International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

ECMC Rule based Software Module Defect Prediction using Support Vector Balanced Data

Kovuru Vijaya Kumar , Ch GVN Prasad

Abstract*: Software module defect prediction (SDP) is one of the promising research area due to its wide range of application across multiple cross domain areas. Early detection of the defect module in a software reduces lot of effort put up in developing that software and it also reduces the cost associated in developing that software. Hence, in a quest to provide a better solution for SDP, we propose a novel rule based software module defect prediction using ECMC method in combination with support vector balanced data. It has been observed from our experiments that the results of this novel method are promising and satisfactory.

***Keywords*: Software module defect prediction, ECMC, rule-based, balancing data.**

I. INTRODUCTION

Software applicability has expanded its horizon from just being confined to computer systems to almost all electronic, mechanical, embedded and Artificial Intelligent(AI) systems. The cheaply availability of the hardware is one of the main reason for the usage expansion of the software to be increased. Due to this, lot of software companies have come into existence and are providing software solutions for various problems in the market. Providing reliable software has become one of the crucial challenge among these software solution providing industries. Here, pinches the research problem of providing accurate reliable predictive model that can predict the defective software as early as possible and works robustly well even for the very extreme class data. According to the literature, maximum software failures are attributed to very less modules of any particular software[16,17]. The dissatisfaction of the customer, increase in maintenance cost happens due to software failure which in turn occurs due to these types of less modules [18].

Hence to address this issue, we hereby propose a novel set of optimal rules which can be used as an early warning predictors in identifying the Software module defects more accurately.

So, the paper's main objective is to propose a set of rules which are user understandable and that can accurately help in identifying the defective modules of the given software metrics. For the experimental purpose PC1 NASA dataset is used.

The remaining part of the paper is organized in the following way. Section II gives the literature survey related to defect prediction in software. In Section III overview of the data description and data preparation is presented. The brief outline of techniques applied are discussed in Section IV, followed by the block diagram of the proposed model and it's experimental setup is described in Section V. Section VI describes results and discussions. Finally, Section VII gives the conclusion of the paper.

II. LITERATURE SURVEY

Lot of research has been done on the Software module defect prediction (SDP) in the literature. Few of the relevant works which has been done in the same domain are discussed in the following.

Yadav, D. K. et. al [2] had worked on size metric of the software using fuzzy logic and found that in the phase of requirement analysis three metrics were critically useful for predicting the fault in the software. In [3], Selvaraj, P. A. et. al had worked on KC1 dataset and applied support vector machine with various kernels for predicting the software defects. They obtained a recall value of 86 percent using polynomial and RBF kernel in hold-out method. Yadav, H. B and Yadav, D. K. had computed the software defect density indicators using fuzzy logic [4]. They computed the defect density at each and every phase of the software development life cycle and concluded that, the identification of the defects at any particular phase would help the developers fix the defects easily.

Agarwal Sonali, et.al. [5] had worked on the CM1 dataset of the software data prediction using the Twin support vector machine. They have achieved a good accuracy on that dataset. Fuzzy Support Vector Regression (FSVR) has been used by Yan, et.al. for predicting the software defects on two different datasets, namely, MIS and RSDIMU[1]. Al-Jamimi[6] with the help of experts defined and converted the software defect data into linguistic values and used Takagi-Sugeno fuzzy model to predict the software defects for various data sets.

Wang, et. al [14] used the Deep Belief Networks for automatically learning the semantics of software defect prediction. They concluded that automatic learning of semantics has improved with the average F1 score variation of 14.2%. Kamei, et. al [13] had done an extensive review of the research works done on SDP and elaborated the methods which had been used till that date and has elucidated the areas in which future work could be done.

In an attempt to propose a novel method for predicting the SDP, Marian, et.al. [7] has applied Fuzzy decision trees on the SDP and has obtained 73.5 percent of AUC.

Manuscript received on February 10, 2020.
Revised Manuscript received on February 20, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Kovuru Vijaya Kumar*, Computer Science and Engineering department, Rayalaseema University, Kurnool, Andhra Pradesh, India. Email: vijay657@yahoo.co.in
Ch GVN Prasad, Department of Computer Science & Engineering, Sri Indu College of Engg & Tech, Hyderabad. India
Email: prasadch204@gmail.com

He had worked on three metrics namely, WMC, CBO and RFC. In [8] Yadav, H. K. et.al tried to improve the defect prediction of software by considering the reliability metrics of software with the help of fuzzy inference systems.

Bennin, et. al [15], tried to address the problem of balancing the SDP data using a novel technique called as MAHAKIL that is an oversampling technique which is diversity based. Bisht, et. al[9], attempted to give a review of all the papers that have published on SDP Models using different data mining techniques. Naufal, et. al tried a combination of balancing methods and few selected complexity metrics to predict the SDP using fuzzy association rule mining[10]. He has obtained 91.63% of accuracy and 85.51% of sensitivity.

Jadhav, et. al [11], has done a survey study on the SDP in cross project context and proposed a new set of software metrics using which we can more accurately predict the software defects. In [12], Li, et.al has done a fusion of fuzzy and GA methods for predicting the SDP. So, as per the literature survey only Naufal, et. al, tried the mixed combination of balancing the data and applying the machine learning techniques. But, no one has till now explored the possibility of balancing and predicting user understandable rules. So, here we try to fill the gap with an attempt to provide a good combination of balancing technique and set of rules which can be used to predict the software defects.

III. DATA DESCRIPTION

One of the NASA's mission critical software projects data called PC1 that is openly available at Repository of PROMISE Software Engineering is used for experimental purpose. The description of the data is as follows.

Table- I: DESCRIPTION OF THE DATASET

Dataset Name	PC1 data
Total number of samples	10885
Number of features	21
Type of features	Numeric
Target classes	Two(1-non defect, 2-defect)
Names of features	5 code measure features, 3 McCabe metrics, base Halstead measures 4, derived Halstead measures are 8 and one branch-count.

Due to the large imbalance among the classes of the data which are having 93 percent of good modules and 7 percent of defect modules, we went for balancing of the data using support vector based balanced data. Otherwise, the Machine learning algorithm may give a biased decision as the dataset is dominated by the good modules.

A. Preprocessing of Data

The preprocessing of the data is done, firstly by segregating the test data into separate file and partitioning the remaining data into training data and validation data. 70 samples have been kept as testing data which contained equal number of class samples. The remaining data which has been divided into training and validation data has been taken in the ratio of 70:30, where 70 percent was used for training and 30 percent was used for validation.

B. Balancing of the Data

For balancing the data, we have used support vector machine. Where, we have used the training data as input to the support vector machine and extracted the support vectors. Thus, obtained support vectors are used for training two different machine learning algorithms, which were able to learn and predict the output very fast due to the less number and good training samples which were essentially the support vector balanced data.

IV. INTELLIGENT TECHNIQUES

All the experiments have been carried out using the Neucom tool.

Decision Tree: Decision tree is a tree based machine learning algorithm which contains the class labels at its leaf and the other important features are represented as an internal node. Essentially, these internal nodes represents the conditions or the ranges which the features of the dataset has to satisfy to reach the leaves. One single path across the branch from root to a single leaf represents the rule.

Evolving Cluster Method Classification: ECMC is a novel semi-supervised classification method proposed by Song, et. al [19] in 2001. This algorithm computes the clustering based on the maximum distances between the samples and computes in a single-pass.

V. BLOCK DIAGRAM OF THE MODEL

The structure or the architecture/block diagram of the proposed model can be viewed below.

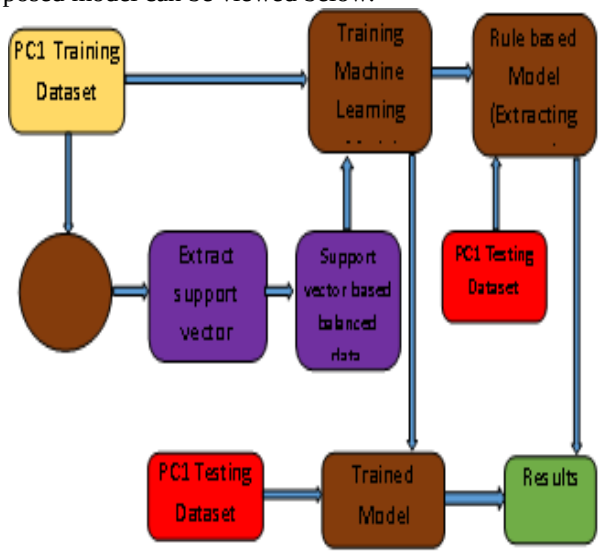

Fig. 1. Block diagram of the experimental process flow.

PC1 NASA dataset is used for carrying out the experiments.

A. First the training data is supplied as it is to the machine learning model to train. On thus trained model we have given the Test data and obtained the first set of results. We also have extracted the rules from these models and the Decision Tree has given 30 rules with accuracy of 77.02 percent and ECMC method has given 68 rules with accuracy of 88.5 percent and recall of 80 percent which can be seen in Fig. 3.

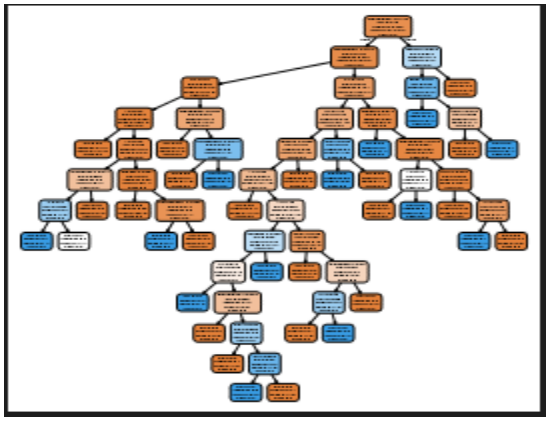

Fig. 2.Decision Tree rules

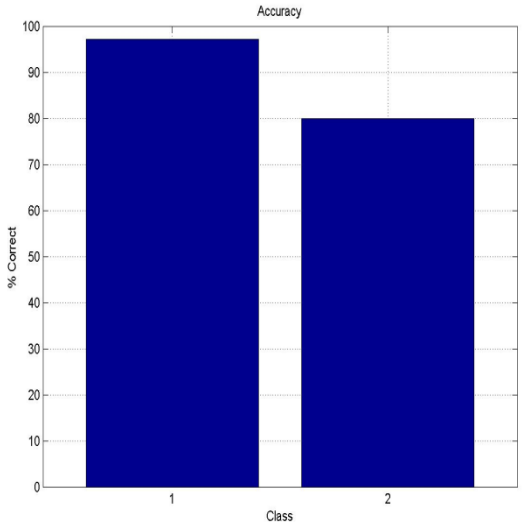

Fig. 3. Accuracy of the ECMC for training data.

B. Next in the case of balanced data, we have supplied the whole training data to the support vector machine and extracted the support vectors from that and used these support vector data to trian the two models we used. In this case, when we trained the decision tree and extracted rules, we obtained only two rules which were giving accuracy of only 28 percent.

Where as in the case of ECMC we obtained eleven rules which have yielded good results of 99 percent of accuracy and 95 percent of recall with respect to target class. The screenshot of the ECMC method, when used with the support vectorized data is shown in Fig. 4. The detail discussion of the results are dealt in the Results and discussion section.

The metrics used for measuring the performance of the machine learning algorithms are discussed in the next section.

C. Units/ Metrics used

The following units/metrics have been used to access the performance of the machine learning techniques used.

Recall(R) is used to identify the correct number of Truepositives(Tp).

$$\mathrm{Re}call(R) = \frac{(Tp)}{(Tp + Fn)} \qquad (1)$$

Where, Fn is Faslenegative.

In the same way, Precision(P) defined as Tp's quality can be shown as,

$$\mathrm{Pr}ecision(P) = \frac{(Tp)}{(Tp + Fp)} \qquad (2)$$

Where, Fp is Falsepositive.

Accuracy(Acc) is defined as total number of Truepositives(Tp) and Truenegatives(Tn) that are correctly predicted among all combination of ratios taken. It is shown as,

$$Accuracy(Acc) = \frac{(Tp + Tn)}{(Tp + Fp + Tn + Fn)} \qquad (3)$$

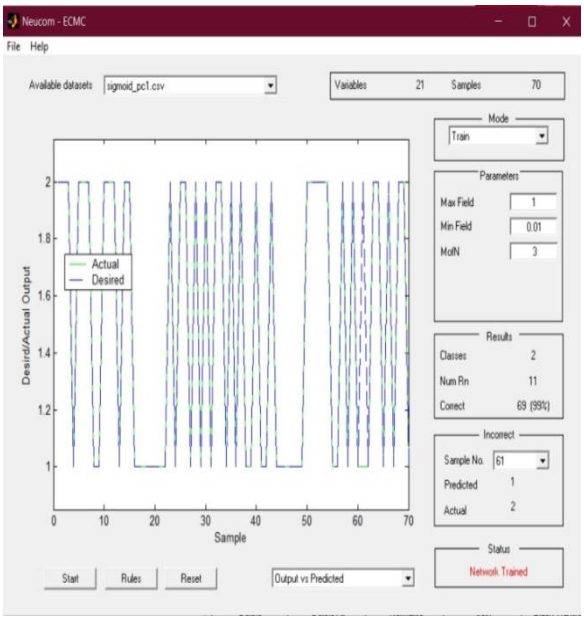

Fig. 4.Screenshot of the results of ECMC method used in Neucom.

VI. RESULT AND DISCUSSION

The dataset contains two classes namely the good modules and defect modules. The good modules are represented using the class 1 label and the defect modules are represented using the class 2 label. Since, the data is largely unbalanced the results of the models used vary drastically when compared between the models used with original data and models used with the support vector balanced data. Since, Class 2 is our target class here, our main concern would be to predict the target class more accurately. That means the recall must be high.

As described in previous section, when the models Decision tree and ECMC where trained using the original data they have given very poor results and the rules generated by both these algorithms were very huge. Hence, to overcome this issue, we have gone for support vector based balanced data which has yielded significantly good results in respect to rules. But, the rules of the Decision tree were very few. Only two rules which deteriorated its performance. Whereas in the case of ECMC, it has given 11 rules which have given a promising results of accuracy and recall. The accuracy of the ECME rules which are generated over the support vector balanced data is around 99 percent and the recall is 97.14 percent. The results can be seen in Fig. 6. The confusion matrix pertaining to this results can be seen in Fig. 5.

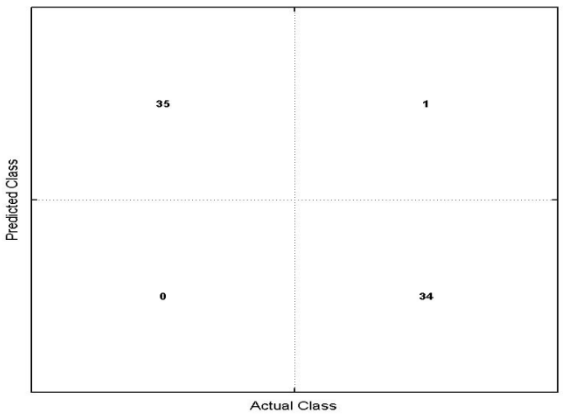

Fig. 5.Confusion Matrix of results of ECMC trained on support vector balanced data.

The 11 rules given by the ECMC are shown in the Table-II, which can be interpreted as "if antecedent then consequent". The third column shows how many of the total seventy samples have been classified based on that specific rule.

These rules are quite interpretable similar to the rules generated by the decision tree. But, the significant difference between these rules and the decision tree rules is that, the decision tree rules specify the rules with the help of features whereas the ECMC method first constructs the clusters with various centres and radius and classifies the samples to one of the rules based on the maximum distance method as described in paper [19].

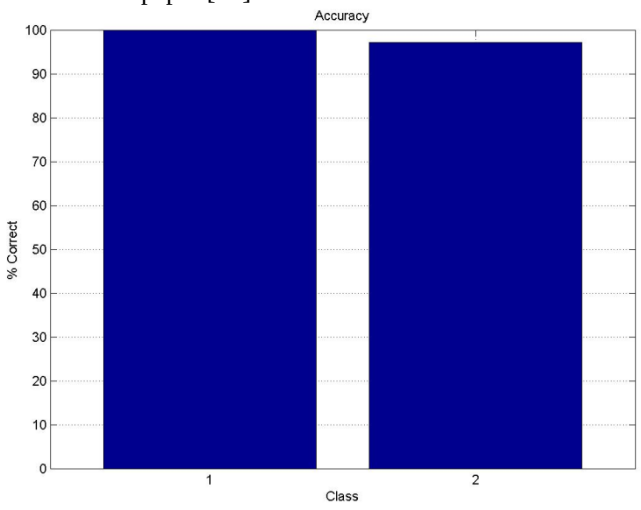

Fig. 6.Accuracy of the ECMC for support vector balanced data

It can be noted that the rules are clearly understandable by the user and serves as an early warning predicting variables. The main advantage of less rules is its readability/understandability, consumption of less computational cost and faster prediction.

Table- II: RULES EXTRACTED BY ECMC METHOD

Rules	Antecedent	Consequent	No. of Samples
Rule 1	If Centre is [0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.24, 0.06, 0.06, 0.05, 0.05, 0.05, 0.05, 0.05, 0.09, 0.07, 0.05, 0.05, 0.05, 0.05] and Radius is 0.40	Then Class is [1]	3
Rule 2	If Centre is [0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.73, 0.07, 0.07, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.10, 0.05, 0.05, 0.05, 0.05] and Radius is 0.31	Then Class is [1]	32
Rule 3	If Centre is [0.08, 0.06, 0.07, 0.07, 0.09, 0.07, 0.20, 0.20, 0.14, 0.05, 0.07, 0.05, 0.06, 0.06, 0.16, 0.27, 0.09, 0.09, 0.09, 0.06] and Radius is 0.85	Then Class is [2]	14
Rule 4	If Centre is [0.13, 0.09, 0.05, 0.11, 0.17, 0.13, 0.12, 0.30, 0.26, 0.08, 0.13, 0.08, 0.12, 0.07, 0.37, 0.38, 0.16, 0.17, 0.16, 0.09] and Radius is 0.98	Then Class is [2]	6
Rule 5	If Centre is [0.19, 0.16, 0.12, 0.13, 0.17, 0.13, 0.11, 0.40, 0.22, 0.09, 0.13, 0.09, 0.22, 0.13, 0.14, 0.47, 0.16, 0.16, 0.17, 0.16] and Radius is 0.92	Then Class is [2]	3
Rule 6	If Centre is [0.19, 0.14, 0.16, 0.20, 0.25, 0.20, 0.10, 0.43, 0.31, 0.15, 0.20, 0.15, 0.24, 0.13, 0.57, 0.57, 0.23, 0.26, 0.24, 0.15] and Radius is 0.81	Then Class is [2]	3
Rule 7	If Centre is [0.31, 0.26, 0.13, 0.37, 0.42, 0.36, 0.10, 0.39, 0.59, 0.23, 0.36, 0.23, 0.09, 0.14, 0.22, 0.53, 0.38, 0.43, 0.40, 0.26] and Radius is 0.97	Then Class is [2]	3
Rule 8	If Centre is [0.23, 0.24, 0.24, 0.32, 0.30, 0.24, 0.08, 0.66, 0.24, 0.25, 0.24, 0.25, 0.18, 0.19, 0.62, 0.66, 0.22, 0.30, 0.30, 0.20] and Radius is 0.89	Then Class is [2]	3
Rule 9	If Centre is [0.26, 0.27, 0.30, 0.18, 0.37, 0.30, 0.07, 0.86, 0.24, 0.38, 0.30, 0.38, 0.34, 0.27, 0.91, 0.65, 0.22, 0.36, 0.40, 0.28] and Radius is 0.66	Then Class is [2]	1
Rule 10	If Centre is [0.33, 0.29, 0.30, 0.25, 0.41, 0.35, 0.07, 0.77, 0.30, 0.41, 0.35, 0.41, 0.51, 0.22, 0.26, 0.76, 0.32, 0.39, 0.46, 0.28] and Radius is 0.00	Then Class is [2]	1
Rule 11	If Centre is [0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.07, 0.65, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.79, 0.95, 0.95, 0.95, 0.95, 0.95] and Radius is 0.00	Then Class is [2]	1

VII. CONCLUSION

For the PC1 dataset of Software module defect prediction, which has highly unbalanced data, the method which we proposed has given a promising result as per our experiments.

Hence, with the help of support vector based balanced data and ECMC rules the software developer would be able to predict and identify the defect modules in very less time and more accurately.

ACKNOWLEDGMENT

The authors would like to thank the PROMISE Software Engineering Repository for providing the PC1 software module defect prediction data set.

REFERENCES

1. Yan, Zhen, Xinyu Chen, and Ping Guo. "Software defect prediction using fuzzy support vector regression." *International symposium on neural networks*. Springer, Berlin, Heidelberg, 2010.
2. Yadav, Dilip Kumar, S. K. Chaturvedi, and Ravindra B. Misra. "Early software defects prediction using fuzzy logic." *International Journal of Performability Engineering* 8.4 (2012): 399-408.
3. Selvaraj, P. A., and P. Thangaraj. "Support vector machine for software defect prediction." *International Journal of Engineering & Technology Research* 1.2 (2013): 68-76.
4. Yadav, Harikesh Bahadur, and Dilip Kumar Yadav. "A multistage model for defect prediction of software development life cycle using fuzzy logic." *Proceedings of the Third International Conference on Soft Computing for Problem Solving*. Springer, New Delhi, 2014.
5. Agarwal, Sonali, and Divya Tomar. "Prediction of software defects using twin support vector machine." *2014 International Conference on Information Systems and Computer Networks (ISCON)*. IEEE, 2014.
6. Al-Jamimi, Hamdi A. "Toward comprehensible software defect prediction models using fuzzy logic." *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2016.
7. Marian, Zsuzsanna, et al. "A novel approach for software defect prediction using fuzzy decision trees." *2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE, 2016.
8. Yadav, Harikesh Bahadur, and Dilip Kumar Yadav. "Early software reliability analysis using reliability relevant software metrics." *International Journal of System Assurance Engineering and Management* 8.4 (2017): 2097-2108.
9. Bisht, Bharti, and Parul Gandhi. "Review Study on Software Defect Prediction Models premised upon Various Data Mining Approaches." *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2019.
10. Naufal, Mohammad Farid, and Selvia Ferdiana Kusuma. "Software defect detection based on selected complexity metrics using fuzzy association rule mining and defective module oversampling." *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, 2019.
11. Jadhav, Rohini, et al. "A Survey on Software Defect Prediction in Cross Project." *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2019.
12. Li, Kewen, et al. "Software defect prediction using fuzzy integral fusion based on GA-FM." *Wuhan University Journal of Natural Sciences* 19.5 (2014): 405-408.
13. Kamei, Yasutaka, and Emad Shihab. "Defect prediction: Accomplishments and future challenges." *2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER)*. Vol. 5. IEEE, 2016.
14. Wang, Song, Taiyue Liu, and Lin Tan. "Automatically learning semantic features for defect prediction." *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016.
15. Bennin, Kwabena Ebo, et al. "Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction." *IEEE Transactions on Software Engineering* 44.6 (2017): 534-550.
16. Fenton, Norman E., and Niclas Ohlsson. "Quantitative analysis of faults and failures in a complex software system." *IEEE Transactions on Software engineering* 26.8 (2000): 797-814.
17. Koru, A. Güneş, and Jeff Tian. "An empirical comparison and characterization of high defect and high complexity modules." *Journal of Systems and Software* 67.3 (2003): 153-163.
18. Koru, A. Gunes, and Hongfang Liu. "Building effective defect-prediction models in practice." *IEEE software* 22.6 (2005): 23-29.
19. Song, Qun, and Nikola Kasabov. "ECM-A novel on-line, evolving clustering method and its applications." *Proceedings of the fifth biannual conference on artificial neural networks and expert systems*. 2001.

AUTHORS PROFILE

Kovuru Vijaya Kumar received his B. Tech (Computer Science &Engineering) degree from Sri Venkateswara University, Tirupati and M.Tech (Computer Science &Technology) degree from GITAM College of Engineering, Andhra University. He worked as Assistant Professor at various engine ering colleges in Hyderabad. He has 7 years of teaching experience. He is currently research scholar at Department of Computer Science and Engineering, Rayalaseema University, Kurnool, Andhra Pradesh, India. His area of interest includes Data Warehousing & Data Mining, Artificial Intelligence, Compiler Design and Network Security.

Dr. Ch GVN Prasad, M.Tech, Ph.D(Experience - 23 years ; 12 years IT industry (8 years in National Informatics Centre, Govt. of India, as Scientist and Software Analyst in AT&T in US) and 11 years Teaching as Professor and HOD of CSE dept). He is currently working as Professor, in Department of Computer Science & Engineering in Sri Indu College of Engg & Tech. Guided many UG, PG and Phd projects as supervisor. Published several papers in international and national journals. Research areas include Data Mining, Image Processing, Neural Networks and Network Security.