Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

K-Means Optimization with Kernel Gaussian Radial Basis Function in C#

Harun Nasrullah[1], Arif Bramantoro[2*], Ahmad A. Alzahrani[3]

[1] Faculty of Information Technology, Budi Luhur University, Jakarta, 12260, Indonesia
[2] School of Computing and Informatics, Brunei University of Technology, Bandar Seri Begawan, BE1410, Brunei Darussalam
[3] Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, 21589, Saudi Arabia
E-mail: arif.bramantoro@utb.edu.bn

Received: 8 June 2025; **Revised:** 11 July 2025; **Accepted:** 15 July 2025

Abstract: *K*-Means is an iterative clustering technique that relies on centroids to organize data into distinct clusters until convergence is achieved. Despite its effectiveness, *K*-Means struggles with non-linearly separable data and is sensitive to initial centroid selection, as data points are assigned to clusters based on proximity to centroids. Moreover, the algorithm's performance hinges on the predetermined number of clusters (*K*), with suboptimal *K* selection leading to inferior clustering outcomes. To address these limitations, this study proposes the use of Kernel *K*-Means, employing the Kernel Gaussian Radial Basis Function to handle non-linearly separable datasets in high-dimensional feature spaces. The Elbow method is employed to determine the optimal *K* value, and the Sum of Squares Error (SSE) method aids in identifying initial centroids. Subsequent analysis of clustering results utilizes the Davies Bouldin Index and Silhouette coefficient techniques. Implementation is conducted using the C# programming language to expand machine learning applications across different programming languages. Our findings indicate that Kernel *K*-Means consistently outperforms traditional *K*-Means, with the Silhouette coefficient's effectiveness varying depending on dataset characteristics and size.

***Keywords*:** elbow, *K*-Means, Kernel Gaussian, Sum of Squares Error (SSE)

MSC: 62H30, 65C20

1. Introduction

K-Means, an iterative clustering algorithm, operates by iteratively reallocating data items across cluster sets until convergence is achieved [1]. Utilizing centroids as representative points for clusters, *K*-Means identifies groups within datasets [2]. However, its effectiveness diminishes when confronted with non-linearly separable data [3], which is common in real-world scenarios where data exhibits diverse structures and shapes.

The sensitivity of *K*-Means to initial centroid selection is well-documented, as varying initial configurations yield different clustering outcomes. Additionally, the algorithm's performance heavily relies on the pre-defined number of clusters (*K*). Consequently, selecting an appropriate *K* value is paramount for achieving optimal clustering results, as an erroneous choice may lead to suboptimal outcomes.

DOI: https://doi.org/10.37256/cm.6520257401

Against this backdrop, the research problem will be how to address the challenge of determining the ideal K value using the Elbow Method, selecting optimal initial centroids via the Sum of Squares Error (SSE) method, and leveraging Kernel K-Means with Kernel Gaussian Radial Basis Function to handle non-linearly separable data in high-dimensional feature spaces. In our experience with service systems, quantitative similarity and Quality of Service (QoS)-aware composition routinely guide decisions in complex environments, complementing clustering-based analysis [4, 5].

Implementation of these computations and algorithms necessitates a suitable programming language that offers flexibility beyond existing tools. C# emerges as a viable choice due to its relative simplicity and modern features such as Exception Handling, Garbage Collection, Extensible Data Types, and robust Object-Oriented features like encapsulation, inheritance, and polymorphism. Moreover, C# supports diverse application development, including word processing, spreadsheets, and mobile applications. However, it is important to acknowledge the limitations of C#, notably its lack of extensive data mining libraries compared to languages like Python and R. Thus, this study advocates for the adoption of C# to broaden the accessibility of machine learning libraries across various programming languages, particularly benefiting C# programmers reluctant to explore alternative languages.

2. Literature review

In their work, Liu et al. in [6] introduced various methods including Fuzzy C-Means, Kernel Fuzzy C-Means, K-Means, Kernel Probabilistic K-Means, Traditional Active Gradient Projection, and Fast Active Gradient Projection. Results indicated that Fast Active Gradient Projection exhibited reduced runtime compared to Traditional Active Gradient Projection, achieving a reduction of 76-95% on real datasets.

Liu et al. in [7] proposed Multiple Kernel Clustering, Simple multiple Kernel K-Means, and multiple Kernel clustering with local Kernel alignment. Their novel algorithm consistently demonstrated the effectiveness of local Kernel alignment criteria.

Rashed et al. in [8] presented Crowding Distance with density measure and Pareto dominance concept, K-Means Clustering, Spectral clustering, Birch, and average-link algorithms. Their findings showcased that Multi-objective Particle Swarm Optimization (PSO) and Crowding Distance effectively addressed optimization problems, with superior average accuracy compared to other techniques.

Rustam and Nadhifa in [9] suggested Spherical K-Means, Kernel Spherical K-Means Gaussian radial basis, and Kernel polynomial. Their study revealed that KSPKM significantly improved accuracy, achieving a level of 98.31% compared to SPKM.

Vankadara and Ghoshdastidar in [10] proposed K-Means Kernel, Gaussian Kernel, and Polynomial Kernel, resulting in distinguishable cluster averages under the Kernel K-Means objective. This approach exhibited smaller errors than random guessing and semi-definite relaxation of the objective Kernel K-Means, achieving partial recovery comparable to the spectral threshold.

Wang et al. in [11] introduced Kernel K-Means, Rank-restricted Nystrom approximation method, and randomized linear algebra. Their Apache Spark implementation of a distributed version of Kernel K-Means on a cluster of 8.1 million vectors demonstrated the utility and simplicity of Kernel K-Means with the rank-limited Nystrom approach for clustering large-scale datasets.

Rustam et al. in [12] explored K-Means, Global K-Means, Kernel K-Means, and Global Kernel K-Means. Their findings indicated that Global Kernel K-Means achieved the highest accuracy rate of 78%. In [13], Ning and Hongyi optimized the algorithm of Kernel K-Means clustering, demonstrating superior performance compared to standard Kernel K-Means in terms of accuracy and stability across multiple subsets.

Tzortzis and Likas in [14] proposed Kernel K-Means, Global Kernel K-Means, and Global Kernel K-Means with convex mixture models, Fast Global Kernel K-Means, and Graph Partitioning. Their research indicated that the Global Kernel K-Means variants outperformed Kernel K-Means with several restarts, with Kernel Global K-Means with convex mixture models and Fast Global Kernel K-Means emerging as favorable alternatives for large datasets.

3. Methodology

Non-linear mapping functions are employed in solving quadratic optimization problems of linear support vector machines by applying the Kernel function $K(X_i, X_j)$ to the original input data [15]. The mapping function, X_i, X_j transforms data from the input space to the feature space, with $\phi(.)$ representing a vector in the input space [16]. This transformation facilitates the use of linear techniques to solve classification problems in the feature space. While $\phi(X)$ may not always be directly accessible, the inner product $\phi(X)$ can still be computed implicitly through Kernel functions in the feature space. Consequently, all calculations are conducted in the original input space, allowing for the utilization of $K(X_i, X_j)$ when $\phi(X_i).\phi(X_j)$ arises during training. The Gaussian Radial Basis Function Kernel [13] is one such kernel function employed in this study, defined as:

$$K(X_i, X_j) = \exp^{-\|X_i - X_j\|^2 / 2\sigma^2} \tag{1}$$

where $\|X - X_i\|^2$ denotes the squared Euclidean distance between two vectors, σ^2 is a specified parameter, and X_i, X_j represent vectors in the input space.

Z-score standardization, a data normalization technique, measures the variance of a given value from the mean value in standard deviation units [17, 18]. The Z-score, also known as the Standard Value, represents a value's deviation from the mean in a normal distribution with an average of zero (0) and a standard deviation of one (1). The Z-score formula is:

$$Z = (x - \mu)/\sigma \tag{2}$$

where σ represents the standard deviation, μ denotes the mean, x signifies the observed value, and Z denotes the Z-score. The standard deviation equation is defined as:

$$\sigma = \frac{\sum (x - \mu)^2}{n - 1} \tag{3}$$

where n represents the number of data points, σ denotes the standard deviation, x signifies the observed value, and μ represents the mean. The mean equation is:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n} \tag{4}$$

where μ signifies the mean, x denotes the observed value, and n represents the number of data points.

Distance measurements, crucial for assessing the similarity between objects, are utilized to evaluate the effectiveness of the clustering process [19]. Euclidean distance, a measure of the relationship between angles and distance, calculates the distance between two points in Euclidean space. For latitude and longitude coordinates, two-dimensional calculations are employed, and the Euclidean distance equation is:

$$d_{(x,y)} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{5}$$

where n represents the number of data points, d denotes the Euclidean distance, x_i represents data x at point i, and y_i represents data y at point i.

The ideal number of clusters in cluster analysis necessitates a balance between compressibility and accuracy. The Elbow Method is commonly utilized to determine the optimal number of clusters [20, 21]. This method involves comparing the SSE calculation results for each cluster value to identify the point at which an "elbow" occurs. The SSE decreases as the cluster number (k) increases. The SSE equation is:

$$\mathrm{SSE} = \sum_{k-1}^{k} \sum_{xi \in Sk} \|X_i - C_k\|^2 \tag{6}$$

where SSE represents the SSE, X_i signifies the attribute value of the i-th data point, and C_k epresents the attribute value of the k-th cluster centroid.

Selecting suitable initial centroids is a critical step in basic K-Means clustering [22]. Random initialization of centroids is a common approach, although it often results in suboptimal clusters. The initial centroid selection problem is typically addressed by running several iterations with different sets of randomly chosen initial centroids and selecting the set that minimizes the SSE. The SSE equation for initial centroid selection is:

$$\mathrm{SSE} = \sum_{k-1}^{k} \sum_{xi \in Sk} \|X_i - C_k\|^2 \tag{7}$$

where SSE represents the SSE, X_i denotes the attribute value of the i-th data point, and C_k signifies the attribute value of the k-th cluster centroid.

The K-Means algorithm identifies the centroid, the average value of points within a cluster [15]. Initially, k are randomly selected as centroids from the dataset D. Objects are then assigned to the most similar cluster based on the Euclidean-like distance calculation between the objects and the cluster mean. The algorithm iteratively reduces the variation within clusters by updating cluster means and reassigning objects to the nearest cluster center until a stable assignment is achieved. The equation for the cluster error is:

$$\varepsilon_K = \sum_{n=1}^{N} \sum_{k=1}^{K} Z_{kn} \|x_n - m_k\|^2 \tag{8}$$

where ε_K represents the cluster error, Z_{kn} signifies the cluster allocation indicator, m_k denotes the group center, and x_n represents the data point.

The Kernel K-Means algorithm differs from traditional K-Means in how it processes input data [23]. In Kernel K-Means, data points are first mapped to a higher-dimensional space using a non-linear function before clustering is performed. This allows for the handling of nonlinear separability by linearly separating the data in the transformed feature space [24]. The mapping function is typically achieved using a Gaussian Kernel function, defined as:

$$K(X_i, X_j) = \exp\left(-\frac{1}{2\sigma^2} \left\|X_i - X_j\right\|^2\right) \tag{9}$$

where X_i, X_j represent vectors in the input space, σ^2 denotes specified parameters, and $\|X - X_i\|^2$ signifies the distance between two vectors.

The Davies Bouldin Index (DBI) is an internal clustering validity metric that evaluates the compactness of data within clusters and the separation between clusters. A lower DBI value indicates better-defined and more distinct clusters [25]. It quantifies the ratio of the average distance between clusters to the distance within clusters, with lower values indicating more cohesive clustering. The DBI is computed as follows:

(1) Sum of Squares Within cluster (SSW): This measures the intra-cluster distance, calculated as the square root of the average squared distance between each data point and the centroid of its cluster:

$$\mathrm{SSW}_i = \left\{ \frac{1}{T_i} \sum_{j=1}^{T_i} \left| X_j - A_i \right|^2 \right\}^{\frac{1}{2}} \tag{10}$$

Here, SSW represents the Sum of Square Within cluster, T_i denotes the number of data points in cluster i, X_j signifies the j-th data point in cluster i, and A_i represents the centroid of cluster i.

(2) Sum of Squares Between clusters (SSB): This measures the intra-cluster distance, calculated as the square root of the average squared distance between each data point and the centroid of its cluster:

$$\mathrm{SSB}_{i,\,j} = \left\| a_i - a_j \right\|^2 \tag{11}$$

where SSB denotes the Sum of Square Between clusters, a_i is the k-th data point from the centroid of dimension c_i, and a_j is the k-th data point from the centroid of dimension c_j for $i \neq j$.

(3) Ratio calculation (R): This computes the ratio (R_{ij}) to assess the comparison value between cluster i and cluster j, using the equation:

$$R_{i,\,j} = \frac{\mathrm{SSW}_i + \mathrm{SSW}_j}{\mathrm{SSB}_{i,\,j}} \tag{12}$$

Here, R represents the Ratio, SSW denotes the Sum of Square Within the i-th or j-th data cluster, and SSB signifies the Sum of Square Between clusters.

(4) DBI computation: The DBI is calculated as the average of the maximum ratio values for each cluster:

$$\mathrm{DBI} = \frac{1}{N} \sum_{i=1}^{k} \max_{i \neq j} \left(R_{i,\,j} \right) \tag{13}$$

where N is the number of clusters used and $R_{i,\,j}$ represents the ratio.

The Silhouette coefficient, another internal validity measure [18, 22], incorporates two methods: the separation method, which quantifies the distance of a cluster from others, and the cohesion method. Positive values indicate effective clustering, while near-zero or negative values suggest suboptimal assignments. To compute the Silhouette coefficient, the following steps are undertaken:

(1) Average distance calculation (a): This computes the average distance between a data point and all other data points within its cluster:

$$a(i) = \frac{1}{\lfloor A \rfloor - 1} \sum_{j \in A,\, j \neq i} d(i,\, j) \tag{14}$$

Here, $a(i)$ represents the average distance of data point i to all other points in cluster A, A denotes the cluster, and i is the data index.

(2) Minimum distance calculation (d): This determines the average distance of a data point from all other data points in each cluster, then identifies the minimum value:

$$d(i, \complement) = \frac{1}{\lfloor C \rfloor} \sum_{j \in \complement} d(i, j) \tag{15}$$

Here, $d(i, \complement)$ signifies the average difference of object i to all other points in cluster C, C represents another cluster distinct from cluster A, and i is the data index.

(3) Minimum distance to objects in other clusters (b): After computing the average distance to objects in each cluster other than cluster A, the minimum value is selected:

$$b(i) = \min \complement \neq A d(i, j) \tag{16}$$

Here, $b(i)$ represents the smallest value, where C and A are clusters, and i is the data index.

(4) Silhouette Coefficient value (SC): Finally, the silhouette coefficient $S(i)$ is calculated using the formula:

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{17}$$

The Silhouette Index, denoted by $S(i)$, measures how well each data point i fits into its assigned cluster compared to other clusters. It is calculated using two values: $a(i)$, the average distance between i and points in other clusters. A higher Silhouette value indicates better clustering quality.

4. Implementation

We have implemented all algorithms and computations using the C# programming language. Several reasons guided our choice of this language. Firstly, C# offers relative simplicity compared to other languages like Java, C++, and Visual Basic. Secondly, it provides modern features such as Exception Handling, Garbage Collection, Extensible Data Types, and Code Security, along with Object-Oriented features like encapsulation, inheritance, and polymorphism. Thirdly, C# supports the development of various applications including word processing, spreadsheets, and mobile applications. However, there are some drawbacks to using C#. Firstly, its numerous operators and coding flexibility can sometimes confuse programmers, especially those inexperienced in pointer processing. Secondly, compared to other data-oriented programming languages like Python and R, C# lacks sufficient data mining libraries. Therefore, we propose the utilization of the C# programming language to expand the usage of machine learning libraries across various programming languages, particularly benefiting C# programmers who may not wish to venture into other languages. Such implementation choices are aligned with smart-city/Internet of Things (IoT) scenarios—e.g., traffic congestion services—where high-dimensional, non-linear patterns often arise and benefit from kernel methods [26].

Before implementing the algorithms and computations in C#, we developed pseudocodes to facilitate understanding in near-human language. The pseudocode for K-Means is outlined as follows:

Input:

• k: number of clusters specified;

• D: $\{d_1, d_2, d_3, \ldots, d_n\}$ collection of n initial-data items;

Output:

• a set of k clusters.

Method:

(1) Validate that each initial data item D is not empty or non-numeric.

(2) Normalize each initial data item D using the Z-Score method to convert it to normal data.

(3) Arbitrarily select 10 unique combinations of k normal data items, considering each combination as candidate data centroids.

(4) Calculate the SSE value from the candidate centroid data.

(5) Identify the minimum SSE value and select the normal data item as the data-fixed centroid.

(6) Repeat:

(7) Assign each normal data item to a cluster based on the closest centroid.

(8) Calculate the average value of n normal data items as a new centroid for each cluster.

(9) Until: No changes occur.

(10) Calculate the Davies Bouldin Index value.

(11) Calculate the Silhouette coefficient value.

Similarly, the pseudocode for K-Means Kernel is delineated as follows:

Input:

• k: number of clusters specified;

• D: $\{d_1, d_2, d_3, \ldots, d_n\}$ collection of n initial-data items;

Output:

• a set of k clusters.

Method:

(1) Validate that each initial data item D is not empty or non-numeric.

(2) Normalize initial data D using the Z-Score method to convert it to normal data.

(3) Transform normal data using the Kernel Gaussian Radial Basis Function into kernel data.

(4) Arbitrarily select 10 unique combinations of kernel data items k in each combination as candidate centroid data.

(5) Calculate the SSE value from the centroid candidate data.

(6) Identify the minimum SSE value and select the kernel data item as the data-fixed centroid.

(7) Repeat:

(8) Assign each kernel data item to a cluster based on the closest centroid.

(9) Calculate the average value of n kernel data items as a new centroid for each cluster.

(10) Until: no change occurs.

(11) Calculate the Davies Bouldin Index value.

(12) Calculate the Silhouette coefficient value.

All source codes along with the datasets are publicly available on GitHub to facilitate the reproduction of the research results (https://github.com/harunnasrullah/ProjectKmeansKernelGRBF_publish). The dataset is obtained from interviews with five experts dealing with student creativity proposals submitted by 981 private and public universities in Indonesia from 2018 to 2021. Data preparation involves defining the dataset, such as ten rows of data with five variables including the number of active students ($X1$), the number of submitted program proposals ($X2$), the number of proposals satisfying the administration requirements ($X3$), the number of funded proposals ($X4$), and the number of proposals allowed to go to the national competition ($X5$). A sample dataset with 981 rows is presented in Table 1.

Table 1. Sample dataset

No	$X1$	$X2$	$X3$	$X4$	$X5$
0	240	0	0	0	0
1	166	1	0	0	0
2	131	5	5	0	0
3	71	3	3	0	0
...	...	...	...	...	...
979	299	1	1	0	0
980	3,771	110	110	7	0

The graphical representation in Figure 1 depicts the values of two distinct variables, each denoted by points on a graph. These points correspond to individual data values, positioned along the horizontal and vertical axes.

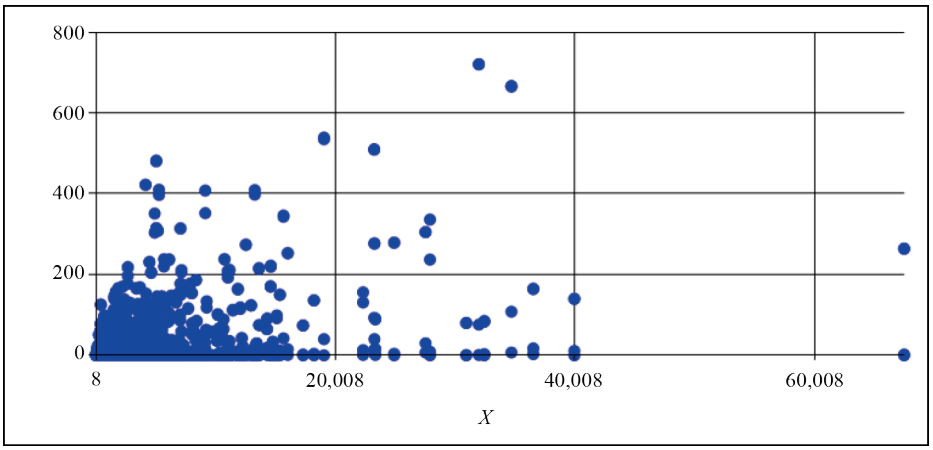

Figure 1. Dataset with 981 rows

Subsequently, employing the Z-score method standardizes the dataset, where positive values exceed the average and negative values fall below it. Table 2 illustrates manual calculations of mean and standard deviation for all attributes, leading to the normalized data showcased in Table 3.

Next, the initial dataset undergoes mapping into a high-dimensional feature space using the Gaussian Radial Basis Function Kernel, resulting in a transformation of data columns from 981 rows and 5 columns to 981 rows and 981 columns, as indicated in Table 4.

Table 2. Mean and standard deviation

	$X1$	$X2$	$X3$	$X4$	$X5$
Mean (μ)	2,641.41	37.24	36.00	1.57	0.08
SD (σ)	4,758.35	69.22	67.78	6.41	0.63

Table 3. *Z*-score normalization result

No	$X1$	$X2$	$X3$	$X4$	$X5$
0	-0.50	-0.53	-0.53	-0.24	-0.13
1	-0.52	-0.52	-0.53	-0.24	-0.13
2	-0.52	-0.46	-0.45	-0.24	-0.13
3	-0.54	-0.49	-0.48	-0.24	-0.13
...	...	...	...	...	...
979	-0.49	-0.52	-0.51	-0.24	-0.13
980	0.23	1.05	1.09	0.84	-0.13

Table 4. Gaussian kernel results

No	0	1	2	3	4	...	979	980
0	1.00	0.99	0.99	0.99	0.78	...	0.99	0.21
1	0.99	1.00	0.99	0.99	0.78	...	0.99	0.21
2	0.99	0.99	1.00	0.99	0.81	...	0.99	0.23
3	0.99	0.99	0.99	1.00	0.80	...	0.99	0.22
...	...	...	...	...	...	...	...	...
979	0.99	0.99	0.99	0.99	0.78	...	1.00	0.22
980	0.21	0.21	0.23	0.22	0.48	...	0.22	1.00

5. Optimization

The Elbow Method is applied to determine the optimal K value for cluster numbers, while the SSE method assists in identifying initial centroids. Table 5 presents Elbow computations for K-Means on Z-score normalization data, performed six times for various K values (2, 3, 4, 5, 6, 7) and three different N values (8, 9, 10), aiding in understanding the data structure.

Table 5. Elbow method results for K-Means

K	SSE			Maximum difference		
	$N = 8$	$N = 9$	$N = 10$	$N = 8$	$N = 9$	$N = 10$
$K = 2$	2.03	1.90	2.06	2.03	1.90	2.06
$K = 3$	0.26	0.65	0.16	1.76	1.24	1.89
$K = 4$	0.72	0.25	0.07	-0.45	0.40	0.09
$K = 5$	0.00	0.01	0.32	0.72	0.23	-0.24
$K = 6$	0.07	0.03	0.91	-0.07	-0.02	-0.59
$K = 7$	0.02	0.08	0.03	0.04	-0.04	0.87
	Max			1.76	1.24	1.89

The Elbow Method, a pivotal technique in cluster analysis, serves to discover the optimal number of clusters (K) within a dataset by scrutinizing the SSE values. Fundamentally, this method operates on the premise that as the number of clusters increases, the corresponding SSE values tend to decrease. However, the identification of the ideal K value

hinges upon the detection of an inflection point on the SSE curve, denoted as the "elbow". This pivotal point, illustrated in Figure 2, delineates a significant decline followed by a phase of relatively stabilized SSE values. Herein lies the essence of optimal cluster determination: the point of maximum curvature, where adding additional clusters does not yield substantial reduction in SSE. In our analysis, this inflection point manifests at $K = 3$, representing the optimal number of clusters for our dataset.

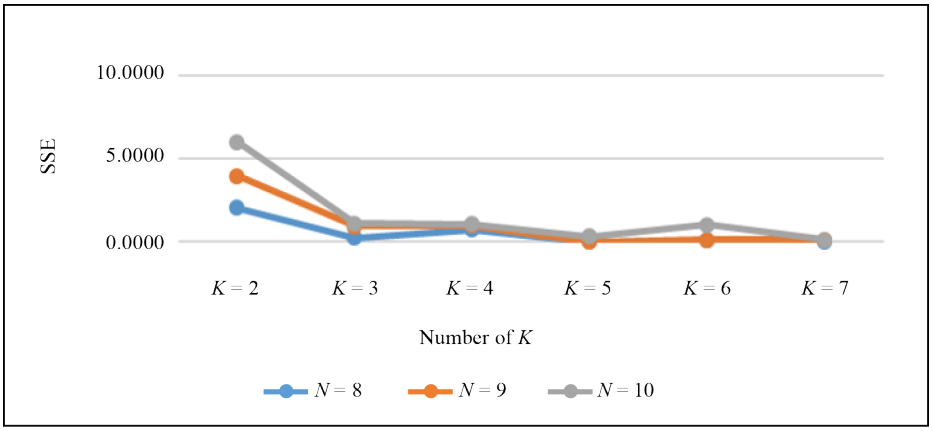

Figure 2. Elbow method result visualization for *K*-Means

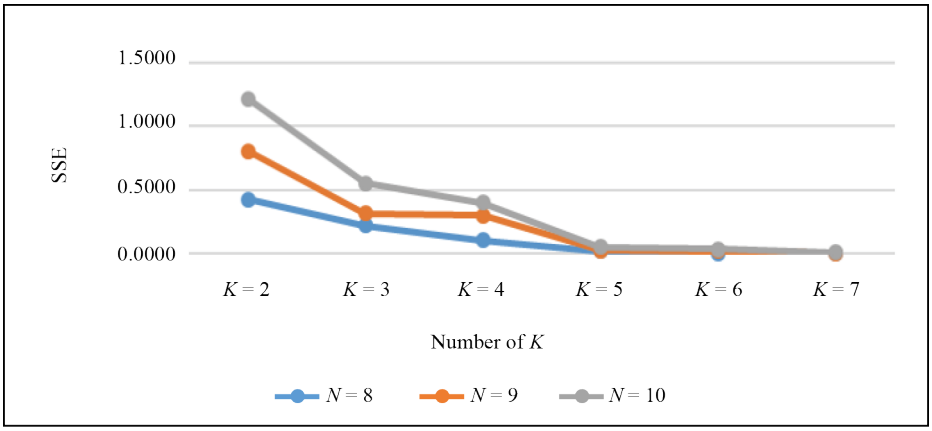

Figure 3. Elbow method result visualization for Kernel *K*-Means

Table 6. Elbow method results for Kernel *K*-Means

	SSE			Maximum difference		
	$N = 8$	$N = 9$	$N = 10$	$N = 8$	$N = 9$	$N = 10$
$K = 2$	0.42	0.37	0.40	0.42	0.37	0.40
$K = 3$	0.21	0.09	0.23	0.20	0.28	0.17
$K = 4$	0.10	0.19	0.10	0.11	-0.10	0.13
$K = 5$	0.01	0.00	0.03	0.08	0.19	0.06
$K = 6$	0.00	0.01	0.01	0.01	-0.01	0.01
$K = 7$	0.00	0.00	0.00	0.00	0.01	0.00
	Max			0.20	0.28	0.17

In Table 6, we present the outcomes of the Elbow Method computation tailored to Kernel K-Means. The identification of the optimal cluster value is visually represented in Figure 3, where the elbow point emerges. This pivotal juncture signifies a significant decrease in the SSE values between adjacent clusters, succeeded by a phase of relative stability. Notably, the elbow manifests at $K = 3$, pinpointing the ideal number of clusters for our analysis.

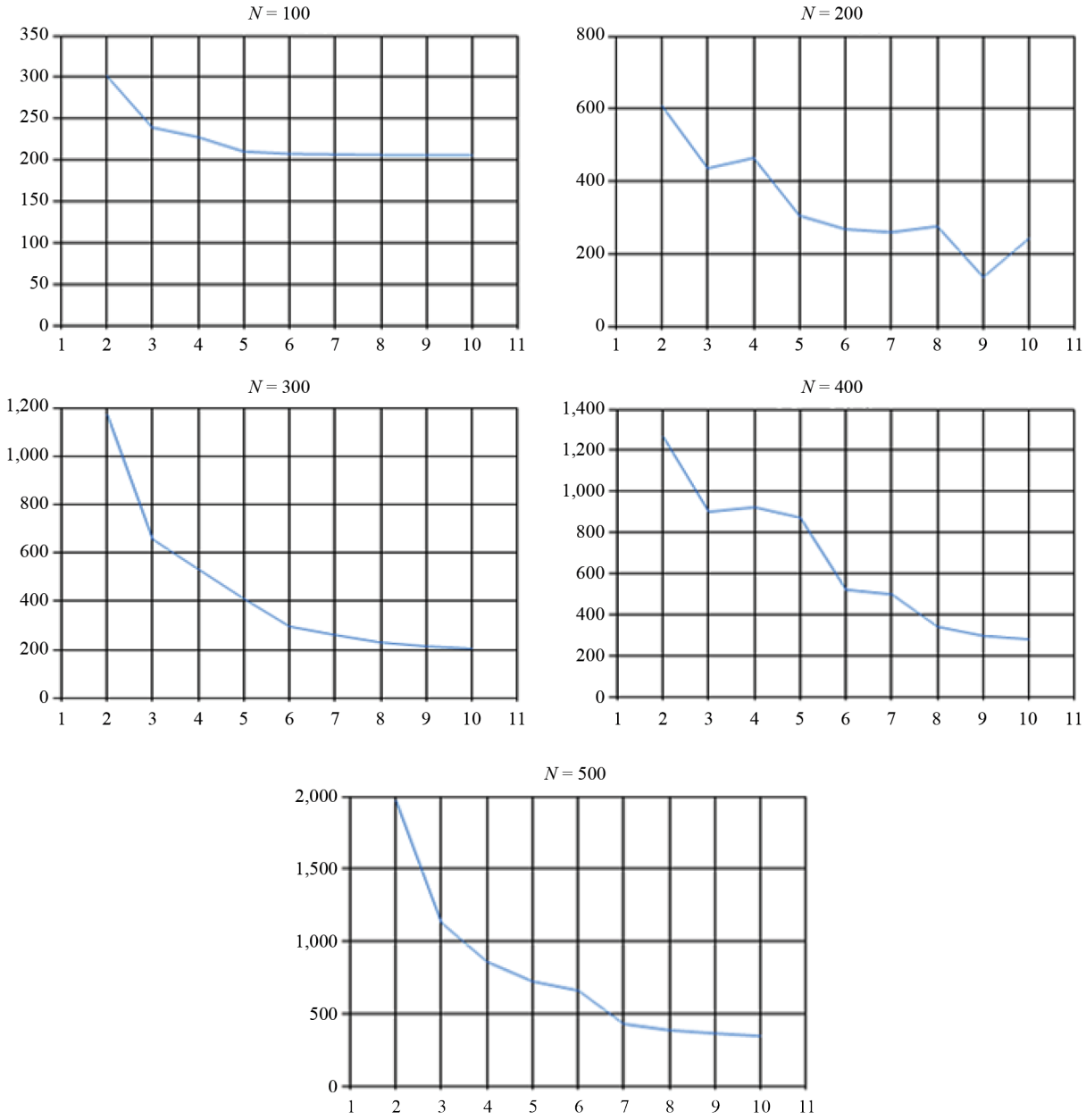

Figure 4. Elbow method for K-Means

We are now ready to execute the Elbow Method computation on our complete dataset, spanning a spectrum of K values (ranging from 2 to 10) and encompassing five distinct N values (100, 200, 300, 400, 500), as depicted in Figure 4. The culmination of this analysis is encapsulated in Table 7, where the results of five distinct SSE computations are presented, each corresponding to varying combinations of K and N values.

Table 7. SSE for *K*-Means

	$N = 100$	$N = 200$	$N = 300$	$N = 400$	$N = 500$
$K = 2$	302.03	608.82	1,172.13	1,267.20	1,973.05
$K = 3$	239.40	436.15	655.89	901.52	1,128.43
$K = 4$	227.77	464.54	532.34	923.50	858.80
$K = 5$	210.35	305.39	409.87	872.78	722.54
$K = 6$	207.63	268.16	295.69	521.55	660.46
$K = 7$	206.91	259.93	260.84	499.09	428.99
$K = 8$	206.22	276.21	229.41	341.53	385.71
$K = 9$	205.99	136.49	214.04	296.71	363.87
$K = 10$	205.94	243.05	204.35	280.44	345.42

Utilizing the SSE data provided in Table 7, we computed the discrepancies between each value of *K* and its successive counterpart, resulting in the generation of differential data encapsulated in Table 8, alongside corresponding experimental outcomes. Remarkably, across varying sample sizes (*N*) ranging from 100 to 500, the most substantial disparities consistently emerged at $K = 3$. This consistent trend underscores the robustness of $K = 3$ as the optimal number of clusters, a conclusion supported by empirical evidence derived from our analysis.

Table 8. Difference in SSE for *K*-Means

$N = 100$	$N = 200$	$N = 300$	$N = 400$	$N = 500$
302.03	608.82	1,172.13	1,267.20	1,973.05
62.62	172.66	516.24	365.67	844.62
11.62	-28.38	123.54	-21.97	269.62
17.42	159.15	122.47	50.71	136.26
2.71	37.22	114.17	351.22	62.07
0.71	8.23	34.85	22.46	231.47
0.69	-16.28	31.42	157.55	43.28
0.22	139.72	15.37	44.82	21.84
0.05	-106.56	9.68	16.26	18.44
62.62	172.66	516.24	365.67	844.62

Furthermore, we leverage kernel transformation data to conduct the Elbow Method for Kernel *K*-Means, aimed at identifying the optimal *K* value for clustering. The SSE data obtained from experiments conducted across different *N* values (100, 200, 300, 400, and 500) are visually represented in Figure 5, providing insights into the ideal cluster configuration.

In Kernel *K*-Means analysis, the optimal cluster configuration is also characterized by the emergence of an elbow point, where the SSE curve experiences a discernible decrease followed by a phase of relative stability. Table 9 presents the outcomes of the SSE computations, conducted across various *K* values (ranging from 2 to 10) and *N* data values (100, 200, 300, 400, 500).

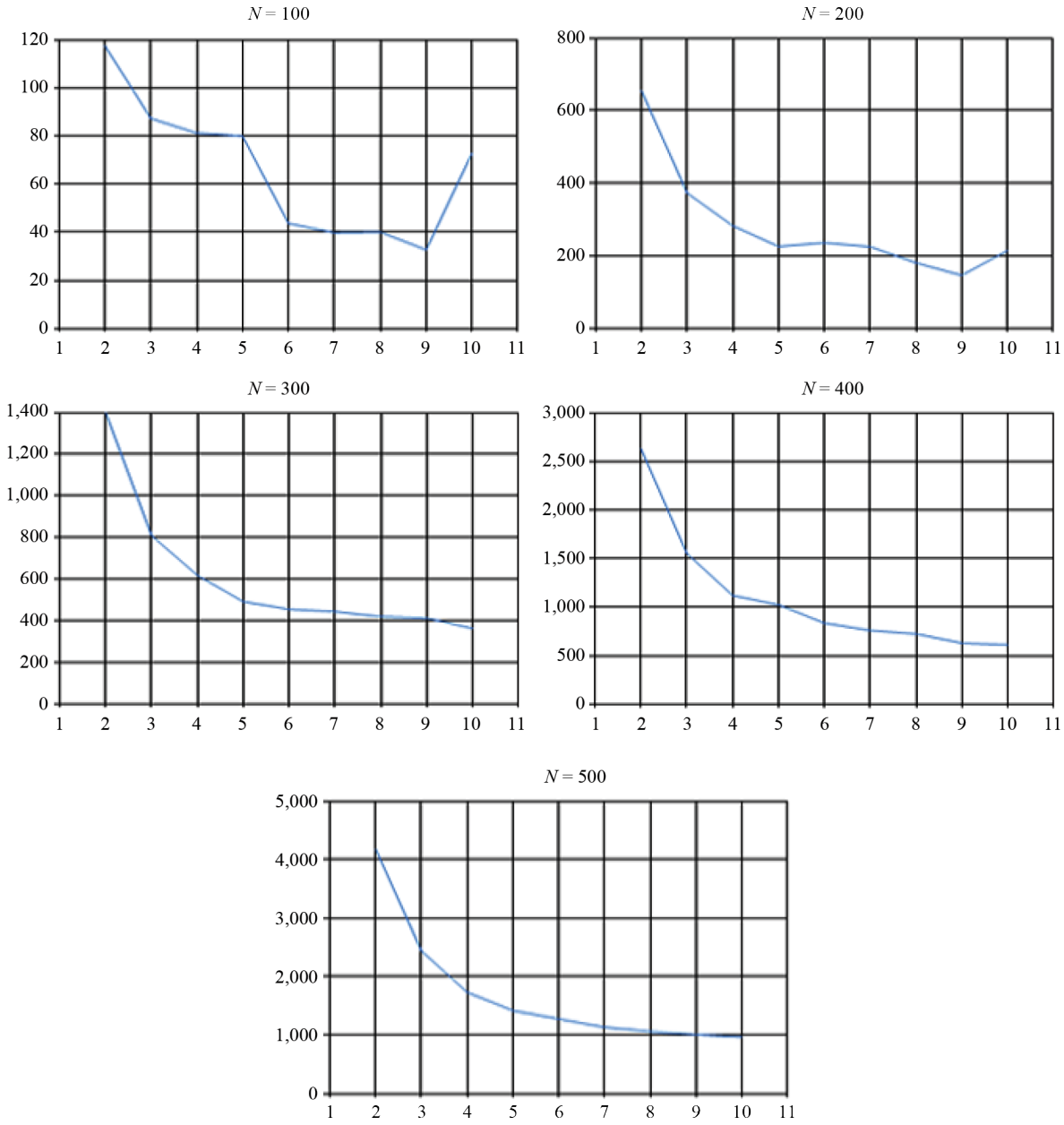

Figure 5. Elbow method for Kernel *K*-Means

Upon examining the test results in Table 9, notably for $N = 100$, which exhibits the largest SSE difference of 36.2828 at $K = 6$, and $N = 200$, which also presents a significant difference, further analysis in Table 10 involves computing the discrepancies between each K value and its succeeding counterpart. Particularly noteworthy is the highest SSE difference observed at $K = 3$, amounting to 282.50. Moreover, for $N = 300$ and $N = 400$, the highest SSE differences occur at $K = 3$, with values of 590.73 and 1,077.1779, respectively, while $N = 500$ reveals identical SSE differences at $K = 3$, further reinforcing the robustness of this cluster configuration.

Table 9. SSE for Kernel *K*-Means

	$N = 100$	$N = 200$	$N = 300$	$N = 400$	$N = 500$
$K = 2$	117.38	654.96	1,399.73	2,629.44	4,173.50
$K = 3$	87.19	372.46	808.99	1,552.26	2,437.47
$K = 4$	81.13	282.32	617.49	1,118.69	1,729.84
$K = 5$	79.85	225.68	491.16	1,023.20	1,419.67
$K = 6$	43.57	235.59	454.29	835.17	1,275.98
$K = 7$	39.76	224.94	443.38	758.32	1,135.84
$K = 8$	40.01	180.78	419.21	724.80	1,060.93
$K = 9$	32.62	146.30	411.51	627.86	1,008.53
$K = 10$	72.75	214.11	362.33	610.23	968.20

Table 10. Difference in SSE for Kernel *K*-Means

$N = 100$	$N = 200$	$N = 300$	$N = 400$	$N = 500$
117.38	654.96	1,399.73	2,629.44	4,173.50
30.19	282.50	590.73	1,077.17	1,736.02
6.05	90.13	191.50	433.57	707.63
1.27	56.64	126.32	95.48	310.16
36.28	-9.90	36.86	188.03	143.69
3.81	10.64	10.91	76.85	140.13
-0.25	44.15	24.17	33.51	74.90
7.39	34.48	7.70	96.94	52.40
-40.13	-67.81	49.17	17.63	40.33
36.28	282.50	590.73	1,077.17	1,736.02

The analysis suggests that $K = 3$ is optimal for the number of clusters, despite the initial trial at $N = 100$ yielding the highest difference at $K = 6$. Subsequent experiments, conducted across $N = 200$, $N = 300$, $N = 400$, and $N = 500$, consistently revealed the largest difference at $K = 3$.

Following the determination of three clusters for either *K*-Means or Kernel *K*-Means, the next step involves identifying the initial centroid. This process relies on the SSE value and entails several steps. Initially, centroids are created and their SSE values computed. Subsequently, these centroids are retained for further analysis, and the process is iterated until the requisite number of centroids has been tested. Finally, the centroid with the lowest SSE value is selected from among all the candidates generated during the initial stage. For instance, the centroid with the lowest SSE value of 3,108.00 is identified in the tenth randomization, with the data row combination 917 | 807 | 692, as detailed in Table 11.

Table 12 presents the centroid data for *K*-Means clusters, with *C*0 comprising 917 data rows, *C*1 comprising 807 data rows, and *C*2 comprising 692 data rows.

Table 11. SSE for *K*-Means

Sequence	Row combinations	SSE
1	291 | 952 | 790 |	3,527.50
2	322 | 459 | 940 |	5,248.47
3	45 | 841 | 320 |	4,288.37
4	237 | 935 | 976 |	3,833.93
5	428 | 49 | 651 |	4,913.58
6	620 | 143 | 326 |	4,861.19
7	343 | 525 | 686 |	3,580.03
8	534 | 619 | 361 |	5,153.60
9	726 | 713 | 36 |	4,595.34
10	917 | 807 | 692 |	3,108.00

Table 12. Initial centroids in *K*-Means

Centroid	No	*X*1	*X*2	*X*3	*X*4	*X*5
*C*0	917	0.08	-0.50	-0.53	-0.24	-0.13
*C*1	807	0.72	2.90	2.95	2.40	1.43
*C*2	692	-0.09	-0.35	-0.33	-0.08	-0.13

Similar to *K*-Means, the *K*-Means Kernel also generates 10 random permutations of three centroids based on the Kernel transformation data to establish the initial centroid. The centroid with the lowest SSE value, identified in the 10th randomization, consists of data rows 765, 796, and 469, as indicated by Table 13. Its SSE value is recorded as 10,459.87. This process ensures the selection of centroids that best represent the data distribution for effective clustering.

Figure 6 presents a scatter plot graphically depicting 981 data points to visualize the centroid of *K*-Means clustering.

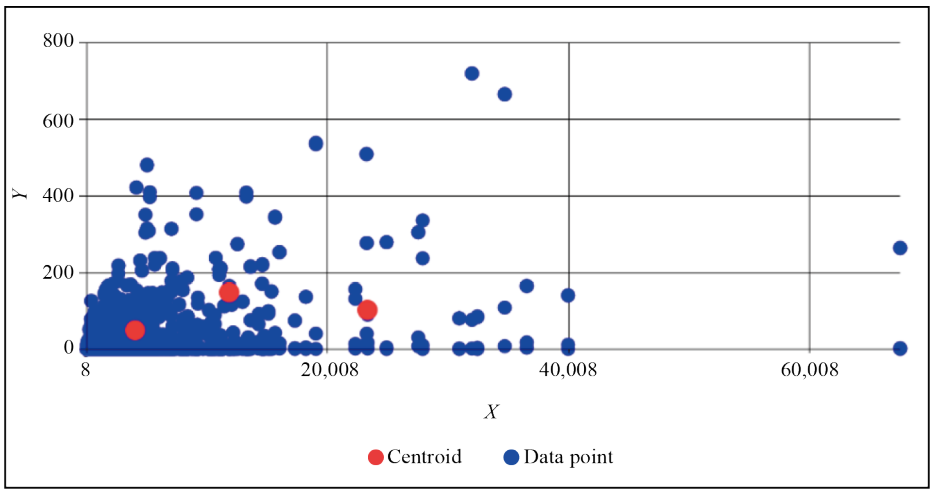

Figure 6. Scatter plot of *K*-Means centroids

Table 13. SSE for initial centroids in Kernel *K*-Means

Sequence	Row combination	SSE
1	437 | 354 | 363 |	80,228.43
2	926 | 131 | 404 |	71,676.11
3	786 | 590 | 626 |	37,626.38
4	923 | 667 | 487 |	15,957.53
5	591 | 51 | 53 |	51,643.64
6	99 | 810 | 94 |	18,413.92
7	940 | 288 | 316 |	74,440.02
8	608 | 653 | 862 |	15,773.25
9	745 | 730 | 723 |	18,649.70
10	765 | 796 | 469 |	10,459.87

This centroid initiates the *K*-Means process, as depicted in Table 14, which illustrates the centroid data for *K*-Means clusters. Specifically, it assigns cluster *C*0 to the 765th data row, cluster *C*1 to the 796th data row, and cluster *C*2 to the 469th data row. These assignments form the basis for further iterations of the *K*-Means algorithm to optimize cluster formation.

Table 14. Initial centroids of Kernel *K*-Means

Centroid	No	1	2	3	…	979	980	981
*C*0	765	0.36	0.36	0.39	…	0.90	0.37	0.73
*C*1	796	0.00	0.00	0.00	…	0.03	0.00	0.20
*C*2	469	0.98	0.98	0.98	…	0.68	0.98	0.24

Figure 7 presents a scatter plot graphically depicting 981 data points to visualize the centroid of Kernel *K*-Means clustering.

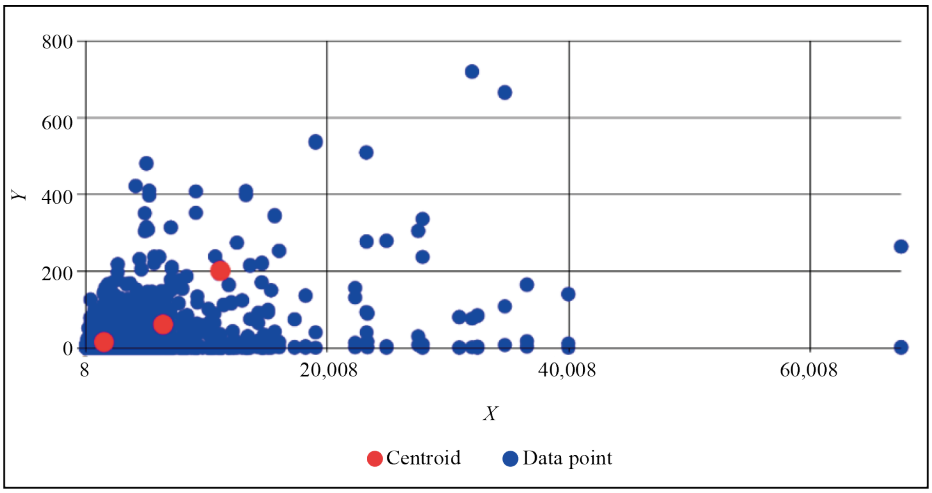

Figure 7. Scatter plot of Kernel *K*-Means centroids

The initial centroid with the combination of 917 | 807 | 692 at the tenth randomization, resulting in a SSE value of 3,108.00, was identified as the optimal cluster K value in the previous stage of the K-Means analysis. However, the detailed iteration process is not displayed as the program only provides a summary of the number of clusters in multiple experiments.

The first step in K-Means computation involves initializing the number of clusters—$C0 = 450$ data, $C1 = 335$ data, and $C2 = 196$ data. Subsequently, the computation undergoes 20 iterations, culminating in the final cluster results: $C0 = 77$ data, $C1 = 6$ data, and $C2 = 898$ data. Table 15 presents the K-Means clustering data for cluster $C0$, consisting of 77 rows of data. Further details of the iterative process are not included but can be inferred from the outcomes provided.

Table 15. K-Means iteration for cluster 0

No	Cluster	No	Cluster	No	Cluster
80	$C0$	741	$C0$	877	$C0$
81	$C0$	749	$C0$	880	$C0$
120	$C0$	750	$C0$	882	$C0$
152	$C0$	752	$C0$	918	$C0$
…	…	…	…	…	…
…	…	…	…	…	…
712	$C0$	860	$C0$		
723	$C0$	865	$C0$		
724	$C0$	875	$C0$		

Table 16 presents the K-Means clustering data for cluster $C1$, comprising 6 rows of data.

Table 16. Results of K-means iteration for cluster 1

No	Cluster
557	$C1$
672	$C1$
782	$C1$
806	$C1$
808	$C1$
812	$C1$

Table 17 displays a portion of the K-Means clustering data for cluster $C2$, consisting of 898 rows of data.

Figure 8 presents a scatter plot graphically depicting 981 data points to visualize the outcomes of K-Means clustering.

In the preceding phase of the Kernel K-Means analysis, it was established that the optimal cluster value was $K = 3$. The initial centroid, determined in the 10th randomization and characterized by the combination of 765 | 796 | 469 |, yielded a SSE value of 10,459.87. Rather than presenting the exhaustive iteration process, the application provides a concise summary of the clustering results across multiple experiments.

Table 17. Results of *K*-Means iteration cluster 2 with 981 rows of data

No	Cluster	No	Cluster	No	Cluster
0	*C*2	30	*C*2	60	*C*2
1	*C*2	31	*C*2	61	*C*2
2	*C*2	32	*C*2	62	*C*2
3	*C*2	33	*C*2	63	*C*2
…	…	…	…	…	…
…	…	…	…	…	…
28	*C*2	58	*C*2	…	…
29	*C*2	59	*C*2	…	…

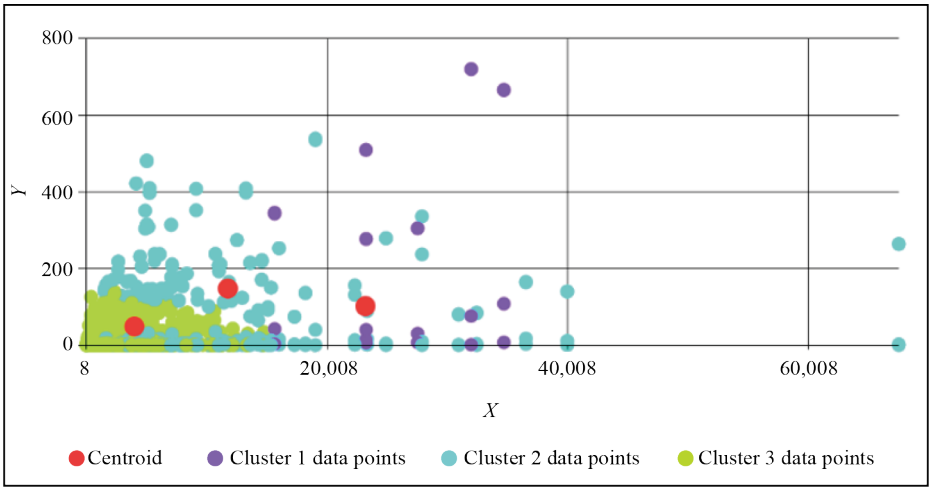

Figure 8. Scatter plot of *K*-Means results

Table 18. Results of Kernel *K*-Means iteration for cluster 0

No	Cluster	No	Cluster	No	Cluster
62	*C*0	593	*C*0	765	*C*0
63	*C*0	600	*C*0	776	*C*0
68	*C*0	602	*C*0	783	*C*0
74	*C*0	603	*C*0	785	*C*0
…	…	…	…	…	…
…	…	…	…	…	…
591	*C*0	762	*C*0	978	*C*0
592	*C*0	764	*C*0		

The first step in the *K*-Means computation involves initializing the number of clusters, with *C*0 comprising 127 data points, *C*1 with 85 data points, and *C*2 with 769 data points. Subsequently, the computation progresses through seven iterations, culminating in the final cluster sizes: $C0 = 131$ data points, $C1 = 104$ data points, and $C2 = 746$ data points.

Table 18 displays the clustering outcomes for Kernel *K*-Means, focusing on cluster *C*0, which encompasses 131 rows of data.

Table 19 shows the Kernel *K*-Means clustering data for cluster *C*1 with 104 rows of data.

Table 19. Results of Kernel *K*-Means iteration for cluster 1

No	Cluster	No	Cluster	No	Cluster
80	*C*1	693	*C*1	817	*C*1
81	*C*1	702	*C*1	832	*C*1
120	*C*1	708	*C*1	836	*C*1
121	*C*1	712	*C*1	844	*C*1
…	…	…	…	…	…
…	…	…	…	…	…
686	*C*1	814	*C*1	980	*C*1
687	*C*1	815	*C*1		

Table 20 shows part of the data from Kernel *K*-Means clustering for cluster *C*2 with 748 rows of data.

Table 20. Results of Kernel *K*-Means iteration for cluster 2

No	Cluster	No	Cluster	No	Cluster
0	*C*2	44	*C*2	99	*C*2
1	*C*2	45	*C*2	100	*C*2
2	*C*2	46	*C*2	101	*C*2
3	*C*2	47	*C*2	102	*C*2
…	…	…	…	…	…
…	…	…	…	…	…
42	*C*2	97	*C*2	…	…
43	*C*2	98	*C*2	…	…

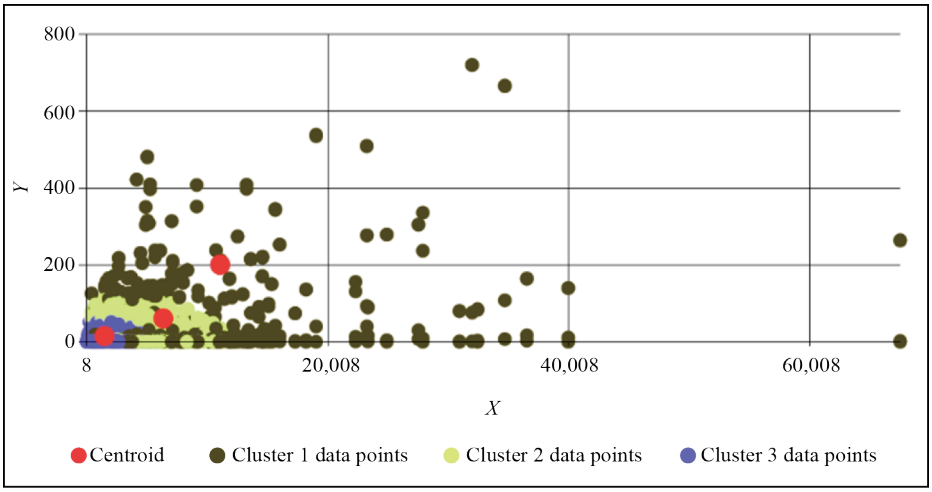

Figure 9. Scatter plot of Kernel *K*-Means

Figure 9 illustrates the results of the Kernel K-Means clustering in a scatter plot, showcasing 981 rows and 981 columns of data on the graph. This visualization provides insights into the clustering patterns and distribution of the data points.

6. Evaluation

The evaluation of clustering results involves three steps: calculating the DBI values through internal testing. This process includes computing the Ratio value, the SSW, and the SSB. Initially, the intra-cluster distance between each data point and the converged cluster centroid is computed to determine the SSW for K-Means, as shown in Table 21. This step provides insights into the compactness of clusters and their separation from each other.

Table 21. SSW results

Cluster	K-Means	Kernel K-Means
$C1$	1.33	0.67
$C2$	0.00	0.00
$C3$	0.72	0.38

Secondly, the centroids listed in Table 22 are utilized to calculate the SSB, indicating the separation between clusters. Additionally, the Ratio metric is computed to compare the i-th and j-th clusters in K-Means. These centroids, having converged, yield the SSB and Ratio comparisons presented below. This analysis provides insights into the distinctiveness and separation of clusters within the dataset.

Table 22. SSB and ratio results

Cluster	K-Means		Kernel K-Means	
	SSB	Ratio	SSB	Ratio
1-2	3.30	0.40	1.52	0.44
1-3	3.20	0.64	2.05	0.51
2-3	6.21	0.11	2.36	0.16

Table 23. Illustration of ratio and DBI matrices

Ratio	K-Means				Kernel K-Means			
	1	2	3	Max	1	2	3	Max
1	0.00	0.40	0.64	0.64	0.00	0.44	0.51	0.51
2	0.40	0.00	0.11	0.40	0.44	0.00	0.16	0.44
3	0.64	0.11	0.00	0.64	0.51	0.16	0.00	0.51
	DBI 0.56				DBI 0.49			

Thirdly, the computation of the DBI involves constructing a ratio value, determining its maximum value, and subsequently averaging the maximum values across matrices, as illustrated in Table 23. This step provides a comprehensive evaluation of the clustering quality by considering the inter-cluster and intra-cluster distances. The resulting DBI values offer insights into the compactness and separation of clusters, aiding in the determination of the optimal number of clusters for the dataset.

The optimal cluster quality is indicated by the lowest DBI index value, which falls between 0 and 1. For K-Means, the DBI value is 0.56, while for Kernel K-Means, it is 0.49. This suggests that Kernel K-Means outperforms K-Means in terms of cluster quality.

The internal testing of clustering results involves four steps to obtain Silhouette coefficient values: (1) computing $a(i)$, which is the average distance of object (i) to all other objects; (2) computing $d(i, C)$, which is the difference in the average distance of object (i) to all other objects; (3) computing $b(i)$, which represents the smallest value; and (4) computing $S(i)$ = Silhouette Index calculation. The interpretation of the Silhouette coefficient is presented in Table 24, providing insight into the quality and cohesion of the clusters.

Table 24. Interpretation of silhouette coefficient

Silhouette coefficient	Interpretation
0.71-1.00	Strong
0.51-0.70	Good
0.26-0.50	Weak
$\leq$ 0.25	Unstructured

Before calculating the Silhouette coefficient for K-Means, the clustered data is sorted, as shown in Table 25. This sorting step ensures that the silhouette values are computed correctly, providing insights into the quality of the clustering results.

Table 25. Sorted K-Means results

No	$X1$	$X2$	$X3$	$X4$	$X5$	Cluster
3	0.40	0.75	0.75	0.39	0.95	$C1$
8	2.26	0.48	0.48	-0.58	-0.59	$C1$
10	-0.63	0.77	0.77	0.43	0.06	$C1$
6	0.20	2.10	2.09	2.60	2.49	$C2$
1	-0.82	-0.97	-0.98	-0.61	-0.37	$C3$
2	-0.84	-0.60	-0.60	-0.58	-0.59	$C3$
4	-0.63	-0.97	-0.96	-0.65	-0.59	$C3$
5	0.98	-0.33	-0.32	-0.26	-0.59	$C3$
7	-0.31	-0.47	-0.49	-0.51	-0.37	$C3$
9	-0.61	-0.75	-0.74	-0.23	-0.37	$C3$

The third data point's value of $b(i)$, representing the lowest value among all $d(i, C)$, is determined to be 2.8354, as observed in the computation results. Following this, the computation of $d(i, C)$ is executed using the same methodology, and the outcomes are detailed in Table 26. This step allows for a comprehensive analysis of the Silhouette coefficient, aiding in the evaluation of the clustering performance.

Table 26. Silhouette *K*-Means calculation results

No	Cluster	$a(i)$	$d(i, C1)$	$d(i, C2)$	$d(i, C3)$	$b(i)$	$S(i)$
3	$C1$	2.00	-	3.30	2.83	2.83	0.29
8	$C1$	2.90	-	5.40	3.14	3.14	0.07
10	$C1$	2.27	-	3.85	2.44	2.44	0.07
6	$C2$	0.00	4.18	-	5.89	4.18	1.00
1	$C3$	0.86	3.22	6.21	-	3.22	0.73
2	$C3$	0.82	2.90	5.96	-	2.90	0.71
4	$C3$	0.82	3.19	6.30	-	3.19	0.74
5	$C3$	1.78	2.18	5.50	-	2.18	0.18
7	$C3$	0.83	2.50	5.62	-	2.50	0.66
9	$C3$	0.78	2.83	5.76	-	2.83	0.72
			Silhouette				0.52

The Silhouette coefficient calculation for Kernel *K*-Means follows the same procedure as for *K*-Means, with the results depicted in Table 27. This evaluation provides insights into the clustering quality and the degree of separation between clusters, facilitating the comparison between different clustering algorithms. Additionally, it aids in determining the optimal number of clusters and assessing the overall effectiveness of the clustering process.

Table 27. Silhouette Kernel *K*-Means calculation results

No	Cluster	$a(i)$	$d(i, 1)$	$d(i, 2)$	$d(i, 3)$	$b(i)$	$S(i)$
3	$C1$	0.92	-	1.52	1.90	1.52	0.39
8	$C1$	1.35	-	1.51	1.98	1.51	0.10
10	$C1$	0.99	-	1.63	1.68	1.63	0.38
6	$C2$	0.00	1.55	-	2.31	1.55	1.00
1	$C3$	0.45	2.00	2.36	-	2.00	0.77
2	$C3$	0.41	1.97	2.39	-	1.97	0.79
4	$C3$	0.43	2.01	2.38	-	2.01	0.78
5	$C3$	1.23	1.27	1.89	-	1.27	0.03
7	$C3$	0.49	1.88	2.40	-	1.88	0.73
9	$C3$	0.40	1.97	2.41	-	1.97	0.79
			Silhouette				0.58

Based on the interpretation of the Silhouette coefficient, the calculation results show a favorable outcome for *K*-Means with a Silhouette coefficient of 0.52 and an even better result for Kernel *K*-Means with a Silhouette coefficient of 0.58. These coefficients indicate a well-defined and distinct cluster structure. Therefore, it can be concluded that the Silhouette coefficient for Kernel *K*-Means is superior to that of *K*-Means, suggesting that the clustering performance of Kernel *K*-Means is more robust and effective.

7. Discussion

The evaluation stage of K-Means and Kernel K-Means was anticipated to yield high-quality findings. The ideal cluster quality is indicated by the DBI value, which falls within the interval (0, 1) with a minimum value close to 0 and a maximum value near 1 for the Silhouette coefficient, indicating the best possible cluster quality.

The experiments conducted revealed that the DBI calculation procedure with a relatively large amount of data can impact the efficiency of the subsequent processes. The DBI values obtained for Kernel K-Means were 0.58 and for K-Means were 0.79, respectively. Thus, it can be concluded that the DBI for Kernel K-Means outperforms that of K-Means.

Similarly, the experiments demonstrated that the sophisticated process of calculating the Silhouette coefficient with a comparatively large dataset can affect the pace of subsequent operations. K-Means yielded a Silhouette coefficient of 0.76, while Kernel K-Means achieved a value of 0.71. Therefore, it can be inferred that the Silhouette coefficient is preferable to DBI for assessing the performance of Kernel K-Means over K-Means.

This discussion indicates that Kernel K-Means exhibits superior performance compared to traditional K-Means, particularly in terms of cluster quality assessment using both DBI and Silhouette coefficient metrics. These findings underscore the effectiveness and robustness of Kernel K-Means in clustering analysis.

While the experimental results presented in this study are promising, it is important to note that they are based on a single, domain-specific dataset related to student creativity proposals from Indonesian universities. The structured nature of this dataset provides a clear and controlled environment for comparing clustering methods. However, we recognize that to establish broader applicability, future evaluations must include multiple datasets with different structures, sizes, and complexity levels. Such datasets would help assess the robustness and adaptability of Kernel K-Means under varying data distributions and domain constraints.

To evaluate the efficiency of the proposed Kernel K-Means approach compared to the K-Means algorithm, we analyse their computational complexities.

The time complexity of K-Means is:

$$O(n \cdot k \cdot i \cdot d) \tag{18}$$

where:

- n = number of data points,
- k = number of clusters,
- i = number of iterations to converge,
- d = number of features/dimensions.

The Kernel K-Means algorithm, employing a Gaussian Radial Basis Function (RBF) kernel, requires computing a full kernel (Gram) matrix, increasing its time complexity to:

$$O\left(n^2 \cdot k \cdot i\right) \tag{19}$$

This quadratic term stems from pairwise similarity computations between all data points, which is especially intensive for large datasets.

In our implementation:

- K-Means converged in 20 iterations, processing Z-score normalized data in a 5-dimensional space.
- Kernel K-Means converged in 7 iterations but required handling a 981×981 kernel matrix, significantly increasing memory and processing cost.

Despite higher computational demands, Kernel K-Means demonstrated superior clustering performance, with better DBI (0.49 vs. 0.56) and Silhouette Coefficient (0.58 vs. 0.52). These improvements highlight a meaningful trade-off

between computational cost and clustering quality, making Kernel K-Means a justified choice for high-dimensional, non-linear datasets.

To assess the robustness of both clustering methods under suboptimal conditions, we simulated a scenario where the number of clusters was deliberately set to $K = 6$, instead of the optimal $K = 3$ determined using the Elbow Method. The evaluation metrics are summarized in Table 28. As expected, the SSE value decreased due to over-clustering, but this was accompanied by an increase in DBI and a drop in the Silhouette Coefficient, especially in traditional K-Means.

Notably, Kernel K-Means exhibited better resilience, with lower DBI (0.62) and higher Silhouette Coefficient (0.49) compared to K-Means. This indicates that Kernel K-Means maintains better cluster cohesion and separation even under incorrect K selection. These results highlight its suitability for high-dimensional or non-linearly separable datasets, especially when the true number of clusters is not known in advance.

Table 28. Evaluation metrics for incorrect K selection ($K = 6$)

Metric	K-Means ($K = 6$)	Kernel K-Means ($K = 6$)
SSE	210.35	43.57
DBI	0.84	0.62
Silhouette coefficient	0.42	0.49

8. Conclusion

In comparing the performance of K-Means and Kernel K-Means, the results demonstrate that mapping non-linearly separable data into high-dimensional feature space using the Kernel Gaussian Radial Basis Function yields significant differences in clustering outcomes.

K-Means converges in the 20th iteration, resulting in three clusters: $C0 = 77$, $C1 = 6$, and $C2 = 898$, with a DBI evaluation value of 0.79 and a Silhouette coefficient of 0.76. On the other hand, Kernel K-Means converges in the seventh iteration, producing three clusters: $C1 = 131$, $C2 = 104$, and $C2 = 748$, with a DBI value of 0.58 and a Silhouette coefficient of 0.71.

The superiority of the Silhouette coefficient for K-Means over Kernel K-Means is evident, as is the superiority of the DBI value for Kernel K-Means. Additional experiments were conducted to validate these evaluation results, varying the values of N, K, and centroids. The findings consistently showed that while the DBI for Kernel K-Means consistently outperformed K-Means, the value of the Silhouette coefficient for Kernel K-Means varied depending on the type of data and the number of rows.

Using the Elbow Method with different K values (ranging from 2 to 10) and varying amounts of N data (ranging from 100 to 500) in five calculations, optimal K cluster values were determined. K-Means yielded optimal K cluster values at $K = 3$, while Kernel K-Means also yielded optimal K cluster values at $K = 3$.

The method for choosing the initial centroid in K-Means involved generating ten random combinations of three centroids to obtain the best initial centroid with a minimum SSE of 3,108.00 on lines 917 | 807 | 692 for Z-score normalized data. For Kernel K-Means, ten random combinations of three centroids were generated on the data obtained from Kernel transformation to obtain the best initial centroid with a minimum SSE of 10,459.87.

All these analyses and evaluation results were obtained using C# as the programming language, providing valuable insights into the effectiveness of both K-Means and Kernel K-Means in clustering analysis.

9. Future work

This study encountered challenges related to data processing, particularly due to the curse of dimensionality inherent in calculating the Kernel Gaussian Radial Basis Function to transform data into higher dimensions. To address this issue, future work should prioritize dimensionality reduction techniques before clustering to refine the dataset.

Additionally, it is recommended to conduct further experiments using a diverse range of datasets to determine the optimal number of K clusters more accurately, leveraging methods such as the Elbow Method. Furthermore, future research should delve into more comprehensive experiments to identify the most suitable centroids, as the choice of centroid significantly influences clustering outcomes.

Given the computational demands of processing high-dimensional data, future work may also involve investing in larger software and hardware resources to support data visualization and analysis, potentially through the development of dedicated systems. By addressing these areas, future studies can enhance the effectiveness and robustness of clustering algorithms in handling complex datasets.

In addition, we intend to evaluate the proposed method on multiple publicly available benchmark datasets from diverse domains (e.g., health, finance, image features) that vary in complexity and dimensionality. This will allow for comprehensive comparative analysis and further validation of the method's performance and scalability.

Acknowledgments

The project was funded by KAU Endowment (WAQF) at king Abdulaziz University, Jeddah, Saudi Arabia. The authors, therefore, acknowledge with thanks WAQF and the Deanship of Scientific Research (DSR) for technical and financial support.

Conflict of interest

The authors declare no competing financial interest.

References

[1] Dunham MH. *Data Mining: Introductory and Advanced Topics*. 1st ed. Pearson; 2006. p.1-89.

[2] Alpaydin E. *Introduction to Machine Learning (Adaptive Computation and Machine Learning Series)*. 2nd ed. The MIT Press; 2010.

[3] Dhillon IS, Guan Y, Kulis B. Kernel *K*-means, spectral clustering and normalized cuts. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery; 2004. p.551-556.

[4] Bramantoro A, Krishnaswamy S, Indrawan M. A semantic distance measure for matching web services. In: *Web Information Systems Engineering–WISE 2005 Workshops (WISE 2005)*. Springer; 2005. p.217-226.

[5] Bramantoro A, Ishida T. User-centered QoS in combining web services for interactive domain. In: *2009 Fifth International Conference on Semantics, Knowledge and Grid*. Zhuhai, China: IEEE; 2009. p.41-48.

[6] Liu B, Zhang T, Li Y, Liu Z, Zhang Z. Kernel probabilistic *k*-means clustering. *Sensors*. 2021; 21(5): 1892. Available from: https://doi.org/10.3390/s21051892.

[7] Liu X, Zhou S, Liu L, Tang C, Wang S, Liu J, et al. Localized simple multiple kernel *k*-means. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, Canada: IEEE; 2021. p.9273-9281.

[8] Rashed AB, Hamdan H, Sharef NM, Sulaiman MN, Yaakob R, Abubakar M. Multi-objective clustering algorithm using particle swarm optimization with crowding distance (MCPSO-CD). *International Journal of Advances in Intelligent Informatics*. 2020; 6(1): 72-81. Available from: https://doi.org/10.26555/ijain.v6i1.366.

[9] Rustam Z, Nadhifa F. Application of kernel spherical *k*-means for intrusion detection systems. *Journal of Physics: Conference Series*. 2019; 1218(1): 012037. Available from: https://doi.org/10.1088/1742-6596/1218/1/012037.

[10] Vankadara LC, Ghoshdastidar D. On the optimality of kernels for high-dimensional clustering. *arXiv:1912.00458*. 2019. Available from: https://doi.org/10.48550/arXiv.1912.00458.

[11] Wang S, Gittens A, Mahoney MW. Scalable kernel *k*-means clustering with nystrom approximation: Relative-error bounds. *Journal of Machine Learning Research*. 2019; 20: 1-49.

[12] Rustam Z, Fitri SG, Selsi R, Pandelaki J. The global kernel *k*-means clustering algorithm for cerebral infarction classification. *Journal of Physics: Conference Series*. 2019; 1417(1): 012027. Available from: https://doi.org/10.1088/1742-6596/1417/1/012027.

[13] Ning C, Hongyi Z. An optimizing algorithm of non-linear *k*-means clustering. *International Journal of Database Theory and Application*. 2016; 9(4): 97-106.

[14] Tzortzis GF, Likas AC. The global kernel *k*-means algorithm for clustering in feature space. *IEEE Transactions on Neural Networks*. 2009; 20(7): 1181-1194. Available from: https://doi.org/10.1109/TNN.2009.2019722.

[15] Han J, Kamber M, Pei J. *Data Mining: Concepts and Techniques*. 3rd ed. Morgan Kaufmann Publishers; 2012.

[16] Liao TW, Triantaphyllou E. *Recent Advances in Data Mining of Enterprise Data: Algorithms and Applications*. World Scientific; 2007.

[17] Berry MJA, Linoff GS. *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*. 2nd ed. John Wiley & Sons, Inc.; 2004.

[18] Larose DT, El Atia S, Ipperciel D. *Data Mining and Learning Analytics: Applications in Educational Research*. 1st ed. John Wiley & Sons, Inc.; 2016.

[19] Gorunescu F. *Data Mining: Concepts, Models and Techniques*. Springer; 2011.

[20] Hartanti NT. Metode elbow dan *k*-means guna mengukur kesiapan siswa SMK dalam ujian nasional [Elbow and *k*-means methods to measure the readiness of vocational school students for national exams]. *National Journal of Technology and Information Systems*. 2020; 6(2): 82-89. Available from: https://doi.org/10.25077/TEKNOSI.v6i2.2020.82-89.

[21] Merliana NPE, Ernawati, Santoso AJ. Analisa penentuan jumlah cluster terbaik pada metode *k*-means [Analysis of determining the best number of clusters using the *k*-means method]. In: *Proceedings of the National Multidisciplinary Seminar & Call for Papers Unisbank*. Indonesia: Stikubank University; 2005. p.978-979.

[22] Tan PN, Steinbach M, Karpatne A, Kumar V. *Introduction to Data Mining*. 2nd ed. Pearson; 2019.

[23] Taylor JS, Cristianini N. *Kernel Methods for Pattern Analysis*. Illustrated ed. Cambridge University Press; 2004.

[24] Girolami M. Mercer kernel based clustering in feature space. *IEEE Transactions on Neural Networks*. 2002; 13(3): 780-784. Available from: https://doi.org/10.1109/TNN.2002.1000150.

[25] Hofmann M, Klinkenberg R. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. 1st ed. CRC Press; 2014.

[26] Al-Majhad HG, Bramantoro A, Syamsuddin I, Yunianta A, Basori AH, Prabuwono AS, et al. A traffic congestion framework for smart Riyadh city based on IoT services. *International Journal of Advanced Computer Science and Applications*. 2018; 9(4): 292-303. Available from: http://dx.doi.org/10.14569/IJACSA.2018.090443.