3. GENETIC K-MEANS CLUSTERING FOR SOFTWARE QUALITY ESTIMATION

This chapter is devoted to studying the potential of clustering – a data mining technique, to solve the complex problems encountered in software engineering.

3.1 CLUSTERING METHODS

Clustering is a division of data into groups of similar objects. Each group called a cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups Berkhin[11]. These clusters correspond to hidden patterns, and the search for clusters is termed “unsupervised learning”. They presented the following categorization of clustering algorithms.

Hierarchical Methods

Agglomerative Algorithms

Divisive Algorithms

Partitioning Methods

Relocation Algorithms

Probabilistic Clustering

K-medoids Methods

K-means Methods

Density-Based Algorithms

Density-Based Connectivity Clustering
Density Functions Clustering

Grid-Based Methods

Methods Based on Co-Occurrence of Categorical Data

Constraint-Based Clustering

Clustering Algorithms Used in Machine Learning

  Gradient Descent and Artificial Neural Networks

  Evolutionary Methods

Scalable Clustering Algorithms

Algorithms for High Dimensional Data

  Subspace Clustering

  Projection Techniques

  Co-Clustering Techniques

3.1.1 Hierarchical Clustering

Hierarchical clustering builds a tree of clusters known as a dendrogram. Every cluster node contains child clusters; sibling clusters partition points covered by their common parent. Hierarchical clustering methods are categorized into agglomerative and divisive. Agglomerative clustering starts with a single point clusters and proceeds by successively merging them. Divisive clustering, on the other hand, starts with one cluster comprising all the data points, and recursively splits the most appropriate cluster. The advantages of this technique include:

  - Applicability to multiple attribute types
Accommodative to a wide variety of similarity or distance measures

Flexibility

The disadvantages of hierarchical clustering, according to Berkhin[11] are:

- Lack of scope for improvement of the intermediate clusters generated
- Vagueness in the termination criteria

A linkage metric is a proximity measure used to generalize the distance between individual points to a distance between subsets. Major linkage metrics include single link, average link and complete link.

### 3.1.2 Partitioning Relocation Clustering

In this category, certain greedy heuristics are used in the form of iterative optimization. Unlike hierarchical methods, relocation algorithms offer scope of improvement of the individual clusters. This lends itself to the possibility of high-quality clusters. One approach in this category is to assume that the data comes from a mixture of several populations whose distributions are to be discovered. This is called “probabilistic clustering”. The major advantage is the concise cluster representation that allows inexpensive computations.

The other approach starts with an objective function. This approach resorts to the usage of cluster representatives in an attempt to reduce the overhead associated with the computation of pair-wise similarities or distances. These algorithms are subdivided into k-medoids and k-means, depending upon the choice of the representative. K-medoid is the most appropriate data point
within a cluster that represents the cluster. A cluster is represented by its centroid in the k-means case. Partitioning Around Medoids (PAM), Clustering Large Applications (CLARA) are examples of k-medoid methods.

### 3.1.3 Density Based Partitioning

An open set in the Euclidean space can be divided into a set of its connected components. The concepts of density, connectivity and boundary are required to implement the above idea for partitioning. Density-based algorithms are capable of discovering clusters of arbitrary shapes. This gives such algorithms an edge over relocation clustering algorithms, but there is an important drawback: the lack of interpretability. There are two major approaches. The first approach pins density to a training data point, and algorithms in this category include DBSCAN and OPTICS. The second approach pins density to a point in the attribute space, and DENCLUE is an algorithm in this category.

### 3.1.4 Grid-Based Methods

Grid-Based methods are based on the idea of inheriting the topology from the underlying attribute space. A segment is defined as a direct Cartesian product of individual attribute sub-ranges. The elementary segment corresponding to single-value sub-ranges is called a unit. While data partitioning is induced by points membership in segments resulting from space partitioning, space-partitioning is based on grid characteristics accumulated from input data. The main advantage is the independence of data ordering achieved by this approach. According to Berkhin[11] density-based methods
tend to work well with numerical attributes, while grid-based methods work well with multiple attribute types.

3.1.5 Dealing Categorical Data

When considering transactions, defined as variable sized entities comprising a finite set of elements called items from a common item universe, two random transactions tend to have very few items in common. Similarity in such cases is measured by Jaccard coefficient. Such data are categorized by high dimensionality, significant amount of zero values and a small number of common values between two objects. Conventional clustering methods based on similarity measures do not work well. Different clustering methods based on the idea of co-occurrence of categorical data have been developed. RCACD (Robust Clustering Algorithm for Categorical Data) shares many features of CURE which is a hierarchical clustering algorithm. RCACD defines a neighbor of a point $x$ as a point $y$ such that $\sin(x,y) \geq \theta$ for some threshold $\theta$ and proceeds to a definition of links $(x,y)$ as the number of common neighbors of $x$ and $y$.

3.1.6 Constraint Based Clustering

The framework for constraint-based clustering is introduced by Tung et al.[84][85]. The taxonomy includes:

- Constraints on individual objects
- Parameter constraints
- Constraints on individual clusters described in terms of bounds on aggregate functions
An existential constraint is a bound from below on a count of objects of a certain subset in each cluster. The most frequent requirement is to bind the number of cluster points from below. K-means algorithm, unfortunately, often ends up with a number of very small clusters. Some modifications have been suggested on the k-means objective function by Bradley et al.\[13\] to incorporate lower limits on cluster volumes. Clustering 2D spatial data in the presence of obstacles is an important constraint-based clustering application. The Clustering with Obstructed Distance (COD) algorithm given by Tung et al.\[84][85\] deals with this problem.

3.1.7 Evolutionary Methods

The Simulation of Near-optima for Internal Clustering Criteria (SINICC) algorithm given by Brown and Huntley\[14\] uses simulated annealing in the context of partitioning. Genetic algorithms are also used in cluster analysis. The Genetically Guided Algorithm (GGA) given by Hall et al.\[33\] and the work of Sarafis et al.\[75][76\] in applying Genetic Algorithm (GA) in the context of k-means objective function are notable improvements in this area.

3.1.8 Scalability Issues

One of the big challenges facing clustering algorithms is scalability. Attempts to extend clustering to very large databases can be classified as:

- Incremental mining – handles one data point at a time and then discards it.
• Data squashing – scanning the data to obtain certain summaries and using the summaries instead of the original data.

• Reliable sampling – using some selected samples instead of all of the data.

3.1.9 Dimensionality Reduction

The objects in data mining often have hundreds of attributes, and clustering in such high dimensional spaces presents enormous difficulties. Dimensionality reduction techniques attempt to address this problem and can be classified into:

• Attribute transformations – simple functions of existent attributes are computed and used. The Principal Component Analysis (PCA) is popular in this arena

• Domain decomposition – divides data into subsets called canopies using some similarity measure.

3.2 AUTOMATIC CLUSTERING OF SOFTWARE SYSTEMS USING A GENETIC ALGORITHM

Doval et al.[20] describe a novel method for automatic clustering of software systems. Designers use directed graphs to represent the inter-relationships between modules. Such graphs are called Module Dependency Graphs (MDGs). They address the problem of partitioning the MDG to find “good” MDG partitions.
MDG’s are often used to represent the structure of complex systems. Modules are represented as nodes, and their relationships are represented as edges. The complexity of the MDG of a modest sized system can itself be quite daunting and therefore, finding good partitions of the MDG where each partition can be relatively simpler compared to the entire system can be extremely useful. The huge dimensionality imposed by MDG’s necessitates the use of a genetic algorithm to perform clustering. The Figure 3.1 shows an MDG and its partition.

Figure 3.1 An MDG and it’s Partition
3.2.1 Genetic Algorithms

Genetic algorithms belong to the class of evolutionary computations that mimic the natural process of evolution to arrive at a solution to a complex problem. GA’s work with a set of individual solutions called a population. Associated with each individual is a fitness measure that determines how good the solution represented is. Each iteration in GA is termed as a generation. Some of the main issues to be addressed while using GA include:

- The choice of the objective function – to assign a fitness to each individual, a mechanism that assigns high fitness values to better solutions is needed
- Encoding – A mechanism needs to be devised to encode the solution to a problem as a set of genes. The choice of this mechanism can have a significant impact on the performance of the GA
- Selection and reproduction – pairs of individuals are chosen from the population based on fitness
- Crossover – A mechanism for combining the contents of two individuals to create a new one
- Mutation – Every individual in the population has a probability to have its contents altered slightly
- Population Size – The number of solutions in a population

3.2.2 The Genetic Algorithm Skeleton

The skeleton of the genetic algorithm can be expressed as:
Step 1: Generate the initial population – in most cases this is random

Step 2: Create a new population by applying selection and reproduction.

Step 3: Apply the cross-over operator to the pairs of strings of the new population

Step 4: Apply the mutation operator to each string in the new population

Step 5: Replace the old population with the newly created population

Step 6: If the number of iterations is less than the maximum, go to Step 2. Else stop the process and display the best result found

3.2.3 Software Clustering using GA

Doval et al.[20] use the concepts of intra-connectivity and inter-connectivity to devise an objective function for the MDG partitioning problem. The intra-connectivity $A_i$ of cluster $i$ with $N_i$ components and $\mu_i$ intra-edge dependencies is defined as:

$$A_i = \frac{\mu_i}{N_i^2}$$

$A_i$ is the number of intra-edge dependencies divided by the maximum number of possible dependencies between the components of cluster $i$. It is 0 when the modules in the cluster are not connected, and 1 when each module in the cluster is connected to every module in the same cluster.

Inter-connectivity is a measure of connectivity between different clusters. The inter-connectivity $E_{i,j}$ between clusters $i$ and $j$, each containing $N_i$ and $N_j$ components respectively with $\epsilon_{ij}$ inter-dependencies is defined as:
The Modularization Quality (MQ) used as the objective function is defined as:

\[
MQ = \begin{cases} 
\frac{\sum_{i=1}^{k} A_i}{k} - \frac{\sum_{i,j=1}^{k} E_{i,j}}{k(k-1)} & \forall k > 1 \\
A_i & k = 1
\end{cases}
\]

MQ establishes a trade-off between inter and intra-connectivity and rewards the creation of highly cohesive clusters and penalizes the creation of too many inter-cluster dependencies.

For encoding, each node in the graph is assigned a unique identifier, and a string \( S \) is defined as

\[
S = s_1s_2s_3...s_N
\]

where \( N \) is the number of nodes in the graph. The various parameters used are: Cross-over rate-80%, Mutation rate-0.004 \( \log_2(N) \) and population size – 10N. The number of generations is capped at 200N. Doval et al.[20] apply their work in an operating system called Mini-Tunis. The partition created by applying their method is found to be very similar to the one found in the original documentation of the operating system.

Sharma and Shikha Raj[79] empirically analyses a derived form of k-means using genetic algorithm approach. This algorithm prevents algorithm to converge towards local minima by considering a rich population of potential solutions.. Lu and Ju[57] described an optimized genetic k-means clustering
algorithm based on genetic algorithm. It uses encoding, initialization, fitness
function selection, crossover and mutation of genetic algorithms into clustering
problem. Chang et al.[16] given a new clustering algorithm based on genetic
algorithm (GA) with gene rearrangement. Karegowda et al.[48][49] investigated the use of two methods namely GA and Entropy based Fuzzy
Clustering (EFC) to assign k-means initial cluster centers for clustering Indian
diabetic dataset. Singh et al.[80] used genetic algorithm with k-means to
improve its efficiency.

3.3 K-MEANS CLUSTERING

The k-means clustering algorithm follows a simple way to classify a
given dataset through a certain number of clusters fixed Apriori. The algorithm
starts by defining k-centroids, one for each cluster. The better choice to place
the centroids is to place them as far as possible from each other. The algorithm
then proceeds to take each point in the dataset and associate it with the nearest
centroid. When all points are done in this way, the first iteration is completed,
and an early groupage is done. Now the algorithm recalculates k new centroids.
After this, a new binding has to be done between the same set of data points
and the new centroids. The k-centroids change step by step until no more
changes are done. The algorithm aims at minimizing an objective function
which is the squared error function.
The objective function

\[ j = \sum_{j=1}^{k} \sum_{i=1}^{n} \|x_i^{(j)} - c_j\|^2 \]

where \(\|x_i^{(j)} - c_j\|^2\) is a chosen distance measure between a data point \(x_i^{(j)}\), and the cluster centre \(c_j\) is an indicator of the distance of the \(n\) data points from their respective cluster centres.

The algorithm works as follows:

Step 1: Place \(k\) points into the space represented by the objects that are being clustered. These points represent initial group centroids.

Step 2: Assign each object to the group that has the closest centroid.

Step 3: When all objects have been assigned, re-calculate the positions of the \(k\)-centroids.

Step 4: Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized is calculated.

The \(k\)-means clustering algorithm is very popular owing to its simplicity, but it does have some weaknesses:

- There is no standard way to initialize the means. The simplest solution is to randomly choose \(k\) samples.
- The results are very sensitive to the initial selected means, and frequently \(k\)-means tends to find sub-optimal solutions.
- The results are sensitive to the metric used to measure \(\|x_i^{(j)} - c_j\|\).
• The results also depend on the value of k. In many cases, there is no idea as to how many clusters can exist within a set of data objects.

3.4 GENETIC K-MEANS CLUSTERING

Krishna and Murty[51] suggest a novel hybrid genetic algorithm that finds a globally optimal partition of a given data into specified number of clusters. They attempt to hybridize GA with the k-means algorithm. Formally stated, the problem is to partition a collection of n given patterns, each pattern is a vector of dimension d, into K groups in such a way that minimizes Total Within Cluster Variation (TWCV). Let \( \{x_i, i = 1, 2, ..., n\} \) be the set of n-patterns.

Let \( x_j \) denote the jth feature of \( x_i \). For \( i = 1, 2, ..., n \) and \( k = 1, 2, ..., K \) let

\[
w_{ik} = \begin{cases} 1, & \text{if } \text{ith pattern belongs to } k\text{th cluster} \\ 0, & \text{otherwise} \end{cases}
\]

If the centroid of the k-th cluster is defined as \( c_k \), then

\[
c_{kj} = \frac{\sum_{i=1}^{n} w_{ik} x_{ij}}{\sum_{i=1}^{n} w_{ik}}. For the k-th cluster, the within cluster variation is defined as
\]

\[
S^{(k)}(W) = \sum_{i=1}^{n} w_{ik} \sum_{j=1}^{d} (x_{ij} - c_{kj})^2
\]

And the Total Within Cluster Variation (TWCV) is defined as

\[
S(W) = \sum_{k=1}^{K} S^{(k)} = \sum_{k=1}^{K} \sum_{i=1}^{n} w_{ik} \sum_{j=1}^{d} (x_{ij} - c_{kj})^2
\]

The objective is to minimize \( S(W) \).
The Genetic K-means Algorithm (GKA) bears several features of any genetic algorithm in that it too starts with a population of encoded solutions, and genetic operators are applied to the population to obtain the next generation. The iterative process is terminated when a specified terminating condition is reached, and this may be a threshold selected for the number of generations. The important aspects of the described GKA are listed below:

- **Coding** – $W$ is encoded into a string $s_w$ by considering a chromosome of length $n$ and allowing each allele to take values $\{1, 2, \ldots, K\}$. Each allele represents a pattern, and the allele value indicates the cluster number to which the pattern belongs. This is called string-of-group-numbers encoding.

- **Initialization** – as with most GA’s the initial population is obtained by initializing each allele in the population to a random number selected from the set $\{1, 2, \ldots, K\}$.

- **Selection** – a chromosome is selected from the previous population according to the distribution: $P(s_i) = \frac{F(s_i)}{\sum_{j=1}^{N} F(s_j)}$ where $F(s_i)$ represents the fitness value.

- **Fitness function** – in order to minimize $S(W)$ – the Total Within Cluster Variation, Krishna and Murty[51] resort to the $\sigma$-truncation mechanism. They define

\[
\sigma_{i,j} = \sum_{k=1}^{K} (w_{i,k} - w_{j,k})^2
\]

where $w_{i,k}$ is the $k$th component of the $i$th pattern and $\sigma_{i,j}$ is the distance between patterns $i$ and $j$. The $\sigma$-truncation mechanism involves discarding the $\sigma$-largest distances and retaining the $\sigma$-smallest distances to prevent the algorithm from converging to local minima.
\[ f(s_w) = -S(W), g(s_w) = f(s_w) - (\chi - c \sigma), \]

where \(\chi\) and \(\sigma\) denote the average value and standard deviation of \(f(s_w)\) in the current population, and \(c\) is a constant between 1 and 3. Fitness is calculated as:

\[
F(s_w) = \begin{cases} 
  g(s_w), & \text{if } g(s_w) \geq 0 \\
  0, & \text{otherwise}
\end{cases}
\]

- **Mutation** – the main issue concerning the mutation operator is that the probability of changing an allele value must be more if the corresponding cluster centre is closer to the data point. In order to define the mutation operator, Krishna and Murty [51] define \(d_j = d(x_i, c_j)\) as the Euclidean distance between \(x_i\) and \(c_j\). The replacement value for an allele is chosen according to the distribution:

\[
p_j = \Pr(s_w(i) = j) = \frac{c_m d_{\max} - d_j}{\sum_{i=1}^{K} (c_m d_{\max} - d_i)}
\]

where \(c_m\) is a constant usually \(\geq 1\) and \(d_{\max} = \max\{d_j\}\). Krishna and Murty[51] also ensure that empty clusters are not formed by proposing to change the allele only if the distance between the data point \(x_i\) and the corresponding cluster centre \(c_{s_w}(i)\) is greater than zero.

- **The k-means operator** – as the GA with just the above steps may take more time to converge, Krishna and Murty[51] presented the k-means operator described as follows: let \(s_w\) be the string. The k-means Operator (KMO) when applied to \(s_w\) produces \(s_w^*\) according to the
following the steps listed below: a) For the given matrix W, cluster centers are calculated b) Each data point is reassigned to the nearest cluster with cluster centre.

Krishna and Murty[51] prove that their presented GKA performs better than the ordinary k-means algorithm, evolutionary strategies and evolutionary programming. They also claim that the method is computationally less expensive.

3.5 SOFTWARE QUALITY ESTIMATION WITH CLUSTERING

Software quality estimation has been a long standing and pressing problem for software developers and managers for a period of time. In the current competitive business environment, the paucity of resources prohibits managers from devoting resources to all modules to ensure quality. There have been attempts to use fault-data from the previous system releases and construct fault prediction models. Such models are then used to predict the fault-proneness of modules in development. Modules that are predicted to be fault-prone are allocated more resources and subject to greater scrutiny and quality assurance techniques.

Zhong et al.[98][99] claim that software quality estimation using supervised techniques like classification works fine only if past data from similar software projects are available. They attempt to use cluster analysis with expert input for predicting the fault-proneness of software modules.
The basic idea behind the work of Zhong et al.[98][99] is as follows: In the absence of software proneness labels, which might result from an organization dealing with a particular type of project it has not dealt with before, unsupervised learning method like cluster analysis represents a viable option. Software modules are grouped into clusters based on the values of various metrics. Modules that are fault-prone will have similar values of metrics and thus be grouped in one cluster. Modules that are not fault-prone will constitute another cluster. After the cluster analysis is completed, an expert can inspect each cluster and label it as “fault-prone” or “not-fault-prone”. Exercising this option saves a lot of effort for the expert who would otherwise have to analyze every module to determine its fault-proneness.

Mertik et al.[63] presented the use of advanced tool for data mining called Multimethod on the case of building software fault prediction model. Azar et.al.[10] given a search-based software engineering approach to improve the prediction accuracy of software quality estimation models by adapting them to new unseen software products. Trivedi and Kumar[83] presented set of software matrix that will check the interconnection between the software component and the application. Seliya and Khoshgoftaar[77] presented a semi-supervised learning scheme as a solution to software defect modeling when there is limited prior knowledge of software quality. Arora et al.[9] have studied three object oriented metrics and given a case study to show, how these
metrics are useful in determining the quality of any software designed by using
object oriented paradigm.

3.5.1 Metrics

The development and usage of sound software metrics has been a big
challenge to the software engineering community[30]. Not withstanding the
many controversies and debates software metrics have been subject to, most
researchers agree that the development and usage of sound software metrics
will lead to considerable improvements in the quality of the engineered
software.

For a module under study, the various metrics associated with it can
provide indications of various aspects of it. For example, the McCabe’s
Cyclomatic Complexity provides insight into the complexity associated with
the module. This complexity may have implications for understandability and
maintainability. For module clustering, the values of these metrics form the
basic input based upon which it is placed in a particular cluster.

3.6 EXPERIMENTAL STUDY

The basic methodology is very similar to the one adopted by
Zhong et.al.[98]. The main objective was to determine whether the genetic
k-means algorithm can deliver more accurate results than the regular k-means
algorithm. A small local software organization involved in providing software
solution to its vendors was subject to the study. The organization is responsible
for the development and maintenance of web sites for its clients. Data
pertaining to three web sites developed by the organization are studied. The software measurement and fault data pertaining to the three applications were collected at the module level – a module in this case was a method as all the web sites were developed using Java Server Pages (JSP). A total of 312 modules from the first application called as A1, 271 modules from the second called as A2 and 134 modules from the third application called as A3 were studied.

The set of metrics used for each module comprises

- Total Lines of Code
- Total Operators
- Total Operands
- Unique Operators
- Unique Operands
- Cyclomatic Complexity
- Halstead’s Program Length
- Halstead’s Program Volume
- Branch Count

The nine metrics taken into account for the study do not in any way fully characterize a module. Many metrics are available and some of them may be even more accurate predictors of the attribute they are attempting to measure. The study concentrates on the values of only these modules. The choice of these metrics was primarily based on the interactions with the expert involved
in labeling. The expert was the senior project manager with 15 years of experience in web site development, maintenance and management.

Both the k-means and the genetic k-means clustering algorithms were implemented in VB.NET under the Windows XP Operating System. The number of clusters to be generated heavily depends on the availability of resources. For the experiment, the number of clusters was fixed at 30, 20 and 15 clusters for the three applications A1, A2 and A3. The choice of the number of clusters is a trade-off between the effort required by the expert for labeling the clusters and obtaining a fine representation of the software data. The more number of clusters, is required by the expert and finer the granularity.

For labeling, the expert is provided with the global mean, minimum, maximum, median, 75th percentile, 80th percentile, 85th percentile and 90th percentile of each metric for each cluster as well as the size of each cluster. Based on these statistics, the expert labeled each cluster as “fault prone” or “not fault prone”. This is the same methodology followed by Zhong et al.[98].

For the comparative analysis, the following measurements are used. These are essentially the same ones used by Zhong et al.[98].

- Mean Squared Error (MSE)
- Cluster Purity – percentage of the most dominated category (fault-prone or not-fault-prone) in the cluster
- Average Purity – Mean purity of all the clusters

The expert’s labeling decision is evaluated using the following criteria:
Over-all classification error

- False Positive Rate (FPR) – percentage of not-fault-prone modules mislabeled as fault-prone
- False-Negative Rate (FNR) – percentage of fault-prone modules mislabeled as not-fault-prone.

For the experiment, a module was considered fault prone if it had more than five faults. This was used only to assess the labeling performance of the expert and not for clustering.

### 3.7 RESULTS AND DISCUSSION

The results of the experiment are shown in the Table 3.1 that follow.

**Table 3.1 Clustering Quality Results for Application 1(A1)**

<table>
<thead>
<tr>
<th>Measure</th>
<th>k-means</th>
<th>Genetic k-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Squared Error</td>
<td>1567.81</td>
<td>1395.32</td>
</tr>
<tr>
<td>Average Purity</td>
<td>0.797</td>
<td>0.877</td>
</tr>
</tbody>
</table>

As can be observed, for the application A1, Genetic k-means is found to yield a lower value for Mean Squared Error and higher cluster purity. There is a 9% improvement observed in cluster purity.

**Table 3.2 Clustering Quality Results for Application 2(A2)**

<table>
<thead>
<tr>
<th>Measure</th>
<th>k-means</th>
<th>Genetic k-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Squared Error</td>
<td>1317.78</td>
<td>1213.45</td>
</tr>
<tr>
<td>Average Purity</td>
<td>0.832</td>
<td>0.882</td>
</tr>
</tbody>
</table>
For the application A2 also, Genetic k-means is found to yield a lower value for Mean Squared Error and higher cluster purity. There is a 5% improvement observed in cluster purity.

**Table 3.3 Clustering Quality Results for Application 3(A3)**

<table>
<thead>
<tr>
<th>Measure</th>
<th>k-means</th>
<th>Genetic k-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Squared Error</td>
<td>877.78</td>
<td>801.21</td>
</tr>
<tr>
<td>Average Purity</td>
<td>0.871</td>
<td>0.893</td>
</tr>
</tbody>
</table>

For the application A3 as well, Genetic k-means is found to yield a lower value for Mean Squared Error and higher cluster purity. There is a 2% improvement observed in cluster purity. The improvement in cluster purity tends to narrow down from A1 to A2 to A3. The Figure 3.2 shows mean squared error obtained by applying K-means and genetic K-means algorithm graphically.

![Figure 3.2 Mean Squared Error when applying k-means and Genetic k-means](image)
The increased average cluster purity obtained by applying genetic k-means is shown in Figure 3.3.

![Figure 3.3](image)

**Figure 3.3** Average Cluster Purity when applying k-means and Genetic k-means

<table>
<thead>
<tr>
<th></th>
<th>False Positive (%)</th>
<th>False Negative (%)</th>
<th>Overall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>15.6</td>
<td>67.2</td>
<td>21.6</td>
</tr>
<tr>
<td>Genetic k-means</td>
<td>15.8</td>
<td>60.1</td>
<td>20.9</td>
</tr>
</tbody>
</table>

Table 3.4 shows that the expert labeling performance, in application A1, genetic k-means records a 0.2% reduction on the number of false-positives. This is a very small improvement. But for false negatives, the improvement goes up to 7.1%. This result is admissible given the importance of false-negatives compared to false-positives.
Table 3.5  Expert Labeling Performance with Application 2 (A2)

<table>
<thead>
<tr>
<th></th>
<th>False Positive (%)</th>
<th>False Negative (%)</th>
<th>Overall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>16.2</td>
<td>58.1</td>
<td>19.8</td>
</tr>
<tr>
<td>Genetic k-means</td>
<td>15.2</td>
<td>53.5</td>
<td>18.6</td>
</tr>
</tbody>
</table>

For the expert labeling performance, in application A2, genetic k-means records a 1% reduction on the number of false-positives. This is a very small improvement. But for false negatives, the improvement goes upto 4.6%. This result is admissible given the importance of false-negatives compared to false-positives.

Table 3.6 shows that for the third application also genetic k-means results in a fewer false positive and false negative percentages. The overall false rate decreases from 19.8% to 18.6%.

Table 3.6  Expert Labeling Performance with Application 3 (A3)

<table>
<thead>
<tr>
<th></th>
<th>False Positive (%)</th>
<th>False Negative (%)</th>
<th>Overall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>13.6</td>
<td>39.2</td>
<td>19.8</td>
</tr>
<tr>
<td>Genetic k-means</td>
<td>12.9</td>
<td>35.7</td>
<td>18.6</td>
</tr>
</tbody>
</table>

The comparison of false positive, false negative and overall percentages for k-means and genetic k-means is shown as a column chart in Figure 3.4.
Figure 3.4 Expert Labeling Performance for A1

The expert labeling performance for the second application is shown graphically in Figure 3.5

Figure 3.5 - Expert Labeling Performance for A2
Figure 3.6 compares the k-means and genetic k-means expert labeling performances.

![Figure 3.6 Expert Labeling Performance for A3](image)

A closer analysis of the results reveals certain interesting properties of the genetic k-means algorithm. First, considering the clustering quality, it is observed that genetic k-means tends to produce clusters with a lower mean squared error value compared to k-means. The difference tends to be more pronounced for the larger application which is A1, and the difference tends to narrow down for the smallest application A3. This leads to the conclusion that genetic k-means has more potential to perform when the number of modules to be clustered is huge as is the case with many real-world projects. The same conclusion holds for the average purity as genetic k-means produces clusters with more average purity, and the difference increases with the increasing size.
of the project considered. This stands in line with the general principle that as the search space tends to become huge, genetic algorithms have more potential to deliver compared with conventional ones.

When analyzing the expert labeling decision, the first thing to be noted is that the feedback from the expert indicated that clusters generated by genetic k-means were more easier to label than those generated by k-means. This tends to be because of the ability of the genetic k-means algorithm to produce more coherent clusters. Narrowing the analysis to false-negative misclassification rates, as the cost of such misclassifications is significantly higher than that of false-positive misclassifications, the genetic k-means is found to perform better with the increasing size of the applications considered. It is very important to consider false-negative misclassifications because if a faulty module is classified as non-faulty, the impact will be severe. On the other hand if a not faulty module is classified as faulty, the only negative outcome will be the increased wasted resources spent on them.

3.8 SUMMARY

The potential of the genetic k-means algorithm to cluster software modules based on their fault proneness was studied. It was found that the genetic k-means algorithm performs significantly better than k-means as the size of the application increases. Given the paucity of resources in most organizations, such ability to predict the fault proneness of modules under development is vital. Clustering has the added advantage that it can perform even in the absence of fault and measurement data of similar past software
projects. In this regard, the results of the conducted empirical study demonstrate that as the size of the application increases, genetic k-means can prove to be more useful and accurate than k-means clustering.