting maintenance performance using object-oriented design complexity metrics. IEEE transactions on Software Engineering, 29(1), 77-87. https://doi.org/10.1109/TSE.2003.1166590

Bean, E. (2008). Defect Prevention and Detection in Software for Automated test Equipment. IEEE Transactions on Instrumentation and Measurement, 11, 16-23. https://doi.org/10.1109/MIM.2008.4579267

Can, H., Jianchun, X., Ruide, Z., Juelong, L., Qiliang, Y., & Liqiang, X. (2013, May). A new model for software defect prediction using particle swarm optimization and support vector machine. In 2013 25th Chinese Control and Decision Conference (CCDC) (pp. 4106-4110). IEEE. https://doi.org/10.1109/CCDC.2013.6561670

Capers, J. (2012). Software Defect Origins and Removal Methods. Namcook Analytics LLC.

Clerc, M. (2007). Particle Swarm Optimization. Wiley. ISBN-10: 1905209045,

Dhanalaxmi, B., Naidu, G. A., & Anuradha, K. (2015). Adaptive PSO based association rule mining technique for software defect classification using ANN. Procedia Computer Science, 46, 432-442. https://doi.org/10.1016/j.procs.2015.02.041

Faizan, M., Khan, M. N. A., & Ulhaq, S. (2012). Contemporary trends in defect prevention: A survey report. International Journal of Modern Education and Computer Science, 4(3), 14-20. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.975.703&rep=rep1&type=pdf

Faizan, M., Ulhaq, S., & Khan, M. N. (2014). Defect prevention and process improvement methodology for outsourced software projects. Middle-East Journal of Scientific Research, 19(5), 674-682. https://doi.org/10.5829/idosi.mejsr.2014.19.5.13669

Ghaffarinasab, N., Zare Andaryan, A., & Ebadi Torkayesh, A. (2020). Robust single allocation p-hub median problem under hose and hybrid demand uncertainties: models and algorithms. International Journal of Management Science and Engineering Management, 15(3), 184-195. https://doi.org/10.1080/17509653.2019.1683479

Goli, A., Tirkolaee, E. B., Malmir, B., Bian, G. B., & Sangaiah, A. K. (2019). A multi-objective invasive weed optimization algorithm for robust aggregate production planning under uncertain seasonal demand. Computing, 101(6), 499-529. https://doi.org/10.1007/s00607-018-00692-2

Gray, D., Bowes, D., Davey, N., Sun, Y., & Christianson, B. (2010, July). Software defect prediction using static code metrics underestimates defect-proneness. In The 2010 International Joint Conference on Neural Networks (IJCNN) (pp. 1-7). IEEE. https://doi.org/10.1109/IJCNN.2010.5596650

Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques. Morgan Kaufmann. ISBN-10: 9780123814791.

Hirmanpour, I., & Schofield, J. (2003). Defect management through the personal software process. Crosstalk, The Journal of Defense Software Engineering, 16(9), 17-20.

Jalote, P., & Agrawal, N. (2005, December). Using defect analysis feedback for improving quality and productivity in iterative software development. In 2005 International Conference on Information and Communication Technology (pp. 703-713). IEEE. https://doi.org/10.1109/ITICT.2005.1609661

Jiang, Y., Li, M., & Zhou, Z. H. (2011). Software defect detection with ROCUS. Journal of Computer Science and Technology, 26(2), 328-342. https://doi.org/10.1007/s11390-011-9439-0

Khan, H. A. (2013). Establishing a defect management process model for software quality improvement. International Journal of Future Computer and Communication, 2(6), 585. https://doi.org/10.7763/IJFCC.2013.V2.232

Kumar, L., & Sureka, A. (2018, February). Feature selection techniques to counter class imbalance problem for aging related bug prediction: aging related bug prediction. In Proceedings of the 11th innovations in software engineering conference (pp. 1-11). https://doi.org/10.1145/3172871.3172872

Menzies, T., Greenwald, J., & Frank, A. (2006). Data mining static code attributes to learn defect predictors. IEEE transactions on software engineering, 33(1), 2-13. https://doi.org/10.1109/TSE.2007.256941

Miholca, D. L., Czibula, G., & Czibula, I. G. (2018). A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. Information Sciences, 441, 152-170. https://doi.org/10.1016/j.ins.2018.02.027

Myrtveit, I., Stensrud, E., & Shepperd, M. (2005). Reliability and validity in comparative studies of software prediction models. IEEE Transactions on Software Engineering, 31(5), 380-391. https://doi.org/10.1109/TSE.2005.58

Nair, T. G., & Selvarani, R. (2012). Defect proneness estimation and feedback approach for software design quality improvement. Information and software technology, 54(3), 274-285. https://doi.org/10.1016/j.infsof.2011.10.001

Nair, T. G., Suma, V., & Tiwari, P. K. (2012). Significance of depth of inspection and inspection performance metrics for consistent defect management in software industry. IET software, 6(6), 524-535. https://doi.org/10.1049/iet-sen.2011.0148

Puška, A., Stojanović, I., Maksimović, A., & Osmanović, N. (2020). Evaluation software of project management used measurement of alternatives and ranking according to compromise solution (MARCOS) method. Operational Research in Engineering Sciences: Theory and Applications, 3(1), 89-102. https://doi.org/10.31181/oresta2001089p

Sharafi, S. M. (2012). SHADD: A scenario-based approach to software architectural defects detection. Advances in Engineering Software, 45(1), 341-348. https://doi.org/10.1016/j.advengsoft.2011.10.012

Sharma, A., Hemrajani, N., Shiwani, S., & Dave, R. (2012). Defect prevention technique in test case of software process for quality improvement. International Journal of Computer Technology and Application, 3(1), 56-61.

Suma, V., & Gopalakrishnan Nair, T. R. (2008). An Effective Defect Prevention Approach in Software Process for Achieving Better Quality Levels. World Academy of Science, Engineering and Technology, 42, 258-262. https://arxiv.org/pdf/1001.3552.pdf

Tirkolaee, E. B., Alinaghian, M., Hosseinabadi, A. A. R., Sasi, M. B., & Sangaiah, A. K. (2019). An improved ant colony optimization for the multi-trip Capacitated Arc Routing Problem. Computers & Electrical Engineering, 77, 457-470. https://doi.org/10.1016/j.compeleceng.2018.01.040

Trivedi, P., & Pachori, S. (2010). Modelling and analysing of software defect prevention using ODC. International Journal of Advanced Computer Science and Applications, 1(3), 75-77. https://doi.org/10.14569/IJACSA.2010.010311

Tu, C. J., Chuang, L., & Yang, J. C. (2007). Feature Selection using PSO-SVM. IAENG International Journal of Computer Science, 33, 1-18. http://www.iaeng.org/IJCS/issues_v33/issue_1/IJCS_33_1_18.pdf

Veysel. G., & Kevin, M. P. (2011) Swarm stability and optimization. Springer Science & Business Media.

Wagner, S. (2006, July). A model and sensitivity analysis of the quality economics of defect-detection techniques. In Proceedings of the 2006 international symposium on Software testing and analysis (pp. 73-84). https://doi.org/10.1145/1146238.1146247

Wagner, S. (2008, July). Defect classification and defect types revisited. In Proceedings of the 2008 workshop on Defects in large software systems (pp. 39-40). https://doi.org/10.1145/1390817.1390829

Wahono, R. S., Suryana, N., & Ahmad, S. (2014). Metaheuristic optimization based feature selection for software defect prediction. Journal of Software, 9(5), 1324-1333. https://doi.org/10.4304/jsw.9.5.1324-1333

Wang, Y., Li, G., Xu, Y., & Hu, J. (2014). An algorithm for mining of association rules for the information communication network alarms based on swarm intelligence. Mathematical Problems in Engineering, 2014. https://doi.org/10.1155/2014/894205

Wang, Z. X., & Li, Q. (2019). Modelling the nonlinear relationship between CO2 emissions and economic growth using a PSO algorithm-based grey Verhulst model. Journal of Cleaner Production, 207, 214-224. https://doi.org/10.1016/j.jclepro.2018.10.010