IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 2, Issue 7, February 2014)

Comparative Analysis of Accuracy Prediction using Fuzzy C-Means and KNN Clasiffier

Anil Kumar Singh
NIET, Greater Noida

Rajkumar Goel
NIET, Greater Noida

Pankaj Kumar
NIET, Gretaer Noida

Abstract –**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 uses two methods; Fuzzy c-means clustering approach and k-Nearest Neighbors Classifier technique, have 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, net reliability, RMSE and MAE values.**

Keywords –**Accuracy, Fuzzy c-means, k-Nearest Neighbors Classifier, NASA MDP, MAE, Reliability, RMSE and Software Fault Prediction.**

I. INTRODUCTION

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.

The main objective of this paper is to design a Software Fault Prediction System using Fuzzy c-means clustering approach and k-Nearest Neighbors Classifier. The results after classification of software fault data come in terms of certain efficiency parameters like Accuracy, Reliability, Mean Absolute Error, and Root Mean Squared Error in order to compare all the approaches.

II. SOFTWARE FAULT PREDICTION SYSTEM

Since it is necessary to have the clear distinctions in faults 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

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 2, Issue 7, February 2014)

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. PROPOSED 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

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 2, Issue 7, February 2014)

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

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

- Fuzzy c-means clustering (FCM) approach.
- K-Nearest Neighbors classifier approach

Figure 1, and 2show flow diagrams for Fuzzy c-means clustering approach amd K-Nearest Neighbors classifier respectively.

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

- In the first method Fuzzy c-means clustering approach is used to detect any fault present in the data.
- In the second method K-Nearest Neighbors classifier approach is used to detect any fault present in the data.

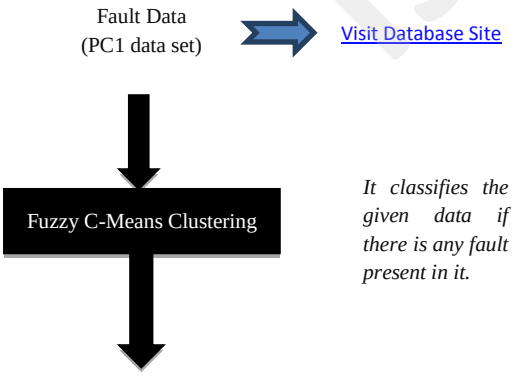

Figure 1: Flow diagram for Fuzzy C-means clustering Approach

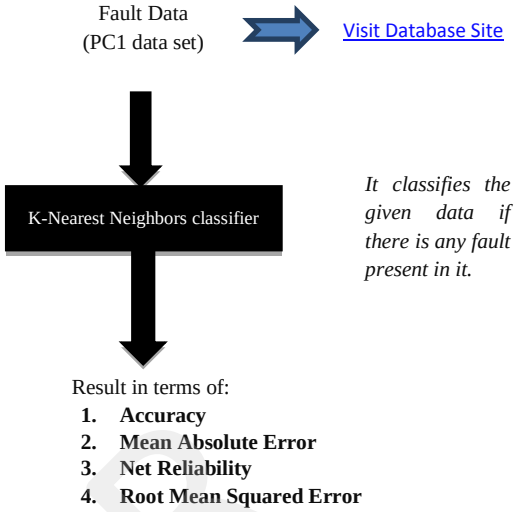

Figure 2: Flow diagram for K-Nearest Neighbors classifier Approach

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) :

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 2, Issue 7, February 2014)

- 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 $X = \{x_1, x_2, x_3 \ldots, x_n\}$ be the set of data points and $V = \{v_1, v_2, \ldots, v_c\}$ 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:

$$v_j = \frac{\left(\sum_{i=1}^{n} {\mu_{ij}}^m x_i\right)}{\sum_{i=1}^{n} {\mu_{ij}}^m} \forall_j = 1,2, \ldots c$$

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).

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

'J' is the objective function.

k-Nearest Neighbors Classifier

In pattern recognition, the k-Nearest Neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

k-Nearest neighbor is an example of instance-based learning, in which the training data set is stored, so that a classification for a new unclassified record may be found simply by comparing it to the most similar records in the training set.

Given a query point x, it is ensured that the instances in a database are not revealed to other databases in the nearest neighbor selection, and that the local classification of each database is not revealed to other databases during global classification.

In order to determine the points in their database that are among the k nearest neighbors of x, each node calculates k smallest distances between x and the points in their database (locally).

After each node determines the points in its database which are within the kth nearest distance from x, each node computes a local classification vector of the query instance where the ith element is the amount of vote the ith class received from the points in this node's database which are among the k nearest neighbors of x.

A very common thing to do is weighted kNN where each point has a weight which is typically calculated using its distance. For eg. under inverse distance weighting, each point has a weight equal to the inverse of its distance to the point to be classified. This means that neighboring points have a higher vote than the farther points.

It is quite obvious that the accuracy might increase on increasing k but the computation cost also increases.

IV. Simulation and Results

In this paper, 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 Fuzzy C-means Clustering and k-nearest neighbour classifier approach 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.

Table 1: Performance comparison for Fuzzy C-means and k-Nearest Neighbors Classifier

Evaluation Parameter	Fuzzy c-means Approach	k-Nearest Neighbors Classifier
Accuracy	79.24	88.31
Reliability	60.07	81.38
Mean Absolute Error (MAE)	0.25	0.11
Root Mean Squared Error (RMSE)	0.083	0.124

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 2, Issue 7, February 2014)

V. CONCLUSION

In this paper, the Software Fault Prediction System is implemented using Fuzzy C-means clustering amd k-Nearest Neighbors Classifier technique. It was found that the k-Nearest Neighbors Classifier 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. Khoshgaftaar, 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.