A Novel Fuzzy-K Means based Support Vector Machine for Software Quality Prediction

Amandeep¹, Abhishek Gupta²

¹Research Scholar, ²Assistant Professor

¹,²Department of Computer Science, Baddi University of Emerging Sciences and Technology
Baddi, Solan. H.P-173205, India

Abstract: Size has been one of the most important factor in software which is increasing day by day, due to which we have seen an enormous amount of increase in complexity as well as in size of software. There has been a quick increase in efforts required in managing the quality of the software. How to preserve and guarantee software quality was always an important issue which was needed to be solved. Now a days there are different sorts of software quality prediction techniques existing which will help us to predict the software quality. Though, all these techniques lack a broad study to assess and relate numerous prediction methodologies so that quality professionals may select a suitable predictor. The key objective of this paper is learn about the diverse techniques which performs best and are an undesirable problem since behavior of a predictor also depends on numerous other specific factors like ambiguity in the availability of data, nature of dataset, problem dominion etc. We have tried to show the some of the several software quality prediction techniques in terms of various evaluation metrics.

Keywords - Software Product, Software Quality, Software Quality Prediction, Software Matrices.

1. INTRODUCTION

When we usually think or hear the word “Software Quality”, we usually imagine of a software product which is of splendid quality and satisfies our expectations. These expectations are based on the intended use of the user, i.e. what the user wishes and how he is going to use the software product. There are a numeral models which has been anticipated for assessment of software quality based on a number of characteristics. Quality of software product is defined in terms of basic components as constituent part of any program or software and proposed a software quality prediction model based on basic components [1]. Software quality prediction is useful for the better use of resources and to minimize faults. Hence it helps in reducing the cost by early assessment of faults. It also helps in planning more thorough tests for modules that might have defects [2]. Many organizations and institutions are involved in the study of software quality prediction techniques and much work has been done to develop and refine models. These models have also been compared and benchmarked using different data sets [3].

Commonly, the quality of software product is well-defined in terms of its fitness of purpose. Even though fitness of the purpose is a satisfactory explanation of quality for hardware products, but it is not satisfactory for software products. To give associate degree example of why this can be therefore take into account, a software product that's functionally correct. That is, if it correctly performs all the functions that have been as per the SRS documents. Even though it's going to be functionally correct, we cannot take into account it to be a quality product, if it has a nearly unusable programme. Therefore, the traditional conception of quality as fitness of purpose for software package product isn't entirely satisfactory [4]. The concept of software quality is not as simply determinable.

There are numerous attainable quality characteristics of software and there is even a global standard for this. For each quality characteristics, a set of attributes, which will be measured, is determined such a definition helps in evaluating the standard of software [5]. Effective software quality analysis and assurance requires models that describe what the software quality is and however will or not it's copied back to the event method.

Two approaches can be followed to make sure of software quality, one is focused on direct specification and assessment of the quality of software product, while the alternative is concentrated on reassuring prime quality of method by that the product is developed. Number of models [5, 6, 7] has been proposed to evaluate the quality of software product, based on various quality characteristics. McCall quality model [4] attempts to bridge the gap between users and developers by focusing, on a number of software quality factors. The evaluation of software has been done by Boehm’s quality model [6], uses a given set of attributes and metrics. More recently, model has been developed by Dromey’s [7], which is focusing on the relationship between the quality attributes and the sub attributes, as well as attempting to connect software product properties with software quality attributes. As we have seen number of attributes reliability, us- ability, efficiency, maintainability and portability are common in number of models [2, 5, 7].

II. LITERATURE

A. Quality

Quality in the language of software engineering as discussed by Lewis [8] means ‘meeting requirements’ and ‘fitness for use’. This implies that the software meets the requirements of the users as stated in the requirements specification, and it does exactly what the user needs. Quality is considered a vital requirement of software products, a business essential, a competitive necessity, and a survival issue for the software industry [9].

B. Quality Management

Quality management entails all planned systematic activities and processes for creating, controlling and assuring quality. It isn’t just a job, but it is a habit that needs to be ingrained into a company’s culture [10].

C. Software Quality Assurance

Software Quality Assurance (SQA), is a well-defined, repeatable process that is integrated with project management and the software development lifecycles to examine the internal control mechanisms and promises adherence to software standards and procedures. The objective of the process is to assure conformance to requirements, reduce risk, assess internal controls and improve quality while conforming to the stated schedule and budget constraints. SQA is the planned and systematic approach to the evaluation of the quality of and adherence to software product standards, processes and procedures. It is a continuous process and assessment [11].

D. Quality Planning

This is the process where a specific quality plan is developed for particular project. It involves a selection of organizational standards that are specific to the software project in question and the development process to be used. It also specifies how the quality assessment process will be carried out.

E. Software Quality

Some large and complex systems typically involve hundreds or even thousands of people in their development over months or even years, and the systems are often to be operated under diverse, and sometimes unanticipated, application environments. One may argue that some systems are unnecessarily large and complex. Accordingly, such “fat software” may be caused by indiscriminately adding non-essential features, poor design, improper choices of languages and methodologies, which could be addressed by disciplined methodologies and return to essentials for “lean software”. Various QA techniques, including many of those covered in this book, can help produce high-quality, lean software.

F. Software Metrics

Software engineering data, like any other data, becomes useful only when it is turned into information through analysis. This information can be used to support investigating different themes in this thesis; thus forming a potential decision-support system. Such decisions can ultimately affect scheduling, cost and quality of the end product. However, it is worth keeping in mind that the nature of typical software engineering data is such that different machine learning techniques, might be helpful in understanding a rather complex and changing software engineering.

Machine learning is a sub-area within the broader field of artificial intelligence (AI), and is concerned with programming computers to optimize a performance criterion using example data or past experience. Within software engineering predictive modelling, machine learning has been applied for the tasks of classification and regression.

G. Software Quality Prediction

Software quality prediction is the process of utilizing software metrics which is measured and collected during software life cycle, to build classification models that are able to estimate quantitative values such as number of faults or a qualitative factor such as risk class.

Initially, the classification model is calibrated based upon training data, which are the software metrics and the risk class of a previously developed system release or similar project. This calibrated classification model can then be used to predict the quality factor of software modules currently under development. Thus, if one supplies the software metrics for a software module, one can predict whether it is likely to be fault-prone or not fault-prone.
H. Software Quality Prediction Techniques

i) Case-Based Reasoning:

Case-Based Reasoning (CBR) [11] uses computational intelligence for automated reasoning processes. A case based repository is made based on the previously made modules and according to this knowledge faults in new modules are predicted. It has a case-base library, which has all the cases already learnt using a predictive algorithm. Then a similarity function can be used to find the nearest neighbours. Then a solution algorithm is used to find the similar solution according to the case-based library built in the first step. The RippleDown Rule [12] is similar to Case-Based Reasoning in which expert knowledge is incrementally added to the system. It handles similar cases and tries to make rules to minimize misclassification.

ii) Bayesian Network:

Bayesian Belief Network (BBN), BayesNet[13] is a probabilistic graphical model used to represent set of variables.is composed of a Directed Acyclic Graph [14] having nodes as variables and edges as conditional independencies. Naive Bayes (NB) algorithm implements Naive Bayes classifier using estimator classes. The choice of numeric estimator precision values is based on study of the training data.

iii) Sequential Minimal Optimization:

Sequential Minimal Optimization algorithm uses John Platt's algorithm [15] for training a Support Vector Classifier (SVC) [15] replaces all missing values globally and converts nominal attributes into binary. Attributes are normalized by default and pairwise classification is used to work out the solutions to multi-class problems.

iv) Voted Perceptron:

Voted Perceptron algorithm globally restores all missing values and converts nominal attributes into binary attributes. This algorithm has been implemented by Freund et al. [16].

v) Instance Based Learning:

Instance Based Learning, IB algorithm [17] classifies according to nearest neighbour. Training instance nearest to the given test instance can be found using normalized Euclidean distance and the first instance found is used in case multiple instances have the same distance. IBk is another variation of IB1. In this algorithm attributes are normalized by default and appropriate values of k are chosen on the basis of cross validation.

vi) Nearest Neighbour Based Learning:

Nearest Neighbour [18] is a nearest neighbour based technique to classify modules as faulty and not faulty. The K Star algorithm [19] classifies a given data instance to the most similar instance which is provided in the training set. The similarity is calculated with the help of some distance function, normally Euclidean distance.

vii) Locally Weighted Learning:

Locally Weighted Learning [20] is a lazy learning algorithm that uses memory based learning and locally weighted linear regression. uses similarity functions to find the nearest neighbours and then by using weighted average it finds the nearby points.

viii) Attribute Selected Classifier:

Attribute Selected Classifier (Attribute Classifier) algorithm reduces dimensionality of training and test data before the data is passed on to a classifier [21].

ix) Regression Method:

Regression Method algorithm performs classifications using Regression. This algorithm is binarized and builds one regression model for each value [22]. Vote algorithm [23] combines classifiers using unweighted average of numeric predictions.

x) Decision Trees:

Decision Stump [24] algorithm uses combination of a boosting algorithm and regression. Decision Stump performs regression based on mean-squared error as well as classification based on entropy while treating missing values separately.48 is an extension of C4.5 decision tree algorithm used for weighted data. Alternating Decision Tree (ADTree) [25] is another type of decision tree consisting of two types of nodes: decision nodes to identify a predict condition and prediction nodes which declare a value to add to the score based on the result of the decision node. Naive Bayes Trees (NBTree) [26] uses decision tree with Naive Bayes [27] classifiers. Quinlan [28] simplified the decision tree to another type named Reduced Error Pruning Tree (REP). It builds a decision tree with least errors and every subtree is replaced by the best leaf node with minimum error and then a bottom up approach is used to prune the nodes until the error does not increase.

xi) Repeated Incremental Pruning:

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) implements a propositional rule learner, which was proposed by Cohen [29] as an optimized version of Reduced Error Pruning (REP). This algorithm can also be used to predict the fault proneness in modules. It gives more efficient results on large samples. Partial Decision Tree, PART [30] is also a decision tree based algorithm to classify modules as faulty and not faulty. It builds a partial C4.5 decision tree in each iteration and makes the “best” leaf into a rule. Classification is done based on these rules. It combines two algorithms C4.5 and Repeated Incremental Pruning to Produce Error Reduction (RIPPER), but does not perform global optimization. UserClassifier [31] builds user defined decision trees. It is an Interactive machine learning techniques which gives the liberty to users build classifiers.

xii) Simple Logistic:

Simple Logistic algorithm is a combination of Linear Logistic Regression and Tree Induction models [32]. It
predict both nominal classes and numeric values. To give more accurate prediction in numeric values it creates trees with linear regression functions at the leaves.

**xiii) Trees:**

Logistic Model Trees [33] algorithm builds a tree dealing with binary and multi-class target variables, numeric and missing values. Random Tree [34] algorithm builds a tree that takes K randomly chosen attributes at each node. Pruning cannot be achieved using this algorithm. Random Forest [35] consists of many decision trees and outputs the class that is the mode of the classes output by individual trees.

**xiv) Conjunctive Rule:**

Conjunctive Rule Learner (ConjunctiveRule) algorithm can predict nominal class labels based on rules constructed earlier by the learner. The rule consist of attribute values ANDed together that result in the class it belongs to [36].

**xv) Decision Table:**

Decision table technique [37] uses majority classifier to predict modules as faulty and non-faulty.

**xvi) ZeroR and OneR:**

Simplistic rule classifiers [38] are used to classify a module as faulty or non-faulty based on the extracted rules. It uses rules based on different values of attributes. The rule with the maximum similarity is chosen to classify a new module. The simplest of these rule classifiers is the majority class classifier, called ZeroR [38]. This classifier takes a look at the target attribute and will output the value that is most commonly found in that attribute. Another modified form of this algorithm is OneR [38], in which an attribute is selected and then the best rule for that value of the attribute predicts the faultiness of the module.

**I. Challenges Inhibiting Implementation of Software quality Assurance**

Software companies frequently face many difficult challenges in their attempt to deliver high-quality software and strive to attain client satisfaction [15].

From different literatures, possible factors that can impair software quality management include: impatient management, strict deadlines, developer ego, extra cost required (e.g., for the purchase of tools), bureaucracy, inadequate tools that can help to automate the process, low level of acquaintance and knowledge of the process, lack of organizational training on quality standards, inexistent framework for quality management in the organization, disapproval by top management, contrary beliefs and opinion, and previous futility of the process.

**J.** The following Qualitative Parameters are used for comparison of our results with the other approaches.

**i) Performance Parameters:**

These are the performance parameters on which our algorithm accuracy, efficiency and complexity would be measured.

**a) Accuracy**

- **Accuracy:** is the proportion of the total number of predictions that were correct
- **Error rate (misclassification rate)= 1 – AC**
- **Accuracy explicitly takes into account the classification of negatives, and is expressible both as a weighted average of Precision and Inverse Precision and as a weighted average of Recall and Inverse Recall:**

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

**b) Precision**

- **Precision or Confidence (as it is called in Data Mining) denotes the proportion of Predicted Positive cases that are correctly Real Positives.**
- **This is what Machine Learning, Data Mining and Information Retrieval focus on, but it is totally ignored in ROC analysis.**
- **It can however analogously be called True Positive Accuracy (tpa), being a measure of accuracy of Predicted Positives in contrast with the rate of discovery of Real Positives (tpr). Precision is defined below:**

\[
\text{Precision} = \text{tpa} = \frac{TP}{TP + FP} = \frac{TP}{TP + FP} = \frac{TP}{TP + FP}
\]

**c) Recall**

- **Recall or Sensitivity (as it is called in Psychology) is the proportion of Real Positive cases that are correctly Predicted Positive.**
- **This measures the Coverage of the Real Positive cases by the +P (Predicted Positive) rule. Its desirable feature is that it reflects how many of the relevant cases the +P rule picks up.**
- **It tends not to be very highly valued in Information Retrieval (on the assumptions that there are many relevant documents that it doesn't really matter which subset we find, that we can't know anything about the relevance of documents that aren't returned).**

\[
\text{Recall} = \text{Sensitivity} = \text{tpr} = \frac{TP}{TP} = \frac{TP}{TP + FP} = \frac{TP}{TP + FP}
\]

**d) F-Score/Measure**

- **The F-Measure computes some average of the information retrieval precision and recall metrics.**
- **Each cluster is considered as if it were the result of a query and each class as if it were the desired set of documents for a query.**
- **We then calculate the recall and precision of that cluster for each given class.**
- **The F-measure of cluster j and class i is defined as follows:**

\[
F_{ij} = \frac{2 \times \text{Recall}(ij) \times \text{Precision}(ij)}{\text{Recall}(ij) + \text{Precision}(ij)}
\]
F-measure is computed using the harmonic mean:
  
  - Given n points, x1, x2, ..., xn, the harmonic mean is:
    \[ \frac{1}{H} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{x_i} \]  
  
  So, the harmonic mean of Precision and Recall is
  \[ \frac{1}{F} = \frac{1}{\frac{1}{R} + \frac{1}{P}} = \frac{P \times R}{P + R} \]  

ii. Comparison Parameter’s

These parameters help us in comparing our algorithm with other algorithms and techniques which are used for software quality prediction.

a. Confusion Matrix

- A table of confusion (sometimes also called a confusion matrix).
- Is a table with two rows and two columns that reports the number of false positives, false negatives, true positives, and true negatives.
- This allows more detailed analysis than mere proportion of correct guesses (accuracy).
- Accuracy is not a reliable metric for the real performance of a classifier, because it will yield misleading results if the data set is unbalanced (that is, when the number of samples in different classes vary greatly).

<table>
<thead>
<tr>
<th>Actual Condition</th>
<th>Predicted Condition</th>
<th>Total Population</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Predicted Condition (Positive)</td>
<td>Predicted Condition (Negative)</td>
</tr>
<tr>
<td>Actual Condition (Positive)</td>
<td>A: True Positive</td>
<td>B: False Negative</td>
</tr>
<tr>
<td>Actual Condition (Negative)</td>
<td>C: False Positive</td>
<td>D: True Negative</td>
</tr>
</tbody>
</table>

### III. PROBLEM FORMULATION

We are trying to implement a hybrid fuzzy K-means algorithm with Support Vector Machines for Software Quality Prediction. Even when the same data mining approach is applied to the same data set, the results may be different since different researchers use different feature extraction and selection methods. It is important that the data is pre-processed before data mining is applied so that redundant information can be eliminated or the unstructured data can be quantified by data transformation. Theoretical guidelines for choosing appropriate patterns and features vary for different problems and different methodologies. Indeed, the data collection and pattern generation processes are often not directly controllable.

Therefore, utilizing feature extraction and selection is the key to simplifying the training part of the data mining process and improving the performance without changing the main body of data mining algorithms.

### A. Proposed Methodology

This section describe different steps to be carried out to achieve the proposed objectives.

The steps are as follows:

**Step 1: Data Collection**

In this step data set is first extracted from the Promise Repository of NASA (http://promise.site.uottawa.ca/SERpository/datasets-page.html) and converted to a readable format.

**Step 2: Preparing the data set**

In the second step we’ll be managing the data set, i.e. applying different rules and taking out redundant data, the data which is not used or some unnecessary data.

**Step 3: Calculate The Euclidian Distance of each point to centroid**

The next step after the dataset used in the research is prepared is used and Euclidian distance of each data point is calculated and a label is allotted to each data point.

**Step 4: Calculate the probability of each point in the cluster using Fuzzy likelihood**

Then Fuzzy K – Means algorithm is applied on the data points and labels, after that a cluster is allotted to the data points depending on the minimum distance from a specific cluster and the probability of each point to be in a cluster.

**Step 5: Select each Cluster as Feature and load to SVM for training**

Once the data and the member functions are calculated then we apply SVM for the training as well as maximization of the hyperplane.

**Step 6: Training and Testing of data**

...
Once the iteration starts the Fuzzy K – Means algorithm calculates the membership of each data point to a specific cluster then keeps on applying it on the whole data set as a part of machine learning. The machine learning is applied in the ratio of 70 to 30. In which the 70% is the training of the SVM and 30 % is the prediction.

**Step 7: Algorithm is trained for already known training labels**

The algorithm is trained as well as implemented but once the process of training is complete the algorithm starts predicting the values those values are checked for accuracy.

**Step 8: Evaluation and validation of the proposed algorithm**

In this final step the results such as accuracy, precision, T1 error and T2 error are checked and compared with the other techniques used by different researchers and methodology used by different.

---

**B. Fuzzy K-means Algorithm**

The traditional k-means clustering algorithm suffers from serious drawbacks like difficulty in finding the correct method for the cluster initialization, making a correct choice of number of clusters (k). Moreover k-means is not efficient for overlapped data set. There have been many methods and techniques proposed to address these drawbacks of k-means. Fuzzy k-means is one of the algorithms which provide better result than k-means for overlapped dataset. Fuzzy k-means was introduced by Bezdek. The fuzzy k-means algorithm is also called fuzzy c-means. Unlike naive k-means which assigns each data point completely belonging to one cluster, in fuzzy c-means each data point has the probability of belonging to a cluster. This allows data point of data set X to be a part of all centers of set C. For example, points on the edges of the clusters might belong to a cluster with lesser degree than those data points belonging to the same cluster at its center. This algorithm is mainly used for datasets in which the data points are between the centers.

**i) Algorithm of Fuzzy K – Means**

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 centers.
2) Calculate the distance between each data point and cluster centers.
3) Calculate fuzzy membership function of each point in each cluster.
4) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers, using the below formula

$$d(x, v) = \frac{e^{-\|x-y\|^2}}{\sqrt{2\pi\sigma}}$$

5) Combine the fuzzy membership to assign clusters to each sample point.
6) Recalculate the new cluster center using:

$$v_i = \frac{\sum_{j=1}^{n} x_j}{\text{number of data points in } i^{th} \text{ cluster}}$$

7) where, $c_i$ represents the number of data points in $i^{th}$ cluster.
8) Recalculate the distance between each data point and new obtained cluster centers.
9) If no data point was reassigned then stop, otherwise repeat from step (3).

---

**Fig. 3 Flow chart of Methodology**

---

**Fig.4 Fuzzy K-means algorithm**
V. RESULTS

As observed from figure 5, the clusters are not separated by visible boundaries and thus decreasing the cluster quality of the given feature. Hence this feature is not selected for our svm training and later on the similarly all features are selected for better prediction through SVM.

As observed from figure 3, the data is clustered and have definite visible boundaries using our proposed Density based K-means Algorithm. These clusters are utilized for selection of features.

Fig. 5 Data into SVM for Machine learning

Fig. 6 Showing distribution of points after clustered data for Feature 1 and 2

Fig. 7 Showing data points using features 1&5

Fig. 8 Showing Confusion Matrix
Figure 6 explains the values of the TP, FP, TN and FN obtained from our prediction algorithm i.e. hybrid Density based Fuzzy K-means aided SVM. As observed the TP and TN values are quite more as compared to the false values. This establishes our efficacy of the algorithm. Where value for TP= 2580, FP=130, TN=1491, FN=190.

Table 2 Comparing Performance of T-SVM and Proposed Algorithm

<table>
<thead>
<tr>
<th>Methods</th>
<th>CCR (%)</th>
<th>T1error</th>
<th>T2error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Algorithm</td>
<td>92.7124</td>
<td>1.3414</td>
<td>3.5254</td>
</tr>
<tr>
<td>SVM</td>
<td>89.00</td>
<td>2.33</td>
<td>8.67</td>
</tr>
<tr>
<td>T-SVM</td>
<td>90.03</td>
<td>2.11</td>
<td>7.86</td>
</tr>
<tr>
<td>PCA+SVM</td>
<td>89.07</td>
<td>2.06</td>
<td>8.87</td>
</tr>
<tr>
<td>QDA</td>
<td>85.49</td>
<td>7.37</td>
<td>7.14</td>
</tr>
<tr>
<td>PCA+QDA</td>
<td>86.53</td>
<td>4.90</td>
<td>7.52</td>
</tr>
<tr>
<td>PCA+CART</td>
<td>83.02</td>
<td>9.59</td>
<td>6.41</td>
</tr>
</tbody>
</table>

As observed the performance of our proposed algorithm on the same data set is better in terms of CCR, Type 1 error and Type 2 error. This can be attributed to the fact that our algorithm includes the application of feature selection which have role in feature selection thereby increasing the accuracy.

VI. CONCLUSION AND FUTURE SCOPE

A. Conclusion

Software Quality is an important area of research and has drawn significant interest recently. Software quality prediction is helpful for better utilization of resources and to minimize error. Hence it helps in reducing the cost by early assessment of faults. It also helps in planning more thorough tests for modules that might have defects. Many organizations and institutions are involved in the study of software quality prediction techniques and much work has been done to develop and refine models. These models have also been compared and benchmarked using different data sets. But still no study has concluded a best technique that always outperforms other techniques overall. This paper proposed a novel Density based Kmeans algorithm aided Support Vector Machines for Software Quality Prediction. The Kmeans was used for feature selection and the selected featured were utilized by SVM for prediction of software quality. The results were analysed in terms of correctly predicted and accuracy, precision, recall and Fscore values.

B. Future Scope

In future other algorithms can be applied for prediction of software quality. The software quality prediction dataset can be changed and verified for accuracy. Also other hybrid algorithm can be developed using Neural Networks and Support Vector Machines. Hierarchical Clustering can also be utilized for feature selection.


