**KNN and Decision Tree Algorithms for Character Recognition of EMNIST Dataset**

# Introduction

|  |
| --- |
|  |
| *Fig. 1: randomly chosen examples of the dataset and corresponding letter* |

Artificial Intelligence plays a fundamental role in the process of developing software to accurately recognise and convert hand drawn letters and characters into machine readable data. In order to train and test said machine learning algorithms, the EMNIST Dataset, consisting of 26000 images of hand written letters alongside labels that each image represents, enables us to present a range of uniquely drawn and labelled characters to the algorithm, akin to real world examples of handwriting. This research project aims to utilise this dataset to understand the impact of two different distance metrics of the K-Nearest Neighbour algorithm, as well as two of the existing classification algorithms pre-existing in the MATLAB software. Each algorithm and distance metric incorporates a unique function, allowing us to see the optimum conditions needed when collecting, managing and digitalising handwritten collected data.

# Data and Preparation

Initially, the dataset consitited of two columns, the first holding vectors of 784 digits representing the intensity data of each image. The second column contained the first column’s corresponding letter. After converting the first column to doubles, we were able to produce the output displayed in figure 1. Evidently the EMNIST dataset contains a variety of handwriting styles, fonts and cases, mirroring the complexity of a real world data sample reaffirming its practicality in the real world. In order to train and test each algorithm, it was appropriate to split the dataset so that a random selection of 1300 instances would each be used for training and testing.

# Methodology

By pursuing a purely quantitative methodological approach to evaluating the four different algorithms, I was able to see directly how changes in the mathematical functions directly attain to the suitability for our desired output. In the context of the EMNIST dataset, I saw it useful to test the Euclidean and Manhattan distance metrics as they react slightly differently to the dataset in question, with the former being more attentive to special distancing and the latter being orientated around geometric shape. Furthermore, I decided to compare MATLAB’s decision tree function to its KNN algorithm as I thought the utilisation of entropy when handling the many non-linear patterns of the EMNIST may prove more preferable.

# Results

After running the randomly assigned testing and training data through each of the four algorithms a total of three times, I recorded the mean time taken for the training and testing to complete, as well as the mean accuracy of the predicted test labels and recorded the results below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Mean Accuracy (%) | Mean Time (Seconds) |
| K-Nearest Neighbour Distance Metrics | Euclidean Distance | 77.49 | 233.94 |
| Manhattan Distance | 75.05 | 234.17 |
| MATLAB Existing Classification Algorithms | KNN Function | 77.49 | 14.64 |
| Decision Tree Function | 56.27 | 1.28 |

The Manhattan Distance and Euclidean distance recorded comparably similar results, with the Euclidean Distance exceeding Manhattan’s mean accuracy of by 2.44%, whilst being 1.87 seconds faster on average. However, MATLAB’s existing classification algorithms recorded varying results with its KNN algorithm measuring an identical accuracy level to the Euclidean Distance variant of self-implemented K-Nearest Neighbour algorithm, whilst being 219.3 seconds faster. That being said, the accuracy of the decision tree function recorded an even smaller average time of 1.28 seconds but only had an accuracy reading of 56.27%. The confusion matrixes revealed that each algorithm experienced high occurrences of incorrect predictions when differentiating between depictions of the letters ‘*I*’ and ‘*J*’.

# Conclusion

The table reveals the algorithm that produced the highest accuracy reading in account of its low testing and training time was the existing MATLAB K-Nearest Neighbour model. Despite this fact, apprehensions prevail when analysing the context of its test. The testing and training data remained static throughout the investigation, thus leading me to assume it may not produce such an accurate reading consistently. Furthermore, the drawbacks of the KNN model remail relevant, with its limitation of dimensionality and reliance on the measuring of spacing. If the dataset was to expand and incorporate letters written cursively or three-dimensional letters, I believe this algorithm would struggle in measuring the distances between the instances with significantly high variability.