International Journal of u- and e- Service, Science and Technology
Vol.9, No. 11 (2016), pp.275-288
http://dx.doi.org/10.14257/ijunesst.2016.9.11.24

Software Fault Prediction Using Unsupervised Learning Technique: A Practical Approach

Ekbal Rashid

Department of Computer Science & Engineering
Jharkhand Rai University, Ranchi, India
ekbalrashid2004@yahoo.com

Abstract

Unsupervised learning techniques such as clustering can be used in software fault prediction, where fault labels are not available. The accurate prediction of faults is likely to occur in coding and that can be rectified early testing phase, which reduces the testing cost as well as maintenance cost and enhance the quality of software. In respect of data mining approach, if training data are not present, then I can not use the supervised learning, this is the biggest problem. To solve this problem, new models using unsupervised learning such as clustering algorithms are quite necessary. This work is the extended version of a Various Strategies and Technical Aspects of Data Mining: A Theoretical Approach, which moves towards practical implementation from theoretical foundation [1]. The main objective of this work to find whether software is faulty or non faulty by using the confusion matrix and also calculating the False Positive Rate (FPR), False Negative Rate (FNR), and Error or fault in a software module. In order to obtain the results I have used an indigenous tool.

Keywords*: Data Mining, Classification, Algorithm, Clustering, Confusion Matrix, software fault prediction*

1. Introduction

It would be quite alright if I say that the methods of knowledge discovery have changed over the days. Previously, people used to start with some hypothesis and would try to prove or disprove this hypothesis. This had given the rise to data analysis to validate or invalidate some particular theory. However, now I am not starting from any previously decided hypothesis. Instead, I let the data do the talking. Say, I do not say previously what the nature of rainfall in a particular location will be. Instead, I mine the historical data related to weather and other relevant things and then look at what interesting things actually surface up. This research methodology is becoming popular among scientific circles also. Scientists are favoring to look at the vast amounts of data available to them at the first phase of research. Then they may set up some experimental requirements on the basis of the results that the data points to. In this paper I have considered the application of clustering in software fault prediction. Software fault prediction research area poses great challenges for application of new techniques in the software industry has been difficult due to unavailability of automated tools [16]. Considering the various practical issues involved in software project development, fault data may not be available for all the software modules in the training data. In such cases unsupervised learning technique such as clustering may be used for fault prediction in software modules. Software fault prediction modeling has become a well-known technique for the early prediction of error pattern. It is an important area of research and the subject of many earlier studies. These studies characteristically produce fault prediction models which allow software engineers to focus development activities on fault-prone module, so improving software quality and building better use of resources [17]. Various Issues in applying data mining approaches

ISSN: 2005-4246 IJUNESST

for defect prediction such as Overfitting, Outliers, High dimensionality, Human interaction, Interpretation of results, Visualization of results, Large datasets, , Multimedia data, Missing data, Irrelevant data, Noisy data. The availability of large amounts of data, the biggest challenge is how to exactly integrate, understand and impose many methods to discover and utilize knowledge from the data. For making better decisions in industry, technology and to predict the future for the purpose, people are very much Igor for the discovery of knowledge of the huge amount of data. With the aim to discover unknown patterns from data, methodologies of data have been derived from machine learning, statistics, and artificial intelligence *etc.*, [2]. However, there are so many researches and important implementation issues were associated with the data mining [3-4]. The main objective of this paper is to predict the faulty module or non faulty module on the basis of confusion matrix which tends to reduce the testing cost and increase the quality of software. Unsupervised learning (*i.e.*, Clustering) techniques have been proposed by Zhong *et. al.*, in [5, 21] with the help of a software engineering human expert. The expert specifies other statistics from the software measurement dataset needed to accurately label each other as either fault prone or not fault prone. The clustering analyst was a professional specializing in data mining and machine learning techniques while the labeling expert was a professional with over fifteen years of experience in software quality and reliability engineering. Machine learning algorithms have proven to be of great practical value in a variety of applications. The field of software engineering turns out to be a fertile ground where many software development and maintenance tasks could be formulated as learning problems and approached in terms of learning algorithms. The strength of machine learning methods lies in the fact that they have sound mathematical and logical justifications and can be used to create and compile verifiable knowledge about the design and development of software artifacts [20]. In general, the major clustering methods that can be classified into the following categories [11-12]. These are given below:

- Partitional Clustering
- Hierarchical Clustering
- Density-Based Clustering
- Grid-Based Clustering
- Model-Based Clustering
- Constraint-Based Clustering
- Graph-Based Clustering
- Sub-space Clustering
- Nearest Neighbor Clustering

2. Related Work

In the presence of the huge amount of data, the important issue is how to exactly understand, integrate, and apply various techniques to discover fault from the software module because software fault prediction or detection is a challenging task. Wide research has been done in this direction. Prediction models based on confusion matrix can predict the number of faults or errors in software modules. Timely predictions of such models can be used to direct quality improvement, cost effectiveness *etc.*

This section enlists and reviews existing work to predict the quality (fault) of the software using unsupervised learning technique.

Du Zhang *et. al.*, in [13] Addressed that the machine learning methods are used to predict or estimate software quality, software size, software development cost, project or software effort etc. Machine learning deals with the issue of how to build programs that improve their performance at some task through experience [18]. Rashid *et. al.*, in [19] emphasized in the Area of Machine Learning and Its Application for Software Quality Prediction. Existing software quality estimation models often involve the use of

supervised learning methods to train a software quality classifier or a software fault prediction model. In such models, the dependent variable was a software quality measurement indicating the quality of a software module by either a risk-based class membership or the number of faults, such a measurement may be inaccurate or even unavailable. In such situations the use of unsupervised learning techniques to build a software quality estimation system with the help of a software engineering human experts, clustering and expert-based software quality estimation is an interactive process. Such an approach was important for software projects where resources were relatively limited and finite. The expert specifies other statistics from the software measurement dataset needed to accurately label each other as either fault prone or not fault prone. The clustering analyst was a professional specializing in data mining and machine learning techniques while the labeling expert was a professional with over fifteen years of experience in software quality and reliability engineering. Venkata U.B. & Raymond A. Paul in [14] proposed a variety of software defect prediction techniques, but none has proven to be consistently accurate. Those techniques include statistical methods, machine learning methods, parametric models and mixed algorithm. This work provides a critical review of software defect prediction techniques with special emphasis on machine learning based methods. These techniques were applied to three real-time defect data sets obtained from NASA's MDP data repository. All defects frequently exhibited non-normal characteristics such as skewness, unstable variances, collinearity and excessive outliers [15].

3. Classification and Clustering of Software Fault Prediction

The basic aim of software fault prediction is to identify error prone tasks as the cost can be minimized with advance knowledge about the errors and this early treatment of error will enhance the software quality [22]. These techniques are used to predict software quality of the system by examining a software module and predicting whether it is faulty or non faulty. In this paper an attempt has been made to propose a model with the help of synthetic data which is used for prediction. The main focus of this work to find whether software is faulty or non faulty by using the confusion matrix and also calculating the False Positive Rate (FPR), False Negative Rate (FNR), and Error or fault in a software module. There are the tasks of discovering a classification or of discovering clusters within a data set.

So, come to the idea of why is fault prediction important? It is the activities connected with the fault detection report for a major part of the project budget.

What is software fault prediction? It investigates or examining a software module and predicting whether it is Faulty or Non Faulty.

What I mean by a faulty module? A module which has not passed a definite number of test cases for example, I may say that a module is faulty if it passes less than 95% of the test cases.

Now, the concepts of classification and clustering have come. Basically, they seem to do the same thing. However, there is a marked difference between the two. Classification maps elements to one of the sets of predetermined classes based on the difference among data elements belonging to different classes [6]. Clustering group's data elements into different groups based on the similarity between elements within a single group [7]. In classification, I know the classes beforehand while mostly in clustering. I do not know how many clusters I am going to get. In data mining, I am interested in discovering classification. For example, suppose I have data about cricket matches that have been played in a particular city. Now, this city is notorious for its frequent changes in weather. Suppose I have data such as if the day is sunny and the temperature is 30°, play is continued. If the day is overcast and temperature is 17°, then play is abandoned; if day is sunny and temperature is 20°, then still the play is continued, so on and so forth. Now, the classification problem is that whether one can classify the weather conditions, that is, how

the day was and the temperature into one of the two classification criteria, whether play will be organized or it will be abandoned.

For classification problem, I use the Hunt's algorithm for decision tree identification.

It goes as follows:

Given N different element types and m different decision classes,
Step 1. For i ← 1 to n do
a. Add element i to the i-1 th element item sets from the previous iteration.
b. Identify the set of decision classes for each item set.
c. If an item set has only one decision class, then it is done, remove that item set from the subsequent iteration.

Now I look into the methods of clustering. Clustering is philosophically different from the classification. Classification is the method of increasing the differences between the elements so as to make them belong to different classes, while clustering is the process of increasing the similarities between different elements so as to form them into different clusters. So, clustering essentially partition data sets into several clusters. In the classification I essentially start from some preconceived notion and then go forward and try to push the data into several classes. That is, I decide ahead of time what the classes should be and then after deciding upon that I actually take one data and search for an appropriate class for it. After I find the appropriate class for it, I actually insert that data into that particular class. Then I take up the next data and after I have decided which class it belongs to I insert that data also into another class. In this way I go on and on till the work has been accomplished with the entire data set. In this process I may use automated tools that can decide on the basis of some pre fed rules as to which class the particular data might belong to. Suppose I am mining the data related to a particular mall. I may classify the data of the mall in several ways. I may classify it into the amount or the volume of purchases that have been made by a particular customer on a particular day. The classification can also be done on the basis of the amount of purchases that a customer makes on a particular day. There may be several classes of several amounts. The inference engine can pick up the records of a particular purchase and then push into the appropriate class. In this way I start with some classes at the beginning and actually build up the classification with the data. This is a type of top down approach that is often employed for segregation of data. On the other hand, if I try to explain clustering on the basis of his example, I may say that I do not start from the beginning with any pre conceived class or classes. Instead, I let the data build up its own set of groups and then at the end of the segregation I have what I can say several clusters. So, this is more or less a kind of bottom up approach to data segregation. What is the property of a particular cluster? The property is that the similarity of different elements in one cluster is much more than the similarity between different elements across clusters. So, members belonging to the same cluster are much more similar to one another than they are to some members of some other cluster. And there are several measures of similarities and most of which are reduced to geometric similarities by projecting these data sets into hypercube or n dimensional spaces and then use some kind of distance measures like Euclidian distance measures or Manhattan distance to compute the similarity. The first kind of clustering algorithm is called the nearest neighbor clustering algorithm. This clustering algorithm takes a parameter called threshold or the maximum distance t between the members of a given cluster [7-8].

Given n elements x1, x2,… xn and given a threshold t,
Step 1. j ← 1, k ← 1, cluster = []
Step 2. Repeat
find the nearest neighbor of xj
let the nearest neighbor be in some cluster m
if distance to nearest neighbor is greater than t, then create a new cluster and increment the number of clusters or assign it to the cluster m
j ← j-1
Step 3. until j > n

The kind of clustering that has been mentioned here actually still has a kind of preconceived notion. Here, the notion is the number of clusters. Here, I see that the numbers of clusters are fixed beforehand. However, many may argue that this is not a proper unbiased method of knowledge discovery. So, there is another kind of clustering technique which is also popular and it is called iterative conditional clustering. This differs from the nearest neighbor technique in the sense that here the number of clusters is fixed beforehand. In the nearest neighbor technique, the numbers of clusters are not fixed beforehand. That means one does not know how many clusters, one is going to get given a particular threshold and a data set.

Take the example of the mall, in the mall, according to the first clustering technique I set the number of clusters before hand and then let the inference engine parse the data. The engine decides whether the next data belongs to the cluster of the previous data or should go into some other cluster. This depends as explained earlier, largely on the difference between the two values and any other value for that matter. On the other hand, I may not give any particular number of clusters and then also go on to actually get the clusters done using the next method that is the method of iterative conditional clustering. Either way the procedure is markedly different from the procedure of classification.

Given n different elements and k different clusters and each with a center, this center means the centroid in the statistical sense.

Step 1. Assign each element to the closest cluster center
Step 2. After all assignments have been made, compute the cluster centoids for each of the cluster. This means one has to compute the average of all the points that have made up this cluster. Possibly, this will shift the centroid to a different location.
Step 3. Repeat the above two steps with the new centroids until the algorithm converges or stabilizes so that the centroids will stop shifting and then we know that we have found the best centroids for each of the clusters.

Iterative conditional clustering is essentially a technique where something like saying that suppose I have a data set, and suppose I want to create ten different clusters out of this data set, where would these clusters lie. On the other hand, nearest neighbor clustering technique would say suppose I have this data set and suppose I have some maximum distance, between elements that can lie between a data set, then how many clusters will I find. This concludes about technical aspects and various strategies of data mining and clustering.

4. Confusion Matrix

In the area of machine learning or data mining, a confusion matrix is also known as error matrix or contingency table that allows visualization of the performance of an algorithm. The confusion matrix is a useful tool for analyzing how well classifier can recognize tuples of different classes. A confusion matrix is a table of at least size x by x. x rows and x columns indicate the number of tuples of class i that were labeled by the classifier as class j. In this research work a confusion matrix has been applied for evaluation of the proposed algorithm. The confusion matrix formed as in Table 1. The actual labels are placed with the rows, while the predicted labels are placed with columns. Every entry x_{ij} in this table shows the number of records from class i predicted to be of class j. For instance, x_{00} is the number of records from class 0 which is correctly predicted as class 0. On the other hand, x_{01} is the number of records from class 0 incorrectly predicted as class 1. The following equations have been used for calculating the Error and Precision [9].

$$\text{Error} = \frac{B+C}{A+B+C+D}$$

$$\text{Precision (P)} = \frac{D}{D+B}$$

Table 1. Confusion Matrix

		Predicted Labels	
Actual Labels		**False (1) (Non-Faulty)**	**True (0) (Faulty)**
	False (1) (Non-Faulty)	A (x_{11})	B (x_{10})
	True (0) (Faulty)	C (x_{01})	D (x_{00})

5. Estimation Parameter

I have used a confusion matrix for evaluation of the algorithm. In this work faults have been decided in terms of False Positive Rate (FPR), False Negative Rate (FNR), and ERROR. Therefore, after getting the value of FPR, FNR, and Error, then I decide whether software module is faulty or non- faulty. See Table 2. [10]

Table 2.

N		**Predicted Labels**	
Actual Labels		**False (Non-Faulty)**	**True (Faulty)**
	False (Non-Faulty)	True Negative A	False Positive B
	True (Faulty)	False Negative C	True positive D

$$\text{FPR} = \frac{FP}{TN+FP} \quad (1)$$

$$\text{FNR} = \frac{FN}{TP+FN} \quad (2)$$

$$\text{Error} = \frac{FN+FP}{TN+FP+FN+TP} \quad (3)$$

6. Dataset

I have conducted experiments on ten datasets for computing FPR, FNR, and ERROR to test my software. The ten data sets are related to software fault prediction. However, Detail information for all the synthetic data set has been given in Table 3.

Table 3. (Synthetic Dataset)

Id	Dataset(cluster)	N	TN	FP	FN	TP
1	AR1	100	30	15	5	50
2	AR2	115	20	10	5	80
3	AR3	135	35	0	0	100
4	AR4	95	20	5	5	65
5	AR5	175	55	0	0	120
6	AR6	165	50	10	5	100
7	AR7	155	60	3	2	90
8	AR8	30	10	0	0	20
9	AR9	45	15	8	3	19
10	AR10	25	8	1	1	15

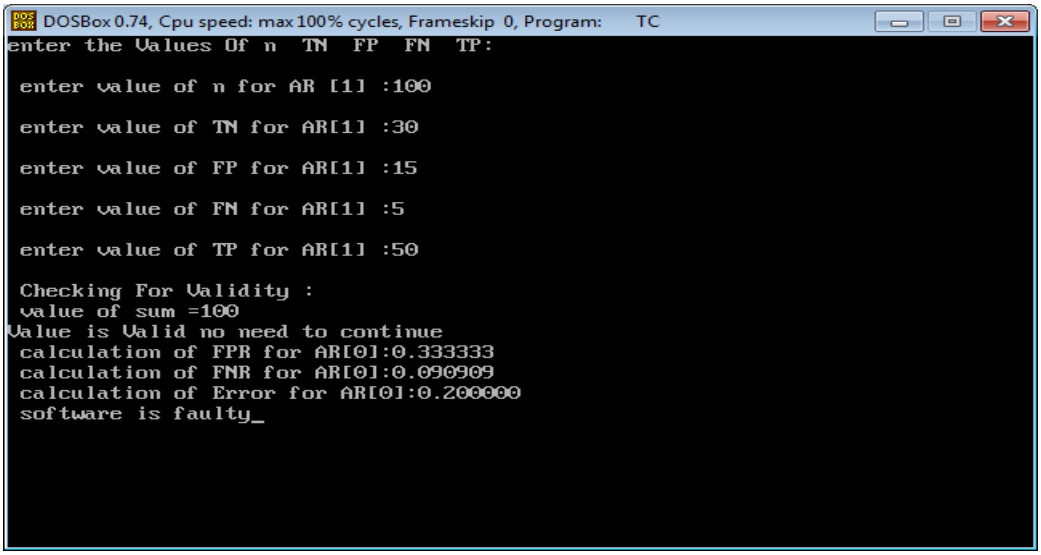
7. Experimental Analysis and Results

I have calculated False Positive Rate (FPR), False Negative Rate (FNR), and Error based on datasets which has been given in Table 3. In this experiment if FPR value is coming high, then testing effort will be wasted while if FNR value is coming high means error prone modules will be escaped testing. It can be seen as a snapshot 1 through 4.
Table 3, shows the synthetic datasets of True Negative (TN**),** True Positive (TP), False Negative (FN), and False Positive (FP). **A**ll these values are used for the calculation purpose. This table also contains ten clusters and these clusters contain more than one attribute. Here, I have given the condition regarding faulty or non faulty of software which has been shown below:
Faulty: -When the values of FN and FP are equal to zero. (*FN and FP=0*).
Non-faulty: -When the values of FN and FP are not equal to zero. (*FN and FP! =0*).

For cluster AR [1]

Table 4. (Values for Cluster AR [1])

n=100		**Predicted Labels**	
Actual Labels		**False (Non-Faulty)**	**True (Faulty)**
	False (Non-Faulty)	TN 30	FP 15
	True (Faulty)	FN 5	TP 50



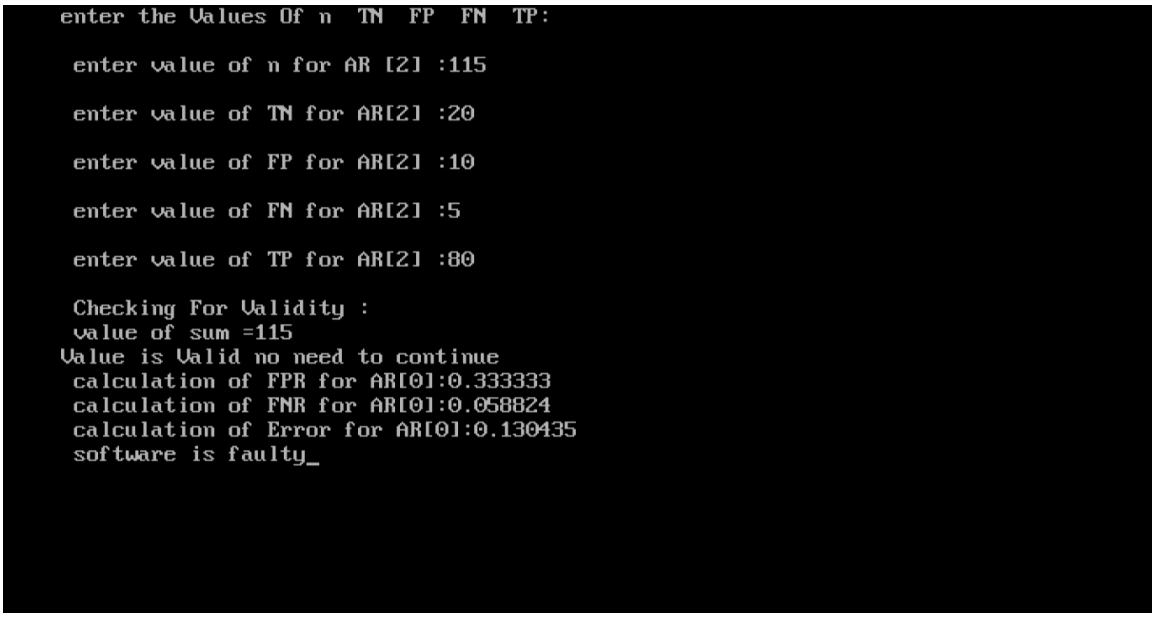
Snapshot 1. (Calculation for AR [1])

Experiment 1 : User input the values for n, TN, FN, TP and FP that are given one by one *i.e.*, 100, 30, 15, 5, and 50 after that it checks the validation of input value and then calculate FPR, FNP and Error that are calculated one by one. The results are coming like 0.33, 0.09, and 0.20 respectively. It can be seen as Snapshot 1, where the value of FPR, FNR, and Error are greater than zero. Therefore, software is Faulty.

For cluster AR [2]

Table 5. (Values for Cluster AR2)

n=115		**Predicted Labels**	
Actual Labels		**False (Non-Faulty)**	**True (Faulty)**
	False (Non-Faulty)	TN 20	FP 10
	True (Faulty)	FN 5	TP 80



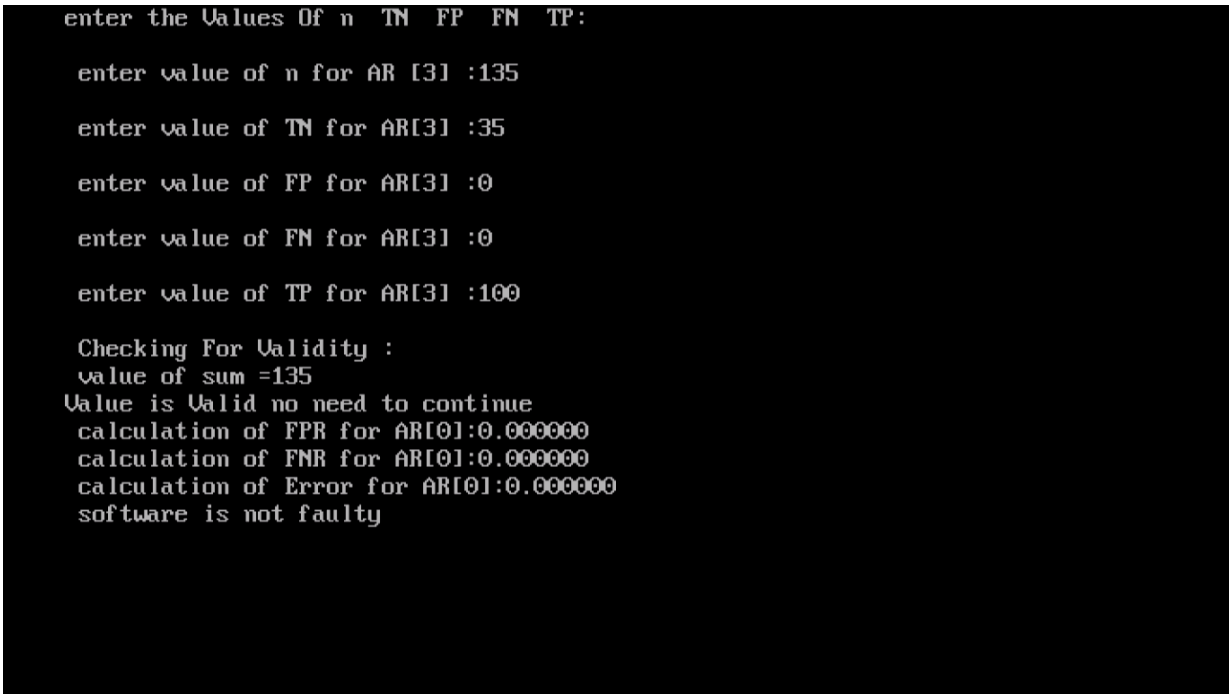
Snapshot 2. (Calculation for AR[2])

Experiment 2: User input the values for n, TN, FN, TP and FP that are given one by one *i.e.*, 115, 20, 10, 5, and 80 it checks the validation of input value and then calculate FPR, FNP and Error that are calculated one by one. The results are coming like 0.33, 0.058, and 0.13 respectively. It can be seen as Snapshot 2, where the value of FPR, FNR, and Error are greater than zero. Therefore, software is Faulty.

For cluster AR [3]

Table 6. (Values for Cluster AR3)

n=135	**Predicted Labels**		
Actual Labels		**False (Non-Faulty)**	**True (Faulty)**
	False (Non-Faulty)	TN 35	FP 0
	True (Faulty)	FN 0	TP 100



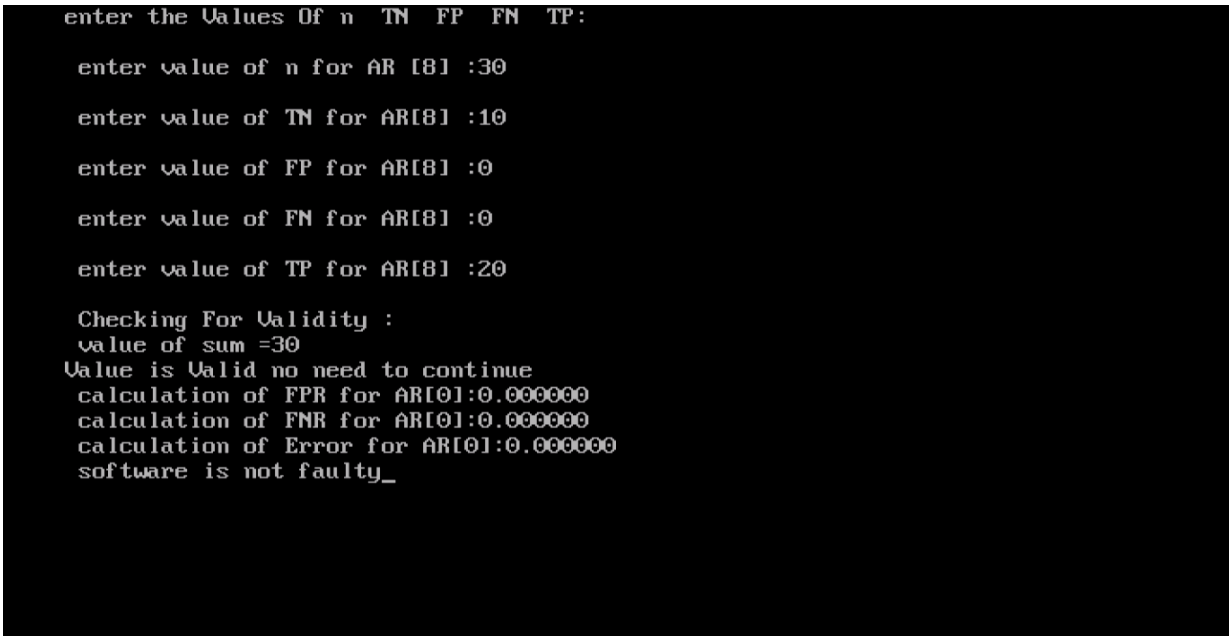
Snapshot 3. (Calculation for AR [3])

Experiment 3: User input the values for n, TN, FN, TP and FP that are given one by one *i.e.,* 135, 35, 0, 0 and 100 it checks the validation of input value and then calculate FPR, FNP and Error that are calculated one by one. The results are coming like 0.0, 0.0, and 0.0 respectively. It can be seen as Snapshot 3, where the value of FPR, FNR, and Error are equal to zero. Therefore, software is non Faulty.

For cluster AR [8]

Table 7. (Values for Cluster AR8)

n=30	**Predicted Labels**		
Actual Labels		**False (Non-Faulty)**	**True (Faulty)**
	False (Non-Faulty)	TN 10	FP 0
	True (Faulty)	FN 0	TP 20



Snapshot 4. (Calculation for AR[8])

Experiment 4: User input the values for n, TN, FN, TP and FP that are given one by one *i.e.*, 30, 10, 0, 0, and 20 it checks the validation of input value and then calculate FPR, FNP and Error that are calculated one by one. The results are coming like 0.0, 0.0, and 0.0 respectively. It can be seen as Snapshot 4, where the value of FPR, FNR, and Error are equal to zero. Therefore, software is non Faulty.

Table 8. Experimental Results on Synthetic Dataset

Dataset	FPR	FNR	Error
AR1	0.33	0.09	0.2
AR2	0.33	0.06	0.13
AR3	0	0	0
AR4	0.2	0.071	0.105
AR5	0	0	0
AR6	0.16	0.476	0.909
AR7	0.047	0.021	0.0322
AR8	0	0	0
AR9	0.34	0.136	0.244
AR10	0.11	0.06	0.08

Table 8, presents the software fault prediction error analyses for ten datasets namely AR1 to AR10. The values of Error, FNR, and FPR are presented for clustering algorithm. The given error labels with the dataset were used to form the confusion matrix for calculating prediction accuracy.

Table 9. Summary of Faulty and non Faulty of Software Prediction

Dataset (cluster)	N	FPR	FNR	Error	Result (Faulty/non-Faulty)
AR1	100	0.33	0.09	0.2	Faulty
AR2	115	0.33	0.06	0.13	Faulty
AR3	135	0	0	0	Non-faulty
AR4	95	0.2	0.071	0.105	Faulty
AR5	175	0	0	0	Non-faulty
AR6	165	0.16	0.476	0.909	Faulty
AR7	155	0.047	0.021	0.0322	Faulty
AR8	30	0	0	0	Non-faulty
AR9	45	0.34	0.136	0.244	Faulty
AR10	25	0.11	0.06	0.08	Faulty

Table 9, presents the detailed summary of faulty and non faulty of software prediction on the basis of values of FPR, FNR and error tested by my software.

8. Conclusion and Future Scope

Software systems are omnipresent in all aspects of society. Software quality is paramount important. Software fault prediction can prevent a big disaster or failure. According to a new federal study: "Software bugs are costing the U.S. economy an estimated $59.5 billion each year, with more than half of the cost borne by end users and the remainder by developers and vender's. Improvement in testing could reduce this cost by about a third or $22.5 billions". The paper presents an Important, Interesting, and an Implementable topic /tool of research to use an unsupervised learning technique for software fault prediction. The main focus is to evaluate the effectiveness of clustering algorithm. In this paper, I have evaluated the potential of clustering algorithm and confusion matrix as well. I compute the defects in terms of FPR, FNR, and ERROR using confusion matrix and then decide the software is faulty or non faulty. The summary of faulty and non faulty has been shown in Table 8. The experiment number 1 to 4 shows the efficiency of confusion matrix and it can be seen as a snapshot (See Snapshot 1 through 4). The future scope is building upon these algorithms and trying to implement these algorithms in different interesting areas. Some of the major areas where these things may be implemented are weather forecasting, political trend finding, scientific analysis, space research, stock market trend study, software growth analysis, software quality prediction and study, *etc.* Each of these cases involves huge sums of data and the data actually build up to such enormous extents say from several hundreds of gigabytes to terabytes and even terabytes. It is very essential in such cases to follow the correct algorithm so as to have a correct analysis of the data. Knowledge discovery in these cases may lead to many interesting discoveries that may set the wheels of progress in an important direction.

References

[1] E. Rashid, S. Patnaik and A. Usmani, "Various Strategies and Technical Aspects of Data Mining: A Theoretical Approach", has been published in Book Title Computational Vision and Robotics and Series Title Advances in Intelligent Systems and Computing Springer. (Indexed in SCOPUS, DBLP, ISI). http://link.springer.com/chapter/10.1007/978-81-322-2196-8_6, vol. 332, **(2015)**, pp. 47-52.
[2] L. Wang and X. Fu, "Data mining with computational intelligence", 1st Indian reprints, Springer, **(2010)**, pp. 1-10.
[3] M. H. Dunhum, "Data Mining- Introductory and Advanced Topics", Pearson Education, 3rd Impression, **(2008)**.
[4] S. K. Pal and P. Mitra, "Pattern recognition algorithms for data mining", Chapman & Hall CRC press, **(2004)**.

[5] S. Zhong, T. M. Khoshgoftaar and N. Seliya, "Unsupervised Learning for Expert-Based Software Quality Estimation" Proc. IEEE Eighth Int'l Symp. High Assurance Systems Eng., **(2004)**, pp. 149-155.
[6] S. Srinivasa, "Data Mining, DataWarehousing and KnowledgeDiscovery Basic Algorithms and Concepts", published by AnilSoft.com.
[7] C. Catal, U. Sevim and B. Diri, "Clustering and Metrics Threshold based software fault prediction of Unlabeled Program Modules", 6th Int. conf. on Information Technology: New generations. IEEE, **(2009)**, pp. 199-204.
[8] J. Han and M. Kamber, "Data Mining Concepts and Techniques", Second Edition, Morgan Kaufmann Publishers.
[9] P. N. Tan, M. Steinbach and V. Kumar, "Introduction to data mining", 4th Impression, Pearson education, **(2009)**.
[10] P. S. Bisnu and V. Bhattacherjee, "Software fault prediction using Quard-Tree based K-Means clustering algorithm", IEEE Transaction Knowledge and data engineering, vol. 24, no. 6, **(2012)**, pp. 1146-1150.
[11] J. Han and M. Kamber, "Data mining concepts and techniques", 2nd edition, Morgan Kaufmann Publishers, **(2007)**.
[12] R. Xu and W. D. Donald, "Survey of clustering algorithms", IEEE Transactions on neural networks, vol. 16, no. 3, **(2005)**, pp. 645-678.
[13] D. Zhang and J. J. P. Tsai, "Machine Learning and Software Engineering", Proceeding of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02).
[14] V. U. B. Challagulla, F. B. Bastani, I.- L. Yen and R. A. Paul, "Empirical Assessment of Machine Learning based Software Defect Prediction Techniques", Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'05).
[15] L. Pickard, B. Kitchenham and S. Linkman, "An Investigation of Analysis Techniques for Software Datasets", Software Metrics Symposium, 1999. Proceedings. Sixth International, **(1999)** November 4-6, pp. 130 -142.
[16] Hall, "A systematic literature review on fault prediction performance in software engineering", IEEE Transaction on Software Engineering, vol. 38, no. 6, **(2012)**.
[17] Catal and Diri, "A systematic review of software fault prediction studies", Expert Systems with Application, vol. 36, no. 4, **(2009)**.
[18] T. Mitchell, "Machine Learning", McGraw-Hill, **(1997)**.
[19] E. Rashid, S. Patnaik and V. Bhattacherjee, "A Survey in the Area of Machine Learning and Its Application for Software Quality Estimation", has been published in ACM SigSoft ISSN 0163-5948, , http://doi.acm.org/10.1145/2347696.2347709 New York, NY, USA, vol, 37, no. 5, **(2012)** September.
[20] D. Zhang and J. J. P. Tsai, "Machine learning and software engineering", Technical Report TR-1, Department of Computer Science, California State University, Sacramento, **(2002)** February.
[21] S. Zhong, T. M. Khoshgoftaar and N. Seliya, "Analyzing Software Measurement Data with Clustering Techniques", IEEE Intelligent Systems, vol. 19, no. 2, March-April **(2004)**, pp. 20-27.
[22] E. Rashid, S. Patnaik and V. Bhattacherjee, "Machine Learning and Software Quality Prediction: As an Expert System", International Journal of Information Engineering and Electronic Business (IJIEEB), DOI: 10.5815/ijieeb.2014.02.02 ISSN: 2074-9023 (Print), ISSN: 2074-9031 (Online), vol. 6, no. 2, **(2014)** April, pp. 9-27.

Author

Ekbal Rashid is working in the Faculty of Computer Science and IT with Jharkhand Rai University, Ranchi, India. He received his BCA, MCA, M.Tech in Computer Science in the year of 2000, 2003 & 2009 respectively from BIT, Mesra and his Ph.D in Computer Science and Engineering in May 2015 from Siksha 'O' Anusandhan University (Deemed University), Bhubaneswar, Orissa. Dr. Rashid has more than 26 National and International publications in journals and conference proceedings of repute including Springer, IEEE, Inderscience. His research area is software engineering, machine learning, data mining and artificial intelligence. He has over more than fourteen years of teaching experience in various reputed Institutes/Universities across the state.

