Classification of Letters Using KNN and Decision Tree Models

Author:

Aditya Gandi

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

Table of Contents

Abstract / Executive Summary

Recognising handwritten letters is a foundational task in pattern recognition with applications ranging from document digitisation to assistive technologies for individuals in need of them. To support the development of accurate classification models, we explore the performance of K-Nearest Neighbours (KNN) and Decision Trees on the Letter Recognition Data set.

The dataset contains 20,000 samples, each representing capital letters from the English alphabet (A-Z) described by 16 numerical features derived from pixel-based statistics such as width, height and edge density. The character images were based on 20 different fonts, and each of the letters within the 20 fonts was randomly distorted to produce the 20,000 unique stimuli. We implement a machine learning pipeline which includes data preparation, data exploration, data modelling and parameter tuning to evaluate the models.

Through comparative analysis, we examine each of the models evaluated against their accuracy, precision, recall and F1-scores. Also, identifying feature ambiguity is a common challenge during the classification of letters. Finally, we suggest a classification model based on the aforementioned metrics and applicability.

1. Introduction

Classification is a fundamental model where the objective is to assign input data to predefined categories. In this project, the focus will be on the classification of handwritten English alphabet letters based on numerical features given in the dataset. This task has practical applications in areas such as optical character recognition, postal address interpretation, and assistive technologies for the visually impaired.

The dataset used for this project includes 20,000 instances of capital letters from (A-Z), each represented by 16 numerical features derived from pixel-based statistics such as width, height, and edge count. The target variable is the letter class.

The goal of this project is to model, tune and evaluate two classification models to determine which performs more accurately on predicting the pixel-based data as one of the 26 capital letters. These models were chosen due to their beneficial characteristics; KNN is a learner with strong locality-based performance (Aymane Hachcham, 2025), while Decision Trees are fast learners with intuitive interpretability (Nnajiofor, 2024). Therefore, the research question being addressed is "**Which classification model — K-Nearest Neighbours or Decision Trees — is more effective in accurately classifying letters based on statistical pixel features?**"

2. Methodology

2.1 Retrieving Data and Exploring Dataset

The data was retrieved using the pandas library from a downloaded dataset, which originated from the UCI Machine Learning Repository. Before modelling the data, the data was explored to understand the structure, relationships and issues that are within the dataset. This is essential to ensure that the dataset and investigation are treated fairly and are not biased, for a clear-cut model selection.

Descriptive statistics of the dataset were printed out for each of the columns, providing insights into tendencies (mean and median), the variability (standard deviation), and the range of the values (min and max). These helped to identify features that may be skewed or contain outliers, and if the scale is variable throughout the different columns.

Visualisations such as a histogram were used to identify the distribution of the different columns, indicating any data handling methods if required. Furthermore, to express the relationships between the columns and better understand the behaviour, a correlation matrix was plotted to quantify the proportionality of the variables. Hypotheses were then drawn and further discussed from the relevant visualisations produced, via both the correlation matrix and scatter plot diagrams.

2.2 Data Pre-Processing and Splitting

The dataset was first checked to ensure that there were no missing values or duplicates within the dataset. Duplicates which were found were removed. There was no additional data handling required. For example, handling outliers was not required as numbers which had lower frequencies were not statistical issues, but valid upper-bound observations. The dataset was split into training and testing sets (80:20) using the sklearn train_test_split, producing 14934 train data and 3734 test data. The training set is used to fit the models, while the training data is unseen and is used to evaluate the model. This was recommended from the relevant datasets information. Furthermore, by plotting the class distribution, nothing was found to indicate any significant class imbalances or irregularities which would need to be addressed.

2.3 Data Exploration Columns

Exploring the individual columns through histograms relayed valuable insights, while quantifiable values were taken from the table generated after using df.describe(). The feature **x-box**, which represents the horizontal position of a character's bounding box, showed a left skewed distribution with the values ranging from 0 to 15. Most of the data was clustered around the mean of ~4. Suggesting relatively compact horizontal positions for most letters. The graph below is an example of the histogram of **x-box**, representative of how the other graphs for the features are displayed as well.

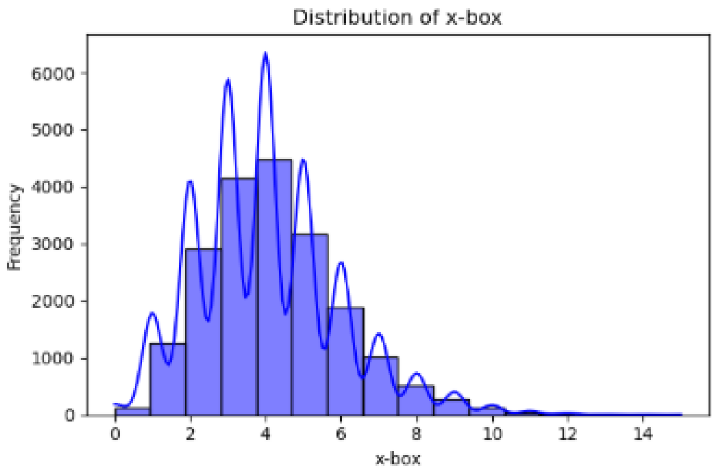


Fig 1 - *x-box histogram*

In contrast, **y-box**, the vertical bounding box position, displayed a larger range and a higher mean of ~7. However, the frequency of larger y-box values drastically dropped after 11, implying skewness. The **width** and **high**, represent the dimensions of each character, both have a mean of ~5 and are left skewed, width being fairly symmetric around the mean. While **high** has a large disparity between the frequencies of 8 and the following numbers, dropping from ~3700 to less than ~300. Indicating skewness once again.

The **onipix** column represents the number of active pixels. The histogram shows peaks between 2 and 4, with the frequency dropping off as the pixel count increases. This suggests that most of the characters are composed of relatively few black pixels, indicating simplicity or sparsity in their structure. The longer tail of the histograms further suggests that there is a smaller group of characters which are far more dense. The **x-bar** and **y-bar are** the horizontal and vertical centroids of pixel intensities, both following a nearly normal distribution with a mean of ~6.8 and ~7.5. This suggests that the average pixel positions are consistently central, as expected from uniformly sized image inputs.

More complex geometric attributes, such as **x2bar** and **y2bar**, which represent the second moments and cross-moment interactions, possess more irregular and skewed distributions. This skewness reflects the diversity of character structures, indicating that only a few characters exhibit asymmetry or curviness. The **x-ege** column, representing horizontal edge counts, was heavily left-skewed, with a higher frequency towards lower values, demonstrating that simpler edge structures are more common within the dataset.

2.4 Exploration of Relationships

Exploring the relationships between the attributes would give more reasoning and insights into the performance of the model. A hypothesis will be stated, and then insights will be discussed based on the visualisation produced. To begin, the plot below (Fig 2) is a correlation matrix, representing the correlation between all the attributes. Ten attribute pairs, which had a mix of different correlation bands (i.e. 0.8-0.6, 0.6-0.4) were selected to be focused on.

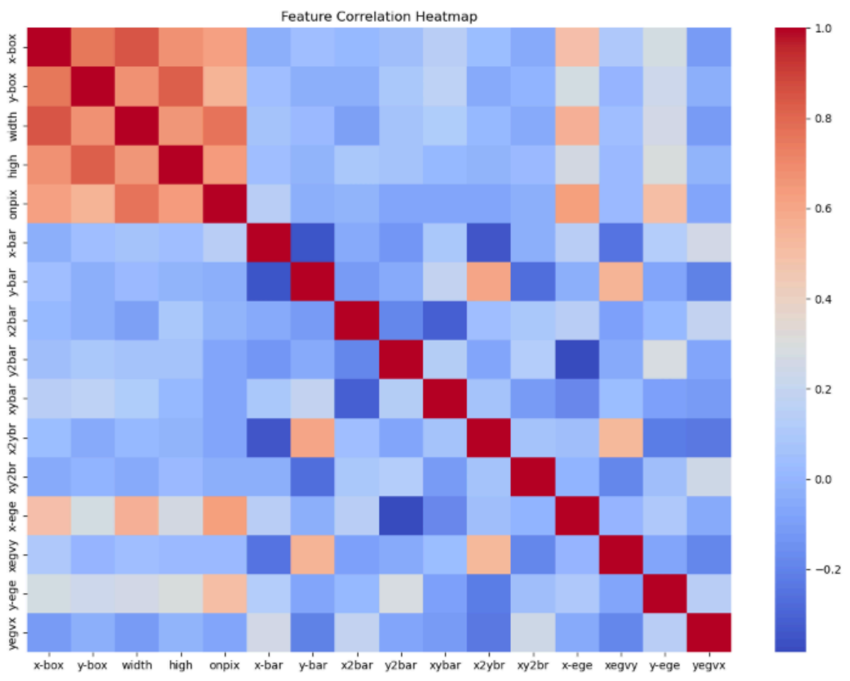


Fig 2 - *Correlation Matrix for all features*

As for the relationships between attributes, **x-box** vs **y-box**. It is hypothesised that wider characters are also likely to be taller, observed through a strong positive linear relationship in the scatter plot, suggesting that large letters expand in both dimensions. Additionally, hypothesised for **x-bar** and **y-bar**, characters with high horizontal centroids would tend to have lower vertical centroids, suggesting a diagonal alignment in structure as observed with most letters. The graph below is a scatter plot along with a trend line, illustrating the relationship of the **x-box** and **y-box**, representative of how the relationships were discerned visually.

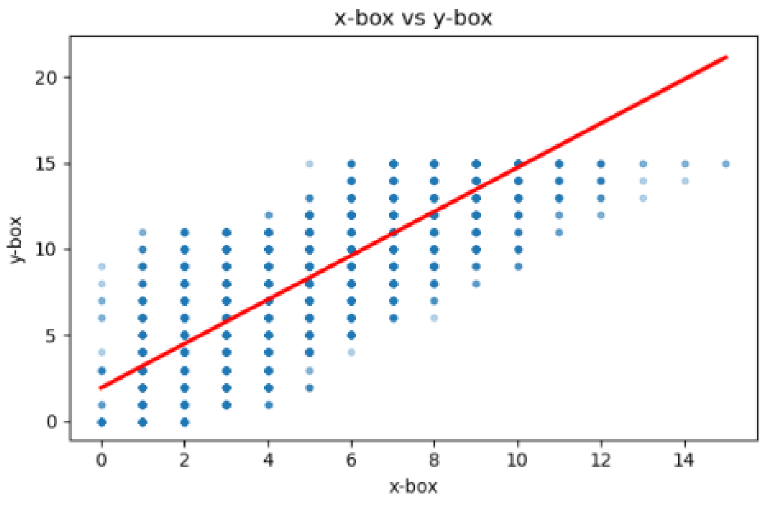


Fig 3 - *x-box vs y-box scatter plot*

For **high** and **width**, it is hypothesised that characters with a greater height value will also have a greater width, as larger characters tend to be larger in both dimensions. The scatter plot indicates a strong positive correlation, reiterating the hypothesis. **Onpix** and **width are** hypothesised to have a positive correlation as wider characters are expected to contain more active (black) pixels, this may be dependent on the font, but should largely be accurate for most fonts. A strong positive correlation is present. Implying the wider letter often has more detail/pixels. Furthermore, **onpix** and **high**, a slightly different hypothesis would be made for taller characters, such as "I", which might not always have higher pixel density, as density could be sparse. A strong positive correlation is observed, indicating that taller characters are denser, suggesting that characters such as "I" are selected cases and do not affect the general trend.

As for **x-bar** and **x2bar**, it is hypothesised that as the horizontal centroid increases (**x-bar**), the horizontal spread (**x2bar**) should remain constant, as the position of the centroid does not necessarily imply more variation. As observed from the graph, there is not much correlation observed, therefore supporting the hypothesis. For **y-bar** and **y2bar**, a similar hypothesis was made whereby the vertical centroid has no correspondence with the vertical variation. This is also supported by the weak/flat trend of the scatter plot.

It is hypothesised that characters with higher horizontal centroids (**x-bar**) will have a higher cross product value (**xybar**), as pixel mass is located more towards the right, it would be more likely to interact with vertical positions to form curved structures or diagonals. This is largely disapproved, as the scatter plot follows a slightly positive trend, demonstrating that a few characters may follow as the hypothesis states. However, they are largely independent due to the almost flat trend line.

For **x-ege** and **xevgy**, it is hypothesised that the **x-ege** will not necessarily have more vertical edge variation (**xegvy**), as vertical variation can exist even with smooth horizontal edges. The hypothesis is largely supported; a weak correlation was observed between the two edge features that inform different structural aspects of the characters.

Xybar and **x2ybr** are hypothesised to show a strong relationship as they both capture spatial complexity. The observed relationship showed no real relationship, generating a flat trend line. This can be attributed to the higher order of the **x2ybr**, capturing deeper structural traits beyond the first order interactions by **xybar**.

2.5 Model Selection

K-Nearest Neighbours and Decision Tree were selected as the two classification models to make predictions. The KNN algorithm predicts based on the most common class among its k-nearest neighbours in the feature space. As it relies on the distance calculations. The decision tree classification model works by recursively partitioning data based on features that provide the greatest information gain. It does not require feature scaling and provides clear and understandable results. Both models were trained on the processed training data and evaluated on the test data using performance metrics, printed out in the classification report.

2.6 Feature Selection and Parameter Tuning

KNN benefits from appropriate feature selection, unlike Decision Tree's which are inherently able to handle irrelevant features due to their hierarchical splitting nature (Fraidoon Omarzai, 2024). To select the appropriate features for the KNN model, a randomised feature selection was implemented, referred to as the hill climbing algorithm. It begins with trying the first feature, the feature is then "selected" if the features and existing already-selected features can improve the accuracy score, from the already-selected features. If so, it is then added to the selected features ("Introduction to Hill Climbing", 2024). Therefore, resulting in a list of appropriate features.

Then parameters including neighbours, distance metric, and weighting scheme were selected to be tuned. The parameters listed are crucial, as neighbours control how many surrounding data points influence the prediction, aiding in balancing variance and bias, ensuring greater generalisation. The distance metric matters as it affects how neighbours are elected, as KNN makes predictions based on nearby neighbours of a data point. There are two distances commonly used and applied to this dataset. Euclidean assumes that differences in all directions are equally important, while Manhattan distance works better when feature differences are independent rather than interacting together. Lastly, weighting dictates how much influence each neighbour has on the prediction.

Similarly, the Decision Tree parameters were selected to be tuned. However, there was no feature selection as noted, due to the nature of Decision Trees. The minimum number of samples required to be in a leaf node was used to reduce overfitting and aid in reducing sensitivity to noise in the training data. Criterion, which evaluates the quality of the split, different criterion leads to different tree structures, which may separate classes better. Finally, maximum depth, to reduce underfitting/overfitting, changes the depth of the tree, essentially controlling how many decisions the tree makes.

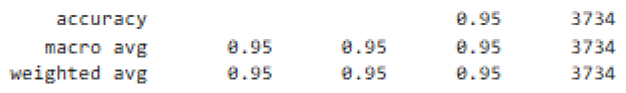
Trial and error with informed decision making was applied to perform the parameter tuning over a set of settings for each parameter for both classification models. Incrementally testing different values and subsequently referring to the performance metrics, parameters were added to or removed from. Concluding when the best metrics were observed and the best models for each classifier were stored.

3. Results

3.1 K-Nearest Neighbours (KNN)

The K-Nearest Neighbours (KNN) models were evaluated on the train dataset. Each model used k=5 but varied in the weighting strategy and distance metric.

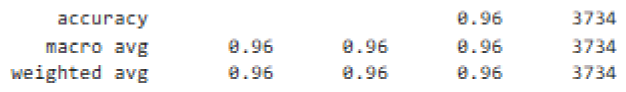
The first KNN model achieved an accuracy of 0.95 with uniform weights and a distance of p=2. The classification report showed high precision and recall for most letters, with few dips in performance for less frequent letters. Letters such as A, L, M, W and Z achieved F1-scores of 0.97 or higher, showing the strong confidence in predictions of these letters. However, a few letters showed lower performance. For example, B had a precision score of 0.90 and K had a score of 0.89.



```
    accuracy                           0.95      3734
   macro avg       0.95      0.95      0.95      3734
weighted avg       0.95      0.95      0.95      3734
```

Fig 4 - *KNN model 1 classification report*

The second model added the use of distance-based weights, producing identical results as shown in Fig 4 with slight variations in performance across particular letters, including C, H, and M. The third model used p=1, here performance was slightly better compared to the first and second KNN models by 0.01 across all metrics, being 0.96.

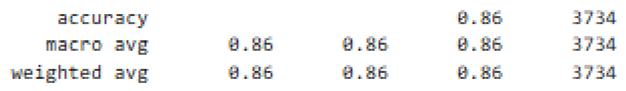


```
    accuracy                           0.96      3734
   macro avg       0.96      0.96      0.96      3734
weighted avg       0.96      0.96      0.96      3734
```

Fig 5 - *KNN model 3 classification report*

3.2 Decision Trees

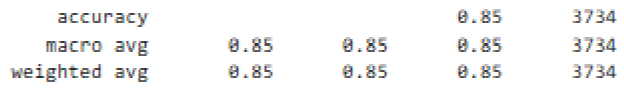
Similar to KNN, the Decision Tree models were evaluated on the train dataset. The first Decision Tree model used default parameters and achieved an overall accuracy of 0.86. The rest of the scores on the classification report are 0.86. The F1-scores for the letters ranged from 0.80 to 0.92, with the strongest performance observed for classes A, L, M, T and W, and lower scores for B, E, F and H.



```
    accuracy                           0.86      3734
   macro avg       0.86      0.86      0.86      3734
weighted avg       0.86      0.86      0.86      3734
```

Fig 6 - *Decision Tree model 1 classification report*

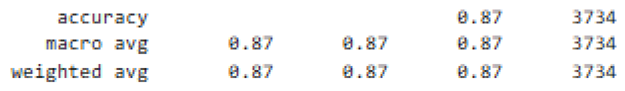
The second Decision Tree model was tuned incrementally. Parameters such as max_depth=15, criterion='entropy', and min_samples_leaf=2. This configuration results in overall metrics accuracy being 0.85, the macro and weighted averages were 0.85 as well. The metrics observed were largely similar to the baseline model for individual letters, decreasing by an overall accuracy of 0.01.



```
    accuracy                           0.85      3734
   macro avg       0.85      0.85      0.85      3734
weighted avg       0.85      0.85      0.85      3734
```

Fig 7 - *Decision Tree model 2 classification report*

The last model, by only using entropy, resulted in 0.87 across all metrics (Fig 8). Therefore, it is the best Decision Tree model.



```
    accuracy                           0.87      3734
   macro avg       0.87      0.87      0.87      3734
weighted avg       0.87      0.87      0.87      3734
```

Fig 8 - *Decision Tree model 3 classification report*

4. Discussion

This section presents a critical evaluation of both classification models, discussing the results gathered and a comparison of both, including a recommendation between the KNN and Decision Tree models.

4.1 Feature Selection and Parameter Tuning

Feature selection reduced the dimensionality of the input space and also eliminated redundant or less informative features. The hill climbing algorithm was efficient in doing so for the KNN model. In the experiment, the algorithm was applied to the baseline model. Applying feature selection to the best model may have returned different results. However, as explored, the various parameters included only caused a slight change in the performance, the difference between the best and the worst model being 0.01. From feature selection, all 16 features were retained, meaning that they are all important in yielding predictions. As for the Decision Trees, due to the aforementioned nature of splitting and disregarding irrelevant features, no feature selection was applied.

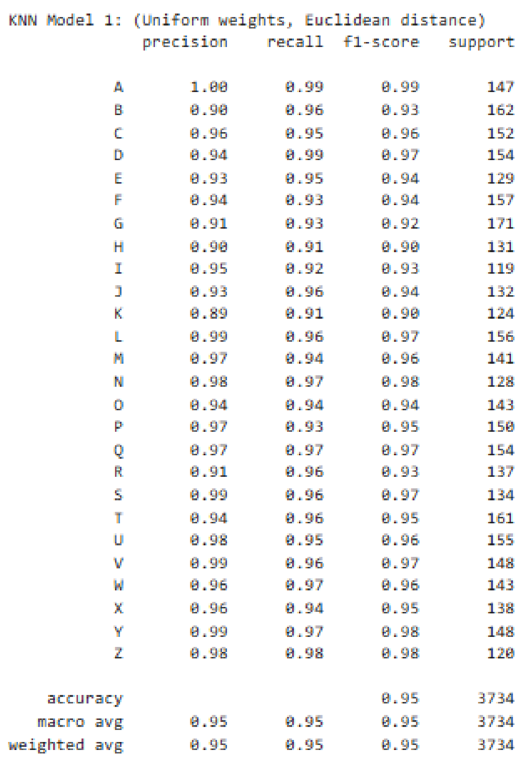
The parameter tuning for the KNN model was incremental. Moving from uniform to distance-based weighting improved the model's predictiveness to local feature variations, reducing misclassification among visually similar letters. Using p=1 yielded the best results. This can be attributed to the structural nature and pixel-based statistics, which align well with orthogonal distance calculations (Lu et al., 2024).

As for parameter tuning, there were interesting observations made. The Decision Tree model had no improvement in performance with the addition of most parameters. For example, maximum depth was tested with different values. The optimal maximum depth was 21, whereby an increase beyond 21 resulted in the same results. Therefore, the value 21 allowed the tree to reach its maximum depth, the same was also observed by the minimum leaf samples producing the best results for 1, which is its default value. A reason why parameters that aid in overfitting were not effective was that with a larger number of samples (20,000) it was harder for overfitting to trigger, as the model had enough data points to be able to learn diverse and general patterns (Rahman, 2023). Furthermore, the default and less restricted decision tree achieved a perfect accuracy on the training data (1.00), indicating potential overfitting. However, when generalisation performance based on the test data improved with deeper trees, this suggests that the restricted trees were in fact, underfitting, failing to capture the complexity of the data. Therefore, allowing the tree to reach a deeper decision is justified, to be able to improve performance.

As highlighted in the results, the only increase in performance was changing the criterion to ‘entropy’. This can be attributed to information oriented splits made by ‘entropy’, which made it easier to separate features for the classes. Especially when working with overlapping features amongst letters, it can make smarter decisions.

4.2 Analysis of Characters

Certain characters perform extremely well across both models, with or without parameter tuning involved. Letters such as ‘Y’, ‘S’, and ‘L’ were consistently predicted with higher scores, while ’B’, ‘E, and ‘H’ had lower scores. For the former, this is likely due to distinct structural features which are well separated descriptors of the characters, from classes such as x-bar, onpix, and x-ege, which represent the edge directions, box sizes and pixel distributions. Whereas the latter potentially suffers from overlapping features (i.e. vertical symmetry and edge orientations). Kowsari et al. (2019) discuss the impact of such ambiguities, noting that classification models often struggle with letters whose shapes lead to similar statistical presentations.



```
KNN Model 1: (Uniform weights, Euclidean distance)
              precision    recall  f1-score   support

           A       1.00      0.99      0.99       147
           B       0.90      0.96      0.93       162
           C       0.96      0.95      0.96       152
           D       0.94      0.99      0.97       154
           E       0.93      0.95      0.94       129
           F       0.94      0.93      0.94       157
           G       0.91      0.93      0.92       171
           H       0.90      0.91      0.90       131
           I       0.95      0.92      0.93       119
           J       0.93      0.96      0.94       132
           K       0.89      0.91      0.90       124
           L       0.99      0.96      0.97       156
           M       0.97      0.94      0.96       141
           N       0.98      0.97      0.98       128
           O       0.94      0.94      0.94       143
           P       0.97      0.93      0.95       150
           Q       0.97      0.97      0.97       154
           R       0.91      0.96      0.93       137
           S       0.99      0.96      0.97       134
           T       0.94      0.96      0.95       161
           U       0.98      0.95      0.96       155
           V       0.99      0.96      0.97       148
           W       0.96      0.97      0.96       143
           X       0.96      0.94      0.95       138
           Y       0.99      0.97      0.98       148
           Z       0.98      0.98      0.98       120

    accuracy                           0.95      3734
   macro avg       0.95      0.95      0.95      3734
weighted avg       0.95      0.95      0.95      3734
```

Fig 9 - *Complete classification report for KNN model 1*

4.3 Comparative Performance of Models

Both the K-Nearest Neighbours (KNN) and Decision Tree models showed strong performance for all the letters of the dataset. Achieving an accuracy score of 0.85+. However, the KNN model outperformed the Decision Tree in accuracy, precision and F1-score across all letters. The line graph below illustrates the performance of the best two models, generated from each of the classifiers, clearly showing how well the KNN model performed compared to the Decision Tree across all letters.

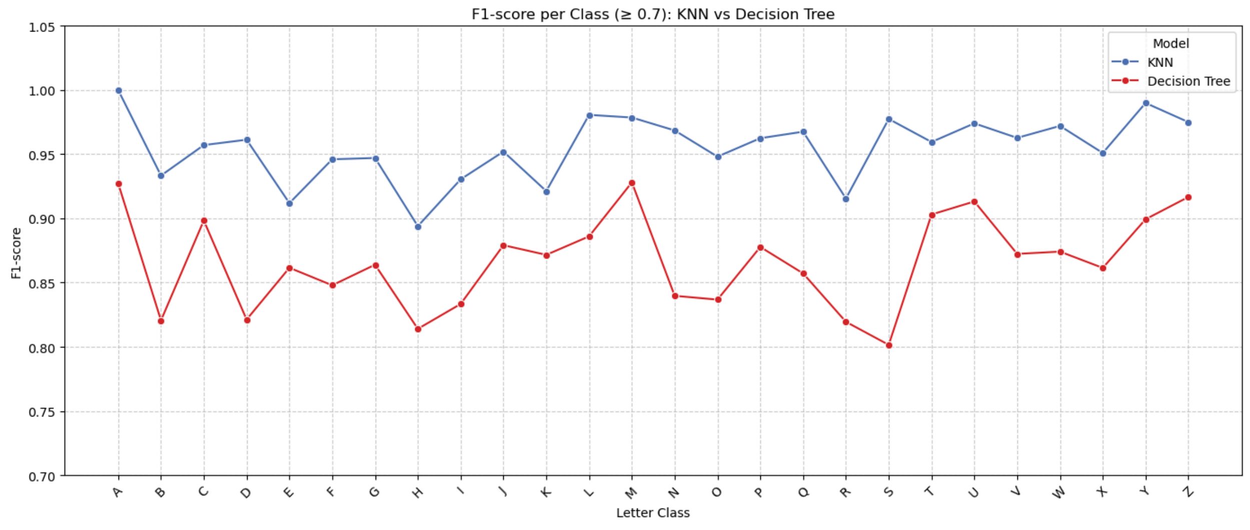


Fig 10 - *KNN vs Decision Tree*

The baseline KNN model with only neighbours set to 5, achieved 0.95 across all metrics. By introducing the distance-based weighting, the model stayed the same at 0.95 across all metrics. The best performing model was with the addition of p=1, achieving scores of 0.96. This model better handled local variability by assigning the neighbours which were closer more importance, therefore it is beneficial when small pixel-based differences distinguish characters, especially in the case of letters.

In contrast, the Decision Tree model with no additional parameters produced an accuracy of 0.86, a difference of 0.09 compared to the baseline KNN model. Changing the criterion, from 'gini' to 'entropy' increased the scores from 0.86 to 0.87. Testing out the parameters for models 2 and 4, the parameters resulted in a decrease in performance, the higher the constraint, with no improvement with any tested values. Therefore, 0.87 is the best overall accuracy, precision and F1-score. This is significantly worse compared to the KNN model, which has a score 0.09 greater in all performance metrics compared to the Decision Tree model.

4.4 Model Strengths and Limitations

As established, the KNN outperforms the Decision Tree model across all models. Discussing the strengths and limitations would be able to give more insight into the reason why, specifically for pixel based data.

KNN's local decision-making allows it to catch subtle variations and learn complex class boundaries. Especially useful in distinguishing characters that differ by only slight pixel changes (e.g O and Q). However, its main drawback lies in computational efficiency, whereby every prediction requires distance calculations, which becomes unfeasible for larger datasets. Decision Trees, on the other hand, lump examples with similar individual features but different global shapes into the same node, being less sensitive to these subtle pixel variations (Kashnitsky, 2025).

Decision Trees are fast to train and predict, and provide clear interpretability through their branching structure. However, as they split based on one feature at a time, it makes it harder for the model to capture complex patterns needed to distinguish visually similar letters. This is justified through the multiple correlated features within the dataset, such as x-box and y-box (**Section 3.4**).

4.5 Recommendation

Based on the discussion and results presented, the best model was the KNN model (k=5, distance-weighted, p=1), which was recommended for classification for this task. It offered the highest overall accuracy, macro average, and weighted average scores from all models. Along with the implementation of feature selection and parameter tuning, the model is robust and able to generalise consistently across all letters. Although the Decision Tree model does offer a faster and interpretable design, its comparatively lower performance makes it less ideal for the pixel-derived data from this dataset.

5. Conclusion

This report, through a structured process, explored the dataset of 20,000 samples and evaluated two classification models, KNN and Decision Trees. The process included steps such as data exploration, data handling, feature selection and parameter tuning. Producing classification reports and comparing results. Analysing the relationships between the attribute pairs played a crucial role in interpreting the performance of the model and the reasons for underperformance as well. It revealed elements such as structural redundancies that affect Decision Trees more compared to the KNN, due to univariate splitting. The relationships guided the reasons why KNN consistently outperformed the Decision Tree models.

KNN emerged as the better performing model for this dataset, achieving the highest accuracy of 0.95 when parameter tuning was applied. The model benefited from feature selection, highlighting the importance of preprocessing in distance-based models. Decision Trees reached a performance of 0.86 and were notably more sensitive to the parameters and ambiguity amongst letters.

In conclusion, while both models were appropriate for the letter classification, the ideal model was the KNN classification, after being tuned, in recognising handwritten letters from pixel-based features.

References

Aymane Hachcham. (2025, April 23). The KNN Algorithm - Explanation, Opportunities, Limitations. Neptune.ai. https://neptune.ai/blog/knn-algorithm

Fraidoon Omarzai. (2024, July 20). Decision Tree In Depth - Fraidoon Omarzai - Medium. Medium. https://medium.com/@fraidoonomarzai99/decision-tree-in-depth-da6ff64d8e54

GeeksforGeeks. (2024, October 10). Introduction to Hill Climbing | Artificial Intelligence. https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

Kashnitsky, Y. (2025). *Topic 3. Classification, Decision Trees and k Nearest Neighbors — mlcourse.ai*. Mlcourse.ai. https://mlcourse.ai/book/topic03/topic03_decision_trees_kNN.html

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). *Text Classification Algorithms: A Survey*. Information, 10(4), 150. https://doi.org/10.3390/info10040150

Lu, Y., Wan, L., Ding, N., Wang, Y., Shen, S., Cai, S., & Gao, L. (2024). *Unsigned Orthogonal Distance Fields: An Accurate Neural Implicit Representation for Diverse 3D Shapes*. ArXiv.org. https://arxiv.org/abs/2403.01414

Nnajiofor, E. (2024). *Understanding the Decision Trees Algorithm: A Guide to Data-Driven Decisions*. Medium. https://medium.com/@emiklad/understanding-the-decision-trees-algorithm-a-guide-to-data-driven-decisions-78259fc7f581

Rahman, A. A. (2023). *Strategies to Mitigate Overfitting in Deep Learning*. www.linkedin.com. https://www.linkedin.com/pulse/strategies-mitigate-overfitting-deep-learning-abdullah-al-rahman