

Building A Classifier For MNIST With K-Nearest Neighbors And Hyperparameter Tuning

Hoang Vinh Nguyen
Lab Group 17
Msc in AI and Automation
University West
Trollhättan, Sweden
hoang-vinh.nguyen@student.hv.se

Indrachapa Gamlath
Lab Group 17
Msc in AI and Automation
University West
Trollhättan, Sweden
lahiru-indrachapa-gamlath.lokugamlathge@student.hv.se

Sachintha Kapuge
Lab Group 17
Msc in AI and Automation
University West
Trollhättan, Sweden
sachintha-wishwajith.kankanam-kapuge@student.hv.se

Abstract—This electronic document is a report to the Assignment 4 of the course Introduction to Artificial Intelligence and Machine Learning. After hyper tuning and modify the original dataset, our *kNN* model achieved 97.9% accuracy score and its learning curve shows sign of more room for improvement. In this classification task, Random Forest Classification is also a good choice to use.

Keywords—*k nearest neighbors, grid search cv, sgd, random forest regressor, accuracy, precision, recall, f1, roc, learning curves, data augmentation, confusion matrix*

I. INTRODUCTION

In this assignment, we will build a classify model to learn from MNIST dataset and have it detected numbers from images. The goal is to achieve 97% accuracy score on the test set. We will focus on the k-Nearest Neighbors (*kNN*), a supervised learning algorithm that works well for both regression and classification tasks.

The next part in this assignment is to find the best combination of hyperparameters for the *kNN* model that can help achieve the accuracy goal. Comparison between different models will be made at the end of the assignment to judge the capability of the *kNN* model.

MNIST dataset is a collection of 70000 images of digits handwritten by high school students and employees of the US Census Bureau[1]. The dataset is widely used in training and testing image classification.

Throughout this assignment 4, we followed the code examples and the content in the book Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurélien Geron[2].

The final code and figures are included in our repository for assignment 4[3].

II. K-NEAREST NEIGHBORS CLASSIFIER

A. Definition

K-Nearest Neighbors is a supervised, lazy, non-parametric, instance-based learning algorithm. It is among the simplest machine learning algorithms and has been widely used in the field of pattern recognition over the last century[4].

The algorithm works by storing all the labeled training examples during its training process, then only when it is

making predictions, will it start processing the training data and gives output[5].

Another variant of k-Nearest Neighbors algorithm is called Nearest Neighbor (NN) algorithm, where k=1. The *kNN* algorithm will use the majority vote mechanism to make predictions, whereas the NN algorithm will use a single nearest neighbor to the query point.

B. Training the *kNN* model

One of many advantages of *kNN* algorithm is it has relatively few hyperparameters to start with, namely k and the distance metric[6]. We started off training the model without using any hyperparameters and measured its precision, recall and accuracy.

The results for a basic *kNN* model are impressive. Using confusion matrix in Fig. 1, we can notice that the model sometimes confuses class 8 with other classes 1, 3, 5 and most notably between class 4 and 9.

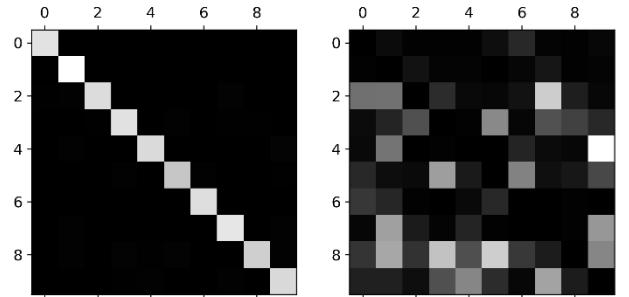


Fig. 1. Confusion matrix shows performance errors of the basic *kNN* model on the train set

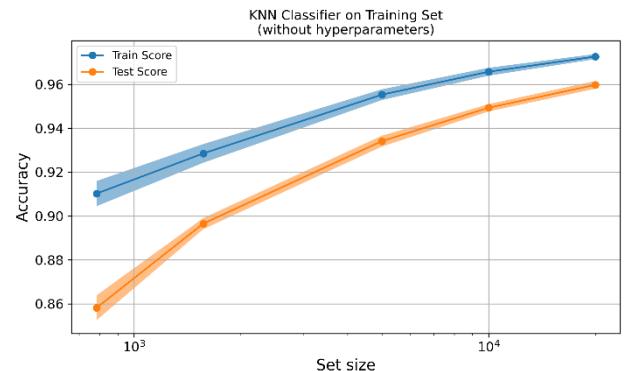


Fig. 2. The basic *kNN* model's learning curve

Fig. 2 shows the accuracy curve of the basic *kNN* model. Both train and test curve are smooth and slowly converge further to the right. The graph shows the model is generalizing well to unseen data.

At this point, we can see the model's accuracy on the test set (using Cross Validation) is barely reached 96% accuracy. We will need to fine tune it for better performance.

C. Hyperparameters Fine-Tune

Using *GridSearchCV* class, we fine-tuned the model to find the best combination of *n_neighbors*=[3,4,5] and *weights*=['distance', 'uniform']. As a result, we found that the value of *n_neighbors*=4 and *weights*='distance' give the better accuracy to the model.

Fig. 3 shows that the model performance on the training set with Cross Validation has reached accuracy of 1. Meanwhile, the performance on the test set of CV has improved better than the previous model. This means that the model has overfitted the training data, but it still has room to improve its generalization capabilities as the test score increases.

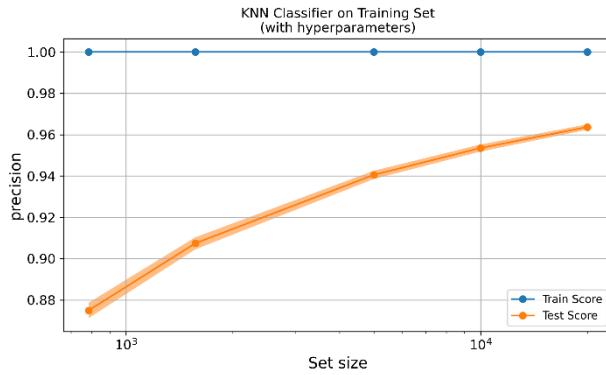


Fig. 3. The learning curve of the *kNN* model after hyper-tuned

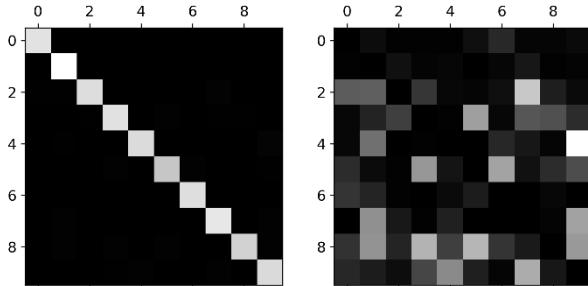


Fig. 4. The confusion matrix of the *kNN* model after hyper-tuned

With the confusion matrix, the model still shows signs of misclassification. At this point, the hyper-tuned *kNN* model scores are shown in Table 1 below.

TABLE I. KNN HYPER-TUNED MODEL SCORES

| | Precision | Recall | F1 | Accuracy |
|-----------|-----------|--------|-------|----------|
| Train Set | 0.972 | 0.972 | 0.972 | 1.0 |
| Test Set | 0.955 | 0.954 | 0.954 | 0.973 |

D. Data Augmentation

To reduce the errors in confusion matrix, we experimented with the method Data Augmentation. By shifting each digit image 1 pixel into different directions top, bottom, left, right, top left, bottom right, we created multilabel dataset of the MNIST.

After training the *kNN* model using the same hyperparameters and with the new augmented dataset, it reached the accuracy score of 0.979.

III. RESULTS AND ANALYSIS

Due to our limited computational power, we chose to train the two SGD and Random Forest Classification models on the original dataset instead of the augmented dataset.

We were still able to fine-tune them and have the result shown on Fig. 5.

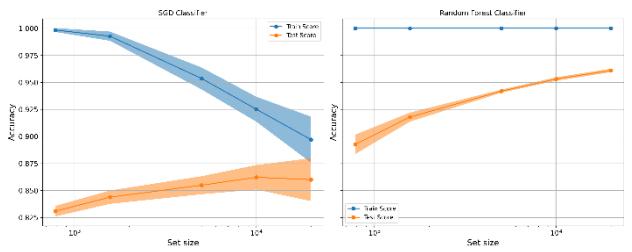


Fig. 5. The learning curve of the *SGD Classifier* and *Random Forest Classifier*

The *SGD Classifier* is a linear model, its score on training set drops drastically the more data feed to the model. At the end of the learning curve, both scores seem to converge but the test score appears to decrease a little. This could be the sign of the model being completely overfitting at first, and after being fed with unseen data, it started to generalize better.

For the *Random Forest Classifier*, the model seems to be overfitting the training data and adding more samples to it only increases its test score slowly. It could be that for the *Random Forest Classification*; we used too many features in the hyperparameter that caused the model to be too complex. Overall, this model also achieves good results as the *kNN* model.

Table II below shows the accuracy scores of each model on both original train set and test set.

TABLE II. EVALUATING RESULT OF EACH MODEL

| | Train Set | Test Set |
|----------------------|-----------|----------|
| kNN | 1.0 | 0.973 |
| SGD | 0.892 | 0.882 |
| Random Forest | 1.0 | 0.968 |

IV. CONCLUSION

The *kNN* algorithm is a simple yet powerful tool to tackle classification problems. In this assignment, we have learned to not only fine tune the model hyperparameters, but also to practice looking at the data and understand what is wrong with it.

To enhance the data quality and enable the kNN model to learn better, we applied the Data Augmentation approach to help increase its score by 0.03, the final value that we achieved is 0.979.

For the MNIST dataset, a linear boundary is not available so that the SGD classification was unable to perform well. Instead, the Random Forest Classification was also a good choice for this classification task.

V. REFERENCES

- [1] US Census Bureau, “MNIST dataset.” Accessed: Oct. 12, 2025. [Online]. Available: https://en.wikipedia.org/wiki/MNIST_database
- [2] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*, 2nd ed. O’Reilly Media, Inc., 2019.
- [3] Hoang Vinh Nguyen, Sachintha Kapuge, and Indrachapa Gamlath, “Github Repository for Assignment 4.” Accessed: Oct. 12, 2025. [Online]. Available: https://github.com/nhv96/iai600_assignment_4
- [4] Sebastian Raschka, “2.1 Introduction,” STAT479: Machine Learning Lecture Notes. Accessed: Oct. 12, 2025. [Online]. Available: https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf
- [5] Sebastian Raschka, “2.2 Key Concepts,” STAT479: Machine Learning Lecture Notes. Accessed: Oct. 12, 2025. [Online]. Available: https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf
- [6] Sebastian Raschka, “2.13 Advantages and Disadvantages of kNN,” STAT479: Machine Learning Lecture Notes. Accessed: Oct. 12, 2025. [Online]. Available: https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf