# Improved point center algorithm for k-means clustering to increase software defect prediction

Riski Annisa [a,b,1,*], Didi Rosiyadi [a,b,c,2], Dwiza Riana [a,3]

[a] Computer Science Master Program of Sekolah Tinggi Manajemen Informatika dan Komputer Nusa Mandiri, Jakarta, Indonesia
[b] Universitas Bina Sarana Informatika, Jakarta, Indonesia
[c] Research Center for Informatics, Indonesian Institute of Sciences (LIPI), Bandung 40135, Indonesia
[1] riski.rnc@bsi.ac.id; [2] didi.rosiyadi@lipi.go.id; [3] dwiza@nusamandiri.ac.id
* corresponding author

ARTICLE INFO

**Article history**
Received March 20, 2020
Revised June 8, 2020
Accepted June 22, 2020
Available online November 30, 2020

**Keywords**
Algorithm
K-means
Cluster
Centroid
Software defect

ABSTRACT

The K-Means is a clustering algorithm that is often and easy to use. This algorithm is susceptible to randomly chosen centroid points so that it cannot produce optimal results. This research aimed to improve the K-Means algorithm's performance by applying a proposed algorithm called point center. The proposed algorithm overcame the random centroid value in K-Means and then applied it to predict software defects modules' errors. The point center algorithm was proposed to determine the initial centroid value for the K-Means algorithm optimization. Then, the selection of X and Y variables determined the cluster center members. The ten datasets were used to perform the testing, of which nine datasets were used for predicting software defects. The proposed center point algorithm showed the lowest errors. It also improved the K-Means algorithm's performance by an average of 12.82% cluster errors in the software compared to the centroid value obtained randomly on the simple K-Means algorithm. The findings are beneficial and contribute to developing a clustering model to handle data, such as to predict software defect modules more accurately.

## 1. Introduction

The current era's characteristic is constant technological advances and information that software is everywhere like Web, mobile, desktop, embedded, or software developed to help achieve goals more easily, quickly, and efficiently [1]. These advances cause the software system to be bigger and more complex than before, so it is necessary to prevent software defects. Therefore, predicting the number of defects in a software module is needed and can help developers allocate limited resources [2]. Furthermore, the predictions software modules' results are categorized as fault-prone and non-fault-prone [2]–[7].

Software defect prediction utilizes software metrics and fault data that originate from previous versions of the current software project or can be retrieved from other similar software projects. Data software contains learning models that significantly influence the efficacy of software Defect Prediction techniques [5][8]. A study investigating 3747 defects from 70 software systems developed by 29 Chinese aviation organizations showed 87% of software defects [9].

Unsupervised machine learning is increasingly being applied to software defect predictions. It is a useful approach for software practitioners because it reduces the need for labeled training data. Various software defect prediction models have been proposed to improve software quality over the past few years, which is increasingly popular using machine learning. This approach can be divided into supervised methods where training data require labels and unsupervised methods, where data does not need to be labeled [10].

Many defect prediction approaches have been proposed; the majority of studies are on defect prediction techniques [7][11]–[18]. Clustering analysis belongs to the unsupervised machine learning technique of patterns into groups. It is widely used in many fields, such as data mining, machine learning, pattern recognition, and image processing. The K-Means algorithm is often used among clustering algorithms because of its simplicity and efficiency [19][20].

One of the clustering grouping techniques is partitional clustering. The most widely used partitional clustering algorithm is the K-Means cluster, where there is n number of instances partitioned into k clusters. An optimal centroid is selected for each cluster to located nearby group instances [21][22]. K-Means begins by choosing the random data point k as the initial set of centroids, which is then increased by the next two steps. Next, each point is inserted into the nearest centroid cluster. Each cluster's center is recalculated as the average of all data points assigned to the cluster [23]–[25]. However, this method's main problem is not ensuring optimal results due to the selection of randomly selected centroid [19][26].

In this study, an algorithm called point center for K-Means clustering was proposed to overcome early random centroid and focus on problems that occur when software data fails for the software's cluster module's error. The proposed point center algorithm finds the initial centroid of the K-Means algorithm, then applied to predict the software defect module's error. The overall error rate of this prediction approach was compared to the K-Means algorithm with the random centroid. The proposed approach was used to get the best cluster center value of the K-Means algorithm to prove its effectiveness.

## 2. Method

### 2.1. Data and experimental design

This study uses NASA MDP datasets because it is very commonly used for predictive software defects and can be obtained in the PROMISE repository. From 2000 to 2013 for 64.79% of software defects research using the NASA MDP dataset [27]. Each NASA MDP dataset consists of several software modules and attributes characteristics. Modules that contain defects are categorized as prone faults, and non-defective ones are categorized as non-fault prone. However, they also consist of McCabe and Halstead complexity attributes in Table 1.

In addition, apart from using NASA MDP datasets (CM1, KC1, KC3, MC2, MW1, PC1, PC2, PC3, PC4), this study also uses iris datasets [28] that often used for clustering algorithm testing. It has three classes (Setosa, Viginica, and Versicolor) and 150 samples. Each class is divided into 50 class data and has four attributes (sepal length, sepal width, petal length, and petal width).

The experiments were using a computer to perform the process of calculation of the proposed method. The hardware and operating system specifications were a DELL laptop with Intel Core (TM) processor i5-3340M CPU @ 2.70GHz, 4.00 GB (RAM) memory, and the Windows 10 Pro operating system 64-bit. Simultaneously, the tools used in this study include Microsoft Excel, RapidMiner, and Rstudio.

### 2.2. Point center algorithm

The K-Means algorithm has a weakness in determining the value of random centroid so that the results are less optimal. This study proposed the algorithm to determine the K-Means centroid's value named point center. This algorithm is based on selecting variables X and Y to determine cluster members. For variable selection, the first stage calculates each attribute's average using Equation (1).

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad j = 1,2,\dots,p \tag{1}$$

where, $\bar{x}_j$ is the average of each attribute ($j$ is attribute), $x_i$ is a data point ($i$ is the data point of the 1 to $n$), and $n$ is the amount of data. Then calculate each standard deviation of each attribute through (2).

$$S(\bar{x}_j) = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}} \tag{2}$$

**Table 1.** List of NASA MDP Attribute

| Atribut | | Dataset NASA MDP | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | *CM1* | *KC1* | *KC3* | *MC2* | *MW1* | *PC1* | *PC2* | *PC3* | *PC4* |
| **LOC counts** | LOC_total | x | x | x | x | x | x | x | x | x |
| | LOC_blank | x | x | x | x | x | x | | x | x |
| | LOC_code_and_comment | x | x | x | x | x | x | x | x | x |
| | LOC_comments | x | x | x | x | x | x | x | x | x |
| | LOC_executable | x | x | x | x | x | x | x | x | x |
| | Number_of_lines | x | | x | x | x | x | x | x | x |
| **Halstead** | Content | x | x | x | x | x | x | x | x | x |
| | Difficulty | x | x | x | x | x | x | x | x | x |
| | Effort | x | x | x | x | x | x | x | x | x |
| | Error_est | x | x | x | x | x | x | x | x | x |
| | Length | x | x | x | x | x | x | x | x | x |
| | Level | x | x | x | x | x | x | x | x | x |
| | Prog_time | x | x | x | x | x | x | x | x | x |
| | Volume | x | x | x | x | x | x | x | x | x |
| | Num_operands | x | x | x | x | x | x | x | x | x |
| | Num_operators | x | x | x | x | x | x | x | x | x |
| | Num_unique_operands | x | x | x | x | x | x | x | x | x |
| | Num_unique_operators | x | x | x | x | x | x | x | x | x |
| **McCabe** | Cyclomatic_complexity | x | x | x | x | x | x | x | x | x |
| | Cyclomatic_density | x | | x | x | x | x | x | x | x |
| | Design_complexity | x | x | x | x | x | x | x | x | x |
| | Essential_complexity | x | x | x | x | x | x | x | x | x |
| **Misc.** | Branch_count | x | x | x | x | x | x | x | x | x |
| | Call_pairs | x | | x | x | x | x | x | x | x |
| | Condition_count | x | | x | x | x | x | x | x | x |
| | Decision_count | x | | x | x | x | x | x | x | x |
| | Decision_density | x | | x | x | x | x | x | x | x |
| | Edge_count | x | | x | x | x | x | x | x | x |
| | Essential_density | x | | x | x | x | x | x | x | x |
| | Parameter_count | x | | x | x | x | x | x | x | x |
| | Maintance_severity | x | | x | x | x | x | x | x | x |
| | Modified_condition_count | x | | x | x | x | x | x | x | x |
| | Multiple_condition_count | x | | x | x | x | x | x | x | x |
| | Global_data_complexity | | | x | x | | | | | |
| | Global_data_density | | | x | x | | | | | |
| | Normalize_cyclo_cmplx | x | | x | x | x | x | x | x | x |
| | Percent_comments | x | | x | x | x | x | x | x | x |
| | Node_count | x | | x | x | x | x | x | x | x |
| **Number of attribute** | | 37 | 21 | 39 | 39 | 37 | 37 | 36 | 37 | 37 |

After calculating the average and standard deviation, then specify the center data point of the dataset variable as in (3).

$$m = [\ \bar{x}_a \bar{x}_b\ ] \tag{3}$$

where $m$ is the first midpoint, $\bar{x}_a$ is the maximum value of the average of the standard deviation (SD) and $\bar{x}_b$ is the average value of the minimum SD. Then, calculate it by the Euclidean distance (4).

$$d_{im} = ((x_i - \bar{x}_a)^2 + (x_i + \bar{x}_b)^2)^{\frac{1}{2}} \qquad i = 1,2,\dots,n \tag{4}$$

where $d_{im}$ is the Euclidean distance for the first midpoint data point. $x_i$ is the calculated data point ($i$ is datapoint of 1 to $n$), $\bar{x}_a$ is the maximum value of the average of matrix $m$, and $\bar{x}_b$ is the average value of the minimum standard deviation of matrix $m$. To calculate the distance between each data point and the starting point. Then, Equation (5) is used to select variables X as a first variable and Y as a second variable of the cluster center member.

$$v = [\ \bar{x}_I \bar{x}_{II}\ ] \tag{5}$$

where $v$ is the matrix or midpoint of the first and second variables, $\bar{x}_I$ is the minimum value of the average of candidate cluster and $\bar{x}_{II}$ is the average of the maximum SD of the candidate cluster. Datapoint based on the selection of variables of the Equation (5) with the highest distance Euclidean data on the Equation (4) selected as the first candidate of the initial center point ($c_1$). Then, calculate the Euclidean distance (6).

$$d_{ic(l)} = \left(\left(x_{iI} - \bar{x}_{c_k I}\right)^2 + \left(x_{iII} + \bar{x}_{c_k II}\right)^2\right)^{\frac{1}{2}} \text{ where } i = 1,2,\dots,n \tag{6}$$

The highest distance data point in Equation (6) is chosen as the second point center candidate ($c_2$). Where $c_k$ obtained based on the point center point, do until the Equation (6) is equal to $c_k$ distance previously to get the k value obtained from the final cluster $c_{(k-1)}$ . Then the cluster members of each point are determined by candidate point center and variable $X$ and $Y$.

### 2.3. K-Means Clustering Algorithm

The K-Means algorithm is a simple method for partitioning a given dataset into a specified number of clusters k. This algorithm has been discovered by several researchers from various disciplines, especially Lloyd (1957, 1982), Forgey (1965), Friedman and Rubin (1967), and McQueen (1967). K-Means at non-convex costs also explain that integration is only for local optimality, and the algorithm is usually quite sensitive to the initial centroid location [29].

K-Means is a fairly simple clustering algorithm that partitioned datasets into clusters of k. This technique's main principle is to compile a partition or centroid/average of a set of data. The K-Means algorithm starts with forming a cluster partition initially, then iteratively clustered the partition is repaired until there is no significant change in the cluster partition [30].

K-Means initializes the cluster by randomly generating k data points, while the proposed method of giving initial K-Means centroid values is not random. This is usually done by producing uniformly random values for each dimension. Each K-Means iteration consists of two steps: i) cluster assignments and ii) centroid updates. Determine the centroid k points, then group the data to form a k cluster, with the centroid points of each cluster being the pre-selected centroid points. Update the centroid point value with (7).

$$\mu_k = \frac{1}{N_k} \sum_{q=1}^{N_k} x_q \tag{7}$$

where $\mu_k$ is centroid point of the $k$-cluster, $N_k$ the amount of data in the $k$ cluster, and $q$ data on $k$ - cluster. Repeat the grouping and update the centroid value until the value from the centroid point no longer changes.

### 2.4. Proposed Method

This research is a proposed method to determine initial centroid value using an algorithm called point center as the determinant of initial centroid value on K-Means clustering (Fig. 1). The steps of the proposed method to determine the initial centroid value by using the point center algorithm are as follows:

**Step 1**: At this step, the preliminary data processing was used to check and eliminate the missing data using RapidMiner 7.3 Library application. Each dataset with empty or missing supporting value need to fill out and ensure it was numerical using the Replace Missing Value operator. Since replacing was only on data and did not impact an attribute, this change could be applied in all data copies. In this case, the value assignment procedure was applied to fill the blank information based on the average data value. Then, the data was stored back in the form of excel for subsequent data processing.

**Step 2**: Calculate the point center algorithm to get $k$ value and point center value as the initial centroid.

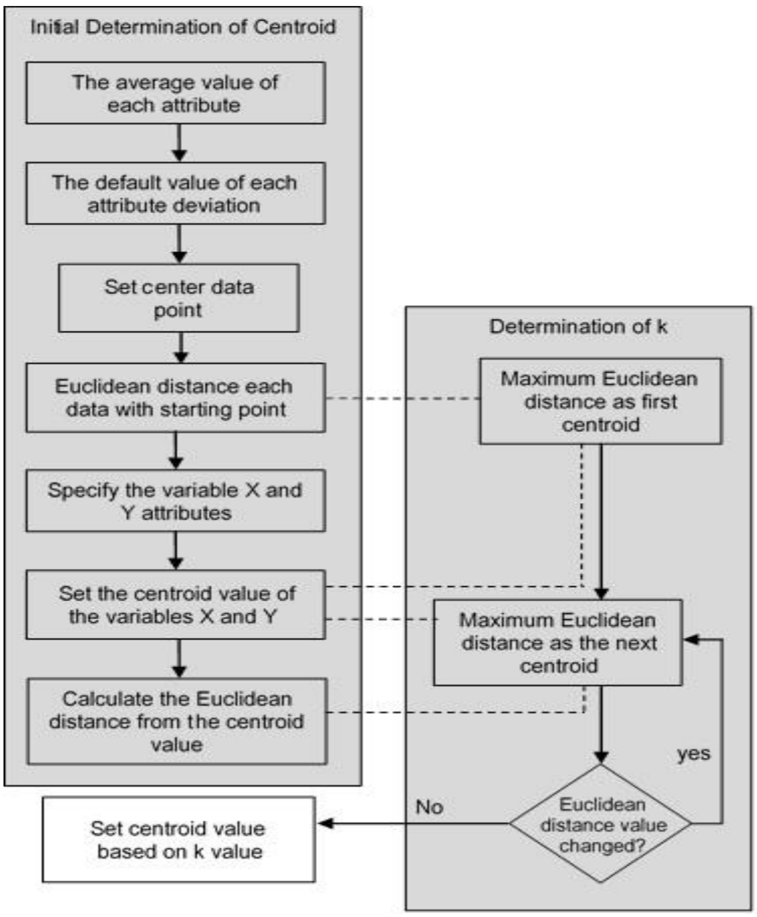
Initial Determination of Centroid
The average value of each attribute
The default value of each attribute deviation
Set center data point
Euclidean distance each data with starting point
Specify the variable X and Y attributes
Set the centroid value of the variables X and Y
Calculate the Euclidean distance from the centroid value
Determination of k
Maximum Euclidean distance as first centroid
Maximum Euclidean distance as the next centroid
yes
No
Euclidean distance value changed?
Set centroid value based on k value

**Fig. 1.** Step of algorithm center point.

**Step 3**: Calculate the cluster value based on the initial centroid on the K-Means algorithm. Then process it to the point center algorithm and the obtained value of $k$.

**Step 4**: Calculate the error rate and the Rand index obtained from the K-Means calculation's confusion matrix. The testing performs by comparing the obtained proposed algorithm clusters and the available clusters by K-Means. The grouping of data obtained using the clustering algorithm was a predicted label, while the dataset label value was actual. The final process was comparing and classifying them.

The K-Means algorithm is an unsupervised learning algorithm without labels; However, in the proposed algorithm, a label was needed as a comparison and to measure the performance of this testing algorithm. The two clusters' appeal's total value could be presented using the confusion matrix in Table 2.

**Table 2.** Confusion Matrix

| | | Actual | |
|---|---|---|---|
| | | ***False (non-faulty)*** | ***True (faulty)*** |
| ***Predicted*** | ***False (non-faulty)*** | True Negative (A) | False Positive (B) |
| | ***True (Faulty)*** | False Negative (C) | True Positive (D) |

Table 2 calculates error value and rand index (accuracy) as in (8) and (9).

$$Error = \frac{B+C}{A+B+C+D} \tag{8}$$

$$Rand\ Index = \frac{A+D}{A+B+C+D} \tag{9}$$

Furthermore, the results obtained are compared with simple K-Means algorithm calculations by the same evaluation technique.

## 3. Results and Discussion

This section discusses the results of the measurements of the results obtained by comparing the measurement results using simple K-Means and K-Means using the proposed method called Point Center K-Means (PCKM). The dataset tested was ten datasets consisting of an iris dataset having 3 clusters and 9 NASA MDP datasets (PC1, PC2, PC3, MW1, CM1, KC1, KC3, and MC2), each of which had 2 clusters.

The Iris dataset experiment used 3 classes (Setosa, Versicolor, Virginica), and 150 sample data. Then, Each sample data has 4 attributes: sepal length, sepal width, petal length, petal width. The proposed algorithm calculation starts with calculating the average value and the standard deviation value of each attribute presented in Table 3.

**Table 3.** Statistics descriptive of the Iris Dataset.

| Statistics | Sepal length | Sepal width | Petal length | Petal width |
|---|---|---|---|---|
| $\bar{x}_j$ | 5.8433 | 3.0540 | 3.7587 | 1.1987 |
| $s(x_j)$ | 0.8281 | 0.4336 | 1.7644 | 0.7632 |

After getting the average value and standard deviation, then set the data center point. The center data point is determined as $m$ = [5.8433, 3.0540] obtained from the sepal length and sepal width attributes then calculate the first center point candidate $(c_1)$ obtained from the maximum Euclidean distance.

Next, determine the variables X and Y, namely $p = [1.1987\ , 3.7587]$ for the center point obtained from the attributes of petal width $(x_I)$ and petal length $(x_{II})$. To determine the second point center candidate $(c_2)$ and the next point center candidate $(c_k)$ was using the euclidean formula. Then, the value of the distance was $d_1 = 2.1878\ ,\ d_2 = 5.6921,\ d_3 = 6.2626$, because the four attributes' distance value was the same as before, the third distance, the number of clusters in this dataset is 3 clusters $(k)$. Then the point center points obtained from the cluster membership were $c_1 = [2;\ \ 6.4], c_2 = [0.2;\ \ 1]$, and $c_3 = [2.3;\ \ 6.9]$. The obtained point center was used to calculate the clustering data using the K-Means. This process was done using the R tool by entering the point center value first and then clustering them based on the $k$ value. The updated centroid value was $c_1 = [1.36;\ 4.3]$, $c_2 = [0.244; 1.464]$, and $c_3 = [2.05;\ 5.63]$.

Fig. 2 presents the division of clusters based on the proposed algorithm, from the grouping the number of $c_1$ was 54 data, $c_2$ was 50 data, and $c_3$ as 46 data. The comparison between the label clusters obtained with the actual class labels could be presented in Table 4. Based on Table 2, equations (8) and (9) could be calculated, and getting the error rate in Table 4 was 5.3%, and the Rand Index was 94.7%. Compared with the simple K-Means, which determine the initial centroid randomly, the results are presented in Table 4 with an error rate of 10.7% and a Rand Index of 89.3%.

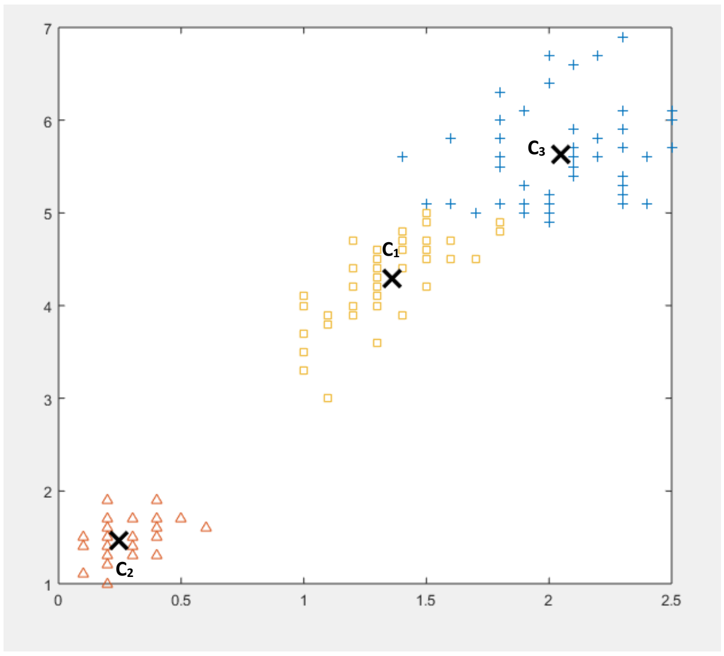

**Fig. 2.** Iris dataset centroid and its cluster using the proposed algorithm.

**Table 4.** Confusion matrix of the Iris dataset using the proposed Method and K-Means.

| | | Actual Proposed method PCKM Iris | | | Actual K-Means Iris | | |
|---|---|---|---|---|---|---|---|
| | | *Setosa* | *Versicolor* | *Virginica* | *Setosa* | *Versicolor* | *Virginica* |
| **Predicted** | *Setosa* | 50 | 0 | 0 | 50 | 0 | 0 |
| | *Versicolor* | 0 | 48 | 2 | 0 | 48 | 2 |
| | *Virginica* | 0 | 6 | 44 | 0 | 14 | 36 |

Furthermore, each NASA MDP dataset is calculated using a point center algorithm to get the initial centroid value, which had been then calculated using the K-Means clustering algorithm (PCKM). A confusion matrix is presented in Table 4. Then each dataset had been recalculated using K-Means. The initial centroid clustering by randomly generating data and the confusion matrix results are also presented in Table 5.

**Table 5.** Confusion matrix NASA MDP using the Proposed method and K-Means.

| **Datasets** | A (True Negative) | | B (False Positive) | | C (False Negative) | | D (True Positive) | |
|---|---|---|---|---|---|---|---|---|
| | *PCKM* | *K-Means* | *PCKM* | *K-Means* | *PCKM* | *K-Means* | *PCKM* | *K-Means* |
| PC1 | 697 | 697 | 1 | 1 | 60 | 60 | 1 | 1 |
| PC2 | **1562** | **1557** | 15 | 15 | **7** | **12** | 1 | 1 |
| PC3 | 984 | 984 | 140 | 140 | 1 | 1 | 0 | 0 |
| PC4 | **1207** | **1202** | 171 | 171 | **14** | **19** | 7 | 7 |
| MW1 | **231** | **210** | **25** | **21** | **6** | **27** | **2** | **6** |
| CM1 | 300 | 300 | 42 | 42 | 2 | 2 | 0 | 0 |
| KC1 | 1740 | 1740 | 283 | 283 | 31 | 31 | 42 | 42 |
| KC3 | 158 | 156 | **31** | **29** | **5** | **8** | **6** | **7** |
| MC2 | 82 | 82 | 38 | 38 | 1 | 1 | 6 | 6 |

The proposed method algorithm can determine the number of clusters ($k$) and the initial centroid value of the K-Means algorithm from the experimental results. Table 6 shows the results of calculating the number of clusters (k) using ten test datasets.

**Table 6.** Number of clusters.

| Datasets | Number of attributes | Data count | Cluster count | Proposed method |
|---|---|---|---|---|
| Iris | 4 | 150 | 3 | 3 |
| PC1 | 37 | 759 | 2 | 2 |
| PC2 | 36 | 1585 | 2 | 2 |
| PC3 | 37 | 1125 | 2 | 2 |
| PC4 | 37 | 1399 | 2 | 2 |
| MW1 | 37 | 264 | 2 | 2 |
| CM1 | 37 | 344 | 2 | 2 |
| KC1 | 21 | 2096 | 2 | 2 |
| KC3 | 39 | 200 | 2 | 2 |
| MC2 | 39 | 127 | 2 | 2 |

Table 4 compares the number of clusters (k) of actual labels with the number of clusters (k) obtained through the proposed method. The number of clusters (k) obtained using the proposed method followed the actual number of cluster labels. Using the proposed method algorithm determined the number of clusters of the K-Means algorithm at first and the initial centroid value. Experiments using R tools were carried out two to three times on the K-Means algorithm that getting a random centroid value. The results showed that each experiment's centroid value was changed, but the error rate was the same.

The proposed method for clustering aimed to determine the initial centroid value. Then, the centroid value is implemented by the K-Means method. A comparison between the proposed method and the K-Means that used the initial randomly are shown in Table 7. The simple K-Means test and the proposed method show that the error results using the proposed method are lower than the K-Means method. The difference in error value with ten datasets between them is 13.1%.

**Table 7.** Method test results based on percentage error level.

| Datasets | K-Means (%) | The proposed method (%) | Difference (%) |
|---|---|---|---|
| **Iris** | **10.7** | **5.3** | **5.4** |
| PC1 | 0.08 | 0.08 | 0 |
| **PC2** | **1.7** | **1.4** | **0.3** |
| PC3 | 12.5 | 12.5 | 0 |
| **PC4** | **13.6** | **13.2** | **0.4** |
| **MW1** | **18.2** | **11.7** | **6.5** |
| CM1 | 12.8 | 12.8 | 0 |
| KC1 | 15 | 15 | 0 |
| **KC3** | **18.5** | **18** | **0.5** |
| MC2 | 30.7 | 30.7 | 0 |
| **Total** | 133.78 | 120.68 | 13.1 |

The comparison result between simple K-Means and the proposed method using ten datasets (Table 7) obtained five datasets (Iris, PC2, PC4, MW1, KC3) showed that the proposed method got the lower errors. The proposed method produced an initial centroid value from the experimental results because it affects the cluster center's fixed value. The error rate is lower than the simple K-Means, and it gets a random centroid value affecting the cluster center result did not fix.

The proposed algorithm had a better Rand Index value on the NASA MDP dataset, such as the PC2, PC4, MW1, and KC3. However, the other NASA datasets were having a level comparable to simple K-Means (Table 8). Rand index is between 0 and 1, a value close to 1 means the perfect rand index, and the value seen from the high rand index.

**Table 8.** Rand Index values testing of the Proposed method and K-Means.

| Datasets | K-Means | Proposed method |
|---|---|---|
| Iris | 0.893 | 0.947 |
| PC1 | 0.92 | 0.92 |
| **PC2** | **0.983** | **0.986** |
| PC3 | 0.875 | 0.875 |
| **PC4** | **0.864** | **0.868** |
| **MW1** | **0.82** | **0.883** |
| CM1 | 0.872 | 0.872 |
| KC1 | 0.85 | 0.85 |
| **KC3** | **0.815** | **0.82** |
| MC2 | 0.69 | 0.69 |

After calculating the proposed method's performance on ten test datasets, 9 of them were NASA MDP datasets that used a sample of clustering software modules as defective and non-defective. The proposed algorithm captured cluster errors of software (Table 9).

**Table 9.** Software defect modules number used in the proposed method testing.

| Datasets | Number of attributes | Module count | Defective module count | Proposed Method Error Rate (%) |
|---|---|---|---|---|
| PC1 | 37 | 759 | 61 | 0.08 |
| PC2 | 36 | 1585 | 16 | 1.4 |
| PC3 | 37 | 1125 | 140 | 12.5 |
| PC4 | 37 | 1399 | 178 | 13.2 |
| MW1 | 37 | 264 | 27 | 11.7 |
| CM1 | 37 | 344 | 42 | 12.8 |
| KC1 | 21 | 2096 | 325 | 15 |
| KC3 | 39 | 200 | 36 | 18 |
| MC2 | 39 | 127 | 44 | 30.7 |
| Total | 329 | 7899 | 869 | 115.38 |

Table 9 presents the number of software modules and the number of defective module labels in the NASA MDP datasets. To capture cluster errors in software with the K-Means point center clustering algorithm (PCKM) can be seen from the obtained error level. From the 9 NASA MDP datasets (PC1, PC2, PC3, PC4, MW1, CM1, KC1, KC3, and MC2), the total error total of 115.38 was calculated, and got approximately 12.82% of the PCKM captured cluster errors of software defect modules.

## 4. Conclusion

This paper proposed an algorithm for calculating the initial centroid value of the K-Means algorithm called the central point algorithm. Besides calculating the initial centroid value, the point center algorithm can also determine the number of clusters. This algorithm was based on selecting variables X and Y to determine the cluster members and opposed ten datasets. The 9 of the approved datasets are datasets for software defect estimates. The proposed method's results in the Iris, PC2, PC4, MW1, and KC3 datasets got lower error results. The higher Rand Index value was shown in some datasets, such as PC2, PC4, MW1, and KC3. Another experimental result showed that the proposed algorithm could improve the performance of K-Means by 12.82% cluster errors in the software defect modules compared to the simple K-Means algorithm. This proposed method could be useful for other data type clustering because it produces better accuracy than the simple K-Means method. Nevertheless, further research will be conducted as the next research stage by applying the proposed K-Means point center algorithm to other research objects, such as biomedical datasets. Also, It still open to compare the results of the proposed K-Means point center algorithm with other clustering algorithms.

## Acknowledgment

We would like to thank the magister of computer Science, STMIK Nusa Mandiri, Jakarta, Indonesia, to commence this research in the first instance, do the necessary research work, and use departmental data.

## Declarations

**Author contribution.** All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.
**Funding statement**. The research has been funded by STMIK Nusa Mandiri, Jakarta, Indonesia.
**Conflict of interest**. The authors declare no conflict of interest.
**Additional information**. No additional information is available for this paper.

## References

[1] M. G. Siavvas, K. C. Chatzidimitriou, and A. L. Symeonidis, "QATCH - An adaptive framework for software product quality assessment," *Expert Syst. Appl.*, vol. 86, pp. 350–366, Nov. 2017, doi: 10.1016/j.eswa.2017.05.060.

[2] L. Qiao, X. Li, Q. Umer, and P. Guo, "Deep learning based software defect prediction," *Neurocomputing*, vol. 385, pp. 100–110, Apr. 2020, doi: 10.1016/j.neucom.2019.11.067.

[3] X. Chen, D. Zhang, Y. Zhao, Z. Cui, and C. Ni, "Software defect number prediction: Unsupervised vs supervised methods," *Inf. Softw. Technol.*, vol. 106, pp. 161–181, Feb. 2019, doi: 10.1016/j.infsof.2018.10.003.

[4] G. K. Rajbahadur, S. Wang, Y. Kamei, and A. E. Hassan, "The Impact of Using Regression Models to Build Defect Classifiers," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 2017, pp. 135–145, doi: 10.1109/MSR.2017.4.

[5] A. Majd, M. Vahidi-Asl, A. Khalilian, P. Poorsarvi-Tehrani, and H. Haghighi, "SLDeep: Statement-level software defect prediction using deep-learning model on static code features," *Expert Syst. Appl.*, vol. 147, p. 113156, Jun. 2020, doi: 10.1016/j.eswa.2019.113156.

[6] R. Moussa and D. Azar, "A PSO-GA approach targeting fault-prone software modules," *J. Syst. Softw.*, vol. 132, pp. 41–49, Oct. 2017, doi: 10.1016/j.jss.2017.06.059.

[7] A. Boucher and M. Badri, "Predicting Fault-Prone Classes in Object-Oriented Software: An Adaptation of an Unsupervised Hybrid SOM Algorithm," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 2017, pp. 306–317, doi: 10.1109/QRS.2017.41.

[8] Z. Sun, J. Zhang, H. Sun, and X. Zhu, "Collaborative filtering based recommendation of sampling methods for software defect prediction," *Appl. Soft Comput.*, vol. 90, p. 106163, May 2020, doi: 10.1016/j.asoc.2020.106163.

[9] F. HUANG and B. LIU, "Software defect prevention based on human error theories," *Chinese J. Aeronaut.*, vol. 30, no. 3, pp. 1054–1070, Jun. 2017, doi: 10.1016/j.cja.2017.03.005.

[10] N. Li, M. Shepperd, and Y. Guo, “A systematic review of unsupervised learning techniques for software defect prediction,” *Inf. Softw. Technol.*, vol. 122, p. 106287, Jun. 2020, doi: 10.1016/j.infsof.2020.106287.

[11] Q. Huang, X. Xia, and D. Lo, “Supervised vs Unsupervised Models: A Holistic Look at Effort-Aware Just-in-Time Defect Prediction,” in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017, pp. 159–170, doi: 10.1109/ICSME.2017.51.

[12] X. Chen, Y. Zhao, Q. Wang, and Z. Yuan, “MULTI: Multi-objective effort-aware just-in-time software defect prediction,” *Inf. Softw. Technol.*, vol. 93, pp. 1–13, Jan. 2018, doi: 10.1016/j.infsof.2017.08.004.

[13] R. Chang, X. Shen, B. Wang, and Q. Xu, “A novel method for software defect prediction in the context of big data,” in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*, 2017, pp. 100–104, doi: 10.1109/ICBDA.2017.8078785.

[14] A. Boucher and M. Badri, “Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison,” *Inf. Softw. Technol.*, vol. 96, pp. 38–67, Apr. 2018, doi: 10.1016/j.infsof.2017.11.005.

[15] S. Singh and R. Singla, “Classification of defective modules using object-oriented metrics,” *Int. J. Intell. Syst. Technol. Appl.*, vol. 16, no. 1, p. 1, 2017, doi: 10.1504/IJISTA.2017.081311.

[16] M. Yan, X. Zhang, C. Liu, L. Xu, M. Yang, and D. Yang, “Automated change-prone class prediction on unlabeled dataset using unsupervised method,” *Inf. Softw. Technol.*, vol. 92, pp. 1–16, Dec. 2017, doi: 10.1016/j.infsof.2017.07.003.

[17] M. Yan, Y. Fang, D. Lo, X. Xia, and X. Zhang, “File-Level Defect Prediction: Unsupervised vs. Supervised Models,” in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2017, pp. 344–353, doi: 10.1109/ESEM.2017.48.

[18] J. Liu, Y. Zhou, Y. Yang, H. Lu, and B. Xu, “Code Churn: A Neglected Metric in Effort-Aware Just-in-Time Defect Prediction,” in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2017, pp. 11–19, doi: 10.1109/ESEM.2017.8.

[19] E. Zhu, Y. Zhang, P. Wen, and F. Liu, “Fast and stable clustering analysis based on Grid-mapping K-means algorithm and new clustering validity index,” *Neurocomputing*, vol. 363, pp. 149–170, Oct. 2019, doi: 10.1016/j.neucom.2019.07.048.

[20] S. Khanmohammadi, N. Adibeig, and S. Shanehbandy, “An improved overlapping k-means clustering method for medical applications,” *Expert Syst. Appl.*, vol. 67, pp. 12–18, Jan. 2017, doi: 10.1016/j.eswa.2016.09.025.

[21] A. Kaur, S. K. Pal, and A. P. Singh, “Hybridization of Chaos and Flower Pollination Algorithm over K-Means for data clustering,” *Appl. Soft Comput.*, vol. 97, p. 105523, Dec. 2020, doi: 10.1016/j.asoc.2019.105523.

[22] A. Fadaei and S. H. Khasteh, “Enhanced K-means re-clustering over dynamic networks,” *Expert Syst. Appl.*, vol. 132, pp. 126–140, Oct. 2019, doi: 10.1016/j.eswa.2019.04.061.

[23] P. Fränti and S. Sieranoja, “How much can k-means be improved by using better initialization and repeats?,” *Pattern Recognit.*, vol. 93, pp. 95–112, Sep. 2019, doi: 10.1016/j.patcog.2019.04.014.

[24] H. Ismkhan, “I-k-means−+: An iterative clustering algorithm based on an enhanced version of the k-means,” *Pattern Recognit.*, vol. 79, pp. 402–413, Jul. 2018, doi: 10.1016/j.patcog.2018.02.015.

[25] S. K. Majhi and S. Biswal, “Optimal cluster analysis using hybrid K-Means and Ant Lion Optimizer,” *Karbala Int. J. Mod. Sci.*, vol. 4, no. 4, pp. 347–360, Dec. 2018, doi: 10.1016/j.kijoms.2018.09.001.

[26] N. Nidheesh, K. A. Abdul Nazeer, and P. M. Ameer, “An enhanced deterministic K-Means clustering algorithm for cancer subtype prediction from gene expression data,” *Comput. Biol. Med.*, vol. 91, pp. 213–221, Dec. 2017, doi: 10.1016/j.compbiomed.2017.10.014.

[27] R. S. Wahono, “A Systematic Literature Review of Software Defect Prediction : Research Trends , Datasets , Methods and Frameworks,” *J. Softw. Eng.*, vol. 1, no. 1, pp. 1–16, 2015, Available at: Google Scholar

[28] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Hum. Genet.*, vol. 7, no. 2, pp. 179–188, 1936, doi: 10.1111/j.1469-1809.1936.tb02137.x.

[29] X. Wu *et al.*, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, Jan. 2008, doi: 10.1007/s10115-007-0114-2.

[30] S. F. Hussain and M. Haris, "A k-means based co-clustering (kCC) algorithm for sparse, high dimensional data," *Expert Syst. Appl.*, vol. 118, pp. 20–34, Mar. 2019, doi: 10.1016/j.eswa.2018.09.006.