ENHANCED KNOWLEDGE IN SCIENCES AND TECHNOLOGY
e-ISSN: 2773-6385

EKST

Vol. 5 No. 2 (2025) 199-209
https://publisher.uthm.edu.my/periodicals/index.php/ekst

Machine Learning Made Visual: An Educational Tool of Enhancing Machine Learning Understanding with Visualisation

Loo Pei Yin[1], Lee Siaw Chong[1]*

[1] *Department of Mathematics and Statistics, Faculty of Applied Sciences and Technology, UTHM Kampus Cawangan Pagoh, Hab Pendidikan Tinggi Pagoh, KM 1, Jalan Panchor, 84600 Pagoh, Muar, Johor, MALAYSIA.*

*Corresponding Author: sclee@uthm.edu.my
DOI: https://doi.org/10.30880/ekst.2025.05.02.023

Article Info

Received: 30 December 2024
Accepted: 17 January 2025
Available online: 19 December 2025

Keywords

Machine Learning (ML), Algorithms, *k*-Nearest Neighbours (*k*NN), *k*-Means Clustering, Visualisation

Abstract

Machine learning (ML) has transformed data-driven industries, but its complexity often makes learning challenging, especially without user-friendly tools to demonstrate how ML algorithms work. Nevertheless, many existing educational resources fail to provide interactive and visual representations of ML concepts, making it difficult for users to grasp algorithmic processes and parameter impacts. To address this gap, this study introduces Machine Learning Made Visual (MLMV), an educational web tool designed to simplify and enhance understanding of ML algorithms through interactive visualisations. It focuses on two key algorithms: *k*-Nearest Neighbours (*k*NN) and *k*-Means Clustering; due to their simplicity and interpretability. With MLMV, users can interactively adjust algorithm parameters, such as *k* values in *k*NN and centroid initialisation in *k*-Means Clustering and observe their effects in real-time visual animations. The tool supports datasets with up to two features to ensure clarity in a 2-dimensional (2D) visual space, utilising technologies like Flask, NumPy, and Matplotlib for backend processing and visualisation. The results demonstrate that MLMV effectively enhances users' comprehension of *k*NN and *k*-Means Clustering by providing clear, step-by-step visual representations of the algorithms in action. Users can explore the influence of parameter changes on algorithm performance in an intuitive and engaging manner. Ultimately, MLMV proves to be a valuable educational resource, offering a quick, responsive, and accessible platform for interactive learning without requiring significant computational resources. It fosters deeper conceptual understanding and facilitates hands-on experimentation with core ML algorithms.

1. Introduction

Machine learning (ML) is a branch of artificial intelligence (AI) that focuses on teaching computers to learn and make decisions like humans [1]. It uses data and algorithms to identify patterns, learn from past experiences, and make predictions or analyse information. ML is generally divided into two main types: supervised and unsupervised learning [2]. Supervised learning involves training a model with labelled data, where each data point is paired with a specific target or output. This allows the algorithm to learn the relationship between inputs and outputs, making it useful to make predictions. Unsupervised learning, on the other hand, works with unlabelled data [3]. Here, the algorithm finds hidden patterns or groupings in the data without being given specific target

This is an open access article under the CC BY-NC-SA 4.0 license.

labels. While ML has become an essential tool in many industries, its algorithms can be challenging to understand, even for professionals. This is because the methods often rely on complex ideas from areas like probability, linear algebra, and optimisation. Although there are tools available to help visualise ML processes, many of them fail to provide clear, step-by-step demonstrations of how the algorithms work. This makes it harder for users to fully grasp the underlying processes behind these powerful techniques. To address this challenge, an interactive educational tool called Machine Learning Made Visual (MLMV) is designed to animate and simplify the ML algorithm processes. MLMV is a web-based interface that seeks to enhance conceptual comprehension and encourage experimentation by concentrating on two core algorithms which are *k*-Nearest Neighbours (*k*NN) and *k*-Means Clustering.

*k*NN and *k*-Means Clustering were selected for their simplicity, interpretability, and ability to effectively illustrate core ML concepts. *k*NN demonstrates the classification process by visualising decision boundaries that adapt to changes in *k* values, making it an excellent choice for teaching with minimal computational requirements. Similarly, *k*-Means Clustering excels at showcasing how data is grouped based on feature similarity. A key aspect of *k*-Means Clustering is the initialisation process, where the selection of initial centroids can significantly impact the clustering results. Users can observe how different initialisations affect the final clusters, providing valuable insights into this important step of the algorithm. By restricting datasets to two features for clear 2-dimensional (2D) visualisation, both algorithms offer intuitive, accessible representations that enhance understanding, making them ideal for an educational tool. It's important to note that neither algorithm is inherently designed to handle missing values and if any are present, it may be unable to execute the algorithms.

MLMV aims to allow users to input data and explore two essential ML algorithms: *k*NN and *k*-Means Clustering. MLMV will enable users to visualise prediction results while providing detailed explanations of each algorithm and its step-by-step process, fostering a deeper understanding of their mechanics. MLMV is designed as a web-based application to make it accessible to users anywhere with an internet connection that allows users to dynamically manipulate key parameters, such as the number of neighbours (*k*) for *k*NN and the initial centroids for *k*-Means Clustering. By observing the real-time effects of these adjustments on results, users can better grasp the algorithms' behaviour and practical applications, making the learning experience both intuitive and engaging.

1.1 Background of the *k*-Nearest Neighbours (*k*NN) Algorithm

The *k*-Nearest Neighbours (*k*NN) algorithm is a basic but effective supervised machine learning (ML) technique used for tasks like regression and classification. The first version of *k*NN algorithm was introduced by Evelyn Fix and Joseph Hodges in 1951 [4]. *k*NN is a foundational method in ML, known for its simplicity and versatility. Initially proposed for statistical applications, it has since gained popularity in a wide range of fields, including image recognition, recommendation systems, medical diagnosis, and text classification. Its ease of implementation and interpretability make it a go-to algorithm for various problem-solving scenarios.

At its core, *k*NN identifies the *k* nearest data points to a testing instance using distance metrics such as Euclidean distance to make predictions. The choice of *k* significantly influences the algorithm's performance; a smaller *k* may result in overfitting, while a larger *k* could oversimplify the decision boundary. By evaluating proximity and similarity between data points, *k*NN mimics human reasoning, making it an intuitive and interpretable method. Visualising *k*NN's decision boundaries enhances user understanding by showing how the algorithm classifies data points and how different *k* values impact the results. This makes *k*NN particularly effective for educational purposes and use cases where simplicity and clarity are essential.

1.2 Background of the *k*-Means Clustering Algorithm

k-Means Clustering is one of the popular unsupervised machine learning (ML) algorithms for grouping data into clusters based on feature similarity. The concept of *k*-Means Clustering was first introduced in 1957 by Stuart Lloyd [5]. The *k*-Means Clustering algorithm processes an unlabelled dataset by dividing it into *k* clusters and iteratively refining the groupings until the optimal clusters are achieved, ensuring data points within each cluster are as similar as possible. The process begins with the initialisation of *k* cluster centroids. Each data point is assigned to the nearest centroid, forming *k* clusters. The centroids are then recalculated as the mean position of all points in their respective clusters. This assignment and recalculation cycle repeats until the centroids stabilise, resulting in well-defined clusters.

This algorithm is particularly effective for datasets with clear and well-separated clusters, offering a simple yet powerful approach to analyse unlabelled data. Due to its efficiency and scalability, *k*-Means Clustering is widely applied in various fields, including customer segmentation, image compression, and anomaly detection. By focusing on similarity and proximity, it provides a robust framework for exploring and interpreting complex datasets. The results of *k*-Means Clustering can be visualised using scatter plots or similar graphical methods, making it easier to understand and interpret the clusters formed. Its efficiency, interpretability, and versatility make it a widely valued tool for clustering tasks.

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 199-209

2. Research Method

Machine Learning Made Visual (MLMV) is an educational tool designed to simplify and enhance understanding of two fundamental ML algorithms: *k*-Nearest Neighbours (*k*NN) and *k*-Means Clustering. It leverages several Python libraries like Flask for backend operations, NumPy for data processing [6], and Matplotlib for visualisations [7]. The frontend is built using Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) for structure and styling, with JavaScript adding interactivity and responsiveness. MLMV enables users to visualise, animate, and interact with ML algorithms, offering a dynamic and intuitive platform to explore their processes and improve comprehension through interactive visualisations.

2.1 The *k*-Nearest Neighbours (*k*NN) Algorithm

The *k*-Nearest Neighbours (*k*NN) is a straightforward and powerful lazy learning algorithm applied to both classification and regression tasks [8]. It operates on the principle that data points that are similar to each other are likely to exhibit similar characteristics. Consequently, the algorithm makes predictions by analysing the *k* closest training examples within the feature space [9]. Unlike many other algorithms, *k*NN does not involve an explicit training phase where a model is built. Instead, it stores the entire training dataset as a reference. It predicts outcomes by measuring the distance between the test data point and every example in the training set. Fig. 1 shows an example of *k*NN in classifying cats and dogs. In this case, with $k = 5$, the unknown data point is classified as a dog based on the majority vote of its five nearest neighbours.

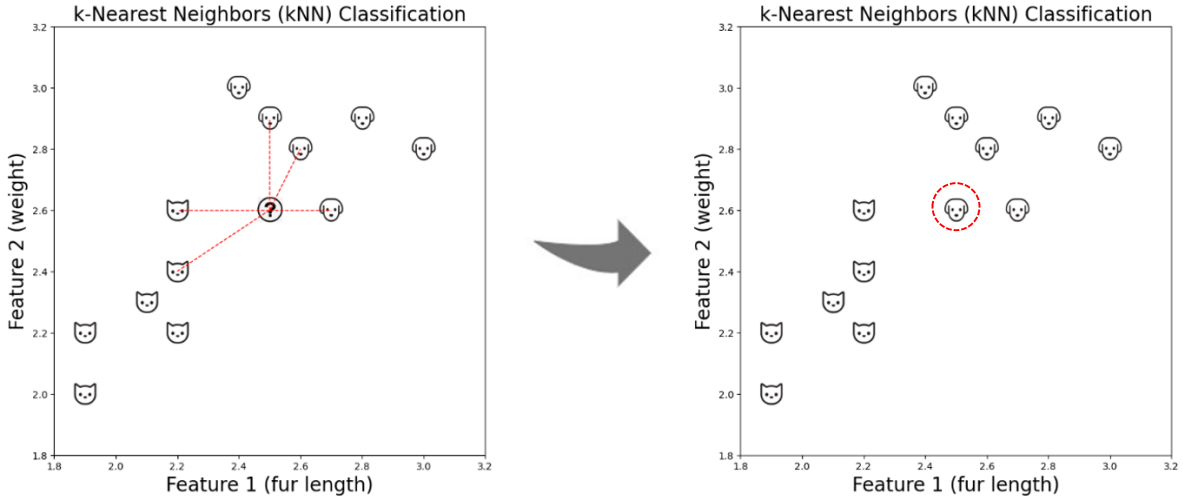


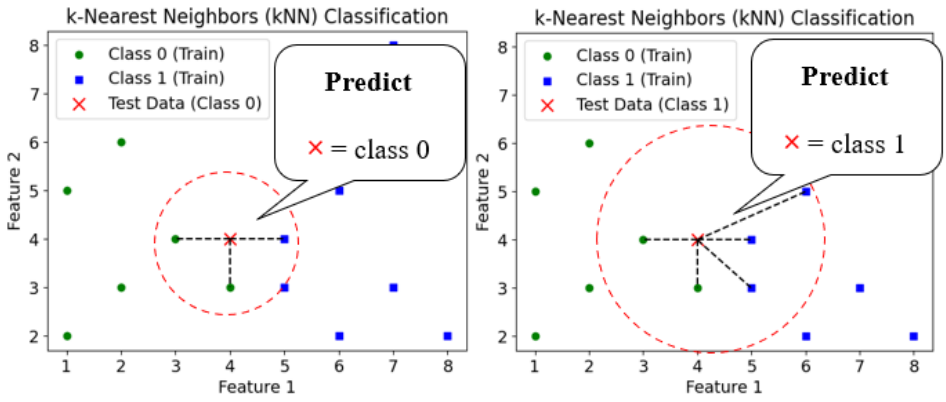
Fig. 1 Example of kNN: Classification of Cats and Dogs

The algorithm operates by first calculating the distance between the test point and all data points in the training set. Common distance metrics include Euclidean distance, Manhattan distance, or Minkowski distance, depending on the nature of the data. Once the distances are computed, the algorithm identifies the *k* nearest data points where *k* is a user-defined parameter indicating the number of closest points to consider. The value of *k* significantly affects the performance of the algorithm, with a smaller *k* being sensitive to noise and a larger *k* potentially overlooking local patterns.

For the *k*NN algorithm to perform effectively in classification tasks, the optimal value of *k* is crucial for balancing model complexity and accuracy.

$$k_{\text{optimal}} = \sqrt{\text{number of data points}} \tag{1}$$

A commonly used heuristic for determining the optimal *k* is given by formula (1) which aims to find a value of *k* that provides good generalisation while avoiding overfitting or underfitting.



(a) (b)

Fig. 2 kNN with distinct k. (a) k = 3; (b) k = 5

For example, Fig. 2 shows how different values of k in kNN affect the classification. When k = 3, the test point has three nearest neighbours (two from Class 0 and one from Class 1). The majority vote classifies it as Class 0. When k = 5, the test point has five nearest neighbours (three from Class 1 and two from Class 0). The majority vote classifies it as Class 1.

In kNN, the selection of a distance metric defines how the "closeness" between data points is calculated. The most common distance metric used in kNN is the Euclidean distance [10]. It measures the straight-line distance between two points in a Euclidean space and is defined mathematically as (2).

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

Here, x and y are two data points in an n-dimensional space, and x_i and y_i represent their coordinates in the i-th dimension. The formula calculates the squared differences between corresponding features, sums them up, and then takes the square root of the result. This makes Euclidean distance an intuitive measure of similarity, reflecting the geometric distance between points. It is widely used because it matches our understanding of physical distances in 2D, or 3D space. Euclidean distance works well with continuous features and is computationally efficient. It is especially useful for small datasets in Machine Learning Made Visual (MLMV), where its simplicity and efficiency are beneficial without overcomplicating the model.

Certainly, visualisation techniques can clarify how the choice of k affects the model's performance. They also assist in identifying patterns, such as the changes in decision boundaries with different values of k, which can guide the selection of the optimal value for the task.

2.2 The k-Means Clustering Algorithm

k-Means Clustering is a commonly used technique for dividing a dataset into k unique and non-overlapping clusters [11]. As a centroid-based partitioning method, each data point is assigned to the nearest centroid, which represents the centre of its respective cluster. The centroids are iteratively updated to minimise the overall distance between data points and their cluster centres, ensuring an optimal grouping of the data. The primary goal of k-Means Clustering is to group similar data points together, making it easier to uncover patterns and relationships within the dataset. Fig. 3 shows the example of k-Means Clustering which displays the final result, where the data is divided into four clusters, each represented by a unique colour and marker shape.

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 199-209

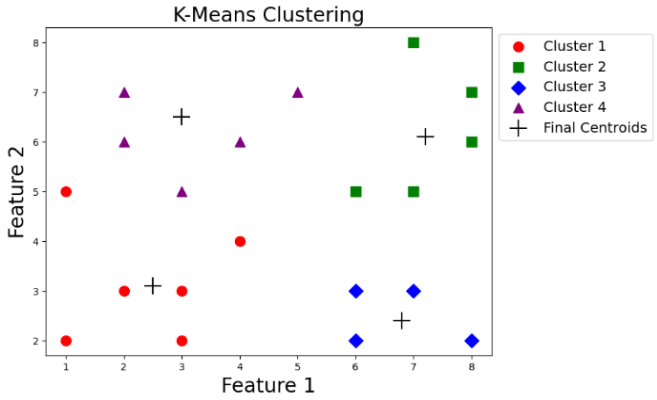


Fig. 3 Example of k-Means Clustering

Initially, k points are randomly selected as the initial centroids for the clusters [12]. Each data point in the dataset is then assigned to the closest centroid, forming k clusters. The distance between each data point and its corresponding centroid is usually computed using the Euclidean distance metric, as shown in formula (2). After all data points have been assigned to clusters, the centroids are recalculated and updated.

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i}^{n} x \tag{3}$$

The new centroid for each cluster is calculated as the average of all the data points within that cluster. Mathematically, this is expressed by (3), where μ_i represents the centroid of cluster C_i, $|C_i|$ is the number of points in cluster C_i, and x denotes a data point within C_i. The assignment and update steps are repeated iteratively until convergence is achieved. Convergence is typically defined when the centroids no longer change significantly, or when a predetermined maximum number of iterations is reached.

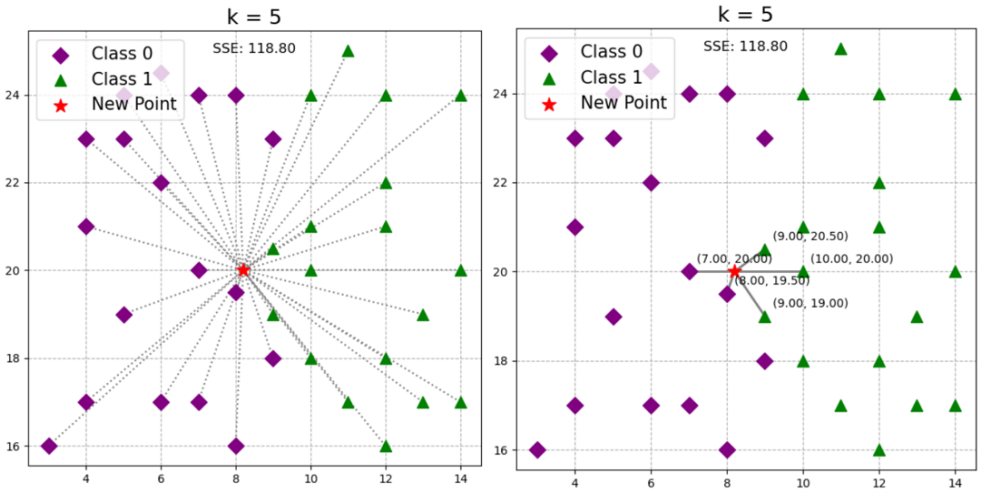
Visualisations can absolutely aid in determining the optimal number of clusters by allowing learners to explore different values of k and evaluate their impact on the clustering results.

3. Result and Discussion

The Machine Learning Made Visual (MLMV) tool offering an interactive platform to visualise two machine learning (ML) algorithms: k-Nearest Neighbours (kNN) and k-Means Clustering. The web-based tool utilises Python's Flask framework for backend functionality, while Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS), and JavaScript are used to create a responsive and engaging user interface. The visualisations, powered by Matplotlib and NumPy, effectively demonstrate the algorithms' iterative processes and decision-making in real time.

3.1 The k-Nearest Neighbours (kNN) Animation

The k-Nearest Neighbours (kNN) algorithm demonstrates how a new data point is classified by evaluating its proximity to existing data points in a 2-dimensional (2D) space. The animation illustrates the step-by-step process of distance calculation, neighbour selection, and final classification, offering an intuitive insight into the algorithm's decision-making process. Fig.5 illustrates the process of the kNN algorithm. This animation step-by-step showcases how the kNN algorithm classifies a new data point based on the proximity and class of its nearest neighbours.

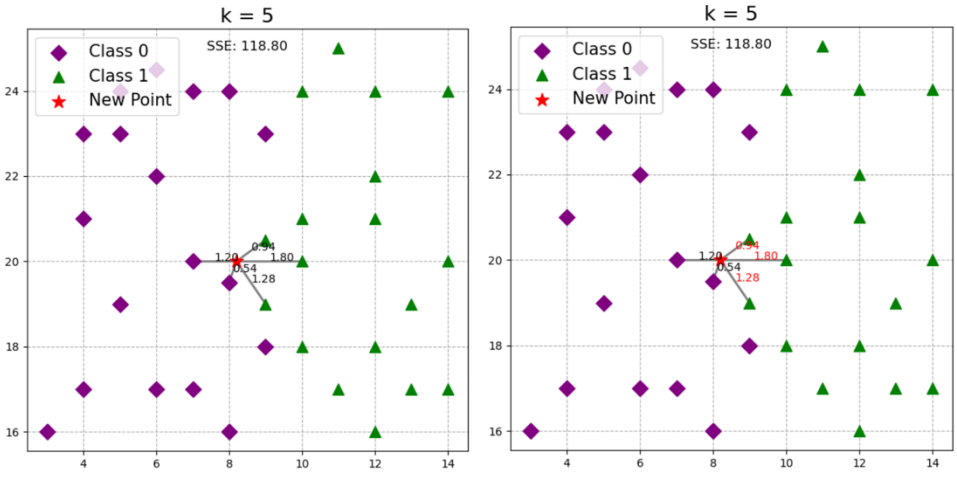


(a) (b)

Fig. 4 kNN Process: (a)Calculate the Euclidean distance between the test data point and all existing data points in the dataset.; (b) Identify the k-nearest data points to the test data point

The animation begins by calculating the Euclidean distance between the test data point and all other points in the dataset, as shown in Fig. 4(a). Each of these distances is visually represented by dotted lines, with the length of each dotted line indicating the proximity of the corresponding point to the test data point. This visual helps to intuitively show the spatial relationships between the test point and all other data points in the dataset.

Next, Fig. 4(b) highlights the identification of the *k*-nearest data points to the test point. To do this, the distances calculated in Step 1 are sorted in ascending order. The top *k* smallest distances correspond to the *k*-nearest neighbours. These are the points that are closest to the test data point, and they will be used to determine the predicted class for the test point. In the visualisation, these *k*-nearest points are marked with solid lines, which connect each of the nearest neighbours to the test point. These solid lines replace the dotted lines from Step 1, indicating that these are the most relevant data points for the classification task. This step helps the viewer visually focus on the nearest neighbours, making it clear which points are influencing the classification decision.



(a) (b)

Fig. 5 kNN Process: (a) Display the distances of the kNN to the test data point.; (b) Highlight the majority class among the kNN

In Fig.5 (a), the distances between the test data point and each of the *k*-nearest neighbours are displayed. Each of the *k*-nearest neighbours is labelled with its respective distance to the test point, providing a numerical indication of proximity. The distance values are positioned along the solid lines and displayed next to the points. This step visually clarifies the relative closeness of the *k*-nearest neighbours to the test point, giving a precise measurement of how far apart the test point is from each neighbour. This helps the viewer understand how the *k*-nearest neighbours were selected, based on their spatial closeness. The *k*-nearest neighbours are identified in

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 199-209

Fig.5 (b), the classification algorithm evaluates the class labels of these *k*-nearest neighbours. For each of the *k*-nearest neighbours, the algorithm checks their class label (e.g., Class 0 or Class 1). It then counts how many data points belong to each class among the *k*-nearest neighbours. The majority class is determined by identifying the class that has the most occurrences among the neighbours.

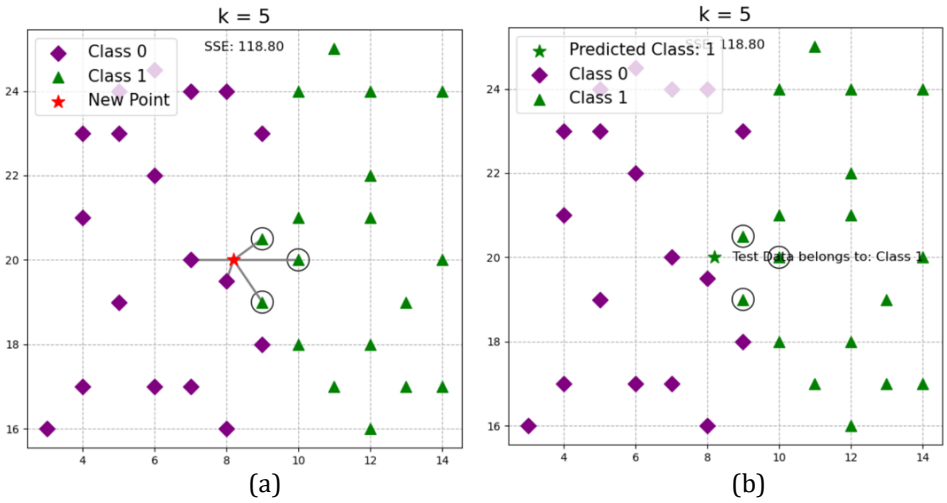


Fig. 6 kNN Process: (a) Draw a circle around the data points of the majority class.; (b) Assign the test data point to the predicted class and update its colour to match the majority class

After determining the majority class, the data points that belong to this majority class are highlighted by circling them as shown in Fig. 6(a). The circled points are the ones contributing to the majority vote for classifying the test data point. In the animation, these points are surrounded by a circle, ensuring that they stand out. This visual cue helps the viewer quickly identify which data points are influential in determining the test point's classification. By marking these points, the process of class voting becomes more transparent. Finally, the test data point is assigned to the predicted class based on the majority class determined in Step 4. The test point's colour is updated to match the colour of the majority class, signalling the completion of the classification process.

In Fig. 6(b), the test point's colour is updated, and the assigned class is displayed as a label next to the test point. This provides a clear and concise visual summary of the *k*NN algorithm's decision-making. The viewer can easily see that the test data point is classified into the class that the majority of its *k*-nearest neighbours belong to. This step solidifies the test point's final classification, providing the viewer with a complete understanding of the process.

3.1.1 Effect of Varying *k* Values in *k*-Nearest Neighbours (*k*NN)

Fig. 7 illustrates the impact of varying *k* values in the *k*-Nearest Neighbours (*k*NN) classification process, highlighting how the choice of *k* influences the classification outcome of a test point. The interactive input fields associated with the system allow users to adjust parameters such as the data points, class labels, test point coordinates, and *k* values, enabling an in-depth exploration of classification behaviour.

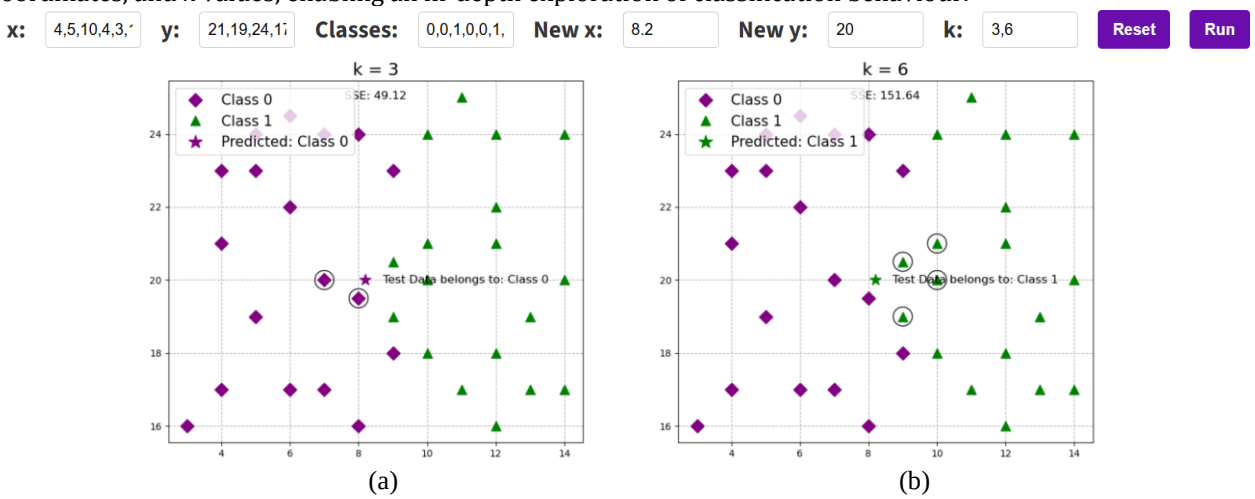


Fig. 7 kNN with varying k values. (a) k = 3; (b) k = 5

In Fig. 7(a) where $k = 3$, the test point (represented by a star) is classified as Class 0 (purple diamonds) because two of its three nearest neighbours belong to Class 0, with only one neighbour from Class 1 (green triangles). This demonstrates how smaller k values focus on local data points, making the algorithm more sensitive to nearby variations but also more susceptible to noise or outliers. Conversely, in Fig. 7(b), with $k = 6$, the test point is classified as Class 1 (green triangles) since four of its six nearest neighbours belong to Class 1, while only two belong to Class 0. Larger k values smooth out local noise by considering a broader set of neighbours, resulting in a more generalised classification, though they may overlook finer local details. This comparison highlights the trade-off between noise sensitivity and generalisation when selecting k, emphasising the importance of choosing an appropriate k value based on the dataset's characteristics and classification objectives.

3.2 The *k*-Means Clustering Animation

The *k*-Means Clustering algorithm showcases how data points are grouped into clusters based on their proximity to centroids in a 2-dimensional (2D) space. The animation visually demonstrates the iterative process of centroid initialisation, cluster assignment, and centroid adjustment, providing a clear understanding of how the algorithm converges.

Fig. 8 illustrates the initial stage of the *k*-Means Clustering algorithm to highlights how it assigns data points to clusters and updates the centroid. This dynamic visualisation helps in comprehending the algorithm's approach to organising data into distinct groups. In this step, data points are displayed along with centroids that have been predefined by the user. The algorithm assigns each data point to the nearest centroid based on the Euclidean distance, forming initial clusters.

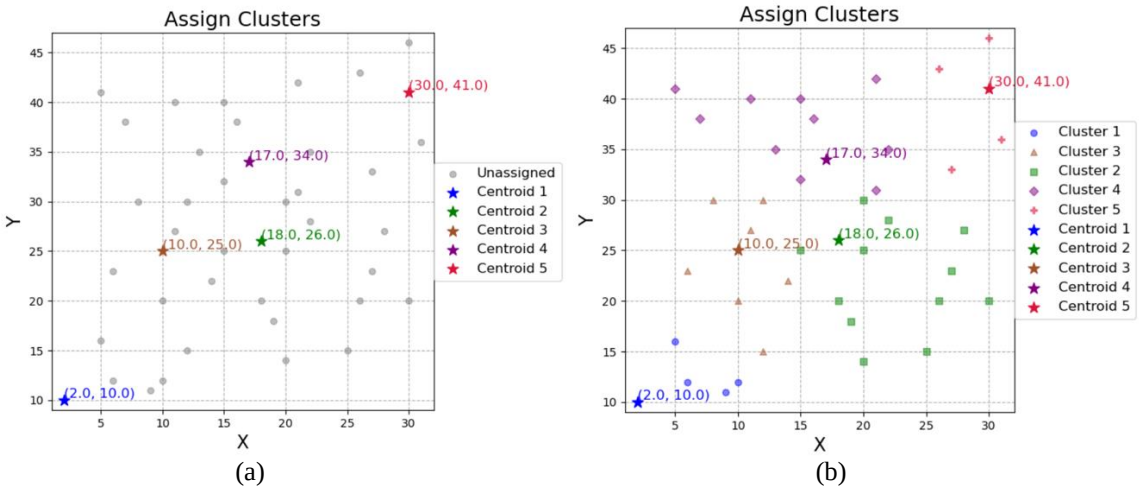
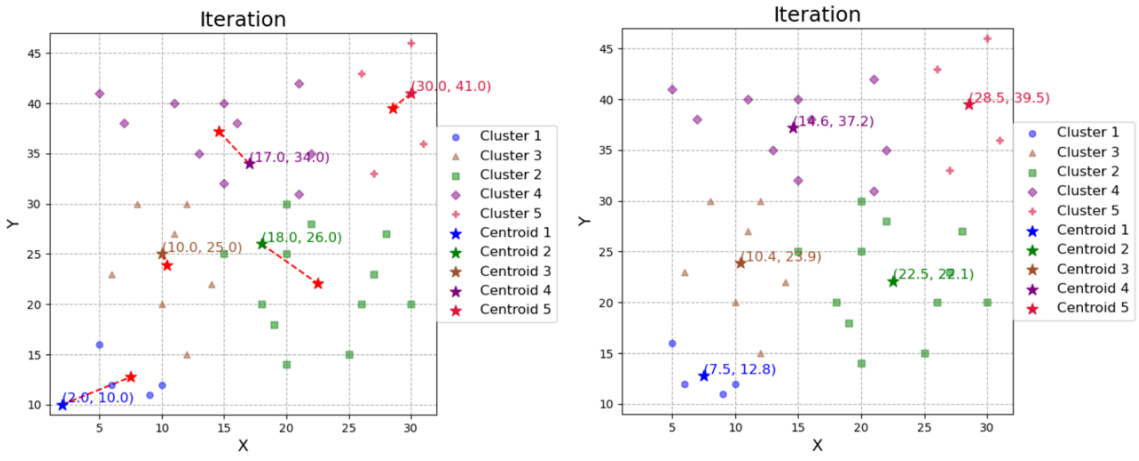


Fig. 8 Initial Cluster Assignments for k-Means Clustering. (a) Unassigned data points with the initial centroids.; (b)Data points assigned to their respective clusters

In Fig. 8(a), the data points are represented as grey dots scattered across the plot, indicating that no clustering has yet been applied. The user-defined centroids, represented by distinct coloured markers (stars), are positioned on the graph to serve as the initial reference points for clustering. At this stage, no data points are associated with any centroid, and they remain uncoloured. In Fig. 8(b), the clustering algorithm begins by calculating the Euclidean distance between each data point and all centroids. Each data point is then assigned to the nearest centroid, forming distinct clusters. This stage effectively demonstrates how the algorithm organises the data into clusters based on proximity, highlighting the centroids' role in defining the cluster boundaries.

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 199-209



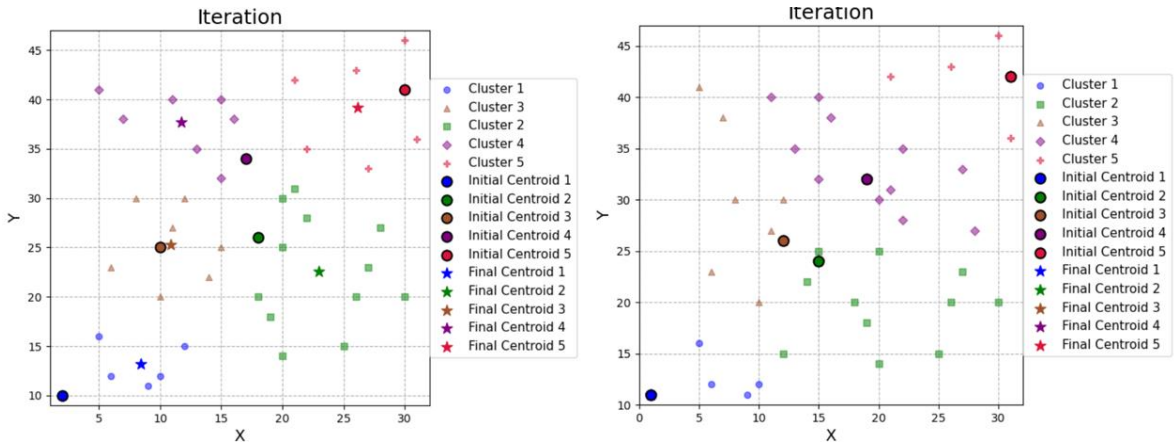
(a) (b)

Fig. 9 Centroid Update Process for k-Means Clustering. (a) Red stars mark the new positions.; (b) Centroids at their updated positions

Fig. 9(a) shows the process of updating the centroids during an iteration of the clustering algorithm. The coloured stars represent the old positions of the centroids before this iteration, while the red stars indicate the newly calculated positions of the centroids. The red lines demonstrate the movement of each centroid from its old position to its new position, which is determined by calculating the mean of the data points assigned to each cluster. This step is a key part of the *k*-Means Clustering algorithm as it refines the centroid locations to better represent their clusters. Fig. 9(b) illustrates the outcome of the update process. The red stars from Fig. 9(a) have now become the centroids' current positions, replacing the old, coloured centroids. Data points remain assigned to their nearest centroids based on these new positions. This figure highlights how the centroids adjust their locations during each iteration to improve the clustering structure, ensuring they better align with the data distribution. This iterative process of recalculating centroids and reassigning data points continues until convergence is reached, meaning the centroids no longer move significantly or the cluster assignments remain stable. At this point, the clustering solution is considered final, representing the optimal grouping of the data based on the *k*-Means Clustering algorithm.

3.2.1 Effect of Distinct Initial Centroids on *k*-Means Clustering Outcomes

Fig. 10 highlights the sensitivity of *k*-Means Clustering to the placement of initial centroids where the circle markers indicate their associated initial centroids. This dependence can lead to different outcomes, as evident from the differing group formations. Although the initial centroids are in relatively similar positions, the final clustering results can differ significantly. For instance, the brown cluster (triangles) demonstrates a notable difference in its final grouping between the two scatter plots. The figure underscores the importance of proper centroid initialisation strategies, such as *k*-Means++, to achieve more consistent and stable clustering results.



(a) (b)

Fig. 10 k-Means Clustering with Distinct Initial Centroids. (a) Initial centroids = [(2, 10), (18, 26), (10, 25), (17, 34), (30, 41)].; (b) Initial centroids = [(1, 11), (15, 24), (12, 26), (19, 32), (31, 42)]

4. Conclusion

In conclusion, the Machine Learning Made Visual (MLMV) tool serves as an effective bridge between theory and practice in machine learning (ML), focusing on *k*-Nearest Neighbours (*k*NN) and *k*-Means Clustering. These algorithms were selected for their simplicity, interpretability, and broad applicability, making them ideal for educational purposes. As a web-based platform, MLMV ensures accessibility by allowing users to access it from any device with a browser, eliminating the need for software installation or high-end hardware.

MLMV offers several impactful features to enhance the user experience. It allows users to input custom datasets and parameters, such as *k* values in *k*NN and the number of clusters in *k*-Means Clustering, to observe how these settings affect results. The tool provides real-time visualisations of processes like distance calculations, cluster assignments, and centroid refinement. Interactive controls, such as pausing, replaying, and stepping through animations, further enrich the learning experience by enabling a step-by-step exploration of the algorithms' inner workings.

*k*NN demonstrates the principles of supervised learning, including distance metrics and classification, while *k*-Means Clustering introduces unsupervised learning concepts like similarity and iterative refinement. Unlike complex algorithms such as neural networks, both *k*NN and *k*-Means Clustering are computationally efficient and easier to visualise, making them ideal for real-time, interactive learning.

MLMV effectively demonstrates how variations in parameters, such as k values in kNN and initial centroid placements in k-Means Clustering, influence the resulting outcomes. This underscores the importance of careful parameter selection and proper initialisation strategies. While the tool is best suited for small datasets due to computational limitations, it remains an engaging and accessible resource for understanding ML concepts.

Looking ahead, MLMV could be expanded to include additional interpretable algorithms, such as Decision Trees (DT), Support Vector Machines (SVMs), and Logistic Regression. These algorithms are not only widely used but also inherently explainable, making them excellent candidates for interactive visualisation.

Acknowledgement

The authors would like to thank the Faculty of Applied Sciences and Technology, Universiti Tun Hussein Onn Malaysia, for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Loo Pei Yin, Lee Siaw Chong;* ***data collection:*** *Loo Pei Yin;* ***analysis and interpretation of results:*** *Loo Pei Yin, Lee Siaw Chong;* ***draft manuscript preparation:*** *Loo Pei Yin, Lee Siaw Chong. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] S. V. Mahadevkar, B. Khemani, S. Patil, K. Kotecha, D. R. Vora and A. Abraham, "A Review on Machine Learning Styles in Computer Vision—Techniques and Future Directions," IEEE Access, vol. 10, pp. 107293-107329, 2022, 10.1109/ACCESS.2022.3209825.

[2] B. Wang and W. Zhang, "Research on Edge Network Topology Optimization Based on Machine Learning," 2023 5th International Conference on Applied Machine Learning (ICAML), pp. 41-46, 2023, 10.1109/ICAML60083.2023.00018.

[3] T. Chauhan, S. Rawat, S. Malik and P. Singh, "Supervised and Unsupervised Machine Learning Based Review on Diabetes Care," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), vol. 1, pp. 581-585, 2021, 10.1109/ICACCS51430.2021.9442021.

[4] E. Fix and J. Hodges, "An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation," International Statistical Review, vol. 3, no. 57, pp. 233-238, 1951, https://doi.org/10.2307/1403796.

[5] S. Lloyd, "Least Squares Quantization in PCM," IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 129-137, 1982, 10.1109/TIT.1982.1056489

[6] A. N. Ziogas, T. Ben-Nun, T. Schneider, and T. Hoefler, "NPBench: a Benchmarking Suite for High-Performance NumPy," Proceedings of the ACM International Conference on Supercomputing, pp. 63-74, 2021, 10.1145/3447818.3460360.

[7] A. Hafeez and A. H. Sial, "Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python," International Journal of Advanced Trends in Computer Science and Engineering, vol. 10, no. 1, pp. 277-281, 2021, 10.30534/ijatcse/2021/391012021.

[8] X. Mao, G. Zhao and R. Sun, "Naive Bayesian Algorithm Classification Model with Local Attribute Weighted Based on KNN," 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), pp. 904-908, 2017, 10.1109/ITNEC.2017.8284867.
[9] J. Xu, K. H. Moon and M. van der Schaar, "A Machine Learning Approach for Tracking and Predicting Student Performance in Degree Programs," IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 5, pp. 742-753, 2017, 10.1109/jstsp.2017.2692560.
[10] G. Putro Dirgantoro, M. A. Soeleman, and C. Supriyanto, "Smoothing Weight Distance to Solve Euclidean Distance Measurement Problems in K-Nearest Neighbor Algorithm," 2021 IEEE 5th International Conference on Information Technology, p. 294–298, 2021, 10.1109/ICITISEE53823.2021.9655820.
[11] X. Ran, X. Zhou, M. Lei, W. Tepsan, and W. Deng, "A Novel K-Means Clustering Algorithm with a Noise Algorithm for Capturing Urban Hotspots," Applied Sciences, vol. 11, no. 23, pp. 11202-11202, 2021, 10.3390/app112311202.
[12] M. Mardi and M. R. Keyvanpour, "GBKM: a New Genetic Based K-Means Clustering Algorithm," 2021 7th International Conference on Web Research (ICWR), pp. 222-226, 2021, 10.1109/ICWR51868.2021.9443113.