Software Fault Prediction with Metric Threshold Using Clustering Algorithm
S.Nandhini Dei
Department of Computer Science, MAM School Of Engineering, Trichy

ABSTRACT
Software fault prediction using clustering algorithm metrics used to build the software fault prediction model software. However there are certain cases when previous fault data are not present level. In software fault prediction using hierarchical Agglomerative Algorithm to predict the fault software and report the prediction in that software using previous version of algorithm kmeans and quad tree based algorithms, the clustering algorithm classified as unsupervised and supervised technique to predict the fault.

Keywords: Clustering, Quad Tree K-Means clustering, Quad Tree, Software Fault Prediction

1. INTRODUCTION
quality models are useful tools toward achieving the objectives of a software quality to assurance initiative. A software model can be used to identify program modules that are likely to be defective. Subsequently, the limited resources are allocated are software modules inspection and improvement can be targeted toward only those program modules, achieving cost-effective resource utilization.

A software quality control estimation model allows the software team to track and detect potential software defects relatively early on during development, which is critical to many high-assurance systems. The trained model is then applied to modules of the current project to estimate their quality. Such a supervised technique assumes that the development organization has experience with systems similar to the current project and that defect data are presented for all program modules in the training data set.

In software development practice, the various practical disadvantages limit of the availability of defect data’s for modules in the training set of data. For example, an organization recorded the recorded or collected software defect data from previous releases or similar projects. In addition, since the organization have experience developing a similar system, the use of software modules and defect data of previous projects for modeling purposes to the future. In current times, where globalization of technology has introduce the advantage to gain the software prediction, distributed to the software development is not uncommon.

Under such conditions, software defect data may not be collected by all development sites depending on the organizational structure and resources of individual sites. The absence of detect the data training data set of data prevents the commonly use Agglomerative software fault prediction algorithm to software quality modeling.

Consequently, the task of software quality measurements or labeling program modules as fault software (fp) or not fault software modules (nfp) falls on the software engineering expert. The process of each p Agglomerative software program module one at a time is a laborious, expensive, and time-consuming effort. proposed semi supervised clustering scheme to aid the expert in the labeling process.

The proposed scheme is contain constraint-based clustering using k-means as the soft prediction algorithm. During the k-means clustering process, the constraint maintains membership of modules (instances) to clusters that are already labeled as either fp or nfp. The proposed technique is to analyze the service of unlabelled S/W modules using two stage approaches.

One is using metrics thresholds and another one is using Agglomerative clustering algorithm and then comparing both in terms of time and time means analysis. The clustering methods create groups of objects and objects in one cluster are similar whereas objects in Clustering technique can be used to group the modules having similar metrics by using similarity measures or dissimilarity measures (distances).

After clustering phase, an expert or an automated approach can check the representative modules of each cluster and then, decide to label the cluster as fault software or not fault-software modules. In this study, we show that software metrics thresholds techniques used to label the clusters instead of using an expert for this time-consuming stage different cluster are dissimilar. Clustering is basically categorized into 2 categories.

1) Partitional Clustering: Given a dataset of n objects, a partitioned clustering algorithm constructs k partitions of the data, where each cluster optimizes a clustering criterion, such as the minimization of the sum of the squared distance from the mean within each cluster.
2.2 Attribute Selection
- Get all Attribute in our dataset
- The dataset contains totally 30 numbers of attributes.
- Attribute Selection is the technique of selecting a subset of relevant features for building robust learning models.
- We take all attributes into our process it takes so much time for processing and increase the work burden.
- So, we reduce the total number of attributes and consider high relevance attributes only.
- Calculate the relevance of an attribute using Attribute Evaluation.
- We use Weka tool for attribute selection and ranker.

RELATED WORK
2.1 Quad based K-means Algorithm and Agglomerative predicting the software modules First, Quad Trees are applied for finding the initial cluster centers to be input to the K-Means Algorithm and agglomerative predicting algorithm. An input threshold parameter governs the number of initial cluster centers and by varying the user can generated the desired initial cluster centers. The concept of clustering gain has been used to determine the quality of clusters for evaluation of the Quad Tree-based initialization algorithm as compared to other initialization techniques. The clusters obtained by Quad Tree-based algorithm were found to have maximum gain values.

Second, the Quad Tree based algorithm is applied for faults in program modules. 2.2 Attribute Selection · Get all Attribute in our dataset · The input dataset contains totally 30 numbers of attributes. · Attribute Selection is the technique of selecting a subset of relevant features for building robust learning models.

We take all attributes into our process it takes so much time for processing and increase the work burden.0 · So, we deducted the number of attributes and consider high relevance attributes only. · Calculate the relevance of an attribute using Attribute Evaluation. · We use Weka tool for attribute selection and ranker. 3.

2. OVERVIEW
OVERVIEW This paper shows the Hierarchical Agglomerative clustering algorithm then proposed system architecture for software fault prediction, result using confusion matrix and conclusion Figure 1. Quad Tree 3.1 Hierarchical Agglomerative Clustering · Clustering algorithms are being successfully applied for solving both classification and regression problems.

It is therefore important to predict the capabilities of this algorithm in predicting software detection. In this study, a Hierarchical agglomerative Clustering Based technique is used for finding faulty Module. · Agglomerative hierarchical clustering is a bottom-up clustering method where clusters have sub-clusters, Bottom-up agglomerative algorithms in each document as a singleton cluster at the outset and then successively merge pairs of clusters until no data's in the dataset. · After the clustering process we separate clustered data's and cluster centroid. Figure2. Hyper Quad Tree 3.1.2. Advantages of Quad Based Kmeans algorithms · Propose a single technique · Hierarchical Agglomerative Clustering.

Processing time is reduced · Clear description of Metric Threshold 3.2 Software Quality Classification Modeling Algorithm Software quality estimation models provide assistance in controlling software development errors by allowing problem areas to be fixed prior to their respective deployment phases.

Such modules that can be used as tools for predicting fault-prone areas of a software system, predicting the quality to software testing and operations can benefit the software development team. Such timely to estimation can be used to direct cost-effective quality improvement efforts to the high-risk modules.

In classification problems, the given a set of records, called a fit data set, where each record consists of attributes. One of the attributes, called the dependent variable, indicates the class to which each record belongs. The goal of classification is to build a model that estimates the dependent variable based upon the other attributes, usually referred to as the independent variables.

Once such a model is built, it can be used to determine the class of future unclassified records. Applications of the classification of the system in a software fault prediction using to track the market status and to seen the software quality of the system to accept the current affairs.

A classification tree and classification of the system based on the recursively partitions the fit data set until each partition consists entirely The tree consists of decision nodes (root and internal nodes) and leaf or terminal nodes. Each decision node contains a split threshold which is a test on one, or more, independent variables and determines how the data is partitioned at that node.

A leaf node consists entirely or predominantly of records in one class, which forms the class label associated with that node. Unclassified records (from test or evaluation data sets) that fall into leaf nodes are classified as per the class label associated with the respective leaf nodes. Predicting the quality of system
modules prior to software testing and operations can benefit the software development team.

Such timely reliability estimation can be used to direct cost-effective quality improvement efforts to the high-risk modules. Tree-based software quality classification models based on software metrics are used to predict whether a software module is fault-prone or not fault-prone. They are white box quality estimation models with good accuracy, and are simple and easy to interpret.

This paper presents an in-depth study of calibrating classification trees for software quality estimation using the SPRINT decision tree algorithm. Many classification algorithms have memory limitations including the requirement that data sets be memory resident. SPRINT removes all of these limitations and provides a fast and scalable analysis.

It is an extension of a commonly used decision tree algorithm, CART, and provides a unique tree-pruning technique based on the Minimum Description Length (MOL) principle. Combining the MOL pruning technique and the modified classification algorithm, SPRINT yields classification trees with useful prediction accuracy. Figure 3. System Architecture

The above performance indicators should be minimized.

A high value of FPR would lead to wasted testing effort while high FNR value means error prone modules will escape testing. In this paper, for calculating the measures, if any metric value of the centroid data point of a cluster was greater than the threshold, that cluster was labeled as faulty and otherwise it was labeled as non-faulty. After this the predicted fault labels will compare with the actual fault labels.

Also the clusters can be labeled according to the majority of its members (by comparing with metrics thresholds) but this increases the complexity of the labeling procedure since all the modules in the cluster need to be examined.

3.1 Hierarchical Agglomerative Clustering

- Clustering algorithms are being successfully applied for solving both classification and regression problems. It is therefore important to investigate the capabilities of this algorithm in predicting software quality. In this study, a Hierarchical Clustering Based Approach is used for finding faulty Module.
- Agglomerative hierarchical clustering is a bottom-up clustering method where clusters have sub-clusters, which in turn have sub-clusters, etc. Bottom-up algorithms treat each document as a singleton cluster at the outset and then successively merge pairs of clusters until no data’s in the dataset.
- After the clustering process we separate clustered data’s and cluster centroid.

```plaintext
SIMPLE-HAC(d_1, ..., d_N)
1  for n ← 1 to N
2    do i ← 1 to N
3      do C[n][i] ← SIM(d_n, d_i)
4      I[n] ← 1 (keeps track of active clusters)
5      A ← [] (assembles clustering as a sequence of merges)
6    for k ← 1 to N − 1
7      do (i, m) ← arg max_{(i,m) \in [1..N] \times [1..N]} \{C[i][m] + I[m] = 1\}
8      A.append((i, m)) (store merge)
9    for j ← 1 to N
10       do C[i][j] ← SIM(i, m_j)
11       C[j][i] ← SIM(i, m_j)
12       I[m] ← 0 (deactivate cluster)
13  return A
```

Figure 1. Quad Tree
3.1.2. Advantages of Quad Based Kmeans algorithms

- Propose a single technique
- Hierarchical Agglomerative is used for Clustering.
- Processing time is reduced
- Clear description of Metric Threshold

3.2 Software Quality Classification Modeling Using The SPRINT Decision Tree Algorithm

Software quality estimation models provide assistance in controlling software development errors by allowing problem areas to be fixed prior to their respective deployment phases. Such models can be used as tools for predicting fault-prone areas of a software system. Predicting the quality of system modules prior to software testing and operations can benefit the software development team. Such timely reliability estimation can be used to direct cost-effective quality improvement efforts to the high-risk modules. In classification problems, we are given a set of records, called a fit data set, where each record consists of attributes. One of the attributes, called the dependent variable, indicates the class to which each record belongs. The goal of classification is to build a model that estimates the dependent variable based upon the other attributes, usually referred to as the independent variables. Once such a model is built, it can be used to determine the class of future unclassified records. Applications of classification are seen in diverse fields, such as retail target marketing, customer retention, medical diagnosis, and fraud detection. The focus of this paper is the application of a classification algorithm to the field of software reliability and quality engineering. A classification tree is a class discriminator that recursively partitions the fit data set until each partition consists entirely, or mostly, of records from one class. The tree consists of decision nodes (root and internal nodes) and leaf or terminal nodes. Each decision node contains a split threshold which is a test on one, or more, independent variables and determines how the data is partitioned at that node. A leaf node consists entirely or predominantly of records from one class, which forms the class label associated with that node. Unclassified records (from test or evaluation data sets) that fall into leaf nodes are classified as per the class label associated with the respective leaf nodes. Predicting the quality of system modules prior to software testing and operations can benefit the software development team. Such timely reliability estimation can be used to direct cost-effective quality improvement efforts to the high-risk modules. Tree-based software quality classification models based on software metrics are used to predict whether a software module is fault-prone or not fault-prone. They are white box quality estimation models with good accuracy, and are simple and easy to interpret. This paper presents an in-depth study of calibrating classification trees for software quality estimation using the SPRINT decision tree algorithm. Many classification algorithms have memory limitations including the requirement that data sets be memory resident. SPRINT removes all of these limitations and provides a fast and scalable analysis. It is an extension of a commonly used decision tree algorithm, CART, and provides a unique tree-pruning technique based on the Minimum Description Length (MOL) principle. Combining the MOL pruning technique and the modified classification algorithm, SPRINT yields classification trees with useful prediction accuracy.

![Software Fault Dataset](image1)

**Figure 2. Hyper Quad Tree**

**Table 1. Confusion Matrix**

<table>
<thead>
<tr>
<th>Actual Labels</th>
<th>False (Non-Faulty)</th>
<th>True (Faulty)</th>
</tr>
</thead>
<tbody>
<tr>
<td>False (Non-Faulty)</td>
<td>True Negative A</td>
<td>False Positive B</td>
</tr>
<tr>
<td>True (Faulty)</td>
<td>False Negative C</td>
<td>True Positive D</td>
</tr>
</tbody>
</table>

The above performance indicators should be minimized. A high value of FPR would lead to wasted testing effort while high FNR value means error prone modules will escape testing.

In this paper, for calculating the measures, if any metric value of the centroid data point of a cluster was greater than the threshold, that cluster was labeled as faulty and otherwise it was labeled as non-faulty. After this the predicted fault labels will compare with the actual fault labels. The clusters can be label corresponds to the majority of its members (by
comparing with metrics thresholds) but this increases the complexity of the labeling. The above performance indicators should be minimized. A high value of FPR would lead to wasted testing effort while high FNR value means error prone modules will escape testing.

In this paper, for calculating the measures, if any metric value of the centroid data point of a cluster was greater than the threshold, that cluster was labeled as faulty and otherwise it was labeled as non-faulty. After this the predicted fault labels will compare with the actual fault labels. Also the clusters can be label according to the majority of its members (by comparing with metrics thresholds) but this increases the complexity of the labeling procedure since all the modules in the cluster need to be examined.

procedure since all the modules in the cluster need to be examined.

5. The Proposed Systems
The proposed system is the Software fault prediction using clustering approach is that classify given data using Hyper Quad-tree algorithm.

The system consists of 3 modules · Create dataset parser · Data set is given as input to the Hyper Quad-tree algorithm in which we Create cells, insert cell, label bucket, split cell, spatial decomposition · Input: Dimension, Data set Output: Centroid · Centroid points obtained using the Hyper Quad-tree is given as an input to the K-Means to get better clusters it Calculates the distance, Shuffle data points according to distance, If centroids are stable then stop.

- The output of this will be set of clusters · approach! that classify given data using Hyper Quad-tree algorithm. The system consists of three modules
- Create dataset parser
- Data set is given as input to the hierarchical algorithm in which we Create cells, insert cell, label bucket, split cell, spatial decomposition
- Input: Dimension, Data set Output: Centroid
- Centroid points obtained using the Hyper Quad-tree is given as an input K-Means to get better clusters it Calculates the distance, Shuffle data points according to distance, If centroids are stable then stop. The output of the cluster FPR.
- Measure the Faults in terms of FPR, FNR.

6. ACKNOWLEDGMENTS
This is a small review of my post graduate project work that I am going to start to implement. I specially thank to my Guide for his assistance.

7. REFERENCES

Finally we have the threshold vector [LoC, CC, UOp, UOpnd, TOp, TOpod]

2. Measure the Faults in terms of FPR, FNR and ERROR using confusion matrix. 5. Metrics thresholds Determine the acceptable metrics thresholds using some parameters. The Parameters are, · Lines of Code (LoC), · Cyclomatic Complexity (CC), · Unique Operator (UOp), · Unique Operand (UOpnd), · Total Operator (TOp), · Total Operand (TOpod). Finally we have the threshold vector [LoC, CC, UOp, UOpnd, TOp, TOpod]