Chapter 3

Review of various Clustering Techniques in Data Mining for Effective Defect Management

Prediction of software defect thrive to develop quality software and to test efficiency by constructing predictive clustering and classification models to enable a timely identification of defect-prone modules. A number of clustering models have been evaluated for this work. Despite a vast number of surveys for clustering algorithms available in chapter 2 for various domains such as machine learning, data mining and information retrieval. It is difficult to decide a priori which algorithm would be the more appropriate for a given data set. This motivated to review of various clustering algorithms in this chapter. Section 3.1 provides a study of an approach to predict software project success by different data mining clustering techniques and section 3.2 shows the analysis of quality of software projects using data clustering techniques such as K-means and fuzzy C-means clustering.

3.1 Introduction

Software has become one of the critical components in any application since three decades. However, the advancement in technology and the increasing demand for software has created a situation where massive amount of data needs to be managed efficiently. Data mining has proved to be one of the most efficient techniques to effectively manage huge amount of data since information is highly expensive and valuable in nature. Retention of high quality software which can predict the success of any project is one of the core challenges in any industry. Hence, predicting software success using the empirical knowledge of projects can be achieved through effective data mining algorithms.

Mining software engineering data has emerged as one of the successful research directions since few decades. Software repositories contain a wealth of valuable information about software projects [21][73][74]. Information stored in these repositories act as predictive pattern for estimating, planning and controlling future projects, Hence the software developing team can now depend less on their intuition and operate more with realistic data. The concept of data mining is gaining acceptance in business as a means of seeking higher profits at lower costs [75]. To classify data mining projects effectively, organizations need to know the important factors for successful data mining. If the crucial success factors are not taken into account adequately or not documented properly, then implementing promising
predictive systems may be risky, [76]. Despite the fact that various studies have listed the advantages and have dealt with the data mining processes, the fact remains that little research has been carried out on the success factors of data mining [77][78][79].

Data mining has emerged as one of the assuring techniques through which valuable information can be mined from the population of empirical projects. Clustering is considered to be one of the most significant techniques of data mining which is applicable in various domains such as life sciences, medical sciences, engineering and so on[80][81]. Clustering technique can be viewed in various perspectives depending on operational environment which includes unsupervised learning in pattern recognition, numerical taxonomy in biology, typology in social sciences, partition in graph theory etc [73].

The most habitually discussed hitch in any software organization is defect prediction to enhance quality and reliability. To accomplish this objective, organizations are integrating data mining techniques which also include software defect management in software engineering process effectively. Therefore clustering is one of the main tasks of explorative data mining and a method for statistical data analysis [82]. One of the vital characteristics of the clustering process is to categorize the software projects into reasonable groups. This grouping facilitates in discovering similarities and differences using which useful conclusions can be drawn from the mined information. This will in turn improve the quality of the software.

Defects do have varying degrees of complexity and severity. All defects do not have the same nature. Hence, in order to deliver defect free software, it is imperative to predict and fix all defects before the product is deployed to the customers. This is yet another key challenge in the software industry. However, software repositories have enormous information which is highly beneficial in assessing the software quality. Hence, data mining techniques and machine learning algorithms can be applied on these repositories to extract the useful information on effective defect management.

Quality is not a stable state but varies depending on applications, nature of projects, industrial environment and so on. Hence, the choice of clustering algorithms is based on availability of type of input data, purpose and application domain of the project. The aim of this chapter is to analyse the effectiveness of popular clustering algorithms in predicting the success of any project. Growing complexities of software and increasing demand of software projects have led to the progress of continual research in the areas of effective project management [75]. Data mining has proven to be one of the established machine learning techniques for effective project management since two decades [83]. Authors in [84] have
presented different clustering algorithms using Waikato Environment for Knowledge Analysis (WEKA) which is the most popularly used data mining tool. In this chapter it focuses on the selection of clustering technique that yields effective results in prediction of project success from the perspective of defect profile. In order to achieve this objective, detailed investigation has been carried out on several projects chosen from various software industries. However, software industries as such are a huge population which necessitates sampling. This work is directed towards formulation of hypothesis to overcome the intrinsic complexities involved within the universe of software industries and in the projects developed by these industries. These are explained in the subsequent sections of this chapter.

3.2 Empirical data analysis using various clustering algorithms

Every algorithm has its unique importance. Authors in [77] conducted an empirical comparison of four clustering algorithms (i.e. CLARA, CLARANS, GAC-R3, and GAC-RARW) over a wide range of data characteristics [77]. Experimental results of CLARANS outperform its counterparts both in terms of clustering quality and execution time. However, demand for effectively mining the valuable information has focused attention on integrating data mining techniques with software engineering. It is an open issue to design a good data mining prediction model [75]. Authors in [78] have shown that there is no particular learning technique that performs very well for all the data sets.

However, IBL (Instance Based Learning) and 1R (1Rule), which are some of the widely used learning techniques have proven to exhibit relatively better consistency in predicting accuracy of data sets when compared with other learning techniques., Authors in [79] have suggested the association rule mining to be an attractive technique for the software engineering community due to its relative simplicity and transparency besides being effective in constructing prediction systems.

Since, defect prediction is one of the most significant activities of software engineering; the application of data mining techniques for effective prediction of software defects is one of the core needs of the day. Authors in [76] have surveyed different data mining algorithms used for defect prediction and have also discussed the performance and effectiveness of data mining algorithms. Authors in [75] have done a comparative analysis of performance of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Neuro-Fuzzy System for prediction of, level of severity of faults present in Java based object oriented software systems. Their results indicate no variations in accuracy of DBSCAN and
Neuro-Fuzzy based systems. Thus, this part of research aims at exploring the choice of right clustering algorithm for effective prediction of software success using defect count as one of the influencing parameters. There exist several clustering techniques to mine the information from massive amount of data [81],[82],[83]. They include partitioning methods, hierarchical methods, density based methods, grid based methods, model based methods, methods for high dimensional data and constraint based clustering. Clustering approach segments data from large data sets into groups based on similarity. In order to comprehend the effectiveness of clustering techniques, it involves comparative analysis of the techniques against the collected project data set.

Empirical data related to defect management aspects were collected from the sampled software industries through their document management repository and defect management centres. Modes of data collection included interactions with project developing team, quality assurance departments, defect management team etc. in the form of interviews, face to face communication and so on. Empirical data analysis includes application of most popular clustering techniques on the software projects to evaluate the efficiency of each clustering algorithm for effective prediction of project success based on defects.

The observational results indicate that K-means clustering is most effective when compared to other clustering approaches in terms of processing time, scalability and efficiency. Table 3.1 provides details about the sampled type of data chosen for the analysis purpose. It further illustrates projects which are sampled for medium category of complexity. The clustering approaches used for comparison are given below.

### 3.2.1 K-means Clustering

K-means algorithm is one of the most popular partitioning clustering techniques. It can be implemented using the following steps [80]:

**Step 1:**
Partition objects into k nonempty subsets.

**Step 2:**
Compute seed points which are the centroid of the cluster in the current partition. Here centroid refers to the centre (mean point) of the cluster.

**Step 3:**
Assign each object to the cluster whose seed point is the nearest.

**Step 4:**
Repeat Step 2, Stop when no more new assignments. The process is iterated until all the data sets of the project are analyzed.

Table 3.1 Attribute with its types

<table>
<thead>
<tr>
<th>Project No</th>
<th>Name of attribute</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Total project time in hours</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>Inspection time scheduled</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>Number of inspectors involved</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>Experience level of inspectors (years)</td>
<td>N</td>
</tr>
<tr>
<td>5</td>
<td>Defect count estimation</td>
<td>N</td>
</tr>
<tr>
<td>6</td>
<td>Number of defects detected</td>
<td>N</td>
</tr>
<tr>
<td>7</td>
<td>Defects actually captured</td>
<td>N</td>
</tr>
<tr>
<td>8</td>
<td>Number of defects not captured</td>
<td>N</td>
</tr>
<tr>
<td>9</td>
<td>Defects due to bad fixes</td>
<td>N</td>
</tr>
<tr>
<td>10</td>
<td>Testing time scheduled</td>
<td>N</td>
</tr>
<tr>
<td>11</td>
<td>Number of testers</td>
<td>N</td>
</tr>
<tr>
<td>12</td>
<td>Class</td>
<td>C</td>
</tr>
</tbody>
</table>

3.2.2 Expectation–Maximization (EM) algorithm

An Expectation–Maximization (EM) algorithm is an iterative method for finding maximum likelihood estimates of parameters in statistical models.

The two stages in this approach alternate between performing an expectation (E) and maximization (M) [83]. It manages E stage by calculating the project grouping probabilities based on success or failure of projects. M stage of the technique is achieved using the computation of the distribution of defect count parameters.

3.2.3 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
DBSCAN is a widely used density based clustering approach. A cluster is having subset of the data points of the database set satisfies these properties: All points within a cluster are equally density connected. When a density point is reachable from any other point of a cluster, it is part of that cluster as well [79]. Euclidean distance metric to form clusters of projects, but it determines the number of clusters automatically, unlike K-means.

3.2.4 Hierarchical clustering

Starting from an initial pair of clusters, recursively considering the option of worth splitting of groups or merging of groups leads to hierarchical clustering technique [77]. Software projects are initially merged based on success and failure count using single linkage clustering principle.

3.2.5 Cobweb

Cobweb algorithm compares four different ways of treating a new instance and chooses the best. It generates hierarchical clustering, where clusters are described probabilistically [79]. In this approach, according to the standard features available in WEKA tool, software projects grouping using defect related data which is based on training and testing procedure. The 66 percent of projects are considered to be the training sets and the remaining projects are considered to be the test sets.

3.2.6 Ordering Points to Identify the Clustering Structure (OPTICS)

OPTICS technique is an extension of DBSCAN in which values are stored with each data object; it is an attempt to overcome the necessity to supply different input parameters. It’s time complexity is $O(n \log n)$. Grouping of software projects is based on density parameters such as epsilon (threshold value) and minpts (minimum number of points). According to the standard WEKA tool, threshold value for the density parameter which is success count of the project is fixed to be 0.9 and minpts to be 6.

3.2.7 Farthest first

Farthest first clustering approach is modelled on K means. It places each cluster centre in turn at the point farthest from the existing cluster centers. This point must lie within the data area. It fastens the clustering in most cases because of the need for less reassignment and
adjustment. As per the WEKA tool, empirical data related to defect profile of projects are subjected to training and cross validation in order to form software project groups for the evaluation of success / failure of projects.

### 3.2.8 MakeDensityBasedClustererAlgorithm

Clusters are defined in density based are having higher density than the rest of the data set. The objects in these sparse areas which are separated from clusters are considered to be noise and border points. In MakeDensityBasedClusterer Algorithm, each cluster is produced by a Simple K-means algorithm.

### 3.3 Observational Inferences

The above mentioned algorithms are implemented using source code in the Weka 3.6.4 version upon which simulations are carried out in order to measure the effectiveness of the algorithms over the empirical datasets. The results are depicted (Refer to Fig. 3.1 through Fig.3.9).

The major strength of K-means is that it is relatively scalable. It also has the ability to process large data sets because computational complexity of the algorithm is O (nkt). It may produce tighter clusters than other clustering, especially if the clusters are global. However, the major deficiency of this approach is the difficulty in comparing the quality of the clusters because different initial partitions can result in diverse final clusters. Figure 3.1 represents clusters having success (cluster 0) and failure (cluster 1) for the projects based on defect count.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Full Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.5</td>
</tr>
<tr>
<td>36</td>
<td>622.2917</td>
</tr>
<tr>
<td>30</td>
<td>598.7083</td>
</tr>
<tr>
<td>15.6</td>
<td>1.2417</td>
</tr>
<tr>
<td>94.64</td>
<td>79.2888</td>
</tr>
</tbody>
</table>

RATE FAILURE SUCCESS FAILURE
Clustered Instances
0 16 (67%)
1 8 (33%)

Figure.3.1 Output of K-means algorithm
The strength of EM is that the technique ensures the results to converge to a local maximum may not be feasible for the global maximum. However, the repetitive process yields better global maximum. Nevertheless, weakness of this approach is its technical difficulties with equating probability value of success/failure. In figure 3.2, log likelihood of the model has shown number of instances assigned to each cluster whenever the learned model is applied to the data as a classifier. The table entries in figure 3.2 show the parameters of numeric attributes (defect count) or frequency of defect counts for the values of nominal attributes (success / failure). Fractional values reveal the soft nature of the clusters produced by EM indicating that any defect can split between several clusters. Figure 3.2 further infers that the algorithm converges towards fixed point but never actually gets there. However, K-means algorithm stops when it reaches a fixed point (does not change from one iteration to the next).
The strength of DBSCAN is that unlike K-means, prior information about the number of clusters is not required. DBSCAN needs only two parameters and is insensitive to the ordering of the points since the edge of two different clusters can exchange when the cluster order is altered. However, the weakness in this approach is that it cannot cluster data sets accurately with large differences in densities, due to high dimensionality. Figure 3.3 depicts the results of DBSCAN, minpts (minimum number of points in an \( \epsilon \)-neighbourhood of that point) and \( \epsilon \) (maximum radius of the neighbourhood) (minpts = 6 and \( \epsilon = 0.9 \)). K-means method is comparatively better than DBSCAN since with increase in project features, finding appropriate value for epsilon (\( \epsilon \)) using DBSCAN algorithm is difficult.

![Figure 3.3 Output for the Hierarchical algorithm](image)

The strength of hierarchical clustering that it is more robust to its input parameters, less influenced by cluster shapes and less sensitive to large differing point densities of clusters. It can represent nested clusters. Weakness of this technique is that clusters are not explicit in the output of the algorithms and have to be determined by dendrogram representation (A dendrogram is a branching diagram that represents the relationships of similarity among a group of entities). Figure 3.4 indicates the results of dendrogram as tree visualise structure.
The result of the algorithm is a tree of clusters called dendrogram, The cluster formation is shown in figure 3.5. Hierarchical clustering is expensive in terms of its computation and storage requirements [85]. Further, merging of dendrogram tree may cause problems for noisy and high dimensional data. These two issues can be resolved by using K-means technique with a predefined cluster number.

The strength of Cobweb is that it automatically adjusts the number of classes in a partition. Cobweb provides merging and splitting of classes based on category utility. It is also capable of performing bidirectional search which is distinct from K-means method. However, the weakness of Cobweb approach is that the probability distribution representation of clusters is expensive whenever it is required for updating information and storage in the form of
clusters. Further, increasing the number of attributes simultaneously increases the time and space complexity. Figure 3.6 infers that K-means method considers the probabilities and independence of clustering and only the Euclidian distance is taken into consideration. Hence it is free of issues.

![OPTICS Visualizer - Main Window](image)

**Figure 3.7 Output for the OPTICS algorithm**

Figure 3.7 indicates that an ordering on the instances according to distance metric, reachability criteria and maximum number of iterations are specified by the user. Since it is very difficult to maintain the order in high dimensional software project data, OPTICS clustering is not efficient when compared to K-means, which does not operate using order of instance.

```
Scheme:     weka.clusterers.FarthestFirst -N 2 -S 1
Relation:   whatever
Instances:  24
Attributes: 6
  1            36
  30
  15.6
  94.64

RATE
Test mode:  evaluate on training data

=== Model and evaluation on training set ===

FarthestFirst
Cluster centroids:
Cluster 0
  11.0 211.0 215.0 -1.8 70.92 FAILURE
Cluster 1
  25.0 1989.0 1851.0 6.9 25.77 SUCCESS

Clustered Instances:
  0  17 ( 71%)
  1   7 ( 29%)
```

**Figure 3.8 Outputs for the Farthest First**
The strength of farthest first clustering algorithm is that it is a heuristic based method, modelled on K-means which is fast and suitable for large-scale data mining applications. Farthest-point heuristic based method has the time complexity $O(nk)$, where $n$ denotes the number of objects in the dataset and $k$ denotes the number of desired clusters. Figure 3.8 Infers that farthest first method is simple and fast.

Farthest point heuristic based method has the time complexity $O(nk)$, where $n$ denotes the number of objects in the dataset and $k$ denotes the number of desired clusters.

**Figure 3.9 Outputs for MakeDensityBasedClusterer**

Clusters are produced using K-means algorithm. Cobweb generates hierarchical clustering, where defects are described probabilistically, EM assigns a probability distribution to each defect instance which indicates the probability of it belonging to each of the clusters. Farthest first depends on lower cost local minim. Hierarchical cluster however provides better result in terms of similarity representation using dendrogram. DBSCAN, OPTICS and MakeDensityBasedCluster perform better. However, K-means algorithm attempts to find the centers of natural clusters in the training data which is an important factor, to be considered for effective defect management.

### 3.4 Analysis of Quality of Software Projects using Data Clustering Techniques

The population of software projects that are developed since the time of evolution of software to today is indicative of wide spectrum of application domains being developed using varied programming languages and in different operating environments. Development of a software product is an inherently difficult endeavour. Managing such a complex venture demands the skilful ability to estimate all project parameters accurately and reliably. Inaccurate estimations lead to unnecessary efforts in continuously modifying the project plan,
preventable delays and in extreme cases the project failures. As already mentioned, K-means clustering is capable of processing large amount of data set. But cluster size should be specified in the beginning. Because of imprecise nature of project parameters and the difficulties involved in determining the size in advance, unsupervised clustering such as fuzzy c-means clustering (FCM) is being considered. Fuzzy Logic is an effective soft-computing technique to solve uncertainties which may arise due to imprecise inputs and for generating linguistic outputs. Fuzzy logic concepts were introduced by Lofti A. Zadeh [24].

Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory to deal with reasoning that is approximate rather than precise. A fuzzy set expresses the degree to which an element belongs to a set. The characteristic function of a fuzzy set is allowed to have values between 0 and 1, which denotes the degree of membership of an element in a given set. Fuzzy systems have been employed in various real life applications. Fuzzy logic modelling techniques such as Fuzzy C-means clustering (FCM) and fuzzy inference have been found to be useful additions to the existing statistical and machine learning techniques which are used for modelling software development [86][87].

Fuzzy Logic has gained popularity in recent history as a sensible technique to achieve improved estimation accuracy of variables in any process. Fuzzy C-means clustering (FCM) is one of the most recent contributions to the field of Artificial Intelligence (AI) and data clustering. Within project management, these variables range from software resource estimation to resource allocations for the completion of a software project [88].

The work is involved with the application of fuzzy logic on imprecise nature of software projects in determining the success or failure rates. The aim of this section is to estimate the level of project success based on defect count as one of the factors with the ultimate objective of determining whether a project is successful or not. In our study, we have applied FCM and K-means clustering for mining of defect related information from the wide range of projects which are developed at various software industries.

Although the idea of applying data mining techniques on software engineering data has existed since 1990s, the idea has attracted large amount of interest in the recent years within software engineering community [59]. Data Mining involves extraction or mining knowledge from large amount of data. It consists of two modes of digging data namely predictive data mining mode and descriptive data mining mode. Predictive data mining deals with extracting information from data and it draws useful analysis for future predictions. The key factor of predictive data mining provides relationship between explanatory variables and the predicted variables from past occurrences and exploiting it to predict future outcomes.
Descriptive data mining describes a data set and presents interesting features of the data without having any predefined target. Both the modes of data mining are used to determine characteristics of association, classification, clustering, prediction and estimation within data sets. However, predictive method is used in our work since the intention of the analysis is to come out with an accurate predictive model for defect estimations which will facilitate effective project management.

Many researchers have contributed towards process and product quality improvement for achieving project success. Fuzzy Logic, Data Mining and Software Engineering domains aim towards the realization of this objective. Authors in [88] have identified the use of fuzzy logic as a reasonable technique to improve the accuracy for ensuring higher software project success rates. Success of any software project heavily depend on the initial estimation of all project parameters. However, information available at different levels of project development phases and desired precision suggest that it can be used in a different way depending on the current phase. Of course, a single model can be used for consistency.

Authors in [89] have suggested data mining clustering and classification algorithms to predict the factors which influence software project success. Authors in [90][91] have shown the power of using soft computing and data mining approach which can indicate the most important factors that lead to quality software development. The added value of visualization provided by different mining model viewers is very crucial for project managers who may not be specialists in data mining. Authors in [91] have presented different data mining clustering algorithms each of which has unique features. In Fuzzy C-means clustering, each data point belongs to a cluster to a degree specified by a membership grade. Fuzzy C-means clustering relies on minimizing a cost function (or objective function) of dissimilarity measure. It has applications in varied fields including pre-processing for system models.

Authors in [79] have explained the practical benefits of clustering. Instead of inspecting and labelling software projects one at a time, the expert can inspect and label a given cluster as a whole, in order to ease out the tediousness of the labelling task, which is compounded when projects are numerous.

The parameters used in the clustering process are shown in table 3.1. These parameters are important to show dynamic balance between the interactions and recommendations which will provide better team configurations for project success. The similarity metric used to compute the similarity between an input value and a cluster centre is the Euclidean distance. Since most similarity metrics are sensitive to the large range of elements in the input values,
each of the input variables must be normalized so that it lies within the unit interval \([0, 1]\). Each clustering algorithm is presented with the training data set, as a result of which two clusters are produced. The data in the evaluation set is then tested against the clusters found and an analysis of the result is performed. Following sections present the results of each clustering technique followed by a comparison of the techniques. These experiments have been conducted using MATLAB.

3.4.1 Analysis of FCM and K-means clustering

In view of the fact that software has strong impact on all applications, it is imperative to develop projects with accurate. Quality has several dimensions the most significant of which is that it should be defect free. Thus, it becomes one of the vital activities of the project manager to accurately predict the defect distribution pattern based on empirical analysis.

Fuzzy clustering and data mining are two significant approaches through which information can be elicited from the huge and varied population of software projects for accurate prediction of defect distribution pattern and the accurate estimation of defects in the subsequent projects. An empirical analysis has been conducted on various projects developed across several software industries for accurate prediction of defect distribution pattern.

Figure 3.10 Comparative analyses of FCM and K-means clustering
Figure 3.10 depicts the mining of software project data for evaluation of software project management. Empirical data which is collected from various software industries and the modes of data collection from the vast empirical projects are done through the Document Management Repository MDP of the respective companies. Sources of data collection additionally include interactions with the project developing team, quality assurance departments, defect management centers, etc. in the form of interviews and face to face communication. Collected data is pre processed by clustering techniques in order to analyse the software project tasks such as defect prediction.

A. Fuzzy C means Clustering (FCM)

Imprecision in the selection of attributes during the early stage of software project development may lead to wrong prediction of the outcomes. For these reasons, it is worthwhile to use fuzzy logic attributes. These models are having reasonably good to provide reduced commitment, interoperability, utilization of knowledge. Since many of the attributes in software models are difficult measure it, therefore using fuzzy attributes seems intuitively appealing.

Project managers will be able to classify projects using fuzzy attributes by considering accuracy and reliability. Defect complexity being a multifaceted concept, well experienced project managers would be able to make fairly consistent classifications of projects in terms of one or a small number of types of defect complexity. This has the added advantage of reducing the number of variables used as inputs into the model. Prediction of defect distribution pattern can be done using FCM technique.

Fuzzy C-means clustering (FCM) has the basic idea of Hard C-means clustering (HCM). However, data points in FCM belong to a cluster with some degree of membership grade, Contrast this with HCM in which every data point belongs to a cluster or falls in some outlier. FCM employs fuzzy partitioning in which a given data point can belong to several groups with the degree of membership specified by a membership grade, a number between 0 and 1. However, FCM uses an objective function which is to be minimized while partitioning the data set. The membership matrix U is allowed to have elements with values between 0 and 1. However, the summation of degrees of belongingness of a data point to all clusters is always equal to unity. The algorithm consists of the following steps [92]:

1. Initialize U = U[ij] matrix U[0]
2. K-step Calculate the center vectors C[k] = [Cj] with U(k) where
3. Update U(k) and U (k+1)

\[ u_{ij}^{(k+1)} = \frac{1}{\sum_{l=1}^{c} \left( \frac{\| x_i - c_j \|}{\| x_i - c_l \|} \right)^{m-1}} \]

4. If \( \| U^{(k+1)} - U^{(k)} \| < \epsilon \) then Stop, Otherwise return to the second step.

Since FCM allows data points to have different degrees of membership to different clusters, it eliminates the effect of hard membership introduced by K-means clustering. This approach employs fuzzy measures as the basis of membership matrix calculation and for the identification of cluster centers. FCM clustering and its different threshold (α-cut) values are employed to classify the projects into different groups based on severity of defects. This enables one to analyse the impact of defects in projects besides being helpful in the prediction of defect distribution pattern. This awareness of accurate defect distribution pattern enables the project manager to accurately plan for the defect management strategies.

B. K-means Clustering

The grouping of data samples (software projects) can be done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. The partitioning of n data sets into k clusters, accordingly to reduce the sum of the squared distances to the cluster centres. K-means clustering is an algorithm based on determining data clusters in a data set so that an objective function of dissimilarity or distance measure can be minimized [81]. The objective function of K-means clustering is shown in equation (3.1).

\[ J = \sum_{j=1}^{k} \sum_{i=1}^{k} \left\| X_i^{(j)} - C_j \right\|^2 \quad (3.1) \]

Here \( \left\| X_i^{(j)} - C_j \right\|^2 \) is a chosen distance measure between a data point \( X_i^{(j)} \) and the cluster centre \( C_j \) and is an indicator of the distance of the n data points from their respective cluster centres.
In most cases, this dissimilarity measure is chosen to be the Euclidean distance. The algorithm consists of the following steps [83].
Step 1: Initialize the cluster center by randomly selecting from the data points.
Step 2: allocate each data to the cluster that has the nearby centroid.
Step 3: after all data have been allocated, recalculate the positions of the k centroids. Stop if either it is below a certain tolerance value or its improvement over previous iteration is below a certain threshold.
Step 4: reiterate the steps 2 and 3 until the centroids no change.
The above steps produce a partition of the data into cluster from which the parameter to be minimized can be computed. The performance of K-means algorithm depends on the initial positions of the cluster centers. Hence it is desirable to run the algorithm several times, each with a different set of initial cluster centers.

K-means clustering is applied on empirical projects based on the steps explained above. The algorithm can be evaluated by testing the accuracy of the evaluation set. Defect count values are assigned to various clusters depending on the distance between each defect value and each of the cluster centers. Root Mean Square Error (RMSE) (the difference between predicted and corresponding observed values are each squared and then averaged over the sample) is used for computing error measure. Accuracy measure is calculated as the percentage of correctly classified projects (both true positives and true negatives).

3.4.2 Experimental Results

In order to evaluate the performance of the cluster experimental results, confusion matrix is being used. In the field of machine learning, a confusion matrix is also known as a contingency table or an error matrix [30] as shown in table 3.2.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Defect</td>
</tr>
<tr>
<td>Defect</td>
<td>TP</td>
</tr>
<tr>
<td>Defect free</td>
<td>FP</td>
</tr>
</tbody>
</table>
The confusion matrix has four categories: True positives (TP) are the projects correctly classified as defect. False positives (FP) refer to defect-free projects incorrectly labelled as defective. True negatives (TN) are the defect-free projects correctly labelled as such. False negatives (FN) refer to defective projects incorrectly classified as defect-free ones indicating the level of inaccurate defect distribution pattern. Various evaluation measures that are being used are:

- **Mean Absolute Error (MAE):** it is a quantity used to measure actual outcomes.

- **Root Mean Square Error (RMSE):** The difference between predicted and corresponding observed values are each squared and then averaged over the sample

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (X_{obs,i} - X_{model,i})^2}
\]

where \(X_{obs}\) denotes observed value and \(X_{model}\) denotes modelled value at time \(i\).

- **Accuracy:** It indicates proximity of measurement results to the true values. Accuracy is the proportion of true results (both true positives and true negatives) in the population. Lower values of MAE and RMSE will give better results.

FCM starts by assigning random values to the membership matrix \(U\), Hence several runs will have to be conducted before higher probability of getting good performance is achieved. The iterations will have to be continued until the value of RMSE becomes constant, there is no change of value between the previous and the current iteration values. The results show no variation in performance when an algorithm was run for more than 35 times using FCM clustering as shown in table 3.3 or more than 15 times using K-means clustering as shown in table 3.3. For testing the results, every input value in the evaluation data set is assigned to one of the clusters with a membership ratio as done in the training set.

However, because the output values are crisp (1 or 0), the evaluation of membership have to be defuzzified so that they can be tested against the actual outputs. Same performance measures used in K-means clustering can be used. Since the effect of random initial membership grades has insignificant effect on the final cluster centers, only the effect of the cluster controlling parameter or weighting exponent \(m\) is analyzed, table 3.3 lists the effects of the tests conducted on project data set by varying the weighting exponent \(m\). \(m\) which is a degree of fuzziness or ‘fuzzification’ metre can have its values ranging from 1 to 5 as shown in table 3.3. It is observed that very low or very high values for \(m\) may reduce the accuracy. High values tend to increase the time taken by the algorithm while determining the clusters.
FCM technique shows no improvement over K-means clustering. Both seem to have close accuracy. FCM has been found to be slower than K-means because of several fuzzy computations involved.

Table 3.3 FCM clustering performance results

<table>
<thead>
<tr>
<th>Performance</th>
<th>Weighting exponent m</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>No. of iterations</td>
<td>15</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.40</td>
</tr>
<tr>
<td>Accuracy</td>
<td>75%</td>
</tr>
</tbody>
</table>

As mentioned in section 3.7, K-means clustering attempts to find cluster centres by trying to minimize the objective function $J$. It alternates between updating the membership matrix and updating the cluster centres until no further improvement in the objective function is noticed. Since the algorithm initializes the cluster centres randomly, its performance is affected by those initial values. Hence it is suggested to conduct several runs of the algorithm so that better results can be obtained.

Table 3.4 K-means clustering performance results

<table>
<thead>
<tr>
<th>Performance</th>
<th>Test iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>No. of iterations</td>
<td>8</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.39</td>
</tr>
<tr>
<td>Accuracy</td>
<td>75%</td>
</tr>
</tbody>
</table>
As can be seen from the results of table 3.4, the best case achieved is 82% accuracy and an RMSE of 0.43. This relatively reasonable performance is due to the high dimensionality (large number of attributes or parameters or variables) of the projects.

Table 3.5 Comparison of Clustering Performance

<table>
<thead>
<tr>
<th>Performance</th>
<th>RMSE</th>
<th>Accuracy</th>
<th>Time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCM</td>
<td>0.45</td>
<td>79%</td>
<td>2.7</td>
</tr>
<tr>
<td>K-means</td>
<td>0.43</td>
<td>82%</td>
<td>0.9</td>
</tr>
</tbody>
</table>

From table 3.5, it is quite apparent that K means is better when compared to FCM in terms of accuracy and time. If the value of m is 2.5, then the accuracy of FCM is found to be 79%. In the case of K-means, RMSE value is 0.43 and the accuracy is found to be 82%.

Figure 3.11 Performance results of RMSE

Figure 3.12 Performance results of Accuracy
It is observed that with increase in attribute list for clustering the projects, K–means has a better accuracy than FCM. Further, the processing time decreases in FCM with increase in attribute list when compared to K-means.

**Highlights**

- Cobweb generates hierarchical clustering in which defects are described probabilistically.
- EM algorithm assigns a probability of distribution to each defect instance which indicates the probability of it belonging to each of the clusters.
- Farthest first depends on lower cost local minima.
- Hierarchical clustering however provides the better result in terms of similarity measure. DBSCAN, OPTICS and MakeDensityBasedCluster perform in terms of cluster instance, that is unlike K-means it is not required for the prior information about the number of clusters.
- However, K-means algorithm attempts to find the centres of natural clusters in the training data. If parameters are large, then K-means is computationally faster than hierarchical clustering. The value of k should be small. is an important factor which needs to be considered for effective defect management.

**3.5 Summary**

This chapter presents a comparative analysis of two significant techniques applied on the empirical software projects for comprehending their suitability in terms of accuracy and time constraints. K-means is more accurate and fast in processing the information when compared to FCM. But for large datasets, FCM has proved to be a suitable clustering technique. This
knowledge of right choice of approach for effective defect distribution pattern enables the project managers make accurate estimation of defect count in their subsequent projects. It further leads to effective defect management which in turn enhances the quality of software.

In this chapter, an empirical analysis of several projects developed at various software industries has been brought out. In this analysis, defect count is considered to be one of the influencing parameters to predict the success of the projects. Various clustering algorithms are applied on the empirical projects to evaluate the most representative clustering algorithm. Observational inferences indicate that K-means is more efficient than other clustering techniques in terms of processing time and efficiency. It is also reasonably scalable. A comparison of the results of K-means with other clustering algorithms has ensured its continued relevance and effectiveness as well.