International Journal of Software Engineering and Its Applications
Vol. 10, No. 12 (2016), pp. 307-318
http://dx.doi.org/10.14257/ijseia.2016.10.12.27

Software Module Fault Prediction using Convolutional Neural Network with Feature Selection

Rupali Sharma* and Parveen Kakkar

Department of Computer Science and Engineering, DAV Institute of Engineering and Technology, Jalandhar, Punjab 144008, India
Department of Computer Science and Engineering, DAV Institute of Engineering and Technology, Jalandhar, Punjab 144008, India
**rupalis910@gmail.com, parveendaviet@gmail.com*

Abstract

Software plays a significant role in technological and economic development due to its utmost importance in day to day activities. A sequence of rigorous activities under certain constraints is followed to come up with reliable software. Various measures are taken during the process of software development to ensure high quality software. One such method is software module fault prediction for quality assurance to discover defects in the software prior to testing. It aids in predicting the software module faults earlier in the development of the software which predicts fault prone modules so that these can be given special attention to avoid any future risk which eventually curbs the testing along with maintenance cost and effort. The literature survey uncovers many findings that had never been focused like dimensionality reduction and feature selection based on individual feature importance which leads to increase in time complexity and chances of false information. This paper addresses these issues and proposes a supervised machine learning based software module fault prediction technique by implementing Convolutional Neural Network (CNN) as classifier model. Feature selection methods used are InfoGain and Correlation. The results obtained are compared with the existing method HySOM (SOM Clustering with Artificial Neural Network Classification) by considering three different feature sets (Fifteen features, Eighteen features and Twenty one features) of PC1 dataset from NASA. The comparative analysis is performed on the basis of accuracy, precision, recall and F1-measure. The results clearly show better performance of the proposed CNN based technique than HySOM. This paper will contribute towards improvement of quality assurance models utilized for software fault prediction by automating this process using machine learning which enhances True Positive Rate and reduces the detection error. This in turn will help project managers, testers and developers to locate and keep track of fault prone modules so that final software is more accurate, consistent and reliable without consuming much of the testing and maintenance resources.

Keywords: *Software Faults; Software Fault Prediction; Feature Selection; Convolutional Neural Network*

1. Introduction

In this modern era, the importance of software can't be neglected as it is used directly or indirectly in many aspects of our day to day activities. Be it an individual or an organization, a good quality software is needed for many important tasks. However, software development is not a single day process. A sequence of rigorous activities under

* Corresponding Author

ISSN: 1738-9984 IJSEIA

certain constraints is followed to come up with reliable software. In order to come up with high-quality software, measures to be taken need to be of utmost quality as well. This indirectly relies on the way how that software is designed, developed and maintained. Major concern remains to develop software with regards to its quality and time-budget constraints. Majority activities are performed directly or indirectly by humans so human errors can't be avoided altogether. To compensate for these overlooked mistakes, software testing and quality assurance activities are conducted to discover defects in the software prior to its delivery to end-customer. Software module defect prediction is one amongst the many quality models that aids to predict the software module faults earlier in the development of the software. By using this prediction model, the project managers and testers can easily classify any software module as faulty or non-faulty which in turn grabs their attention towards these faulty modules so that any later unforeseen serious risk can be avoided. This will eventually curb the testing and maintenance cost. Thus, software developed by following high-quality software module fault prediction techniques is likely to be more accurate, consistent and reliable without consuming much of the testing resources.

Software fault prediction is a process of identifying various fault prone components of the software early in the software development process, before testing, so that these trouble spots can be focused upon to avoid any later serious consequences. Figure1 explains a machine learning based software fault prediction process. The complete process can be separated into two sets of activities namely Training and Prediction. Training begins by extracting the inputs from available archive data sources. Next, the process of labelling generates associated labels for these extracted input instances. Feature selection on the input data helps to refine the input instances based on general features for defect prediction. Once this is complete, the generated labels and the features of instances combine to produce a training set. This training set is supplied to the machine learning algorithm for the training of the prediction model. During prediction, a prediction model is generated which takes in the features selected from input data and the training set to predict the label for a software module which tells whether a software module is prone to be faulty or not.

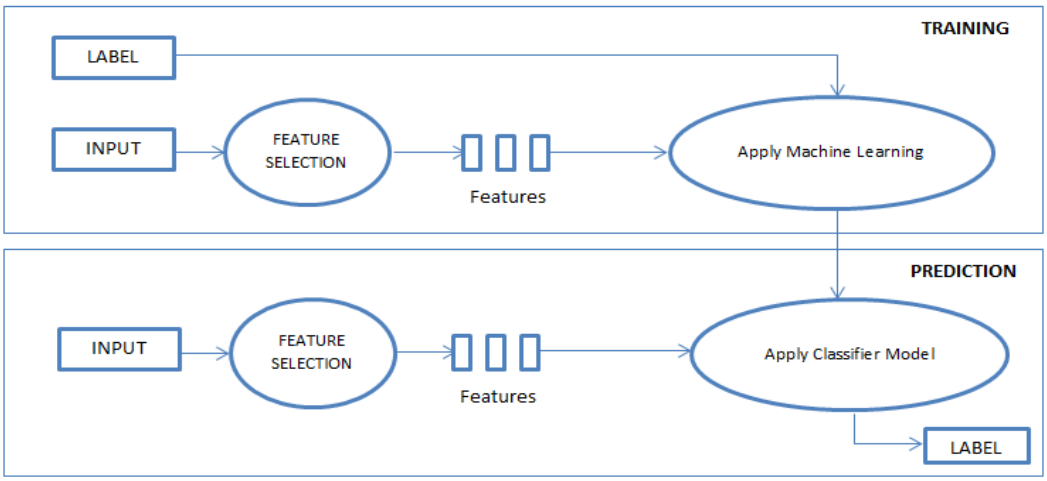

Figure 1. Software Fault Prediction Process

This paper proposes a software fault prediction technique by implementing a Convolutional Neural Network classifier model by utilizing InfoGain and Correlation based feature selection. The proposed methodology will automate the process of software fault prediction using machine learning thereby improving True Positive Rate and reducing error which will greatly improve the performance of classification of fault-prone

or non-fault-prone modules. The feature selection part will aim at removing irrelevant and redundant features for better prediction. This in turn will contribute a great deal in streamlining the software fault prediction process which will assist project managers, testers and other stakeholders to better assess and track those modules which may be troublesome in future. This will improve the software product quality as well as reduce cost, time, effort and resources for testing and maintenance of the software.

2. Literature Survey

Many approaches to software defect prediction have been proposed in the past. The relevant literature was collected from various sources like research papers (journals and conferences), books and websites. It was studied thoroughly to understand the trends in the area of software fault prediction. The major highlights are discussed below.

There are many factors which influence the prediction of defects as described in Table 1 [1]. Software complexity affects the defect prediction to a great deal which includes factors like requirement and design pages, programming language type and code size. Developer and tester knowledge also aids in the prediction process. Test process (number of test cases, test case coverage, test automation rate, test case execution productivity, effort spent in test case design and prior activities to system testing), errors (errors in requirement, design, code, test plan and test case), faults (faults in requirements, design, code, integration and test cases), defect information (like number, defect severity, type and validity) and software type like whether the software is component based or web based greatly impacts the prediction of defects.

Table 1. Possible Factors in Defect Prediction

Factors	Area
Number of requirement pages Number of design pages Type of programming language Size of Code	Software Complexity
Developer knowledge Tester knowledge	Knowledge
Test case coverage Total test cases Test automation rate Test case execution productivity Total effort in test case design Total effort in phases prior to system testing	Test Process
Requirement error Design error Code error Test plan error Test cases error	Errors
Severity of defect Type/category of defect Validity of defect Total defects logged	Defect
Requirement fault Design fault Code fault Integration fault Test cases fault	Fault
Component-based Web-based	Type of Software

[1]

[2] implemented two techniques namely, k-means and neural gas clustering to perform the task of fault prediction in the absence of quality labels. Then classification of faulty and non-faulty modules was performed by a human expert by labelling the clusters. Results were evaluated on the basis of False Positive Rate (FPR), False Negative Rate (FNR) and overall error rate. The results showed that labelling was done much easily and overall error rate was better in neural gas however k-means built clusters more rapidly. Overall, neural gas outperformed k-means clustering technique.

[3] focused on fault prediction approach when inadequate fault data was present and used Expectation Maximization (EM) approach to build a classifier model. Performance of the classification model was analyzed on the basis of Type-I error, Type-II error, and overall misclassification rate. This technique delivered satisfactory results while working with a limited fault data. However, both the above techniques are unautomated and dependent on an expert for a subset of operations.

[4] proposed software prediction technique that integrated clustering and metrics-thresholds methods so as to expel the accountability of human expert and thus resulted in a fully automated prediction model. Turkish data set was used for experimental studies. False Positive Rate (FPR), False Negative Rate (FNR) and overall error rate was used as an evaluation metrics. [5] also used the same dataset and evaluation metrics. The research focused on an automated technique for fault prediction in software modules by incorporating X-means clustering (an automatic generation of cluster numbers) and metrics threshold values (mean vector of each cluster was checked against the metrics threshold).

[6] studied the effectiveness of neural networks in identifying the faulty modules in Telecommunications Company (TELCO) data source in order to figure out these modules early in the development. Further, a comparison was made with a non-parametric discriminant model and results showed neural network outperformed the discriminant model.

[7] investigated various approaches for software fault prediction and proposed a technique based on extension of decision tree learning; that is, random forest in which numerous trees were generated from the corresponding data set and then voting was performed to classify the modules to be fault prone or not. The proposed technique was compared with logistic regression, discriminant analysis, C5.0 classifiers and several Waikato Environment for Knowledge Analysis (WEKA) classifiers. Prediction and classification accuracy of the proposed method was identified which came out to be much more significant as compared to others.

[8] performed the fault prediction techniques by implementing C4.5 classifier which is an extension of ID3 (Iterative Dichotomiser 3) and Naive Bayes algorithms on National Aeronautics and Space Administration (NASA) dataset and revealed the high performance of Naive Bayes.

[9] mainly emphasized on the verification-validation efforts by identifying fault-prone modules in the early project stages. Analysis of the results was done by implementing State Vector Machine (SVM), Extension of an Iterative Dichotomiser-3 (C4.5), Logistic Regression (LogReg) and Neural Network (NN) on NASA dataset. The results concluded that C4.5 was better among these.

[10] targeted SVM in the prediction of faulty modules in the software and, analyzed and compared it with other 8 prediction models (Logistic Regression, k-Nearest Neighbors, Naive Bayes, Random Forest, Decision Trees, Multilayer perceptrons, Radial

Basis Function Networks and Bayesian Belief Network) specific to four datasets from NASA (CM1, PC1, KC1 and KC3). Results showed better performance of SVM.

[11] proposed prediction approach using Quad-tree based k-means clustering and the results showed that clusters built by implementing this approach possessed higher gain values. Also the overall error rate was considered superior as compared to the other approaches like Naive Bayes, discriminant analysis, Catal et al. two stage and single stage methods.

[12] investigated various issues that may arise when training data for prediction model is not available. Generation of cluster numbers proved to be more challenging than others. Expectation-Maximization (EM) and X-means algorithms were was proposed to predict the faultiness of the module. X-means outperformed EM.

[13] carried out experimental analysis of statistical methods (linear regression and logistic regression) and machine learning methods (Artificial Neural Network and its variants) using Chidamber and Kemerer (CK) Metric Suite to identify faults in the modules. Comparison analysis was done for a case study of Apache Integration Framework (AIF) and results showed that Weighted Method per Class (WMC) metric is the key factor for fault classification.

[14] proposed Adaptive Neuro Fuzzy Inference System (ANFIS) as one of the applications that could predict the number of faults in the software module. Comparison was built against State Vector Machine (SVM) and Artificial Neural Network (ANN) that showed high predictive performance of ANFIS. All these were implemented on data brought by PROMISE (Predictor Models in Software Engineering) Repository.

[15] studied various techniques of fault prediction and then came to the result that the method of directly optimizing the model performance outperforms the existing methods. A new Learning To Rank (LTR) approach was proposed to directly enhance the model performance by ranking.

[16] investigated various feature selection approaches with noise tolerance and proposed FECS (FEature Clustering with Selection strategies) approach on the data collected from Eclipse and NASA. The proposed approach first performed feature clustering followed by the feature selection phase with three search strategies. Empirical studies showed the effectiveness of the proposed method.

This literature survey uncovers many findings that had never been focused. However, in the present literature, all the features have been used and only limited efforts have been made in the area of dimensionality reduction for software module defect prediction. Ignoring dimensionality reduction leads to more time complexity. Previous works have ignored separate importance of the individual features leading to either taking all the features or ignoring them altogether. This leads to increase in the false information at the time of classifier learning. Classifier models in the previous work have ignored overlapping of the features which increases the training error at time of learning. Also, present literature suggests that weightage of neural network relies on Self Organizing Map (SOM) clustering approach which depends on the static threshold values which does not reduce the detection error in the real world.

3. Proposed Methodology

Proposed methodology aims at performing the task of feature selection before the classifier model can classify the features as faulty or non-faulty. With this additional step of feature selection, only those features will be nominated that have more importance in predicting the faults. Thus, proposed methodology tries to reduce the time complexity and error rate for the fault prediction by reducing the feature set. Feature selection method

used are InfoGain and Correlation. Then, overlapping of features is reduced by utilizing the Convolutional Neural Network (CNN). Results of the proposed methodology (CNN) are compared with the existing method HySOM (SOM Clustering with Artificial Neural Network Classification) by considering three different feature sets (Fifteen features, Eighteen features and Twenty one features) of PC1 dataset [17] and analysis is made by considering accuracy, precision, recall and F1-measure parameters.

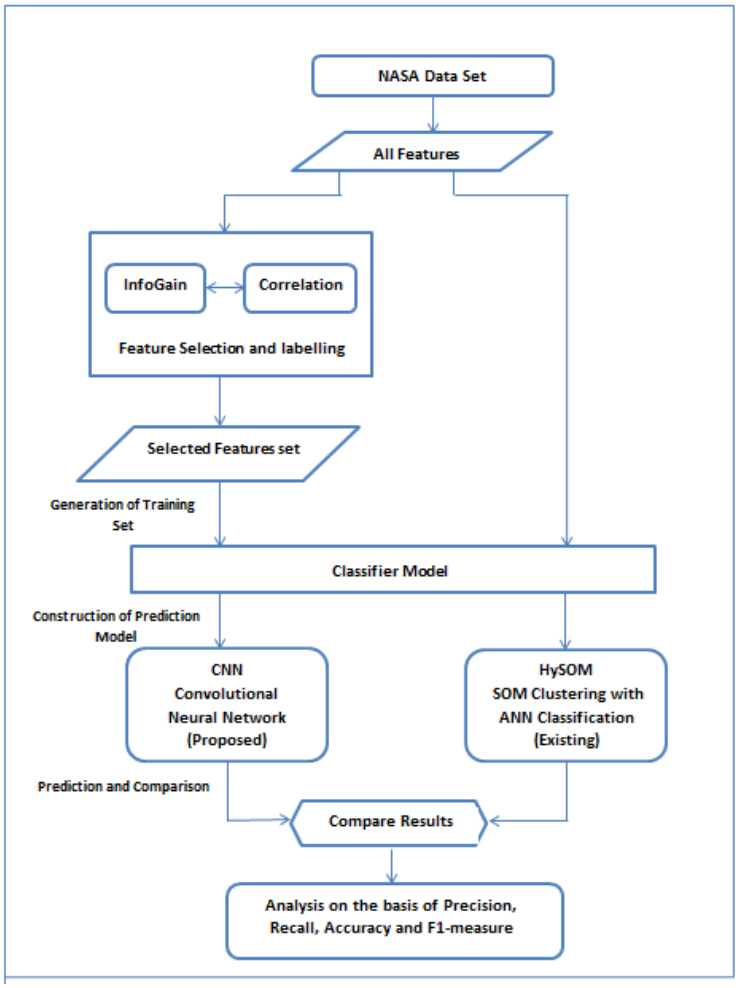

Figure 2. Proposed Methodology

Figure 2 shows the step by step flow of activities in the proposed methodology. The overall methodology is carried out in two phases: Training and Prediction. During training phase, firstly PC1 dataset from NASA containing faulty and non-faulty data is taken from PROMISE Software Engineering Repository. As it contains various redundant and irrelevant features, so InfoGain and Correlation based feature selection methods are performed to refine the input instances. Initially a base feature set containing some features (twenty one features in this case) is taken. InfoGain as well as Correlation is applied on the set of these twenty one features. Further implementation is applied on twenty one features; features appearing in the union of the results of InfoGain and Correlation (Eighteen features); and; features appearing the intersection of the results (Fifteen features). This allows to consider separate importance of the individual features. Then, these instances are labeled as faulty or non-faulty. Further, these features of instances along with the generated labels produce a training set that will be further supplied to the machine learning algorithm to train the prediction model. Once the

training is done, the prediction model is constructed for the fault prediction task. Features selected from input data along with the training set is provided as an input to the prediction model to predict the label for a software module that will classify whether the module is faulty or non-faulty.

Once this is complete, a comparative analysis is performed between the proposed methodology (CNN) and the existing method (HySOM) on the basis of accuracy, precision, recall and F1-measure which are calculated by using following formulae:

$$\text{Precision} = \frac{\sum \text{True Positive}}{\sum \text{True Positive} + \sum \text{False Positive}}$$

$$\text{Recall} = \frac{\sum \text{True Positive}}{\sum \text{True Positive} + \sum \text{False Negative}}$$

$$\text{Accuracy} = \frac{\sum \text{True Positive} + \sum \text{True Negative}}{\sum \text{True Positive} + \sum \text{False Positive} + \sum \text{True Negative} + \sum \text{False Negative}}$$

$$\text{F1 Measure} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

where, True Positive is correctly predicted positive, True negative is correctly predicted negative, False Positive is falsely predicted positive, and False Negative is falsely predicted negative.

4. Results and Analysis

Classifier model proposed above was implemented using MATLAB. The research was carried out on PC1 data set of NASA. The results were classified as True Positive, True Negative, False Positive and False Negative. The following set of values and resultant graphs were obtained by carrying out experimental analysis on the PC1 data set from NASA. Base feature set is taken to be twenty one. On the application of InfoGain and Correlation eighteen features appeared in either of the results (union) and fifteen features appeared in common in both the results (intersection).

Table 2. Comparative Analysis for PC1 Data Set of NASA

Prediction Model	Parameters	Fifteen Features	Eighteen Features	Twenty one Features
CNN	Accuracy	91.66	89.70	88.27
	Precision	89.48	79.50	88.57
	Recall	81.07	99.08	89.27
	F1-Measure	85.07	88.22	88.92
HySOM	Accuracy	89.27	85.75	84.24
	Precision	89.24	78.26	86.72
	Recall	80.62	97.24	83.24
	F1-Measure	84.71	86.72	84.94

Table 2 shows the values of Precision, Recall, and Accuracy obtained by implementing classifier models-CNN (Convolutional Neural Network) and HySOM (SOM Clustering with ANN Classification) for three different feature sets (Fifteen features, Eighteen features and Twenty One features) on PC1 data set.

Figures 3 to 6 compares values of precision, recall, accuracy and F1-measure of the proposed technique, CNN (color coded as blue) with HySOM (color coded as red) in the form of a bar chart for easy assessment.

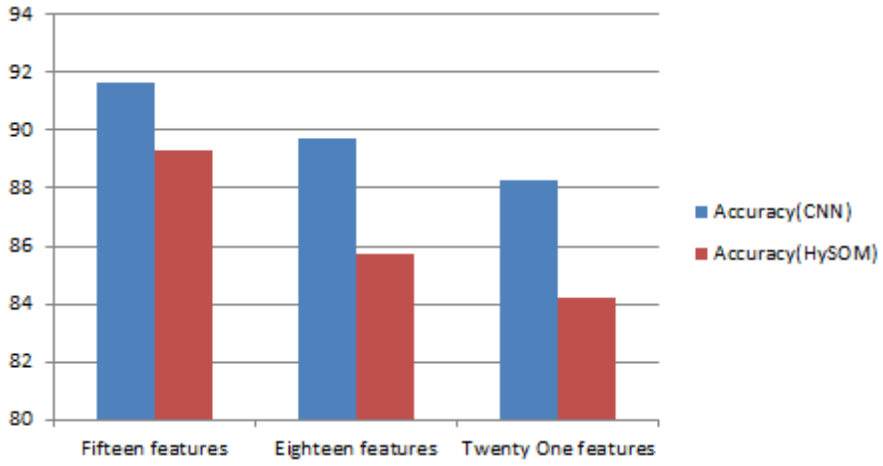

Figure 3. Analysis of Accuracy

Bar chart shown in Figure 3 represents Accuracy values for CNN and HySOM classifier models. For CNN, accuracy values are 91.66%, 89.70% and 88.27% and for HySOM accuracy values are 89.27%, 85.75% and 84.24% for Fifteen features, Eighteen features, Twenty one features set which clearly show an increase in Accuracy by 2.39% , 3.95% and 4.03% respectively for various sets of features.

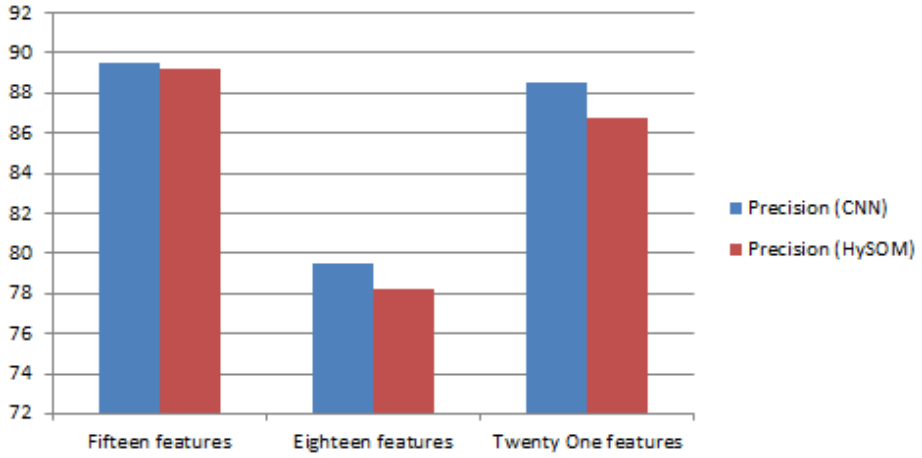

Figure 4. Analysis of Precision

Figure 4 represents Precision values for CNN and HySOM classifier models. For CNN precision values are 89.48%, 79.50% and 88.57% and for HySOM precision values are 89.24%, 78.26% and 86.72% for Fifteen features, Eighteen features, Twenty one features set which clearly show an increase in precision by 0.24%, 0.24% and 1.85% respectively.

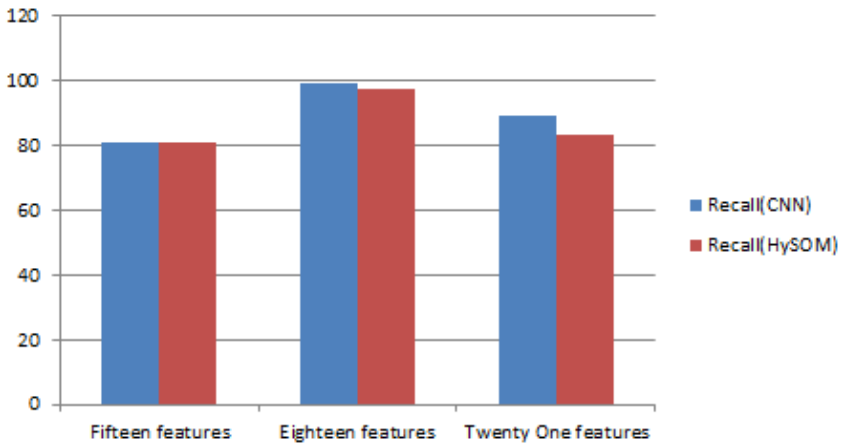

Figure 5. Analysis of Recall

Figure 5 represents Recall values for CNN and HySOM classifier models. For CNN recall values are 81.07%, 99.08% and 89.27% and for HySOM recall values are 80.62% , 97.24% and 83.24% for Fifteen features, Eighteen features, Twenty one features set which shows increase in recall by 0.45%, 1.84% and 6.03% respectively.

Figure 6 represents F1-Measure values for CNN and HySOM classifier models. For CNN F1-measure values are 85.07% , 88.22% and 88.92% and for HySOM, F1-measure values are 84.71%, 86.72% and 84.94% for Fifteen features, Eighteen features, Twenty one features set which clearly show an improvement of 0.36%, 1.50% and 3.98% respectively. The results clearly show that the proposed CNN method is better than HySOM.

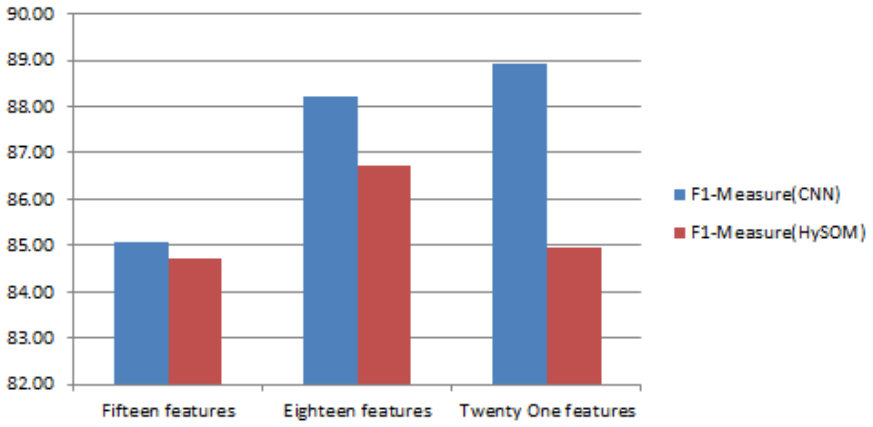

Figure 6. Analysis of F1-Measure

5. Conclusion

Software has become an indispensable part of our day to day activities and is very significant in technological as well as economic development. Due to its substantial importance and use, software development process needs to be continuously improved to provide the best-quality and most reliable software in minimum amount of time and resources. Quality assurance activities during software development help to avoid any faults in the software and to minimize the testing effort when software is developed. Software module defect prediction is one of the quality models that aids the prediction of the software module faults earlier in the development of the software so that rigorous attention can be given to the fault prone modules to avoid any serious future risk and to

reduce both the effort and cost involved in testing and maintenance. Many approaches to software defect prediction have been proposed in the past but ignore many important aspects in software fault prediction. This paper considers many of these aspects and proposes a machine learning based software fault prediction technique by incorporating feature selection using Infogain and Correlation. Further, the overlapping of features is reduced by implementing the Convolutional Neural Network (CNN) classifier model. Results of the proposed methodology (CNN) are compared with the existing method HySOM (SOM Clustering with Artificial Neural Network Classification) by considering three feature sets (Fifteen features, Eighteen features and Twenty One features) for PC1 dataset from NASA. Result analysis is performed on the parameters of accuracy, precision, recall and F1-measure. The results show CNN to be a clear winner against HySOM. The research conducted will contribute largely towards improvement of quality assurance activities for software module fault prediction by automating this process and improving TPR and reducing error using machine learning approach. The proposed technique will be helpful to the project managers, developers and testers to keep a close eye on fault-prone modules during development which can be problematic in near future. This will eventually curb the testing and maintenance cost along with contributing towards reliable and high-quality software. The future research may be focused towards reducing the effect of noise in features by kernel which would increase the information in feature set without losing the features and reduce classification error by graphical models like Hidden Markov Model (HMM), Conditional Random Fields (CRFs) *etc*.

References

[1] M. D. M. Suffian and S. Ibrahim, "A Prediction Model for System Testing Defects using Regression Analysis", arXiv preprint arXiv:1401.5830, **(2014)**.
[2] S. Zhong, T. M. Khoshgoftaar and N. Seliya, "Analyzing software measurement data with clustering techniques", IEEE Intelligent Systems, vol. 19, no. 2, **(2004)**, pp. 20-27.
[3] N. Seliya and T. M. Khoshgoftaar, "Software quality estimation with limited fault data: a semi-supervised learning perspective", Software Quality Journal, vol. 15, no. 3, **(2007)**, pp. 327-344.
[4] C. Catal, U. Sevim and B. Diri, "Clustering and metrics thresholds based software fault prediction of unlabeled program modules", Sixth International Conference on Information Technology: New Generations, IEEE, (**2009)**, pp. 199–204.
[5] C. Catal, U. Sevim and B. Diri, "Metrics-driven software quality prediction without prior fault data", Electronic Engineering and Computing Technology, Springer, (**2010)**, pp. 189–199.
[6] T. M. Khoshgoftaar, E. B. Allen, J. P. Hudepohl and S. J. Aud, "Application of neural networks to software quality modeling of a very large telecommunications system", IEEE Transactions on Neural Networks, vol. 8, no. 4, **(1997)**, pp. 902-909.
[7] L. Guo, Y. Ma, B. Cukic and H. Singh, "Robust prediction of fault-proneness by random forests", 15th International Symposium on Software Reliability Engineering, **(2004)**, pp. 417-428.
[8] T. Menzies, J. Greenwald and A. Frank, "Data mining static code attributes to learn defect predictors", IEEE Transactions on Software Engineering , vol. 33, no. 1, **(2007)**, pp. 2-13.
[9] E. Arisholm, L. C. Briand and M. Fuglerud, "Data mining techniques for building fault-proneness models in telecom java software", The 18th IEEE International Symposium on Software Reliability, ISSRE'07, IEEE, **(2007)**, pp. 215-224.
[10] K. O. Elish and M. O. Elish, "Predicting defect-prone software modules using support vector machines", Journal of Systems and Software, vol. 81, no. 5, **(2008)**, pp. 649-660.
[11] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm", IEEE Transactions on Knowledge and Data Engineering, vol. 24, **(2012)**, pp. 1146–1150.
[12] P. Mikyeong and H. Euyseok, "Software Fault Prediction Model using Clustering Algorithms Determining the Number of Clusters Automatically", International Journal of Software Engineering and Its Applications, vol. 8, no.7, **(2014)**, pp. 199-204.
[13] S. Yeresime, K. Lov and R. Santanu, "Statistical and Machine Learning Methods for Software Fault Prediction Using CK Metric Suite: A Comparative Analysis", Hindawi Publishing Corporation, ISRN Software Engineering Volume, **(2014)**.
[14] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction", Expert Systems with Applications, vol. 42, no. 4, **(2015)**, pp. 1872-1879.
[15] X. Yang, K. Tang and X. Yao, "A learning-to-rank approach to software defect prediction", IEEE Transactions on Reliability, vol. 64, no.1, **(2015)**, pp. 234-246.

[16] W. Liu, S. Liu, Q. Gu, X. Chen and D. Chen, "FECS: A Cluster Based Feature Selection Method for Software Fault Prediction with Noises", Computer Software and Applications Conference (COMPSAC), vol. 2, IEEE, **(2015)**.
[17] G. Boetticher, T. Menzies and T. Ostrand, "Software defect prediction, PROMISE Repository of Empirical Software Engineering Data", West Virginia University, Dept. of Computer Science, [Online], Available: http://promise.site.uottawa.ca/SERepository/datasets/pc1.arff, **(2004)**.

Authors

Rupali Sharma received her B. Tech degree in Computer Science and Engineering from CT Institute of Technology, Jalandhar, India in 2014 and is currently pursuing M. Tech degree in Computer Science and Engineering from DAV Institute of Engineering and Technology, Jalandhar, India. Her research area includes software engineering and machine learning.

Parveen Kakkar is an assistant Professor in DAV Institute of Engineering and Technology, Jalandhar, India. He has completed his B. Tech and M.Tech in Computer Science and is currently pursuing PHD. His main areas of interest is information security. He has 29 numbers of publications in good journals and conferences.