Performance Analysis of Selected Clustering Techniques for Software Defects Prediction

Abdullateef O. Balogun[1], Rufus O. Oladele[2], Hammed A. Mojeed[3], Barakat Amin-Balogun[4], Victor E. Adeyemo[5], Taye O. Aro[6]
[1,2,3,4,5]Department of Computer Science, University of Ilorin, PMB 1515. Ilorin, Nigeria.
Email: [1]balogun.ao1@unilorin.edu.ng, [2]roladele@yahoo.com, [3]mojeed.ha@unilorin.edu.ng, [4]barakatbalogun@yahoo.com, [5]v.e.adeyemo@gmail.com,
&
[6]Department of Mathematical and Computing Sciences, Kola Daisi University, Ibadan, Oyo State,
Nigeria.
Email:[6]taiwo774@gmail.com

ABSTRACT
Classification algorithms that help to predict software defects play a major role in the software engineering process. This study investigated the application and performance of clustering techniques in software defect prediction (SDP). Seven clustering techniques; Farthest First Clusterer, K-Means, X-Means, Sequential information Bottleneck, Hierarchical Clusterer, Make-Density Clusterer, and Expectation Maximization were used for the classification of 8 software defect datasets from NASA repository. Experimental results revealed that the use of clustering technique as a classification process is well established as it gave a good predictive performance. Based on average accuracy across the 8 datasets, Farthest First had the best performance of 86.16%, Hierarchical clustering had 85.50% while K-Means Clustering techniques had 72.33% respectively. Expectation Maximization (EM) (33.52%) and X-Means (48.84%) gave rather poor results and Sequential Information bottleneck (SIB) (63%) and Density-based clustering techniques (71.08%) had average performances. In addition, further comparison of classification via clustering techniques with selected standard classification techniques; k-Nearest Neighbor (kNN), Naïve Bayes (NB), and Decision Tree (DT) showed that some classification via clustering techniques (Farthest First and Hierarchical Clustering Techniques) performed considerably well and outperforms some standard classification algorithms. With this, classification via clustering techniques can be considered as an alternative approach to standard classification methods in SDP. It produced good and competitive predictive performance in SDP with an advantage of not necessarily training a predictive model and using annotated datasets while developing the predictive model. Consequently, SDP models developed using classification via clustering techniques models can be transferred from one project to another as no training of model is involved. This will help reduce and manage the available resources during the software development process.

Keywords: *Classification Technique, Clustering Technique, Software Defects prediction, Software Engineering*

African Journal of Computing & ICT Reference Format:

Abdullateef O. Balogun, Rufus O. Oladele, Hammed A. Mojeed, Barakat Amin-Balogun, Victor E. Adeyemo and Taye O. Aro (2019), Performance Analysis of Selected Clustering Techniques for Software Defects Prediction, *Afr. J. Comp. & ICT*, Vol.12, No. 2, pp. 30 - 42.

© Afr. J. Comp. & ICT, June 2019; ISSN 2006-1781

I. INTRODUCTION

A defect in the software module occurs when incorrect programming logic or code are used to develop a software system. This further produces the wrong output and leads to a poor quality software product. Defects in the software modules may be rejected by a user or terminate the contracted agreement with the software company. The high cost of development and maintenance with customer dissatisfaction are the end results of defective software [1, 2]. Early detection of these software defects during software developments improves the reliability and quality of the software. This also helps to make better software maintenance within the period of time in a cost-effective way, which leads to prompt software release as well as high customer satisfaction [3]**.**

Presence of software defects leads to degradation of software quality which might be the underlying cause of failure. As a process, software quality assurance (SQA) ensures production of high quality and reliable software systems. This concept is drawn in throughout the software development life cycle (SDLC). Activities such as performance analysis, functional tests, quantifying time and budget along with measurement of metrics are used to ensure software quality based on this concept [4]. Software Metrics are tools generally used to analyze the process efficiency, reliability, and quality of software products. These metrics are effectively utilized for defect prediction which helps to identify and improve the quality of software project [5]. In a software system, every software module is described as a set of features (metrics value) and a class label. The class label denotes whether a module is defective and the derived metric values are used to build predictive models. In other words, software defect prediction (SDP) process utilizes historical data mined from software repositories to determine the quality of software modules for software quality assurance. Furthermore, SDP ensures software engineers pay adequate attention to software development activities which enhances the software quality and minimize the cost and time to develop software systems [6].

Researchers have proposed and developed several SDP prediction models which are used to filter the software defects [5, 7-9]. These models can be used to predict the response variable which can either be defective or not defective or a quality factor (for example number of defects) for a particular software module. Statistical methods, machine learning method, and soft computing techniques are widely used in literature to predict software faults [1, 2, 5, 10-12].

This paper is focused on how clustering techniques are used for SDP. Clustering involves finding natural groupings in a dataset [13]. Clustering algorithms can group software modules according to the values of their software metrics. Unsupervised learning methods such as clustering techniques should be seen as the primary choice for analyzing software quality in the absence of class labels. The underlying software engineering assumption is that defective class label of software modules will have similar software metrics and so will likely form clusters. Similarly, non-defective software modules will likely group together. A clustering approach offers practical benefits to software engineers who must decide the class labels. Instead of inspecting and labeling software modules one at a time, the expert can inspect and label a given cluster as a whole; he or she can assign all the modules in the cluster the same quality label. Selecting an appropriate clustering algorithm for SDP problem is worth investigating due to the nature of clustering algorithms not needing to train datasets unlike other classification and regression techniques [1].

II. RELATED WORKS

Although there are many existing kinds of literature on SDP, there are not so many extensive benchmarking studies regarding classification via clustering for SDP. The reason for this is that most existing studies consider only supervised methods for SDP. However, studies in other research domain such as network security, space weather, and decision systems have successfully used classification via clustering for prediction [12, 14, 15]. That is why good benchmarking studies are needed on classification via clustering for SDP.

Lopez, et al. [16], reported the use of classification via clustering methods in predicting final marks in a university course. Heidrich-Meisner and Wimmer-Schweingruber [15] in their study also deployed the k-means technique for solar wind classification. They tried to determine if classification via clustering methods can obtain similar performance as traditional classification algorithm in the context of educational data mining (EDM) and space weather. Their respective experimental results revealed that given sufficient data, classification via clustering method can be as good as traditional classification methods. However, these performances cannot be generalized since these studies were limited to EDM and space weather.

El-Manzalawy, et al. [17], in their study introduced a novel approach for integrating unsupervised learning algorithms and domain knowledge heuristics, based on statistical properties of clustered sleep and wake epochs, to develop reliable sleep/wake state prediction models using unlabelled wrist actigraphy data. Their results laid the groundwork for developing fully automated machine learning models for sleep/wake state prediction and sleep parameters estimations by eliminating the need for costly and labor-intensive expert annotations of PSG recordings

for labeling actigraphy data. Rana and Lal [12] also looked into the handling of missing attribute values using Rough Set Theory (RST) and classification via clustering techniques. Classifications via clustering techniques were used for predicting learning styles from a given dataset. Their results showed classification via clustering to be a formidable approach in such task. Furthermore, Yadav and Pal [18] in their study, integrated classification algorithms (J48 graft, LADTree, and BayesNet) with a clustering method (K-means). Their experimental results showed that the successful integration of classification algorithm (Bayes Net) with clustering algorithm (Kmeans) had a good performance.

In the context of SDP, to the best of our knowledge, no study has explored classification via clustering techniques in SDP. Hence, this study seeks to empirically validate and compare the performance of classification via clustering techniques for software defect prediction.

III. METHODOLOGY

This section entails the methodology used in this study. This study is aimed at evaluating the performance of clustering techniques as classifiers for SDP. The algorithms used are k-Means (KM), Expectation Maximization (EM), Density-Based clusterer, Farthest-First clusterer, Hierarchical clusterer, Sequential Information Bottleneck (SIB) clusterer, and X-Means clustering algorithm in software defect prediction.

3.1 Classification via Clustering Techniques

3.1.1 K-Means Clustering

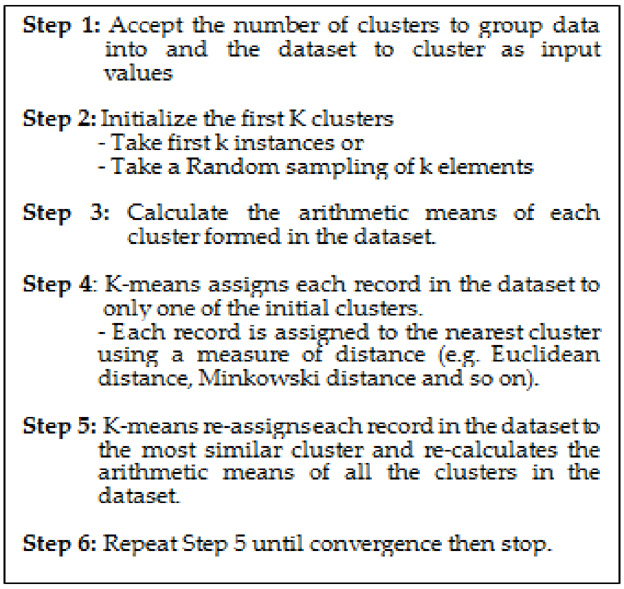
K-means is a well-known partitioning method where objects or entities are classified as belonging to one of the k groups with k chosen a priori. Each object is assigned to a group based on cluster membership which is determined by calculating the centroid for each group (the multidimensional version of the mean) and assigning each object to the group with the closest centroid [19]. The algorithm operates on a set of d-dimensional vectors, D = $\{x_i \mid i = 1, \ldots, N\}$, where $X_i \exists R^d$ denotes the i^{th} data point and R^d the dataset.

Techniques for selecting these initial seeds include sampling at random from the dataset, setting them as the solution of clustering a small subset of the data or perturbing the global mean of the data k times. Then the algorithm iterates between two phases till convergence:

Phase 1: Data Assignment. Each data point is assigned to its closest centroid, with ties broken arbitrarily. This results in a partitioning of the data.

Phase 2: Relocation of "means". Each cluster representative is relocated to the center (mean) of all data points assigned to it. If the data points come with a probability measure (weights), then the relocation is to the expectations (weighted mean) of the data partitions. The algorithm converges when the assignments no longer change.

Below is a generalized traditional K-means algorithm procedure:



```
Step 1: Accept the number of clusters to group data
        into and the dataset to cluster as input
        values

Step 2: Initialize the first K clusters
        - Take first k instances or
        - Take a Random sampling of k elements

Step 3: Calculate the arithmetic means of each
        cluster formed in the dataset.

Step 4: K-means assigns each record in the dataset to
        only one of the initial clusters.
        - Each record is assigned to the nearest cluster
        using a measure of distance (e.g. Euclidean
        distance, Minkowski distance and so on).

Step 5: K-means re-assigns each record in the dataset to
        the most similar cluster and re-calculates the
        arithmetic means of all the clusters in the
        dataset.

Step 6: Repeat Step 5 until convergence then stop.
```

Algorithm 1: Procedure for K-Means Algorithm [19]

3.1.2 Expectation Maximization Clustering

Expectation Maximization (EM) is a model-based approach to solving clustering problems. It is an iterative algorithm that is used in problems where data contains latent variables or is considered incomplete. Unlike distance based or hard membership algorithms (such as K-Means), EM is known to be an appropriate optimization algorithm for constructing proper statistical models of the data. EM is widely used in applications such as computer vision, recommender systems, and decision systems [20-22]. Unlike K- Means, in clustering via EM, the numbers of clusters that are desired are predetermined. It is initialized with values for unknown (hidden) variables. However, the EM algorithm works well on clustering data when the number of clusters is known [20]. EM is a very general algorithm for parameter estimation in the case where certain data are unknown. The inputs to this algorithm are the data set (x), the total number of clusters

(M), the accepted error to converge (e) and the maximum number of iterations. The algorithm can be subdivided into two stages, namely the initialization stage and the iterative stage which consists of two steps, expectation step (E-step) and a maximization step (M-step) executed iteratively until some form of convergence is reached. The E-Step estimates the probability of each point belonging to each cluster, followed by the M-step which re-estimates the parameter vector of the probability distribution of each

class. The algorithm finishes when the distribution parameters converge or reach the maximum number of iterations.

3.1.3 Farthest First Clustering

Farthest first clustering algorithm as proposed in [15] has the same procedure as k-means. It also chooses centroids and assigns the objects in the cluster but with its max distance and initial seeds as values that are at the largest distance to the mean of values. In this case, cluster assignment is different; at initial cluster, the link with high Session Count is derived, like at cluster-0 more than in cluster-1, and so on. Farthest first actually solves the problem of k-center and it is very efficient for a large set of data. In farthest first algorithm, the mean for calculating centroid will not be computed. It takes an arbitrary centroid and calculates the distance of one centroid from other as maximum cluster assignment using farthest –first. When outlier detection is performed on the dataset, objects that are outliers are detected. This places the cluster center at the point further from the present cluster. This point must lie within the data area. The points that are farther are clustered together first. This feature of farthest first clustering algorithm speeds up the clustering process in many situations as fewer reassignment and adjustment is needed.

The steps of the algorithm are as follows:

```
Step 1: Choose a random data as the center
        point first.
Step 2: Finding the data that is the farthest point
        from the first point
Step 3: Finding a third point which is the
        farthest point from two existing points.
        Henceforth i= 3, 4, …, n
Step 4: Converge and Stop when there are no
        more data points
```

Algorithm 2: Procedure for Farthest First Algorithm [23]

3.1.4 X-Means

X-Means is K-Means extended by an Improve-Structure Part. In this part of the algorithm, the centers are attempted to be split in its region. The decision between the children of each center and itself is done comparing the Bayesian Information Criterion (BIC) values of the two structures[24].

```
Step 1: Initialize K = $K_{min}$
Step 2: Rub K-Means algorithm
Step 3: FOR k = 1, . . . ,k: Replace each centroid $\mu_k$ by two
        centroids $\mu_{(1)}$ and $\mu_{(2)}$.
Step 4: Run K-means algorithm with K = 2 over the
        cluster k. Replace or retain each centroid based
        on the model selection
Step 5: IF convergence condition is not satisfied, go to
        Step (2). Otherwise Stop
```

Algorithm 3: Procedure for X-Means Algorithm [24]

3.1.5 Density-Based Clustering

Density-based clustering algorithm has played a vital role in finding non-linear shapes structure based on the density. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is the most widely used density-based algorithm. It uses the concept of density reachability and density connectivity [25]. In density-based clustering, clusters are identified by looking at the density of points. Regions with a high density of points depict the existence of clusters whereas regions with a low density of points indicate clusters of noise or clusters of outliers. A cluster is defined as a maximal set of density-connected points. The main feature of density-based clustering is that it discovers features of arbitrary shape and it can handle noise:

```
Input: DB: Database
Input: ε: Radius
Input: minPts: Density threshold
Input: dist: Distance function
Data: label: Point labels, initially undefined
foreach point p in database DB do
    if label (p) undefined then continue
    Neighbors N ← RangeQuery (DB, dist, p, ε)
    If | N | < minPts then
        label (p) ← Noise
        continue
    c ← next cluster label
    label (p) ← c
    Seed set S ← N \ {p}
    foreach q in S do
        if label (q) = Noise then label (q) ← c
        if label (q) undefined then continue
        Neighbors N ← RangeQuery (DB, dist, q, ε)
        label (q) ← c
        if | N | < minPts then continue
        S ← S ∪ N
return S
```

Algorithm 4: Procedure for DBSCAN Algorithm [26]

3.1.6 Sequential Information Bottleneck Clustering

The sequential information bottleneck method (SIB) produces crisp clusters, p(t\x)∃(0,1), where each document belongs to a single cluster. This corresponds to clustering maximizing the dependency between the clusters T and the words Y, measured by the mutual information (TY). It may, therefore, be seen as a margin-contingency table algorithm in this sense [18].

```
Input:
    |X| objects to be clustered
    Parameters: K, n, maxL, ε
Output:
    A partition T of X into K clusters
Main Loop:
    For i = 1, ..., n
        T_i ← random partition of X.
        c ← 0, C ← 0, done = FALSE
        While not done
            For j = 1, ..., |X|
                draw x_j out of t(x_j)
                t^new(x_j) = arg min_t' d_F({x_j}, t')
                If t^new(x_j) ≠ t(x_j) then c ← c + 1
                Merge x_j into t^new(x_j)
            C ← C + 1
            if C ≥ maxL or c ≤ ε · |X| then
                done ← TRUE
    T ← arg max_{T_i} F(T_i)
```

Algorithm 5: Pseudocode for Sequential Information Bottleneck Clustering Algorithm [27]

3.1.7 Hierarchical Clustering

In hierarchical clustering, the goal is not to find a single partitioning of the data, but a hierarchy (generally represented by a tree) of partitions which may reveal interesting structure in the data at multiple levels of granularity. The most widely used hierarchical methods are the agglomerative clustering techniques; most of these techniques start with a separate cluster for each point and then progressively merge the two closest clusters until only a single cluster remains. In all cases, we assume that we have a measure of similarity between pairs of objects, but the different schemes are distinguished by how they convert this into a measure of similarity between two clusters. For example, in the single linkage, the similarity between two clusters is the maximum similarity between points in these two different clusters. In a complete linkage, the similarity between two clusters is the minimum similarity between points in these two different clusters. Average linkage defines the similarity between two clusters as the average similarity between points in these two different clusters [28, 29]. Different algorithms can be given for the same hierarchical clustering method. However, a general agglomerative algorithm for hierarchical clustering may be described informally as follows:

```
Input: M: Dataset
Input: LMS: Linkage Methods
Result: Best Agglomeration Correspondence

Truncate (M);
dmatrix ← CalculateDistances (M)
bestAgglomeration ← { }
ccoef ← 0
foreach linkageMethod lm in LMS do
    linkageLevelList ← { }
    while Size(dmatrix)> 0 do
        linkageResult ← ComputeLevelLinkage (lm, dmatrix)
        linkageResult ← linkageLevelList ∪ {LinkageResult}
        UpdateDistanceMatrix (dmatrix, linkageResult)
    Restore(dmatrix)
    ccoefAux ← CompCopheneticMatrix (LinkageLevelList,
    dmatrix)
    if ccoef < ccoefAux then
        ccoef ← ccoeftAux
        bestAgglomeration ← linkageLevelList
return bestAgglomeration
```

Algorithm 6: Pseudocode for Hierarchical Clustering Algorithm[30]

The number of variables obviously affects the calculation time required, but they are usually considered constant for any particular set of data. In Steps 2 and 3, it might be worthwhile to consider keeping a sorted list of all dissimilarities under consideration (taking O (N2 log N) time).

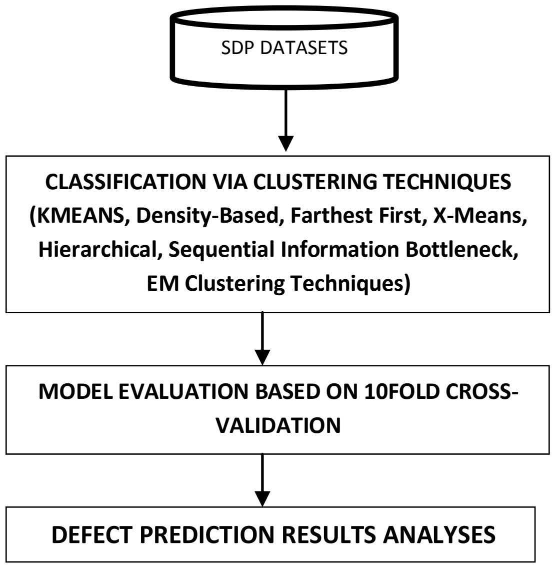

SDP DATASETS
CLASSIFICATION VIA CLUSTERING TECHNIQUES (KMEANS, Density-Based, Farthest First, X-Means, Hierarchical, Sequential Information Bottleneck, EM Clustering Techniques)
MODEL EVALUATION BASED ON 10FOLD CROSS-VALIDATION
DEFECT PREDICTION RESULTS ANALYSES

Fig 1. Experimental Architecture.

Fig 1 showed the experimental architecture used in this study. The SDP datasets were classified by the classification via clustering techniques (K-Means, Farthest-first, X-Means, Hierarchical, SIB, EM, and Make density Clusterer). The respective classification via clustering models was evaluated based on 10-fold cross-validation. Cross-validation method is used so as to avoid overfitting and in line with existing studies [2, 31]. The defect prediction results based on the performance metrics (accuracy, precision, and recall) were analyzed to empirically validate and compare the performance of classification via clustering methods in SDP. The algorithms were based on the Waikato Environment for Knowledge Analysis (WEKA) API library. Default parameter settings were used for both clustering and standard classification techniques.

3.2 Software Defect Datasets

The datasets used in this study are 8 public-domain software defect datasets provided by the National Aeronautics Space Administration (NASA) repository. The datasets used in this study are; KC1, KC2, KC3, MC1, MW1, PC1, PC3, and PC4 respectively. A brief description of these datasets is provided in Table 1.

Table 1: Software Defect Datasets

Dataset	Number of Instances	Number of Attributes
KC1	2109	22
KC2	522	22
KC3	458	40
MC1	161	40
MW1	403	38
PC1	679	38
PC3	1053	38
PC4	1270	38

3.4 PERFORMANCE EVALUATION METRICS

There are a number of ways to evaluate the performance of a classifier model but the commonly used performance metrics in SDP are accuracy, precision, and recall [11]. The metric values were all computed using the statistical values of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

- Accuracy: Accuracy is the percentage of correctly classified instances and it is one of the widely used measures.

$$Accuracy = (TP + TN)/(TP + FP + FN + TN) \quad (1)$$

-

- Precision: This is the number of classified fault-prone modules that actually are fault-prone modules.

$$Precision = TP/(TP + FP) \quad (2)$$

- Recall: This is the number of fault-prone modules that are correctly classified.

$$Recall = TP/(TP + FN) \quad (3)$$

IV. RESULTS AND DISCUSSIONS

4.1 Experimental Results

Tables 1, 2 and 3 present the prediction performance results for each of the classification via clustering algorithms with respect to accuracy, precision, and recall.

From Table 1, on the average, Farthest First clusterer had the best result of 86.16%, followed by hierarchical clustering technique with 85.50%. K-means also had a good result with 72.33% and Make Density clusterer had 71.08% accuracy on the average. EM and X-Means had the worst performance with an average accuracy of 33.52% and 48.84% respectively. SIB clusterer had an average performance with 63% average accuracy. Hierarchical clustering algorithm gave the best result in terms of accuracy over some of the datasets used (KC1 (84.46%), KC3 (90.39%) and MW1 (92.31%)), while Farthest first also gave a good result in some datasets too (PC4(86.90%), PC3(99.68%) and PC1(94.96%)).

However, in Table 2, Make density clusterer had the highest average precision value of 0.83. Farthest first clusterer had an average precision value of 0.8 while K-means and X-means both had 0.81 precision values respectively. Hierarchical clustering technique had 0.79, SIB clusterer had 0.78 and EM recorded 0.72 average precision values respectively. Table 3 presents the performance of the clustering techniques based on their respective recall values. Clearly, both farthest first and hierarchical clusterer had the highest average recall values of 0.86 respectively. EM also had an average recall value of 0.77.

This further strengthens the point that low accuracy values do not necessarily mean a predictive model is bad as in the case of EM with an average accuracy of 33.52%, average precision of 0.72 and average recall of 0.77. Same also can be said in the case of X-means clusterer with an average accuracy of 48.84%, average precision of 0.81 and average recall of 0.68. In relations to the classification via clustering techniques having average precision values greater than 0.7 and average recall value greater than 0.65 respectively, it can be stated that the respective models from this clustering techniques are of a good fit. Though, some of them had poor accuracy value. This can be attributed to the data quality problems with the SDP datasets such as high dimensionality and class imbalance.

Moreover, farthest first and hierarchical clustering techniques both had good accuracy, precision and recall values.

The aforementioned findings indicated that classification via clustering technique can also work well for SDP most especially hierarchical and farthest first clustering algorithms.

Table 1: Accuracy Results of each Clustering Technique Algorithm

CLUSTERERS	**KC1**	**KC2**	**KC3**	**MC2**	**MW1**	**PC1**	**PC3**	**PC4**	**Average**
EM	31.06	40.42	45.63	35.40	25.81	23.62	37.68	28.53	33.52
HIERARCHICAL	84.46	79.70	90.39	68.32	92.31	93.24	88.70	86.90	85.50
MAKE-DENSITY	77.96	64.41	54.49	69.09	81.02	86.74	65.35	69.56	71.08
SIB	52.96	58.43	55.24	44.10	60.05	80.88	73.70	78.60	63.00
K-MEANS	81.46	80.65	60.26	62.73	81.64	88.46	65.64	57.82	72.33
X-MEANS	45.76	46.94	42.95	49.37	56.20	64.19	42.00	43.35	48.84
FARTHEST FIRST	83.12	80.23	88.46	65.45	90.51	94.96	99.68	86.90	**86.16**

Table 2: Precision Results of each Clustering Technique Algorithm

CLUSTERERS	**KC1**	**KC2**	**KC3**	**MC2**	**MW1**	**PC1**	**PC3**	**PC4**	**Average**
EM	0.70	0.79	0.69	0.62	0.73	0.54	0.86	0.84	0.72
HIERARCHICAL	0.72	0.84	0.62	0.78	0.85	0.94	0.79	0.76	0.79
MAKE-DENSITY	0.81	0.85	0.90	0.69	0.90	0.93	0.77	0.83	**0.83**
SIB	0.86	0.82	0.76	0.54	0.82	0.87	0.79	0.75	0.78
K-MEANS	0.82	0.84	0.91	0.59	0.90	0.90	0.80	0.75	0.81
X-MEANS	0.75	0.85	0.89	0.60	0.90	0.94	0.79	0.75	0.81
FARTHEST FIRST	0.69	0.64	0.82	0.78	0.84	0.91	0.99	0.76	0.80

Table 3: Recall Results of each Clustering Technique Algorithm

CLUSTERERS	**KC1**	**KC2**	**KC3**	**MC2**	**MW1**	**PC1**	**PC3**	**PC4**	**Average**
EM	0.89	0.87	0.93	0.62	0.88	0.63	0.67	0.63	0.77
HIERARCHICAL	0.85	0.80	0.91	0.68	0.92	0.93	0.89	0.87	**0.86**
MAKE-DENSITY	0.79	0.64	0.55	0.69	0.81	0.87	0.65	0.70	0.71
SIB	0.53	0.58	0.67	0.44	0.66	0.81	0.74	0.85	0.66
K-MEANS	0.82	0.81	0.60	0.63	0.82	0.89	0.66	0.58	0.72
X-MEANS	0.74	0.59	0.53	0.61	0.80	0.71	0.75	0.71	0.68
FARTHEST FIRST	0.83	0.80	0.89	0.66	0.92	0.95	1.00	0.87	**0.86**

Table 4: Performance Comparison of Classification via clustering methods and standard classification methods based on Accuracy

CLUSTERERS	**KC1**	**KC2**	**KC3**	**MC2**	**MW1**	**PC1**	**PC3**	**PC4**	**Average**
EM	31.06	40.42	45.63	35.40	25.81	23.62	37.68	28.53	33.52
HIERARCHICAL	84.46	79.70	**90.39**	68.32	**92.31**	93.24	88.70	86.90	85.50
MAKE-DENSITY	77.96	64.41	54.49	69.09	81.02	86.74	65.35	69.56	71.08
SIB	52.96	58.43	55.24	44.10	60.05	80.88	73.70	78.60	63.00
SIMPLE K-MEANS	81.46	80.65	60.26	62.73	81.64	88.46	65.64	57.82	72.33
X-MEANS	45.76	46.94	42.95	49.37	56.20	64.19	42.00	43.35	48.84
FARTHEST FIRST	83.12	80.23	88.46	65.45	90.51	**94.96**	**99.68**	86.90	**86.16**
KNN	84.40	80.46	87.77	67.08	88.09	92.06	87.46	87.11	84.30
NB	82.36	**83.52**	85.15	**73.91**	83.87	89.18	48.69	87.04	79.22
DT	**84.54**	81.42	88.86	68.94	91.56	93.33	88.36	**88.13**	85.64

4.2 Results Discussion

Comparing the clustering algorithms with other classifiers used for SDP, it is quite evident that some of the clustering algorithms outperform some of the classifiers. From Table 4, on datasets KC3 and MW1, Hierarchical-clustering technique had the best accuracy value (90.39% and 92.31%). Farthest First clusterer gave the best accuracy result on PC1 (94.96%) and PC3 (99.68%) respectively. On the individual dataset, DT had the best accuracy value (84.54%) on KC1 dataset. On KC2 dataset, k-Means gave an accuracy result of 80.65% respectively, which is less than the accuracy of NB (83.52%) for that dataset. NB also had an accuracy of 73.91% in MC2 dataset, which is the highest of all algorithms, applied to the dataset.

On PC4 dataset, DT outperforms all clustering algorithms with an accuracy result of 88.13% with the best clustering algorithm in this case Farthest First and Hierarchical clusterer both having an accuracy of 86.90%. However, on the average, Farthest First Clusterer had the highest accuracy value of 86.16%, followed by DT (85.64%) and then hierarchical clusterer (85.50%). In this case, the clustering techniques are as efficient as the standard classifiers. This, in turn, means that the classification via clustering technique can be used in place or alongside other classifiers in SDP since they gave good and competitive performance results in the experiments.

5. CONCLUSION AND FUTURE WORKS

There are different algorithms for classification processes via clustering techniques. However, every algorithm has its own advantages and disadvantages. It is obvious that some algorithms have higher accuracy than others which makes it necessary to point out the difference. In this study, clustering techniques such as EM, Hierarchical, X-means, k-means, SIB, Farthest First, and Make density clusterer were applied on eight NASA Datasets (KC1, KC2, KC3, MC2, MW1, PC1, PC3, and PC4). From the experimental results and comparison with standard classifiers, it can be concluded that clustering techniques can work well for SDP but caution must be exercised on the choice of a clustering algorithm. Hierarchical and Farthest first clusterer can be deployed for SDP since they gave a very good result in the experiments. The researchers intend to extend this study by investigating the effect of feature selection on classification via clustering technique and also deploying multi-criteria decision-making methods on their respective results.

REFERENCES

[1] G. Abaei, A. Selamat, and H. Fujita, "An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction," *Knowledge-Based Systems,* vol. 74, pp. 28-39, 2015.

[2] A. G. Akintola, A. O. Balogun, F. Lafenwa-Balogun, and H. A. Mojeed, "Comparative analysis of selected heterogeneous classifiers for software defects prediction using filter-based feature selection methods," *FUOYE Journal of Engineering and Technology,* vol. 3, no. 1, pp. 134-137, 2018.

[3] N. Fenton and J. Bieman, *Software metrics: a rigorous and practical approach*. CRC press, 2014.

[4] R. Malhotra, *Empirical research in software engineering: concepts, analysis, and applications*. Chapman and Hall/CRC, 2016.

[5] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Information Sciences,* vol. 264, pp. 260-278, 2014.

[6] E. J. Braude and M. E. Bernstein, *Software engineering: modern approaches*. Waveland Press, 2016.

[7] A. O. Balogun, A. O. Bajeh, V. A. Orie, and A. W. Yusuf-Asaju, "Software Defect Prediction Using Ensemble Learning: An ANP Based Evaluation Method," *FUOYE Journal of Engineering and Technology,* vol. 3, no. 2, 2018.

[8] R. Jimoh, A. Balogun, A. Bajeh, and S. Ajayi, "A PROMETHEE based evaluation of software defect predictors," *Journal of Computer Science and Its Application,* vol. 25, no. 1, pp. 106-119, 2018.

[9] J. Nam and S. Kim, "Clami: Defect prediction on unlabeled datasets (t)," in *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on,* 2015, pp. 452-463: IEEE.

[10] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Transactions on Software Engineering,* vol. 34, no. 4, pp. 485-496, 2008.

[11] M. Mabayoje, A. Balogun, A. Bajeh, and B. Musa, "SOFTWARE DEFECT PREDICTION: EFFECT OF FEATURE SELECTION AND ENSEMBLE METHODS," *FUW Trends in Science & Technology Journal,* vol. 3, no. 2, pp. 518-522, 2018.

[12] H. Rana and M. Lal, "A comparative study based on rough set and classification via clustering approaches to handle incomplete data to predict learning styles," *International Journal of Decision Support System Technology (IJDSST),* vol. 9, no. 2, pp. 1-20, 2017.

[13] A. Balogun, M. Mabayoje, S. Salihu, and S. Arinze, "Enhanced Classification Via Clustering Using Decision Tree for Feature Selection," *International Journal of Applied Information Systems (IJAIS),* vol. 9, no. 6, pp. 11-16, 2015.

[14] A. O. Balogun, A. M. Balogun, V. E. Adeyemo, and P. O. Sadiku, "A Network Intrusion Detection System: Enhanced Classification via Clustering Model," *Computing, Information System Development Informatics & Allied Research Journals,* vol. 6, no. 4, pp. 53-58, 2015.

[15] V. Heidrich-Meisner and R. F. Wimmer-Schweingruber, "Solar Wind Classification Via k-Means Clustering Algorithm," in *Machine Learning Techniques for Space Weather*: Elsevier, 2018, pp. 397-424.

[16] M. I. Lopez, J. Luna, C. Romero, and S. Ventura, "Classification via clustering for predicting final marks based on student participation in forums," *International Educational Data Mining Society,* 2012.

[17] Y. El-Manzalawy, O. Buxton, and V. Honavar, "Sleep/wake state prediction and sleep parameter estimation using unsupervised classification via

clustering," in *Bioinformatics and Biomedicine (BIBM), 2017 IEEE International Conference on*, 2017, pp. 718-723: IEEE.

[18] D. Yadav and S. Pal, "An Integration of Clustering and Classification Technique in Software Error Detection," *African Journal of Computing & ICT,* vol. 8, no. 2, pp. 2006-1781, 2015.

[19] R. Marapareddy and S. W. Saripalli, "Runway Detection Using K-Means Clustering Method Using UAVSAR Data," in *Proceedings of the International Conference on Artificial Intelligence and Applications (AIAPP), Geneva, Switzardland*, 2017, pp. 25-26.

[20] J. Garriga, J. R. Palmer, A. Oltra, and F. Bartumeus, "Expectation-maximization binary clustering for behavioural annotation," *PLoS One,* vol. 11, no. 3, p. e0151984, 2016.

[21] A. Kujawińska, M. Rogalewicz, and M. Diering, "Application of expectation maximization method for purchase decision-making support in welding branch," *Management and Production Engineering Review,* vol. 7, no. 2, pp. 29-33, 2016.

[22] M. Nilashi, O. bin Ibrahim, N. Ithnin, and N. H. Sarmin, "A multi-criteria collaborative filtering recommender system for the tourism domain using Expectation Maximization (EM) and PCA–ANFIS," *Electronic Commerce Research and Applications,* vol. 14, no. 6, pp. 542-562, 2015.

[23] A. Dharmarajan and T. Velmurugan, "Lung cancer data analysis by k-means and farthest first clustering algorithms," *Indian Journal of Science and Technology,* vol. 8, no. 15, 2015.

[24] M. Herald, "X-Means Clustering Implementing the Gap Statistic for Multiple Positron Emission Particle Tracking," 2018.

[25] R. D. H. Devi and P. Deepika, "Performance comparison of various clustering techniques for diagnosis of breast cancer," in *Computational Intelligence and Computing Research (ICCIC), 2015 IEEE International Conference on*, 2015, pp. 1-5: IEEE.

[26] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN," *ACM Transactions on Database Systems (TODS),* vol. 42, no. 3, p. 19, 2017.

[27] P. Gayathri, S. Punitha, and M. Punithavalli, "Document clustering using sequential information bottleneck method," *arXiv preprint arXiv:1004.1796,* 2010.

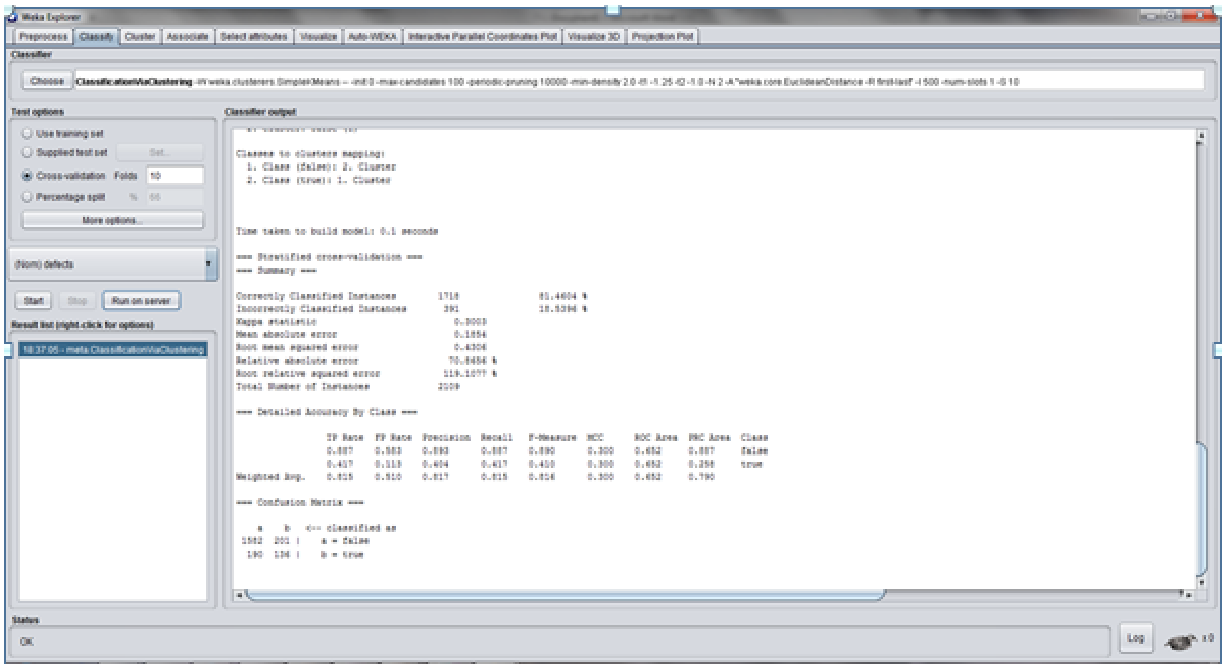
[28] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, and A. Song, "Efficient agglomerative hierarchical clustering," *Expert Systems with Applications,* vol. 42, no. 5, pp. 2785-2797, 2015.

[29] N. Gillis, D. Kuang, and H. Park, "Hierarchical clustering of hyperspectral images using rank-two nonnegative matrix factorization," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 53, no. 4, pp. 2066-2078, 2015.

[30] R. Romero, E. Iglesias, and L. Borrajo, "A linear-RBF multikernel SVM to classify big text corpora," *BioMed research international*, vol. 2015, 2015.

[31] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology,* vol. 58, pp. 388-402, 2015.

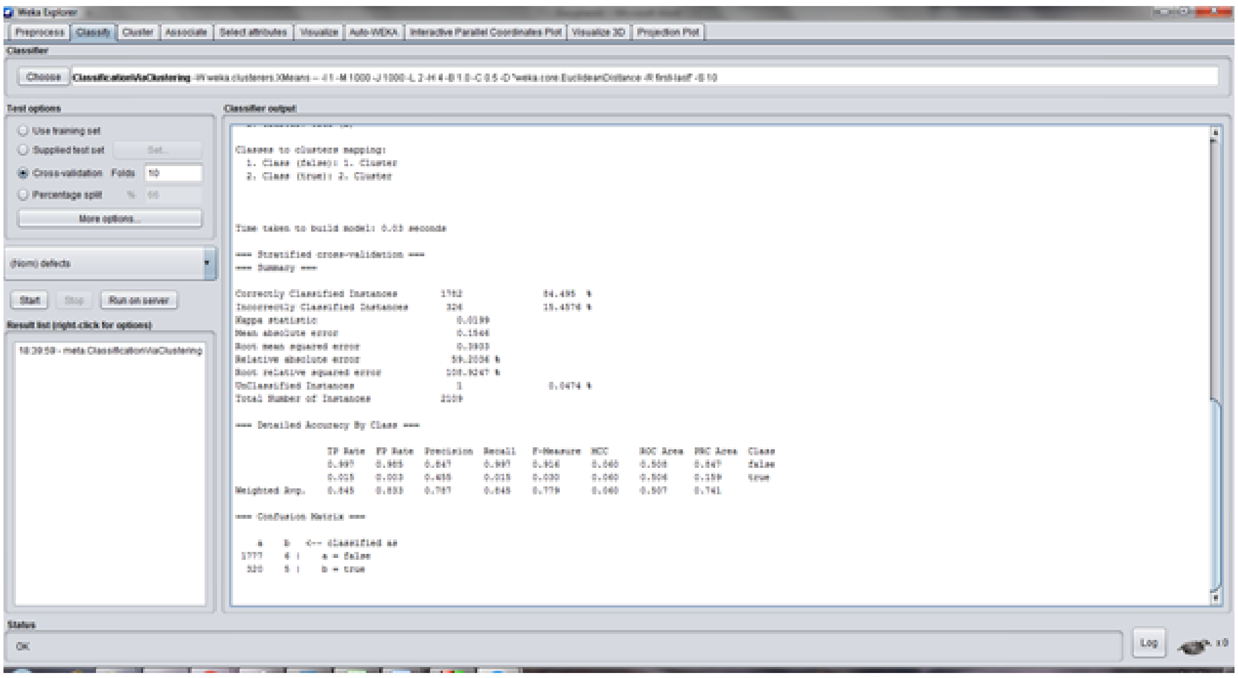
APPENDIX A: SELECTED SCREENSHOTS OF THE EXPERIMENTS USING WEKA DATA MINING TOOL



```
ClassificationViaClustering -W weka.clusterers.SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
```

```
Classes to clusters mapping:
  1. Class (false): 2. Cluster
  2. Class (true): 1. Cluster



Time taken to build model: 0.1 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1718               81.4604 %
Incorrectly Classified Instances       391               18.5396 %
Kappa statistic                          0.3003
Mean absolute error                      0.1854
Root mean squared error                  0.4306
Relative absolute error                 70.8456 %
Root relative squared error            119.1077 %
Total Number of Instances             2109

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.887    0.583    0.893      0.887    0.890      0.300    0.652     0.887     false
                 0.417    0.113    0.404      0.417    0.410      0.300    0.652     0.258     true
Weighted Avg.    0.815    0.510    0.817      0.815    0.816      0.300    0.652     0.790

=== Confusion Matrix ===

    a    b   <-- classified as
 1582  201 |    a = false
  190  136 |    b = true
```

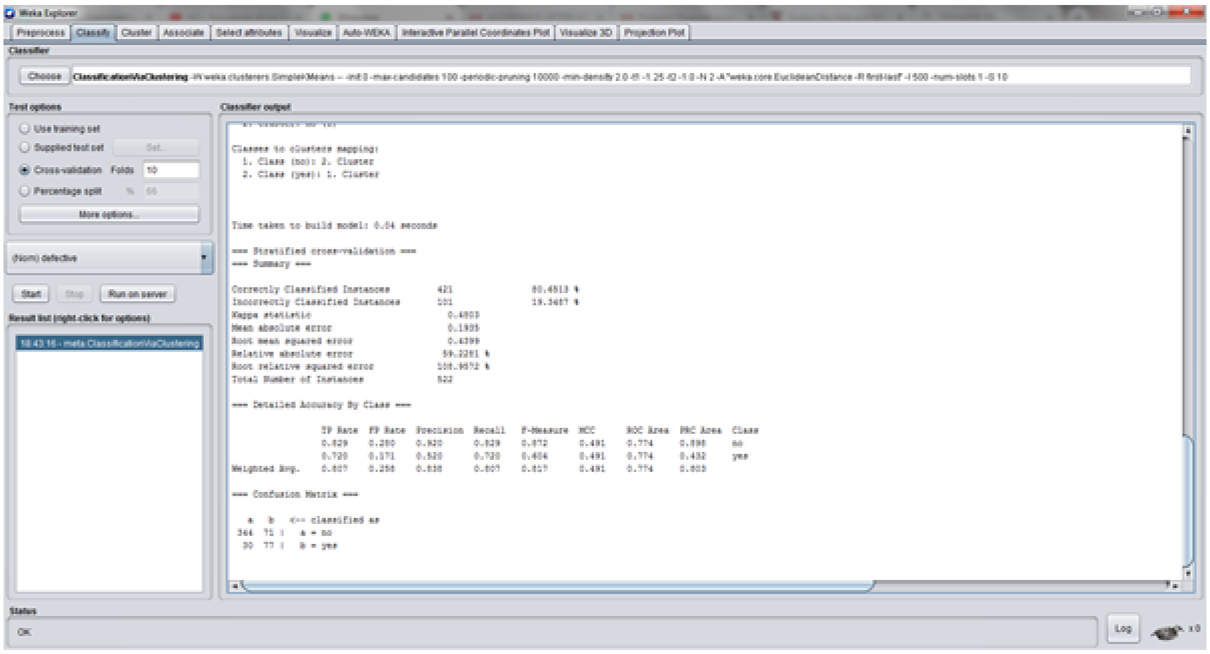
Diagram 1: Kmeans clustering algorithm on KC1 dataset



```
ClassificationViaClustering -W weka.clusterers.XMeans -- -I 1 -M 1000 -J 1000 -L 2 -H 4 -B 1.0 -C 0.5 -D "weka.core.EuclideanDistance -R first-last" -S 10
```

```
Classes to clusters mapping:
  1. Class (false): 1. Cluster
  2. Class (true): 2. Cluster



Time taken to build model: 0.03 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1782               84.495  %
Incorrectly Classified Instances       326               15.4576 %
Kappa statistic                          0.0199
Mean absolute error                      0.1546
Root mean squared error                  0.3933
Relative absolute error                 59.2036 %
Root relative squared error            108.9247 %
UnClassified Instances                   1                0.0474 %
Total Number of Instances             2109

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.997    0.985    0.847      0.997    0.916      0.060    0.508     0.847     false
                 0.015    0.003    0.455      0.015    0.030      0.060    0.506     0.159     true
Weighted Avg.    0.845    0.833    0.787      0.845    0.779      0.060    0.507     0.741

=== Confusion Matrix ===

    a    b   <-- classified as
 1777    4 |    a = false
  320    5 |    b = true
```

Diagram 2: Xmeans clustering algorithm on KC1 dataset

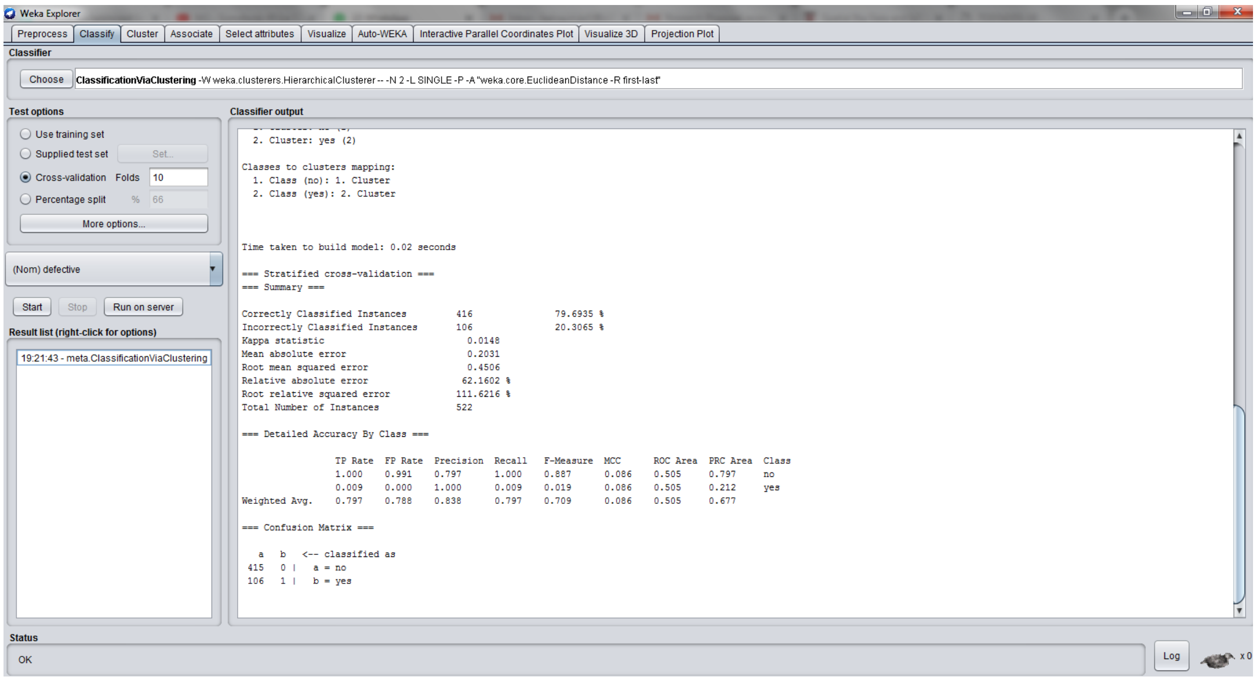

Classifier: ClassificationViaClustering -W weka.clusterers.SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Test options: Cross-validation Folds 10
(Nom) defective
Result list: 18:43:16 - meta.ClassificationViaClustering

```
Classes to clusters mapping:
  1. Class (no): 2. Cluster
  2. Class (yes): 1. Cluster



Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         421               80.6513 %
Incorrectly Classified Instances       101               19.3487 %
Kappa statistic                          0.4503
Mean absolute error                      0.1935
Root mean squared error                  0.4399
Relative absolute error                 59.2281 %
Root relative squared error            108.9572 %
Total Number of Instances              522

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.829    0.280    0.920      0.829    0.872      0.491    0.774     0.898     no
                 0.720    0.171    0.520      0.720    0.604      0.491    0.774     0.432     yes
Weighted Avg.    0.807    0.258    0.838      0.807    0.817      0.491    0.774     0.803

=== Confusion Matrix ===

   a   b   <-- classified as
 344  71 |   a = no
  30  77 |   b = yes
```

Status: OK

Diagram 3: Kmeans clustering algorithm on KC2 dataset

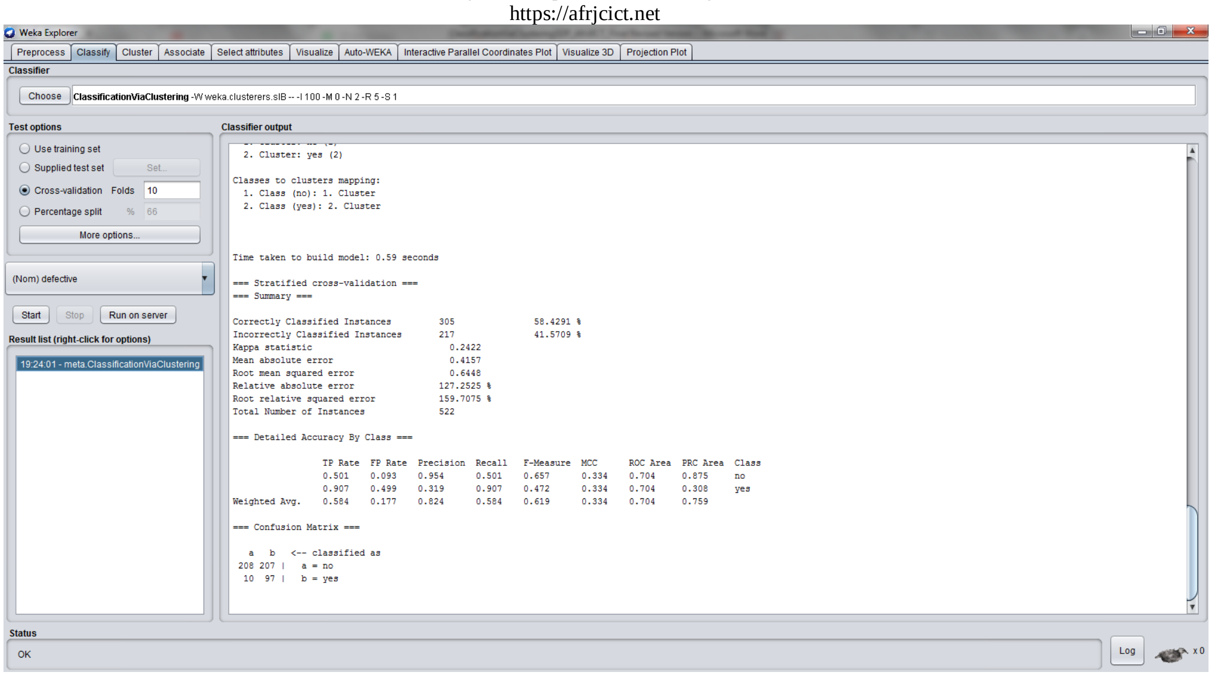

Classifier: ClassificationViaClustering -W weka.clusterers.HierarchicalClusterer -- -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"

Test options: Cross-validation Folds 10
(Nom) defective
Result list: 19:21:43 - meta.ClassificationViaClustering

```
  2. Cluster: yes (2)

Classes to clusters mapping:
  1. Class (no): 1. Cluster
  2. Class (yes): 2. Cluster



Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         416               79.6935 %
Incorrectly Classified Instances       106               20.3065 %
Kappa statistic                          0.0148
Mean absolute error                      0.2031
Root mean squared error                  0.4506
Relative absolute error                 62.1602 %
Root relative squared error            111.6216 %
Total Number of Instances              522

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.991    0.797      1.000    0.887      0.086    0.505     0.797     no
                 0.009    0.000    1.000      0.009    0.019      0.086    0.505     0.212     yes
Weighted Avg.    0.797    0.788    0.838      0.797    0.709      0.086    0.505     0.677

=== Confusion Matrix ===

   a   b   <-- classified as
 415   0 |   a = no
 106   1 |   b = yes
```

Status: OK

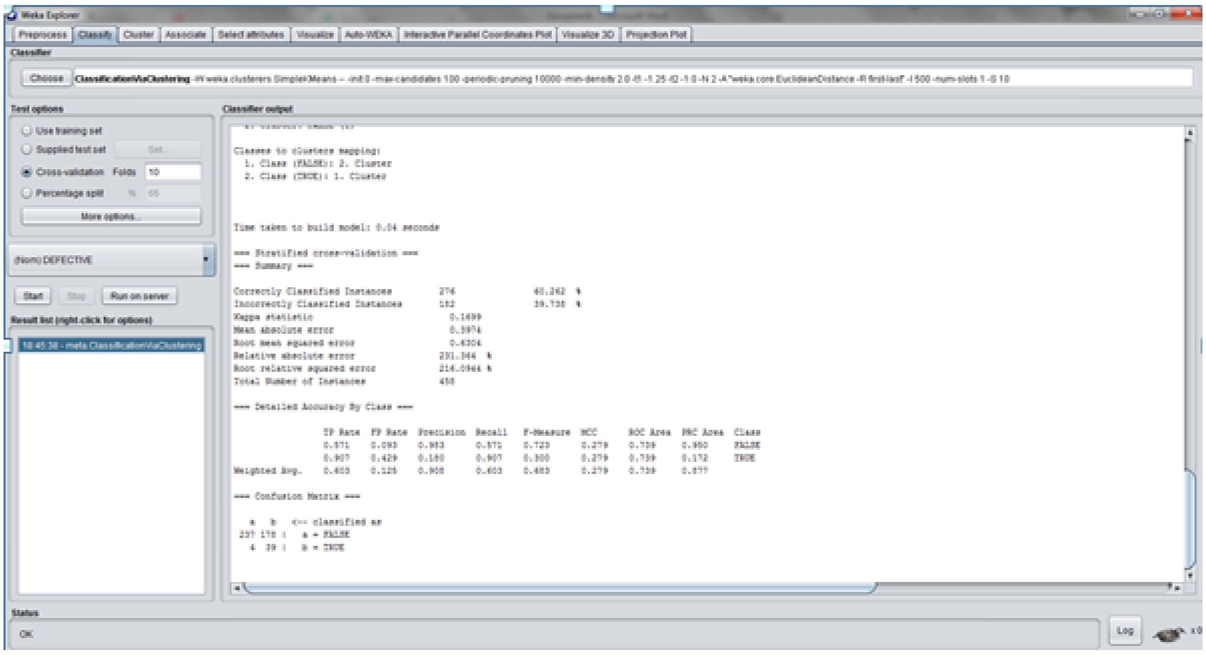
Diagram 4: Hierarchical clustering algorithm on KC2 dataset



ClassificationViaClustering -W weka.clusterers.sIB -- -I 100 -M 0 -N 2 -R 5 -S 1

```
  2. Cluster: yes (2)

Classes to clusters mapping:
  1. Class (no): 1. Cluster
  2. Class (yes): 2. Cluster



Time taken to build model: 0.59 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         305               58.4291 %
Incorrectly Classified Instances       217               41.5709 %
Kappa statistic                          0.2422
Mean absolute error                      0.4157
Root mean squared error                  0.6448
Relative absolute error                127.2525 %
Root relative squared error            159.7075 %
Total Number of Instances              522

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.501    0.093    0.954      0.501    0.657      0.334    0.704     0.875     no
                 0.907    0.499    0.319      0.907    0.472      0.334    0.704     0.308     yes
Weighted Avg.    0.584    0.177    0.824      0.584    0.619      0.334    0.704     0.759

=== Confusion Matrix ===

   a   b   <-- classified as
 208 207 |   a = no
  10  97 |   b = yes
```

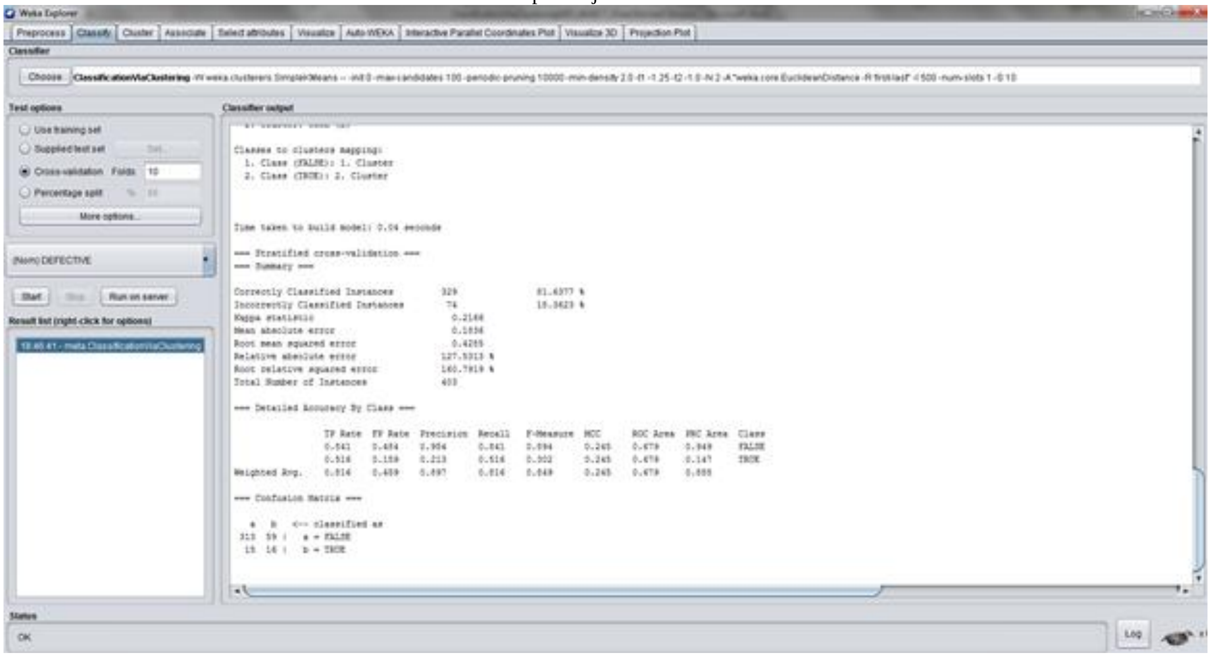
Diagram 5: Sequential Information Bottleneck (SIB) clustering algorithm on KC2 dataset



ClassificationViaClustering -W weka.clusterers.SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

```
Classes to clusters mapping:
  1. Class (FALSE): 2. Cluster
  2. Class (TRUE): 1. Cluster



Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         276               60.262  %
Incorrectly Classified Instances       182               39.738  %
Kappa statistic                          0.1499
Mean absolute error                      0.3974
Root mean squared error                  0.6304
Relative absolute error                231.344  %
Root relative squared error            214.0944 %
Total Number of Instances              458

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.571    0.093    0.983      0.571    0.723      0.279    0.739     0.950     FALSE
                 0.907    0.429    0.180      0.907    0.300      0.279    0.739     0.172     TRUE
Weighted Avg.    0.603    0.125    0.908      0.603    0.683      0.279    0.739     0.877

=== Confusion Matrix ===

   a   b   <-- classified as
 237 178 |   a = FALSE
   4  39 |   b = TRUE
```

Diagram 6: Kmeans clustering algorithm on KC3 dataset



Classifier: ClassificationViaClustering -W weka.clusterers.SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Test options: Cross-validation Folds 10

(Nom) DEFECTIVE

Result list: 18:46:41 - meta.ClassificationViaClustering

```
Classes to clusters mapping:
  1. Class (FALSE): 1. Cluster
  2. Class (TRUE): 2. Cluster



Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         329               81.6377 %
Incorrectly Classified Instances        74               18.3623 %
Kappa statistic                          0.2166
Mean absolute error                      0.1836
Root mean squared error                  0.4285
Relative absolute error                127.5313 %
Root relative squared error            160.7819 %
Total Number of Instances              403

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.841    0.484    0.954      0.841    0.894      0.245    0.679     0.949     FALSE
                 0.516    0.159    0.213      0.516    0.302      0.245    0.679     0.147     TRUE
Weighted Avg.    0.816    0.459    0.897      0.816    0.848      0.245    0.679     0.888

=== Confusion Matrix ===

   a   b   <-- classified as
 313  59 |   a = FALSE
  15  16 |   b = TRUE
```

Status: OK

Diagram 7: Kmeans clustering algorithm on MW1 dataset