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

EKST

Vol. 5 No. 2 (2025) 221-229
https://publisher.uthm.edu.my/periodicals/index.php/ekst

Performance Evaluation of Machine Learning Algorithms on Letter Recognition Task

Yeoh Lay Qi[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, 86400 Pagoh, Muar, Johor, MALAYSIA.*

*Corresponding Author: sclee@uthm.edu.my
DOI: https://doi.org/10.30880/ekst.2025.05.02.025

Article Info

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

Keywords

Letter Recognition, *k*-Nearest Neighbours , Random Forest, EMNIST Dataset, Performance Evaluation

Abstract

Machine Learning (ML) algorithms have become integral in addressing various technological challenges, including image recognition and text classification. A critical application is handwritten letter recognition, which demands efficient algorithms to handle diverse handwriting styles and complex data. This study applies and evaluates two supervised learning algorithms, *k*-Nearest Neighbours (*k*NN) and Random Forest, on the Extended Modified National Institute of Standards and Technology (EMNIST) Letters dataset, comprising 124,800 training samples and 20,800 testing samples of uppercase and lowercase letters. Our goal is to determine the optimal hyperparameters for both algorithms, namely k for *k*NN, and the number of trees and tree depth for Random Forest. For *k*NN, the optimal number of neighbours is $k = 5$, achieving an accuracy of 85.33%. For Random Forest, the optimal hyperparameters include a tree depth of 14, a minimum sample split of 2, and 29 trees, yielding an accuracy of 81.84%. Our results show that by selecting the optimal hyperparameters for both algorithms, *k*NN outperforms Random Forest in terms of accuracy.

1. Introduction

Machine Learning (ML), a subfield of Artificial Intelligence, focuses on developing algorithms and models that enable computer systems to learn from data and make predictions or decisions without explicit programming [1]. It encompasses various tasks, including classification, regression, clustering, and recommendation. ML approaches are broadly categorized into Supervised, Unsupervised, Semi-Supervised, and Reinforcement Learning, each suited for various applications. Among these, classification, as an aspect of supervised learning, aims to categorize data into predefined labels. This project specifically applies ML to handwritten letter recognition, which involves identifying and classifying letters from diverse sources such as handwritten documents, printed texts, and digital images.

Handwritten letter recognition poses unique challenges due to the variability in human writing styles, letter shapes, and inconsistencies in scanned images. While deep learning algorithms like Convolutional Neural Networks (CNNs) excel in such tasks, traditional machine learning approaches like *k*-Nearest Neighbours (*k*NN) and Random Forest remain relevant for their simplicity, computational efficiency, and interpretability. These methods can deliver strong performance with appropriate parameter tuning, making them ideal for applications with limited resources or training data.

This project employs the Extended Modified National Institute of Standards and Technology (EMNIST) database, a more comprehensive and challenging alternative to the traditional Modified National Institute of Standards and Technology (MNIST) dataset. The EMNIST dataset contains grayscale images of handwritten

characters, each represented as a 28 × 28 pixel matrix with intensity values ranging from 0 (black) to 255 (white) [8].

While the EMNIST dataset includes both handwritten digits and letters, this project focuses exclusively on the balanced set of uppercase and lowercase letters categorized into 26 alphabetical groups (A to Z). It comprises 145,600 samples, with 124,800 used for training and 20,800 for testing [9]. This extensive and diverse dataset ensures a robust foundation for evaluating the performance of ML algorithms in handwritten letter recognition.

For preprocessing, the images in the EMNIST dataset were initially represented as grayscale values ranging from 0 to 255. The pixel values were binarised to improve model efficiency and performance, with all values above 0 set to 1 and all values equal to 0 remaining 0. This transformation reduced the data's complexity and allowed the models to focus on the presence or absence of features rather than the intensity of pixels, which simplified the recognition task. Fig. 1 shows some examples of the preprocessed letters dataset.

Fig. 1 Example of dataset

Overall, this project aims to develop and optimize *k*NN and Random Forest algorithms for recognizing handwritten characters using the EMNIST dataset. This research focuses on tuning hyperparameters for the *k*NN and Random Forest to achieve a balance between accuracy and computational efficiency. Increasing hyperparameter values, such as the number of neighbours in *k*NN or tree depth, minimum sample split and number of trees in a Random Forest may result in small changes in accuracy while significantly increasing computational time. Therefore, while the model accuracy remains a critical performance metric, computational efficiency is equally important.

For the *k*NN algorithm, parameter *k* represents the number of nearest neighbours considered when predicting a data point. This project evaluates different *k* values ranging from 1 to 10. Each *k* value was tested by computing the model's accuracy, and the results were plotted to generate an elbow curve. Three hyperparameters were optimised for the Random Forest algorithm to improve the model's performance: tree depth, minimum sample split, and number of trees. Tree depth determines the maximum number of levels a decision tree can have. It controls how the model performs, balancing complexity and risk of overfitting. Minimum sample split specifies the minimum number of samples required to split a node. A lower value allows the model to capture finer details in the data but increases the risk of overfitting. The number of trees defines the amount of decision trees in the Random Forest algorithm. More trees generally improve accuracy but also increase computational cost.

To achieve these objectives, several libraries in Python are used to implement and evaluate the performance of the Machine Learning (ML) algorithms on the EMNIST dataset. These libraries include NumPy, Matplotlib, Time and EMNIST.

2. Research Method

The research method integrates the strengths of *k*-Nearest Neighbours (*k*NN) and Random Forest to address the challenges of handwritten letter recognition. *k*NN provides simplicity and intuitive classification by comparing a query point to its nearest neighbours in the training dataset. However, its computational cost increases significantly with large or high-dimensional datasets, which reduces its efficiency. Random Forest mitigates these limitations by constructing an ensemble of decision trees that aggregate their outcomes to improve accuracy and robustness. This method handles large datasets effectively and is less sensitive to parameter variations. However, these advantages come with higher computational demands, requiring more processing power and longer

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 221-229

evaluation times. The trade-offs between *k*NN's simplicity and Random Forest's robustness make them complementary approaches for this project.

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

The *k*-Nearest Neighbours (*k*NN) algorithm is a foundational yet powerful method for classification and regression tasks in ML. Proposed by Fix and Hodges in the 1950s, it categorizes objects based on their proximity to others in the dataset [2]. Its popularity grew in the 1960s and 1970s as computational power increased, and Cover and Hart explored its theoretical properties in 1967 and demonstrated its error rate to be bounded by twice the Bayes error rate under ideal conditions [3].

*k*NN is a classification method based on similarity. It assumes that nearby data points in the feature space are likely to share the same class. Consequently, to predict the output for a new data point, the *k*NN algorithm collects the closest *k*NN neighbours from the training data and uses the label of the most similar neighbours to predict the class of the new data. The *k*NN algorithm begins with an existing set of example data with its corresponding class labels, known as the training dataset. When a new data point without a label is received and the parameter k that refers to the number of nearest neighbours is determined, the algorithm compares this new data point to every piece of data in the training dataset using chosen distance metrics.

Several distance metrics can be used in this algorithm, including Euclidean distance, Manhattan distance, and Minkowski distance. In this project, Euclidean distance, d is used to calculate the distance between two points, $\vec{x}$and $\vec{y}$ with n features. The Euclidean distance can be mathematically represented as:

$$\boldsymbol{d} = \sqrt{\sum_{i=1}^{n} (\boldsymbol{x_i} - \boldsymbol{y_i})^2}$$

where x_i and y_i represent the coordinates of the points $\vec{x}$ and $\vec{y}$ in the ith dimension.

The distances are then sorted from smallest to largest, and the algorithm takes a majority vote on the class labels of the *k*NN. The class with the most votes is assigned to the new data point. For instance, if $k = 3$ and the three nearest neighbours include two from Class B and one from Class A, the algorithm classifies the point as Class B. Fig. 2 illustrates these concepts.

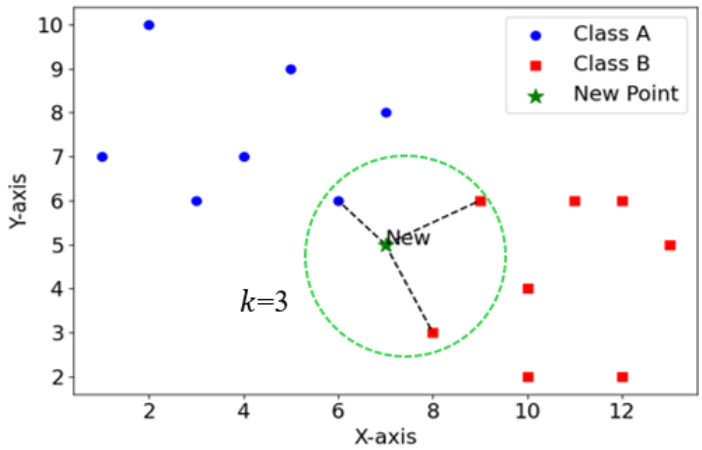

Fig. 2 Concepts of kNN when k=3

The value of k is a positive integer, typically under 20, which decides how many nearest neighbours are considered for the majority vote. Next, the value of k is recommended to be an odd number because a tie may happen if k is an even number. The k value significantly influences the algorithm's performance and behaviour. This is because the algorithm predicts the class of the new data point based on the number of votes. Hence, the number of closest neighbours will affect the number of votes. Fig. 3 shows an example of how the k value influences the classification of the class on a new data point. Based on the figure, the *k*NN algorithm would classify the new point as Class A because among the 7 nearest points to the new point consist of 4 blue points and 3 red points. This classification is different from Fig. 2, which classifies the new point as Class B.

The choice of k significantly affects the algorithm's performance. A small k makes the model sensitive to noise and outliers, leading to overfitting, while a large k reduces sensitivity but may lead to underfitting. In this project, k was determined by evaluating values in the range of 1 to 10. accuracy was calculated for each value of k, and the accuracy was plotted to form an elbow curve.

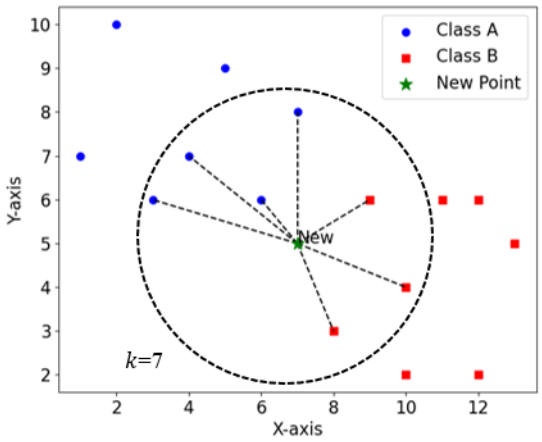

Fig. 3 Example of the value of k=7

2.2 Random Forest

Random Forest, introduced by Leo Breiman and Adele Cutler in the early 2000s, is an ensemble learning technique that builds multiple decision trees to improve prediction accuracy and robustness [4]. Over the years, its capabilities have been enhanced to address challenges like high-dimensional data, imbalanced datasets, and missing values. Variants such as Extremely Randomized Trees and Rotation Forest [5] have been developed to extend their flexibility.

This algorithm has applications in diverse fields, including genetics, ecology, and medicine. For instance, Random Forest has been used to infer demographic histories from genetic polymorphisms, classify frailty in seniors using plantar pressure measurements [6], and predict mother-to-child HIV transmission based on medical history [7]. It remains a widely used algorithm due to its robustness, simplicity, and versatility across data types and problem domains. Ongoing research continues to improve its efficiency, interpretability, and applicability to emerging ML challenges.

Random Forest improves predictive accuracy by combining the results of multiple decision trees. A decision tree divides a dataset into subsets using decision nodes (splitting criteria) and leaf nodes (outcomes). Starting from a root node representing the entire dataset, the tree branches out based on features until a stopping criterion is met or the subsets are sufficiently homogeneous. The splitting is guided by metrics like Gini Impurity, Mean Squared Error (MSE), or Information Gain (IG). IG measures the reduction in entropy, which evaluates the impurity or uncertainty in class labels. A feature with the highest IG is selected as the root node, ensuring an optimal split. However, decision trees can overfit the training data and favour features with more split points, neglecting other significant features. A sample structure of the Decision Tree algorithm is shown in Fig. 4 [10].

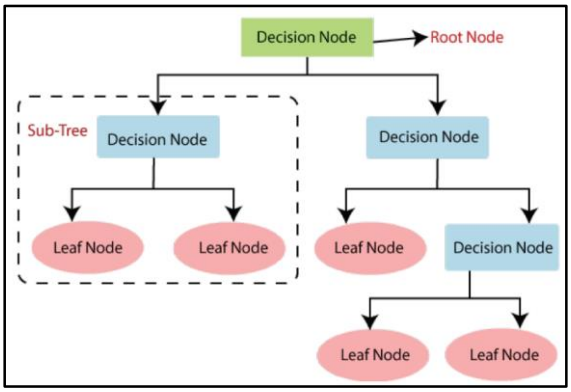

Fig. 4 A Sample structure of the Decision Tree

The core idea behind decision trees is choosing the feature that best divides the dataset into different classes or predicts the target variable. This is achieved using various metrics. The measurement used in this project is IG, which measures the change in entropy after the dataset is split on an attribute. Entropy is defined as a metric used to evaluate the impurity in the given dataset or the variance of the dataset. The term "impurity" in a dataset indicates the level of mixing or lack of clarity in the class labels of the data points. A dataset with low impurity has separated the class labels more clearly, indicating most of the data points fall within the same class. On the other hand, a highly impure dataset has a lot of mixing between the class labels, meaning there is a lot of uncertainty

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 221-229

regarding the classification of a randomly selected data point. The training dataset consists of n classes, and the entropy is given by [10]

$$\boldsymbol{Entropy} = -\sum_{i=1}^{n} \boldsymbol{p}(\boldsymbol{x_i})\boldsymbol{log_2 p}(\boldsymbol{x_i}),$$

where $p(x_i)$ is the probability of randomly choosing an element from the class, x_i.

After calculating entropy, IG is identified by subtracting the weighted entropies of every branch from the initial entropy, with the feature yielding the highest IG selected for splitting. In addition, IG is computed for every feature in a dataset with various features. The feature with the highest IG will be the most important feature, and it will be the root node for the decision tree. However, the decision tree tends to overfit the training data when the trees are grown to their maximum depth. Besides that, it also tends to prioritise features that have more potential split points, possibly leading them to neglect features with fewer split points that could be significant.

Random Forest overcomes these limitations by introducing randomness and aggregation. It begins by generating multiple bootstrap samples from the original training dataset, where each sample is created by randomly selecting data points with replacements. This ensures that each sample differs in composition, even though they have the same size as the original dataset. Decision trees are then built on these samples using random subsets of features for splitting nodes. This random selection of features introduces diversity among the trees, prevents overfitting and ensures varied tree structures.

Once the forest of decision trees is constructed, predictions for new unlabelled data are made by aggregating results from all trees using majority voting, where the class with the highest number of votes becomes the final prediction. This ensemble approach balances bias and variance, creating a model that is robust, accurate, and less prone to overfitting than a single decision tree. Fig. 5 illustrates a Random Forest algorithm sample. By leveraging the strengths of multiple trees, Random Forest ensures that its predictions are reliable and effective, even for complex datasets.

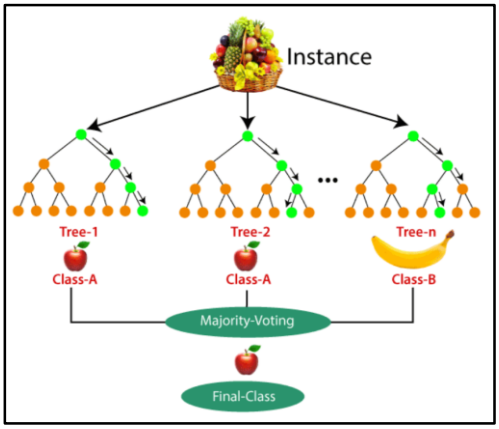

Fig. 5 Example of idea of Random Forest [11]

3. Results and Discussions

Optimizing the parameters of machine learning models is crucial for achieving high performance. For *k*-Nearest Neighbours (*k*NN), determining the ideal *k* value is essential for accurate predictions. Similarly, fine-tuning the hyperparameters of the Random Forest algorithm, such as tree depth, minimum samples required to split a node, and the number of trees in the forest, ensures robust and reliable outcomes. This study uses the Extended Modified National Institute of Standards and Technology (EMNIST) Letters dataset, consisting of 124,800 training samples and 20,800 testing samples of handwritten uppercase and lowercase letters, to evaluate the performance of these algorithms. The performance can be visualized by calculating the accuracy for each parameter value and plotting the elbow curve to observe the trend.

3.1 Optimal *k* for *k*-Nearest Neighbours (*k*NN) Algorithm

To address this, a range of smaller *k* values (1 to 10) were considered, and the Elbow method was employed to identify the optimal *k*. The Elbow method involves plotting the accuracy of the *k*-Nearest Neighbours (*k*NN) model against different values of *k* and observing the point where the increase in accuracy starts to taper off, resembling an "elbow" shape. Fig. 6 illustrates the results of this analysis. The accuracy of the *k*NN model increased

significantly from $k = 1$ to $k = 5$, reaching a peak at $k = 5$ with an accuracy of 0.8533. Beyond $k = 5$, the accuracy plateaued, showing only marginal improvements or slight declines. Thus, $k = 5$ was selected as the optimal value.

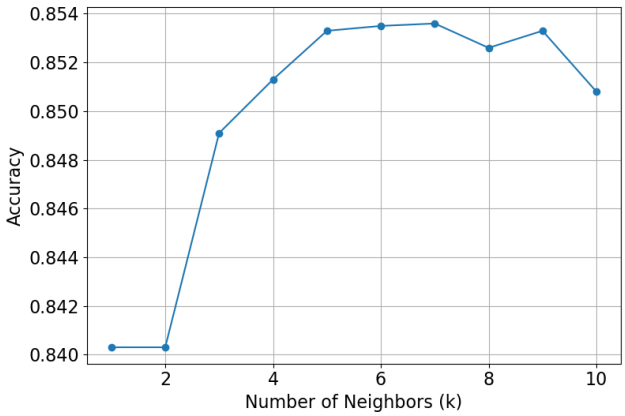

Fig. 6 Elbow curve for optimal k in kNN

3.2 Optimal Parameters for Random Forest

The optimal hyperparameters for Random Forest, including tree depth, minimum samples split, and the number of trees, were determined through a multi-step process. Initially, individual decision trees were employed to analyse and establish baseline values for tree depth and minimum samples split by plotting an elbow curve using accuracy against parameter values. Similarly, the elbow method was applied to the Random Forest algorithm to fine-tune the number of trees. This approach provided a straightforward and visual way of selecting well-suited hyperparameters for individual trees and the ensemble model.

3.2.1 Tree Depth

Fig. 7, which depicts the analysis of tree depth, revealed that accuracy steadily increased with deeper trees, rising from just above 0.0737 at a depth of 1 to approximately 0.7257 at a depth of 20. However, the improvement in accuracy began to plateau after a depth of 14. Hence, a tree depth of 14 is selected as the optimal value, as depths greater than 14 provided minimal accuracy gains but took significantly longer to compute, increasing the model's complexity and training time without a corresponding performance improvement.

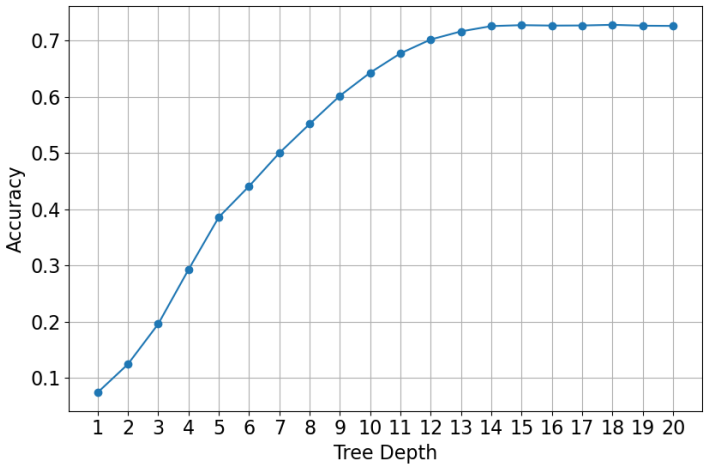

Fig. 7 Elbow curve for optimal depth in Random Forest

3.2.2 Minimum Samples Split

Fig. 8 represents the elbow graph of the minimum sample split. The decision tree approach revealed that the highest accuracy of 0.7258 occurred at a minimum sample split of 6. However, increasing the minimum sample split further led to longer computation times without significant improvements in accuracy. Therefore, a minimum sample split of 2, with an accuracy of 0.7254, was selected as the optimal value, providing a good balance of performance and computational efficiency.

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 221-229

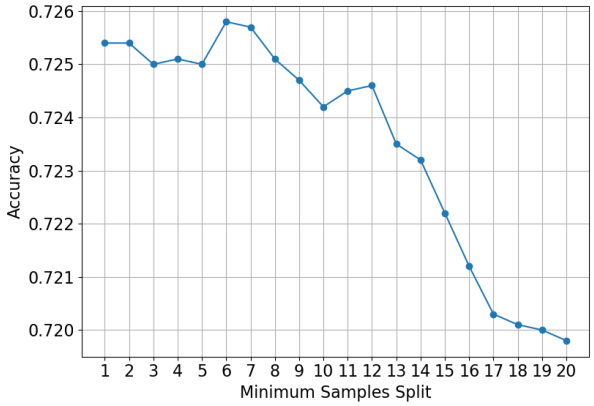

Fig. 8 Elbow curve for optimal minimum samples split in Random Forest

3.2.3 Number of Trees

The number of trees in the Random Forest was tuned after determining the optimal tree depth and minimum sample split, which are 14 and 2, respectively. The elbow method was used to identify the point at which additional trees no longer significantly improved the model's accuracy. The number of trees in the Random Forest plays a critical role in balancing the trade-off between model performance and computational efficiency.

The results of this analysis are visualized in Fig. 9. The accuracy of the Random Forest model shows a sharp increase as the number of trees increases from 1 to approximately 12. After this point, the rate of improvement slows, with the curve gradually flattening. By the time the number of trees reaches 29, the accuracy has nearly stabilized, and further increases in the number of trees yield only marginal gains in accuracy.

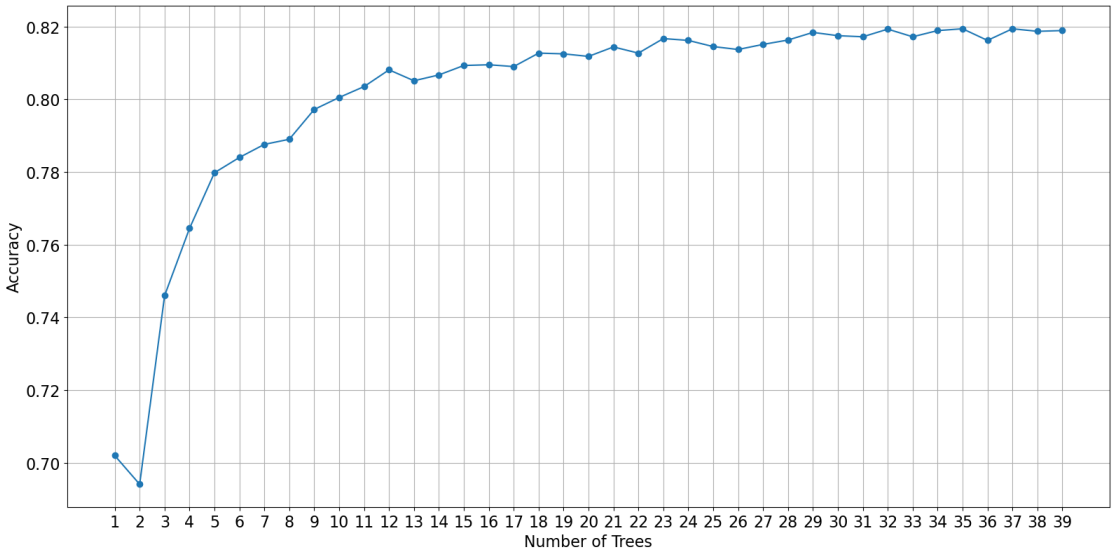

Fig. 9 Elbow curve for optimal number of trees in Random Forest

In terms of computational efficiency, approximately 3 hours are needed in computing with 29 trees, while increasing the count to 39 trees took over 4 hours. Despite the slight accuracy improvements observed with more trees, the rise in computation time made them impractical. Thus, the optimal number of trees was selected to be 29 for optimal performance and efficiency. This point ensures the model achieves high performance without unnecessary overhead from including more trees than required.

3.3 Analysis of Misclassified Samples

Upon evaluating the prediction results, several misclassified samples were observed for both *k*-Nearest Neighbours *(*kNN) and Random Forest. For *k*NN, misclassification often occurred when the nearest neighbours included ambiguous or poorly written letters, leading to incorrect predictions. Similarly, in Random Forest, most misclassifications arose in cases where the features of the handwritten letters were too similar, causing overlapping decision boundaries. Fig. 10 and Fig. 11 showcase examples of misclassified letters for both

algorithms. These samples highlight the inherent challenges in distinguishing between similar letters, especially in cases where the writing style introduces ambiguity.

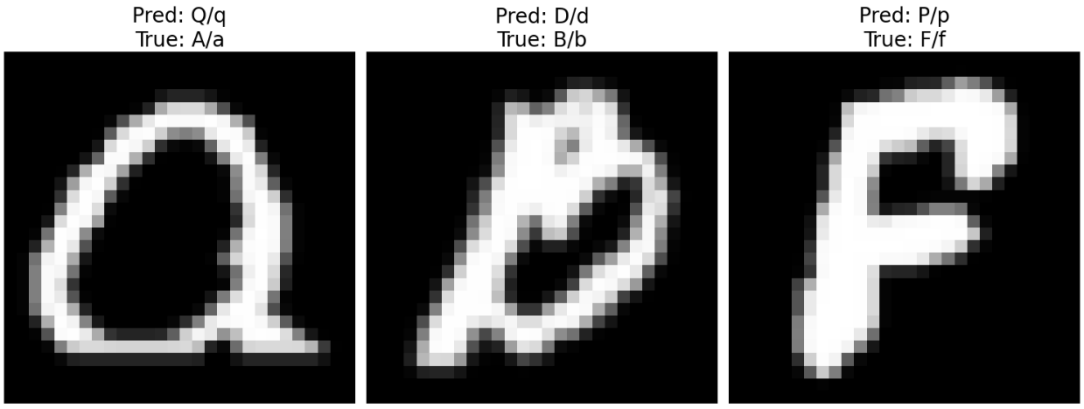

Fig. 10 Misclassified letter for kNN algorithm

Fig. 10 demonstrates misclassifications made by the *k*-Nearest Neighbors (*k*NN) algorithm on the EMNIST dataset. In the first sample, kNN predicted the character as "Q/q" when the true label was "A/a." This misclassification is likely due to the similarity in the looped structures of the two letters, which caused kNN to match with nearby neighbours with similar shapes but different details. In the second case, kNN predicted "D/d" while the actual label was "B/b." Both letters share a round structure, and kNN likely struggled to differentiate between their subtle differences. Lastly, the model predicted "P/p" instead of "F/f." This could be because both characters have vertical strokes, and kNN may have focused more on their shared structural features rather than distinguishing between the loop in "P" and the open structure of "F".

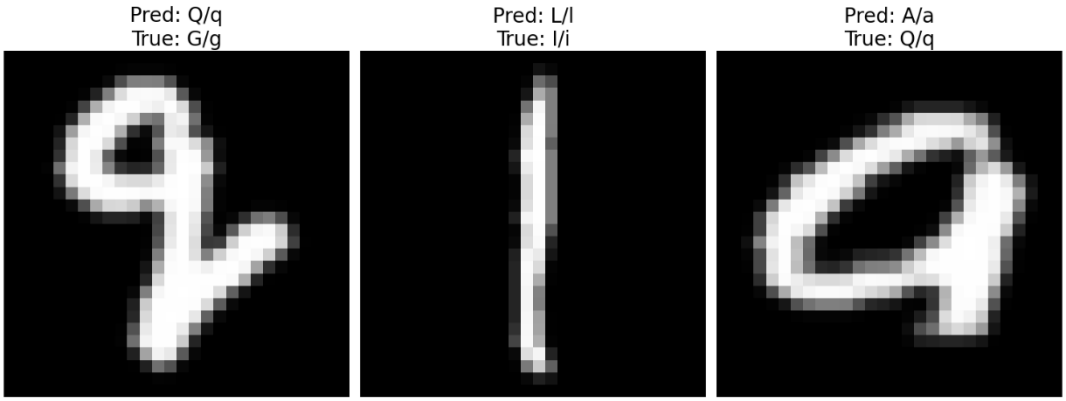

Fig. 11 Misclassified letters for Random Forest algorithm

Fig. 11 highlights incorrect predictions made by the Random Forest model. The model predicted "Q/q" in the first example when the true label was "G/g." The confusion here is likely due to the similar curve patterns at the top of both characters, with the model giving undue importance to less distinctive features. In the second sample, the prediction was "L/l," while the true label was "I/i." Both letters are simple vertical lines, and the Random Forest model appears to have struggled to distinguish between their slight differences, such as thickness or stroke placement. In the third instance, the model predicted "A/a" when the actual label was "Q/q." Similar to the kNN error, the loop in "A/a" may have been mistaken for the loop in "Q/q," with the model failing to weigh the distinguishing features correctly.

4. Conclusion

This study utilized the Extended Modified National Institute of Standards and Technology (EMNIST) database, which includes a balanced set of uppercase and lowercase letters with 145,600 samples. The dataset is divided into 124,800 training samples and 20,800 test samples. The training dataset was used to train the machine learning models, while the test dataset was kept aside for evaluating the models' performance on unseen data. The dataset's diversity and structure provide a strong foundation for evaluating model performance in handwritten character recognition.

The results demonstrate the importance of tuning k in the k-Nearest Neighbours (kNN) algorithm to match the dataset's characteristics. The elbow curve reveals that the accuracy rapidly increased from $k = 1$ to $k = 5$, and waned afterwards. Therefore, $k = 5$ was chosen as the optimal value with an accuracy of 85.33%.

For the Random Forest algorithm, the optimal hyperparameters were determined as tree depth = 14, minimum sample split = 2, and the number of trees = 29, resulting in an accuracy of 0.8184. Training using these parameter settings took approximately 3 hours, but if the number of trees is increased to 39, the accuracy rises by only 0.05% at the cost of an additional hour of computation. Hence, we conclude that the number of trees = 29 is optimal, considering the trade-off between accuracy and computational efficiency.

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:*** *Yeoh Lay Qi, Lee Siaw Chong;* ***data collection:*** *Yeoh Lay Qi, Lee Siaw Chong;* ***analysis and interpretation of results:*** *Yeoh Lay Qi, Lee Siaw Chong;* ***draft manuscript preparation:*** *Yeoh Lay Qi, Lee Siaw Chong. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] GeeksforGeeks, "Types of Machine Learning," GeeksforGeeks, n.d. [Online]. Available: https://www.geeksforgeeks.org/types-of-machine-learning/?ref=lbp. [Accessed 17 May 2024].

[2] P. J. Grother, "National Institute of Standard and Technology (NIST)," Headquarters, n.d. [Online]. Available: https://www.nist.gov/srd/nist-special-database-19. [Accessed 11 May 2024].

[3] G. Cohen, S. Afshar, J. Tapson and A. Van Schaik, EMNIST: Extending MNIST to Handwritten Letters, Anchorage, AK, USA: IEEE, 2017, pp. 2921-2926. Available: 10.1109/IJCNN.2017.7966217

[4] E. Fix and J. L. Hodges, "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties," *International Statistical Review / Revue Internationale de Statistique,* vol. 57, no. 3, p. 238, 1989. Available: https://doi.org/10.2307/1403797

[5] K. Rojek, R. Ochorok and M. Wiencis, "Enhancing Movie Recommenders by Means of KNN-based Algorithms," *System (Linköping),* pp. 49-54, 2022. Available: https://ceur-ws.org/Vol-3360/p07.pdf

[6] M. Schonlau and R. Y. Zou, "The Random Forest Algorithm for Statistical Learning," *Stata Journal,* vol. 20, no. 1, pp. 3-29, 2020. Available: https://doi.org/10.1177/1536867X20909688

[7] J. Xia, P. Ghamisi, N. Yokoya and A. Iwasaki, "Random Forest Ensembles and Extended Multiextinction Profiles for Hyperspectral Image Classification," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 56, no. 1, pp. 202-216, 2018. Available: 10.1109/TGRS.2017.2744662

[8] E. Anzai, D. Ren, L. Cazenille, N. Aubert-Kato, J. Tripette and Y. Ohta, "Random Forest Algorithms to Classify Frailty and Falling History in Seniors using Plantar Pressure Measurement Insoles: A Large-scale Feasibility Study," *BMC Geriatrics,* vol. 22, no. 1, 2022. Available: https://doi.org/10.1186/s12877-022-03425-5

[9] R. Chaula and G. Justo, "A Robust Random Forest Prediction Model for Mother-to-Child HIV Transmission Based on Individual Medical History," *Tanzania Journal of Engineering and Technology,* vol. 41, no. 3, pp. 64-70, 2022. Available: https://doi.org/10.52339/tjet.v41i3.845

[10] M. Hafeez, M. Rashid, H. Tariq, Z. Abideen, S. Alotaibi and M. Sinky, "Performance Improvement of Decision Tree: A Robust Classifier Using Tabu Search Algorithm," *Applied Sciences,* vol. 11, no. 15, p. 6728, 2021. Available: https://doi.org/10.3390/app11156728

[11] E. R. Sruthi, "Understand Random Forest Algorithm With Examples (Updated 2024)," Analytics Vidhya, n.d. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/. [Accessed 19 May 2024].