CHAPTER 5

SOFTWARE REUSABILITY PROCESS THROUGH MACHINE LEARNING TECHNIQUES

5.1 INTRODUCTION

Software reusability is the process of developing or upgrading the software components using existing or previously developed software components. A good software reusability approach increases the productivity, reliability and quality of the software being developed. In parallel, it also reduces the overall development cost and time. In the initial level an investment is needed to form a repository of reusable software components. This chapter deals with the development of reusable components to improve the quality, minimizes the amount of work needed for developing future components and thus ultimately reduces the risk of developing new components which depends on the reusable software repository.

5.2 SOFTWARE REUSABILITY

Among the various software components, the simplest form of reusability is exhibited by subroutines or functions. A part of code is organizing into layers using modules or namespaces. The objects of the object-oriented programming offer an advanced form of reusability. But it is highly difficult to measure the level or the score of reusability. The capacity to build larger software modules from smaller components and to identify the similarities among those components mainly decides the reusability. Reusability is an essential characteristic of a software component. It requires compulsory
management of building, packaging, configuration, installation, deployment and maintenance issues and if these are not managed properly the software may appear to be reusable from the design point of view but practically the software components cannot be reused. Reuse is a process of conceiving solution to a problem using the solutions already available for the sub-problems.

![Diagram](image)

**Fig. 5.1 Flow of Reusability Process**

The process of reuse can be completed by following the six major steps (Kyo C. Kang et al. 1992) is shown in Fig. 5.1. In each step preparation for the next phase is carried out. Initially a reuse plan is developed by studying the given problem and available solutions to it, after that a solution structure is identified. Next the structure of the solution is reconfigured to improve the reusability in the upcoming phases. Then the existing reusable components are acquired, instantiated and/or modified. The reused
software components are then integrated together with the newly developed components and finally the complete product is validated and evaluated.

The software reusability has a several benefits (Ian Sommerville, 2004) and they are summarized below.

The reusable software components are tested and tried in already working systems and when they are used to build new software module the reliability is increased. During the initial usage of the software components the bugs in the design and implementation are identified and removed. So, when reused the chance of failures are greatly reduced. When the software components are reused it reduces the uncertainty in the cost estimation of the software. This is an essential factor as it reduces the error in the overall project cost estimation which is particularly true when majority of the software components are built from reusable components. It reduces the requirement of coding specialist doing the same work on various software projects. Instead the specialist can develop reusable software components that can be used whenever similar problem arises during the software development life cycle.

Some standard or conventional software components can be made reusable so that there will not be change in the functioning or user interface of them. To mention few example the menus in a user interface can be developed and maintained as a reusable software component so that a common menu format can be presented to the user. Building and introducing the system to the market as early as possible is more important rather the reducing the overall development cost. The software reusability can increase the overall production speed as both the development and validation time required for most of the component is greatly reduced.
The IT professionals have started realizing the importance of software reusability to manage the situation called as software crisis (Smith et al. 1998, Victor R. Basili 1989, Victor R. Basili and Dieter Rombach 1988, Boehm and Ross 1989, Boehm 1988). As discussed previously software reuse increase the productivity and quality (Boehm 1999, William B. Frakes and Christopher J. Fox, 1996) of the software product. The concept of software reuse introduced in 1968 which introduced new scope for software design and development (Gomes and Bento, 2001). The software development people are slowly drifting towards the software reusability by which new software system can be built using the existing systems without much effort and time.

As the software reuse poses many benefits there has been much discussion on the topic of reusability including the research directions for software reusability. The software developers are following many of the software reuse approaches in software designs, patterns, and templates and also in searching, matching and modeling tools. The entire software industry is moving towards large scale software re-usage. So, at this juncture it is essential to predict or identify whether a software component is reusable or not while it is being developed and maintained.

5.3 PROPOSED SOFTWARE REUSABILITY MODEL

A model for software reusability is proposed after considering the various metrics along with the three classifiers namely SVM, GMM and BN. This proposed model is further applied to mixed kernel models to improve the performance for software reusability.
5.3.1 List of Metrics

The metrics used in this work are derived based on the answers given by industry people for a set of questionnaire. For each of the software project data points were coded in to either categorical or numerical data and can be divided in to two categories namely state metrics and control metrics. The state metrics composed of attributes for which the company has no control such as the size and application domain. The control metrics composed of attributes such as type of reusability approach followed, modification carried out to the existing development cycle, commitments from the management. The list of the metrics representing software reusability is given in Table 5.1.

5.3.2 Feature Selection

Information theory techniques can be used for feature selection in time series prediction or pattern recognition. These techniques focus on maximizing the mutual information between the input and output data. But this procedure computational complexity is high as the joint entropy is to be calculated. To calculate joint entropy, the joint probability distribution has to be estimated. To eliminate this computational effort, the feature selection task can be accomplished using the principle of minimum redundancy and maximum relevance. This method maximizes the mutual information between the input and output data indirectly with a low computational cost.
<table>
<thead>
<tr>
<th>S. No.</th>
<th>Name of the Attribute</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Project ID</td>
<td>Project identifier</td>
</tr>
<tr>
<td>2</td>
<td>Software Staff</td>
<td>Number of staffs involved in software development</td>
</tr>
<tr>
<td>3</td>
<td>Overall Staff</td>
<td>Total No. of staffs involved in the software project</td>
</tr>
<tr>
<td>4</td>
<td>Software Production</td>
<td>Represents the similarity of the software with other</td>
</tr>
<tr>
<td>5</td>
<td>Software and Product</td>
<td>Represents either the software is embedded in a product or process</td>
</tr>
<tr>
<td>6</td>
<td>SP maturity</td>
<td>Represents certification or standard of the S/W</td>
</tr>
<tr>
<td>7</td>
<td>Application Domain</td>
<td>Represents the domain of the Software product</td>
</tr>
<tr>
<td>8</td>
<td>Type of Software</td>
<td>Represents the nature of the software i.e. real time, embedded, or other.</td>
</tr>
<tr>
<td>9</td>
<td>Size of base line</td>
<td>The size of software module on which reusability was imposed</td>
</tr>
<tr>
<td>10</td>
<td>Development approach</td>
<td>Represents the analysis and design approach used.</td>
</tr>
<tr>
<td>11</td>
<td>Staff Experience</td>
<td>Represents the experience of the staff in code development</td>
</tr>
<tr>
<td>12</td>
<td>Reuse approach</td>
<td>Represents the coupling between the reusable and other components of the software</td>
</tr>
<tr>
<td>13</td>
<td>Work-products</td>
<td>Represents the type of components reused.</td>
</tr>
<tr>
<td>14</td>
<td>Domain Analysis</td>
<td>Tells whether or not domain analysis was performed</td>
</tr>
<tr>
<td>15</td>
<td>Origin</td>
<td>Represents whether components are developed from scratch or re-engineered.</td>
</tr>
<tr>
<td>16</td>
<td>Independent team</td>
<td>Tells whether development and reuse team are same or not.</td>
</tr>
<tr>
<td>17</td>
<td>When assets developed</td>
<td>Tells when the reusable components were developed</td>
</tr>
<tr>
<td>18</td>
<td>Qualification</td>
<td>Tells whether reusable components undergo a qualification process or not.</td>
</tr>
<tr>
<td>19</td>
<td>Configuration management</td>
<td>Tells whether the reusable components have a configuration management and change control or not.</td>
</tr>
<tr>
<td>20</td>
<td>Rewards policy</td>
<td>Tells whether there is reward to staff if reusability is adopted.</td>
</tr>
<tr>
<td>21</td>
<td>Assets</td>
<td>Number of reusable assets in the software repository.</td>
</tr>
</tbody>
</table>
But still the combinatorial optimization problem i.e. the check for all possible combination of features requires high computational effort. Because of these high computational costs, a simple and efficient method using incremental search which can produce a quasi-optimal solution was proposed in the previous literatures. With this background, to reduce the computational cost the combinatorial optimization was performed using the Genetic Algorithms. The algorithm output a vector of indices of the features that forms the optimal set of features. Here the order of features has no relation with their importance.

The parameters of the Genetic Algorithm are fixed as follows:

Size of the population - 100
Maximum Number of generation - 50
Crossover Probability - 0.7
Mutation Probability - 0.4
Crossover strategy - Random single point

In the initial stage, the chromosome population is generated randomly. The number of chromosome in the initial population decides the rate of convergence. If the population is larger the problem converges slowly else if the population is small, it converges quickly and explores only a smaller portion of the search space. The optimal feature selected by the genetic algorithm consists of the following attributes namely Software Production, Software and Product maturity, Application Domain, Type of Software, Size of base line, Development approach, Reuse approach, Work-products, Domain Analysis, Origin, when assets developed, Rewards policy, and Assets. Out of 29
attributes (28 attributes and 1 class label) of the dataset 14 attributes were selected which constitute the optimal feature set.

The feature set was then used to train the classifiers. The fitness function of the genetic algorithm for various Iterations is shown in Fig. 5.2.

![Fitness function over iterations](image)

**Fig. 5.2 Evaluation of Fitness Function Over Iterations**

### 5.3.3 Classifiers for Software Reusability

The prediction of success or failure is a kind of binary classification problem. The three classifiers used in this work are Support Vector Machine, Bayesian Networks, and Gaussian Mixture Model. All three classifiers don’t have built-in feature selection ability and are commonly used in the machine learning applications. SVM is a mathematical model which has a supervised learning algorithm capable of analyzing data and identifying patterns in it. The classification process is done in two steps one
is training phase and the other one is testing phase. The labeled feature vectors are fed as input during the training time and the data to be classified is given in the testing phase. The accuracy of classification depends on the efficiency of the trained model. In this work the support vector machine with Gaussian kernel is used and it is denoted by the Equation (5.1),

$$K(x, y) = \exp\left(-\frac{||x - y||^2}{\sigma^2}\right)$$

(5.1)

where $||x - y||^2$ may be recognized as the squared Euclidean distance between the two feature vectors. $\sigma^2$ is a free parameter. An equivalent, but simpler, definition involves a parameter $\gamma = \frac{1}{2\sigma^2}$.

Bayesian Networks (BNs) are probabilistic graphical models used to represent knowledge regarding an uncertain domain. Each node in the graph denotes a random variable and the edges connecting the nodes denote the dependency among the respective random variables. The dependencies in the graph are calculated using statistical and computational methods. Therefore, the concepts of BNs are combined from graph, probability, computer science and statistical theories. Bayesian Network can be used even when the value of some of the attributes are missing (Oswaldo Ludwig and Urbano Nunes, 2010).

The GMM is a useful supervised learning classification algorithm which can be used for classification of $N$-dimensional dataset. During the training phase GMM is built for each class of data. In this work two GMM has to be constructed to fit the defective and non-defective data samples. There will not be any interactions between GMM of
different classes. At the classification phase the unknown-class data is given as input to GMM of each class. The predicted class is the one associated with the GMM with the maximum probability.

5.4 EXPERIMENTS AND RESULTS

In this training dataset, out of 100% records nearly 75% records were belonging to software components that can be reused i.e. the class label for those records were marked as ‘1’ and 25% records belong to non-reusable category. This causes imbalanced data. This problem is crucial in classification or clustering because it is needed to maximize the instances of recognizing the minority class. Two methods for dealing with class imbalances are: oversampling and downsizing. Oversampling consists of re-sampling the small class at random until it contains as many examples as the other class. Downsizing consists of the randomly removed samples from the majority class population until the minority class becomes some specific percentage of the majority class.

Using the Synthetic Minority Over-Sampling TTechnique (SMOTE) the non-reusable components record was made 40%. The SMOTE is an oversampling method that works by creating synthetic samples from the minor class instead of creating copies. The algorithm selects two or more similar instances using a distance measure and perturbing an instance one attribute at a time by a random amount within the difference to the neighboring instances.
5.4.1 Dataset

For an IT concern, which has not implemented software reuse concepts, there exists a chance to develop reusable software components from the scratch or they can identify reusable components from their software repository. The prediction or identification of reusability of the software components present in the repository is accomplished by using a set of software metrics and a mathematical model like SVM or ANN. This work is focused on building such a mathematical model to predict the reusability of the software. In this work, PROMISE software engineering repository data is used which contains 29 attributes describing the reusability of the software (Menzies et al. 2016). Among the various software engineering research datasets, the PROMISE repository is a specialized research dataset repository. The majority of the attributes are of categorical type. So, it is essential to encode the categorical features using one-of-K coding scheme before giving them as input to the SVM for training.

5.4.2 Results and Discussion

The accuracy of all the classifiers used for the prediction is presented in the Fig. 5.3. The predictive models constructed using the SVM are computationally more expensive when compare to the GMM and Bayesian Network. As the number of records is limited it is obvious that the SVM outperforms the GMM and Bayesian Networks. If the number of records available is huge the GMM can perform much better as the estimates of the model parameter are better. In the case of SVM the availability of more data will not change the position of the support vectors. The experiments are conducted in this work to exhibit the suitability of the classifiers for a given optimal feature set where the performance of all the classifiers are almost same.
5.4.3 Certain Improvements of the SVM Classifier

The final experiment is conducted on feature data set consisting of feature points within two-dimensional space. It is assumed that two distinct regions of points are separated by a parabolic boundary, where vector points of class (success) are below and failure are above the separating curve. The linear nor the quadratic kernels alone are able to resolve the SVM classification problem. The Fig. 5.4 and Fig. 5.5 below illustrates that feature points of both classes are scattered onto overlapping regions in the quadratic kernel space. It indicates that for this case the sole utilization of the quadratic kernel is not enough to resolve the classification problem. The proposed solution composes mixed kernels out of base kernel functions. This is perhaps similar to how a vector can be composed out of its coordinate base vectors or a function can be assembled in functional Hilbert space. A new mixed kernel out of the linear and the quadratic kernels $K_1$ and $K_2$ is formed.
The mixed Kernel is given in Equation (5.2),

\[ K_M(x, z) = \rho_1 K_1(x, z) + \rho_2 K_2(x, z) \]  \hspace{1cm} (5.2)

The results demonstrate that the mixed kernel successfully solves the classification problem even though each individual base kernels are insufficient on its own. In experimentation, the actual values of the mixture weights \( \rho_1 \) and \( \rho_2 \) are not critical. The weights are chosen according to how much each individual base kernel on its individual aligned with the raw classification matrix \( Y = y^T y \) composed out of the classifier vector 'y'.

Alignment is based on the observation that a perfectly selected kernel matrix \( K_{i,j} = K(x_i, x_j) \) would trivially solve the classification problem if its elements are equal to the \( Y \) matrix. In that case choosing an arbitrary vector \( x_i \) as support the kernel \( K(x_i, x_i) \) would give 1if \( x_i \) and \( x_i \) are in the same category and -1 otherwise.

Similar to the concept of correlation, kernel alignment between two kernels is defined by the expectation over the feature space, resulting in matrix products. The Kernel alignment is given in Equation (5.3),

\[ \rho(K_1^c, K_2^c) = \frac{< K_1^c, K_2^c >}{\sqrt{< K_1^c, K_1^c > < K_2^c, K_2^c >}} \]  \hspace{1cm} (5.3)

with the product\( < A > = T_r[A^T B] \). The expectation is take assuming centered \( n \times n \) kernel matrices according to \( K^c = \left(I - \frac{11^T}{n}\right) K \left(I - \frac{11^T}{n}\right) \).
Fig. 5.4 Parabolic Feature Set (Mixed Kernel SVM)

Fig. 5.5 Non-Linear Separable in Quadratic Kernel Coordinates
The experiments proved that among all of the existing kernels the mixed kernel has the best performance on the chosen dataset for classification, and the kernel mix of polynomial and quadratic kernel has greatest performance than the RBF kernel. The Fig. 5.6 shows the comparison between RBF Kernel SVM and Mixed Kernel SVM. According to the theory of structural risk minimization, to get structural risk minimization of study machine, the advantage of both the ability of study and generalization must be utilized. If choose a Poly kernel and Quadratic kernel to make up a new kernel, an improvement on SVM classification accuracy was witnessed. The Fig. 5.6 shows the improved accuracy of the mixed kernel when compared to the standard RBF kernel. As the size of the dataset is varied the mixed kernel performs better when compared to the RBF kernel. It is observed that dataset with 14 attributes the accuracy of the mixed kernel was 2.66% greater than RBF kernel’s accuracy (Mixed kernel - 94.06%, RBF kernel - 91.4%).
5.5 SUMMARY

This chapter presents a method to predict the success or failure of a software project based on the software metrics which are related to the reusability of the software components. The projects developed in the various domains of the software industry uses an object oriented or procedural approach for software development and are expected to be successful. But around one-third of the software projects were classified as failed because of lack of reuse processes, and non-modification of non-reuse processes. The main cause for the failure of software components are due to the lack of commitment from the top management people or lack of awareness of the importance of these factors. All the projects that are classified as success implements the concept of reusability in their respective project development. This is reflected from the attributes related the reusability of the software modules. Among the three classifiers used for the prediction, the SVM with Gaussian kernel performed better than the other classifiers as the Gaussian kernel perfectly fits for these kind of training data. A mixed kernel combining the Poly kernel and Quadratic kernel to make up a new kernel make an improvement on SVM classification accuracy.