

Software Fault Prediction using Fuzzy Clustering & Genetic Algorithm

Saurabh Bhattacharya
babu.saurabh@gmail.com

Dr.Sourabh Rungta
sourabh@rungta.ac.in

Naresh Kar
nareshkar@gmail.com

Abstract –Modern systems are primarily based on the software based systems. Software quality and reliability have become the main concern during the software development. It is very difficult to develop software without any fault. The fault-proneness of a software module is the probability that the module contains faults and a software fault is a defect that causes software failures in an executable project. Early detection of fault prone software components enables verification experts to concentrate their time and resources on the problem areas of the software systems under development. In this paper, performance comparison of a Software Fault Prediction System using Fuzzy c-means clustering approach and a hybrid technique (combination of Fuzzy c-means and Genetic Algorithm) has been performed with the real time data set named PC1, taken from NASA MDP software projects. The performance is recorded on the basis of accuracy, reliability, RMSE and MAE values.

Keywords – Accuracy, Fuzzy c-means, NASA MDP, MAE, Reliability, RMSE and Software Fault Prediction.

I. INTRODUCTION

A software system consists of various modules and any of these modules can contain the fault which can adversely affect the reliability of the system. But early prediction of faulty modules can help in producing fault free software. So, it is better to categorize the software module in faulty / non - faulty module just after completing the coding phase. After categorizing more efforts can be put on the faulty modules to produce reliable software.

A fault is a defect, an error in source code that causes failures when executed. A fault prone software module is the one containing more number of expected faults. Accurate prediction of fault prone modules enables the verification and validation activities focused on the critical software components. Clustering is defined as the classification of data or object into diverse groups. It can also be mentioned to as partitioning of a data set into diverse subsets. In hierarchical clustering the data are not partitioned into a particular cluster in a single step. But a series of partitions takes place, which may vary from a single cluster comprising all objects to n clusters each containing a single object.

A software fault is a defect that causes software failure in an executable product. For each

execution of the software program where the output is incorrect, we observe a failure. Software engineers distinguish software faults from software failures. Faults in software systems continue to be a major problem. Various systems are delivered to users with excessive faults. This is despite a huge amount of development effort going into fault reduction in terms of quality control and testing. It has long been recognized that seeking out fault-prone parts of the system and targeting those parts for increased quality control and testing is an effective approach to fault reduction.

An inadequate amount of valuable work in this area has been carried out previously. Regardless of this it is difficult to identify a reliable approach to identifying fault-prone software components. Using software complexity measures, the techniques build models, which categorize components as likely to contain faults or not.

In the last five decades data from various natural and social sources are stored in massive and complex databases for fast access of information and communication technologies. The clustered data holds various significant parameters to make it compatible in many range. For example, the code of biological information is stored in the sequence of DNA and RNA [1]. While the web documents are structured in the format of XML and HTML [2]. However it is nearly impossible to analyse the data by handwork considering speed and accuracy hence various data mining algorithms are designed in order to fetch data by pre-defined computational work.

K-means is referred to as a hard c-means clustering. It is the classical and the basic method [3]. Hard c-means clustering divides a set of objects in c-crisp clusters by a very simple procedure. In this method an object is assigned with one cluster exclusively. In contrast to this the fuzzy c-means clustering is an extension of k-means using the concepts of fuzzy sets. In c-means clustering every object is provided with degree of belonging to clusters.

The main objective of this paper is to design a Software Fault Prediction System using Fuzzy c-means clustering approach and a hybrid technique (combination of Fuzzy c-means and Genetic Algorithm). The results after classification of software fault data come in terms of certain



International Journal of Digital Application & Contemporary research

Website: www.ijdacr.com (Volume 2, Issue 5, December 2013)

efficiency parameters like Accuracy, Reliability, Mean Absolute Error, and Root Mean Squared Error in order to compare both approaches.

II. SOFTWARE FAULT PREDICTION

Faults, errors, defects and failures are terminologies that are inter-related with each other but have considerable disagreement in definitions. Since it is necessary to have the clear distinctions in these terms hence IEEE Standard Glossary of Software Engineering Terminology is followed. According to the library definitions if the mistake is human made that generates an incorrect result, a software fault will occur with the manifestation of error, that would lead to the software failure which results into an inability of system or component to perform its operations within specified parameters [6]. A defect and the fault term is common in hardware and engineering systems. Here, we are using the term fault for software errors caused generally by coding segment. These mistakes are surfaced in the testing portion at system and unit level. Although the anomalies reported by the testing process could mark the software as a fail, but we are focusing on the term fault (it is expected that all reported anomalies are tracked in coding level). In other words, software faults are referred to the pre-release faults, which is similar to the approach proposed by Fenton and Ohlsson [4].

Software fault prediction is one of the quality assurance activities in Software Quality Engineering such as formal verification, fault tolerance, inspection, and testing. Software metrics [14, 15] and fault data (faulty or non-faulty information) belonging to a previous software version are used to build the prediction model. The fault prediction process usually includes two consecutive steps: training and prediction. In the training phase, a prediction model is built with previous software metrics (class or method-level metrics) and fault data belonging to each software module. After this phase, this model is used to predict the fault-proneness labels of modules that locate in a new software version. Recent advances in software fault prediction allow building defect predictors with a mean probability of detection of 71% and mean false alarm rates of 25% [16]. These rates are at an acceptable level and this quality assurance activity is expected to quickly achieve widespread applicability in the software industry.

III. METHODOLOGY

The methodology consists of the following steps:

1. Find the structural code and requirement attributes

The first step is to find the structural code and requirement attributes of software systems i.e. software metrics. The real time defect data sets are taken from the NASA's MDP (Metric Data Program) data repository, [online] Available: <http://mdp.ivv.nasa.gov> in named as PC1 dataset which is collected from a flight software from an earth orbiting satellite coded in C programming language, containing 1107 modules and only 109 have their requirements specified. PC1 has 320 requirements available and all of them are associated with program modules. All these data sets varied in the percentage of defect modules, with the PC1 dataset containing the least number of defect modules.

2. Select the suitable metric values as representation of statement

The Suitable metric values used are fault and without fault attributes, we set these values in database create in MATLAB R2010 A as 0 and 1. Means 0 for data with fault and 1 for data without fault. The metrics in these datasets (NASA MDP dataset) describe projects which vary in size and complexity, programming languages, development processes, etc. When reporting a fault prediction modelling experiment, it is important to describe the characteristics of the datasets. Each data set contains twenty-one software metrics, which describe product's size, complexity and some structural properties. We use only fault and without attributes to classify the selected NASA MDP PC1 dataset. Also the product metrics and product module metrics available in dataset which can also be use are the product requirement metrics are as follows:

- Module
- Action
- Conditional
- Continuance
- Imperative
- Option
- Risk_Level
- Source
- Weak_Phrase

The product module metrics are as follows:

1. Module
2. Loc_Blank
3. Branch_Count
4. Call_Pairs
5. LOC_Code_and_Comment
6. LOC_Comments
7. Condition_Count
8. Cyclomatic_complexity
9. Cyclomatic_Density

10. Decision_Count
11. Edge_Count
12. Essential_Complexity
13. Essential_Density
14. LOC_Executable
15. Parameter_Count
16. Global_Data_Complexity
17. Global_Data_Density
18. Halstead_Content
19. Halstead_Difficulty
20. Halstead_Effort
21. Halstead_Error_EST
22. Halstead_Length
23. Halstead_Prog_Time
24. Halstead_Volume
25. Normalized_Cyclomatic_Complexity
26. Num_Operands
27. Num_Operators
28. Num_Unique_Operands
29. Num_Unique_Operators
30. Number_Of_Lines
31. Pathological_Complexity
32. LOC_Total

1. In the first method Fuzzy c-means clustering approach is used to detect any fault present in the data.
2. In the hybrid approach, Genetic Algorithm optimizes the 'exponent' and 'number of clusters' for FCM approach and then FCM classifies the dataset in order to make an efficient and supervised classification model.

Figure 1 and 2 show flow diagrams for Fuzzy c-means clustering approach and hybrid approach respectively.

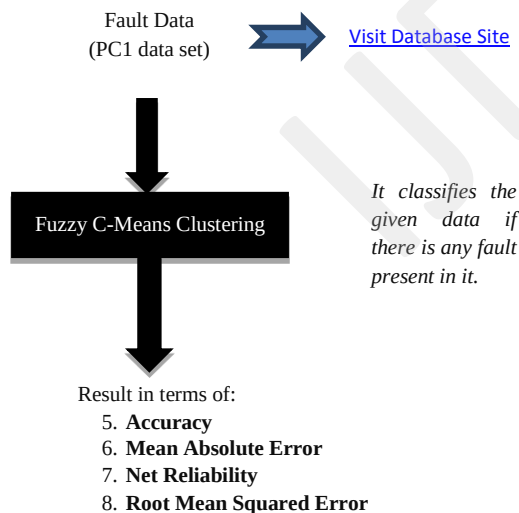


Figure 1: Flow diagram for Fuzzy C-means clustering Approach

In this research we have developed a software fault prediction module using two methods:

- Fuzzy c-means clustering (FCM) approach.
- Hybrid technique (combination of Fuzzy c-means and Genetic Algorithm).

The PC1 software fault database is used which is available at NASA's research website.

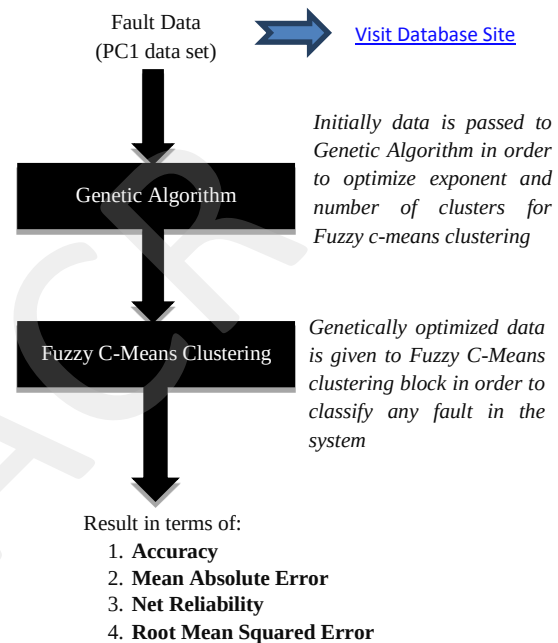


Figure 2: Flow diagram for hybrid technique

Fuzzy C-Means Clustering

Objective function based fuzzy clustering algorithms such as the fuzzy c-means (FCM) algorithm has been used extensively for different tasks such as pattern recognition, data mining, and image processing and fuzzy modeling.

In fuzzy clustering, each point has a degree of belonging to clusters, as in fuzzy logic, rather than belonging completely to just one cluster. Thus, points on the edge of a cluster, maybe in the cluster to a lesser degree than points in the centre of cluster. An overview and comparison of different fuzzy clustering algorithms are available.

With fuzzy c-means, the centroid of a cluster is the mean of all points, weighted by their degree of belonging to the cluster:

$$c_k = \frac{\sum_x w_k(x)x}{\sum_x w_k(x)}$$

The degree of belonging, $w_k(x)$, is related inversely to the distance from x to the cluster center as calculated on the previous pass. It also depends on a



parameter m that controls how much weight is given to the closest center. The fuzzy c-means algorithm is very similar to the k-means algorithm:

1. Choose a number of clusters.
2. Assign randomly to each point coefficients for being in the clusters.
3. Repeat until the algorithm has converged (that is, the coefficients' change between two iterations is no more than, the given sensitivity threshold) :
 - Compute the centroid for each cluster, using the formula above.
 - For each point, compute its coefficients of being in the clusters, using the formula above.

The algorithm minimizes intra-cluster variance as well, but has the same problems as k-means; the minimum is a local minimum, and the results depend on the initial choice of weights. Using a mixture of Gaussians along with the expectation-maximization algorithm is a more statistically formalized method which includes some of these ideas: partial membership in classes.

Algorithmic steps for Fuzzy c-means clustering

Let $X = \{x_1, x_2, x_3 \dots, x_n\}$ be the set of data points and $V = \{v_1, v_2, \dots, v_c\}$ be the set of centers.

- 1) Randomly select 'c' cluster center.
- 2) Calculate the fuzzy membership μ_{ij} using:

$$\mu_{ij} = 1 / \sum_{k=1}^c (d_{ij}/d_{ik})^{(2/(m-1))}$$

- 3) Compute the fuzzy center v_j' using:

$$v_j = \frac{(\sum_{i=1}^n \mu_{ij}^m x_i)}{\sum_{i=1}^n \mu_{ij}^m} \quad \forall j = 1, 2, \dots, c$$

- 4) Repeat step 2) and 3) until the minimum 'J' value is achieved or $||U^{(k+1)} - U^{(k)}|| < \beta$.

Where,

'k' is the iteration step.

' β ' is the termination criterion between [0, 1].

' $U = (\mu_{ij})_{n \times c}$ ' Is the fuzzy membership matrix.

'J' is the objective function.

Genetic Algorithm

A genetic algorithm is a probabilistic search technique that computationally simulates the process of biological evolution. It mimics evolution in nature by frequently altering a population of candidate solutions until an optimal solution is found.

The GA evolutionary cycle starts with a randomly selected initial population. The changes to the population happen through the processes of selection based on fitness, and alteration using mutation and crossover. The application of selection and alteration leads to a population with a higher proportion of improved solutions. The evolutionary

cycle carry on until an acceptable solution is found in the current generation of population, or some regulator parameter such as the number of generations is exceeded.

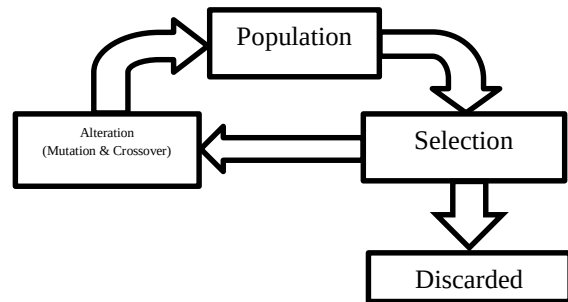


Figure 3: Genetic algorithm evolutionary cycle

The smallest unit of a genetic algorithm is called a gene, which denotes a unit of information in the problem domain. A series of genes, recognized as a chromosome, signifies one possible solution to the problem. Each gene in the chromosome signifies one component of the solution pattern.

The most common form of representing a solution as a chromosome is a string of binary digits. Each bit in this string is a gene. The procedure of converting the solution from its unique form into the bit string is known as coding. The specific coding system used is application dependent. The solution bit strings are cracked to enable their evaluation using a fitness measure.

A. Selection

In biological evolution, only the fittest survive and their gene pool contributes to the creation of the succeeding generation. Selection in GA is also based on a similar process. In a common form of selection, recognized as fitness proportional selection, every chromosome's likelihood of being selected as a decent one is proportional to its fitness value.

B. Alteration to improve good solutions

The alteration step in the genetic algorithm refines the good solution from the current generation to produce the next generation of candidate solutions. It takes place by acting out crossover and mutation.

C. Crossover

Crossover may be regarded as artificial mating in which chromosomes from two individuals are combined to create the chromosome for the next generation. This is carried out by splicing two chromosomes from two different solutions at a crossover point and swapping the spliced parts. The fact is that some genes with good characteristics from one chromosome may as a result combine with

some good genes in the other chromosome to create a better solution represented by the new chromosome.

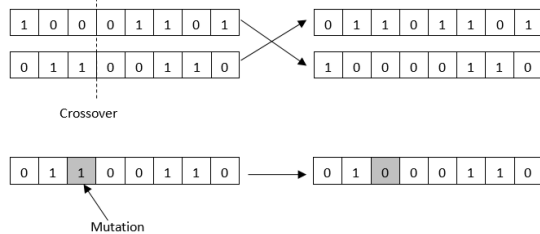


Figure 4: block representation of Crossover and Mutation

D. Mutation

Mutation is a random adjustment in the genetic composition. It is beneficial for announcing new characteristics in a population – something not achieved through crossover alone. Crossover only reorders prevailing characteristics to give new combinations. For instance, if the first bit in every chromosome of a generation happens to be a 1, any novel chromosome created through crossover will also have 1 as the first bit.

The mutation operator changes the current value of a gene to a different one. For bit string chromosome this modification amounts to flipping a 0 bit to a 1 or vice versa. Mutations can be counterproductive, and applied only randomly and infrequently.

The steps in the typical GA for finding a solution to a problem are listed below:

1. Generate an initial solution population of a certain size randomly.
2. Calculate each solution in the current generation and assign it a fitness value.
3. Select “good” solutions based on fitness value and discard the rest.
4. If satisfactory solution(s) found in the current generation or maximum number of generations is exceeded then stop.
5. Change the solution population using crossover and mutation to create a new generation of solutions.
6. Go to step 2.

IV. SIMULATION AND RESULTS

In this study, training and testing methodology is being used, wherein a project is chosen for training the system. The NASA MDP dataset named PC1 is used in this. Then our Fuzzy C-means Clustering and Genetic Algorithm based hybrid approach (GA-FCM) is applied on the same project and the final calculated values are then used to classify the modules of project as fault prone or fault free.

Simulation is carried out using MATLAB 2010a.

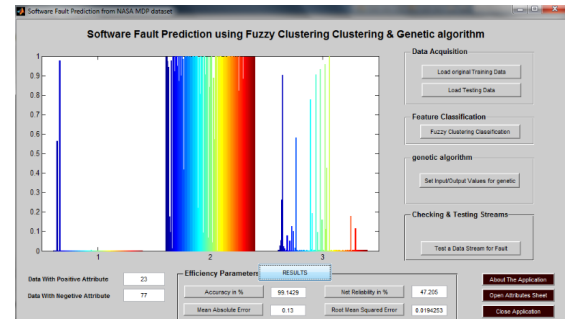


Figure 5: Graphical User Interface (GUI) for proposed work

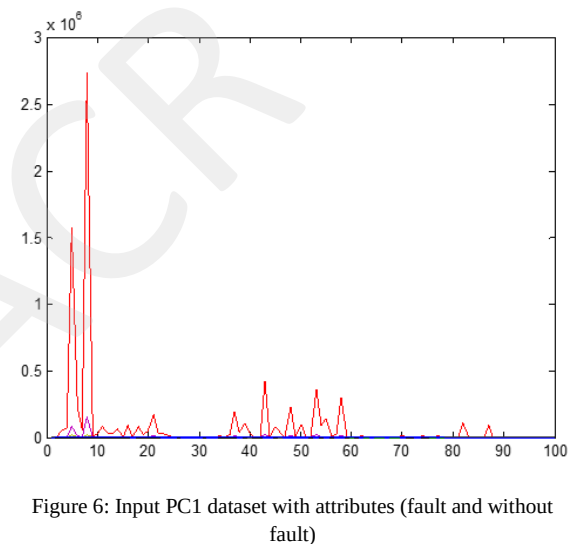


Figure 6: Input PC1 dataset with attributes (fault and without fault)

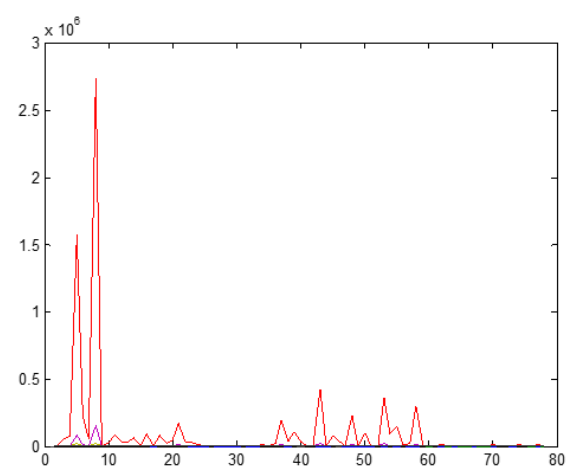


Figure 7: Input PC1 dataset with fault attributes when separating fault attributes from input data

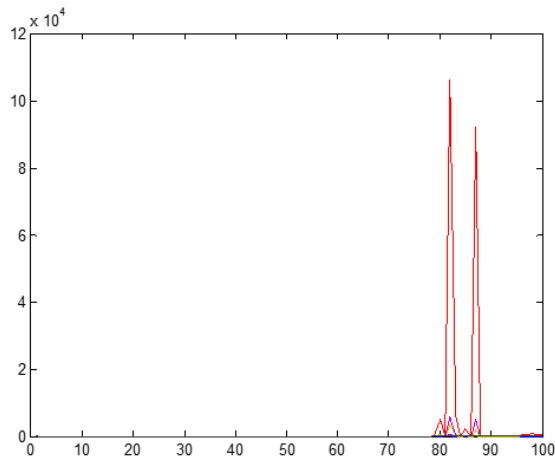


Figure 8: Input PC1 dataset with without fault attributes when separating without fault attributes from input data

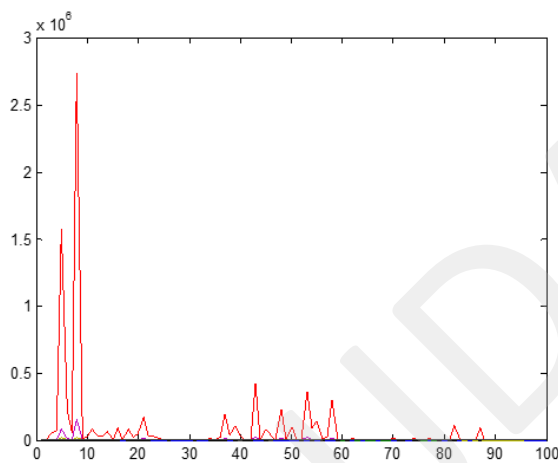


Figure 9: Data Input (PC1 database without attributes)

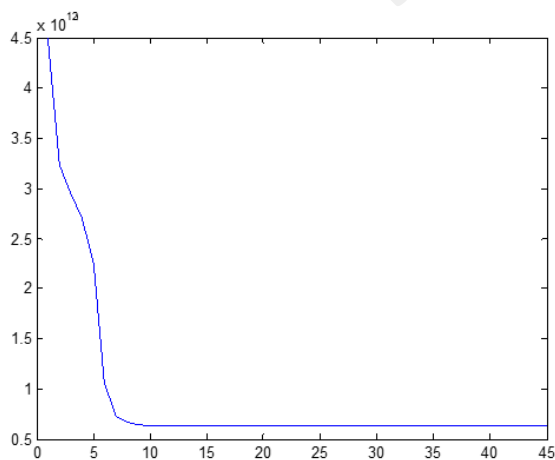


Figure 10: Variations in objective function with respect to iterations count

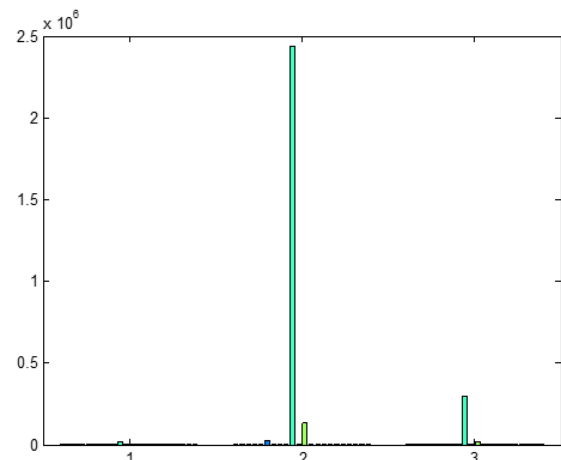


Figure 11: Value of centroid for the number of clusters

Table 1: Performance comparison for Fuzzy C-means and Hybrid (G.A-Fuzzy C-means) technique

Evaluation Parameter	Fuzzy c-means Approach	Hybrid (GA-FCM) Approach
Accuracy	76.20	99.14
Reliability	56.00	47.20
Mean Absolute Error (MAE)	0.28	0.13
Root Mean Squared Error (RMSE)	0.108889	0.019425

V. CONCLUSION

In this paper, the Software Fault Prediction System is implemented using Fuzzy C-means clustering and hybrid (Fuzzy c-means + Genetic Algorithm) techniques. Fuzzy clustering Based techniques are discussed for the comparative analysis in order to predict level of impact of faults in NASA's public domain defect dataset. Predicting faults in the software life cycle can be used to improve software

International Journal of Digital Application & Contemporary research

Website: www.ijdacr.com (Volume 2, Issue 5, December 2013)

process control and achieve high software reliability. It was found that the hybrid method gives more accuracy and less errors as compared to Fuzzy C-means clustering method on the basis of evaluation parameters: accuracy, reliability, MSE and RMSE.

REFERENCE

- [1] Jaakkola T., and Haussler D., "Exploiting generative models in discriminative classifiers", In *Advances in Neural Information Processing Systems 1*, MIT Press, pp. 487-493, 1998.
- [2] Kazama J., and Tsujii J., "Evaluation and extension of maximum entropy models with in equality constraints", *Proceedings of 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP2003)*, pp. 137-144, 2003.
- [3] Anderberg M. R., *Cluster Analysis for Applications*, Academic Press, New York, 1973.
- [4] N. E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. *IEEE Transactions on Software Engineering*, 26(8):797-814, 2000.
- [5] Michael R. Anderberg. *Cluster analysis for applications*. Academic Press, 1973.
- [6] Section 13.4 of Mardia et al., 1979, and Section 2 of Veltkamp and Hagedoorn, 2000
- [7] Sections 13.4 and 14.2.3 of Mardia et al., 1979
- [8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise" *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*.
- [9] Nam Hun Park Won Suk Lee, "Statistical Grid-based Clustering over Data Streams". *SIGMOD Record*, Vol. 33, No. 1, March 2004.
- [10] Paul S. Bradley, Usama Fayyad, and Cory Reina. *Scaling EM (Expectation-Maximization) Clustering to Large Databases*. Technical Report MSR-TR-98-35, Microsoft Research, Redmond, WA, USA, October 1999.
- [11] Binu Thomas, Raju G., and Sonam Wangmo, "A Modified Fuzzy C-Means Algorithm for Natural Data Exploration" *World Academy of Science, Engineering and Technology* 25 2009.
- [12] Bose S.K., "Presenting a Novel Neural Network Architecture for Membrane Protein Prediction", *Intelligent Engineering Systems, INES'06, Proceedings. International Conference*, 2006.
- [13] Yuan Chen, "Research on software defect prediction based on data mining", *IEEE 2nd International Conference on Computer and Automation Engineering (ICCAE)*, 2010.
- [14] N. E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. *IEEE Transactions on Software Engineering*, 26(8):797-814, 2000.
- [15] N. E. Fenton and M. Neil. A critique of software defect prediction models. *IEEE, Transactions on Software Engineering*, 25(5):675-689, 1999.
- [16] Pei Wang, "Software Defect Prediction Scheme Based on Feature Selection", *IEEE, International Symposium on Information Science and Engineering (ISISE)*, 2012.
- [17] Qinbao Song, "Software defect association mining and defect correction effort prediction", *IEEE Transactions on Software Engineering*, Vol. 32, Issue: 2, February 2006.
- [18] Lanubile F., Lonigro A., and Visaggio G., "Comparing Models for Identifying Fault-Prone Software Components", *Proceedings of Seventh International Conference on Software Engineering and Knowledge Engineering*, 1995, pp. 12-19, 1995.
- [19] T.M. Khoshgafaar, E.D. Allen, J.P. Hudepohl, S.J. Aud, "Application of neural networks to software quality modeling of a very large telecommunications system", *IEEE Transactions on Neural Networks*, 8(4), pp. 902-909, 1997.
- [20] Saida Benlarbi, Khaled El Emam, Nishith Geol, "Issues in Validating Object-Oriented Metrics for Early Risk Prediction", by Cistel Technology, Ontario Canada, 1999.
- [21] Runeson, Claes Wohlin, Magnus C. Ohlsson, "A Proposal for Comparison of Models for Identification of FaultProneness", *Dept. of Communication Systems, Lund University, Profes* 2001, pp. 341-355, 2001.