CS-451 Assignment-3 Report

1. K Nearest Neighbors Classifier

1. Description

i. In k nearest neighbors' classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. KNN uses Euclidian distance to calculate k neighbors.

2. Parameters

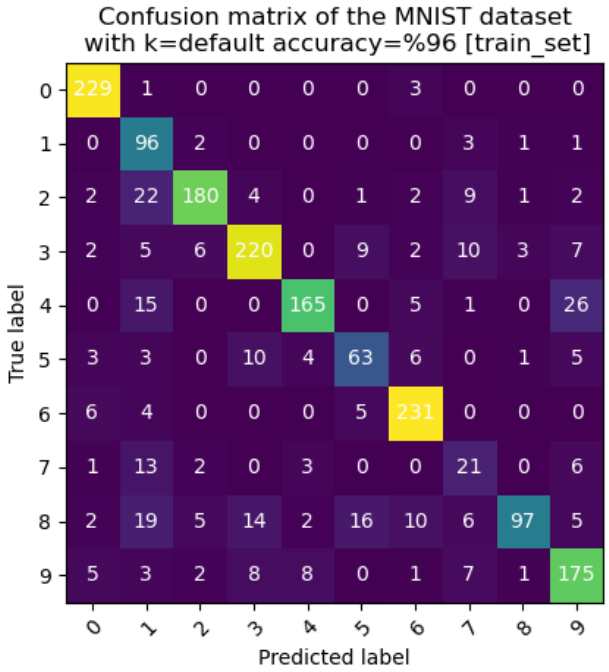
i. n_neighbors => Number of neighbors used to classify
ii. weights => uniform (All points in each neighborhood are weighted equally), distance (closer points have higher weights for the votes)
iii. Algorithm => auto, I did not play with this parameter in results
iv. P => p= 1 (Manhattan distance), p= 2 (Euclidian distance), p = l_p (minkowski distance)

3. Steps

i. Determine parameter K
ii. Calculate the Euclidian distance between the instance and all training samples
iii. Sort the distance and determine k neighbors
iv. Gather the category of the k neighbors
v. According to the majority vote put the instance in that category

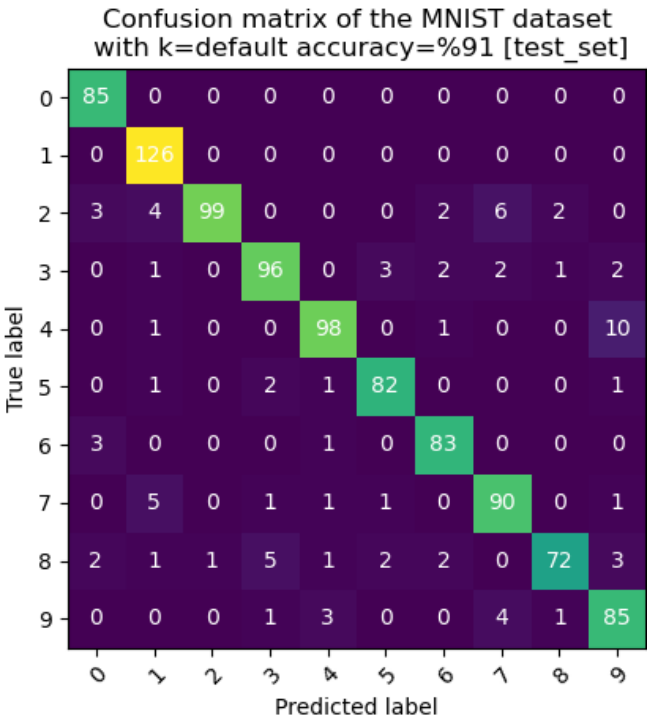
4. Results

i. Confusion matrix of train and test dataset with default parameters



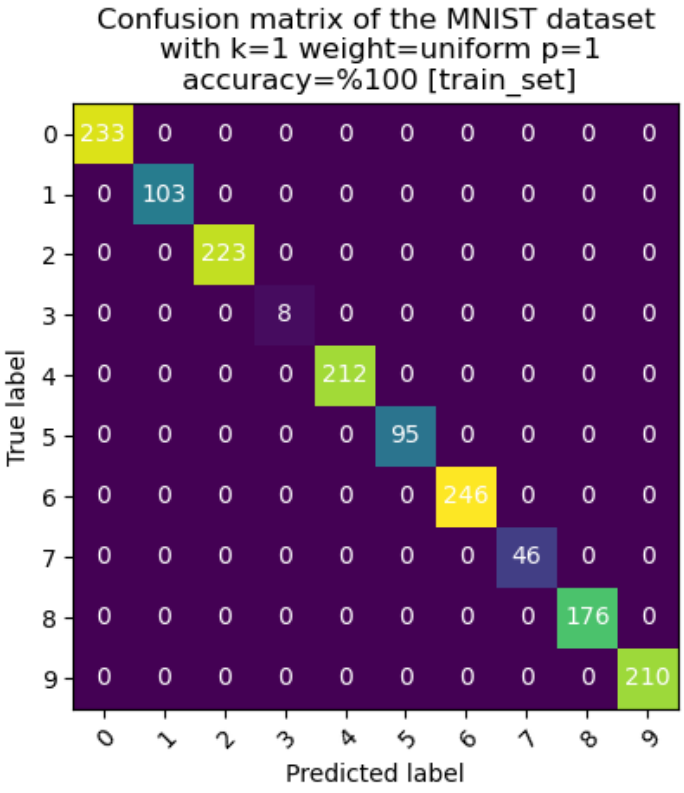
1.

2.

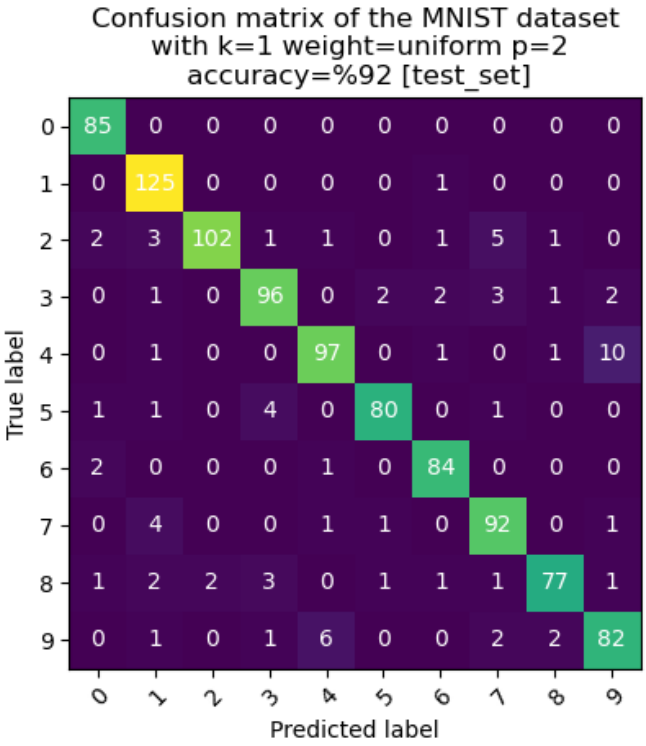


3.

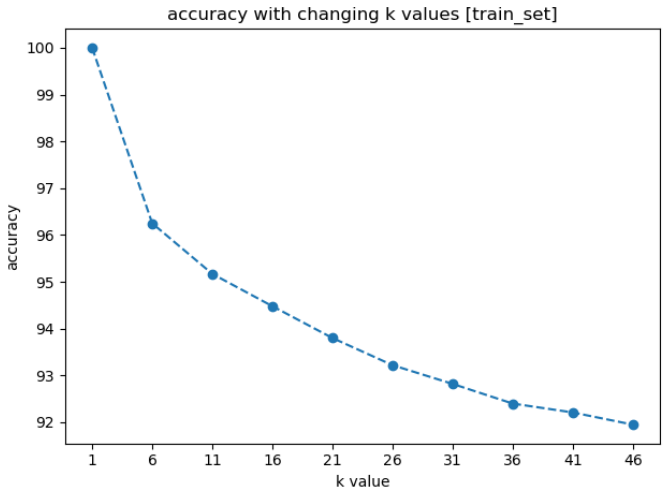
ii. Confusion matrix of train and test dataset with tuned parameters



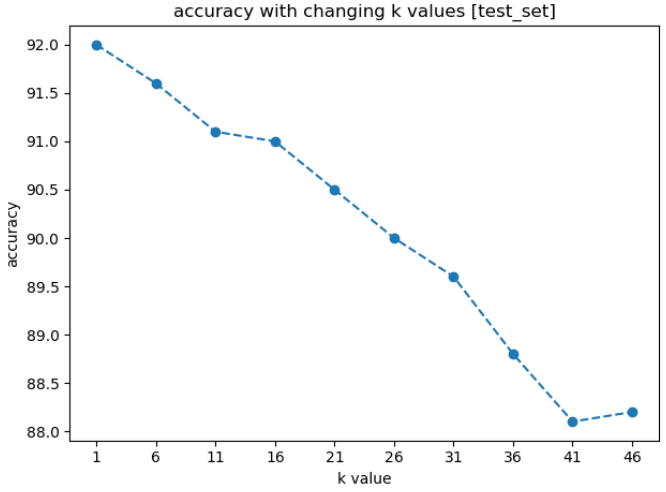
1.


Confusion matrix of the MNIST dataset
with k=1 weight=uniform p=2
accuracy=%92 [test_set]
True label
Predicted label
0 85 0 0 0 0 0 0 0 0 0
1 0 125 0 0 0 0 1 0 0 0
2 2 3 102 1 1 0 1 5 1 0
3 0 1 0 96 0 2 2 3 1 2
4 0 1 0 0 97 0 1 0 1 10
5 1 1 0 4 0 80 0 1 0 0
6 2 0 0 0 1 0 84 0 0 0
7 0 4 0 0 1 1 0 92 0 1
8 1 2 2 3 0 1 1 1 77 1
9 0 1 0 1 6 0 0 2 2 82
0 1 2 3 4 5 6 7 8 9

2.


accuracy with changing k values [train_set]
accuracy
k value
100 99 98 97 96 95 94 93 92
1 6 11 16 21 26 31 36 41 46

3.


accuracy with changing k values [test_set]
accuracy
k value
92.0 91.5 91.0 90.5 90.0 89.5 89.0 88.5 88.0
1 6 11 16 21 26 31 36 41 46

4.

5. **Parameter Effect Discussion**
    i. With default hyperparameters (neighbors =5, weights= uniform, p= Euclidian distance) training set accuracy was %96 percent. With tuned parameters (neighbors =1, weights= uniform, p= Euclidian distance) %100 accuracy could be reached normally I would have expected n=1 to be overfitting but surprisingly it worked well with the test set with %1 increase as well on accuracy %91 to %92. With Manhattan distance best training accuracy was reached but with closest neighbors' algorithm giving best results I don't believe it would have made a difference. As seen with the test results Euclidian distance gave the best result, in this case Manhattan distance parameter created an overfitting.

2. Naïve Bayes Classifier

1. Description

i. In AI, naïve Bayes classifiers are a group of basic "probabilistic classifiers" based on applying Bayes' theorem with solid (naïve) independence assumptions between the features. They are among the least difficult Bayesian system models. They could be combined with Kernel density estimation and accomplish higher accuracy levels. In this assignment Gaussian Naïve Bayes classifier is used

2. Parameters

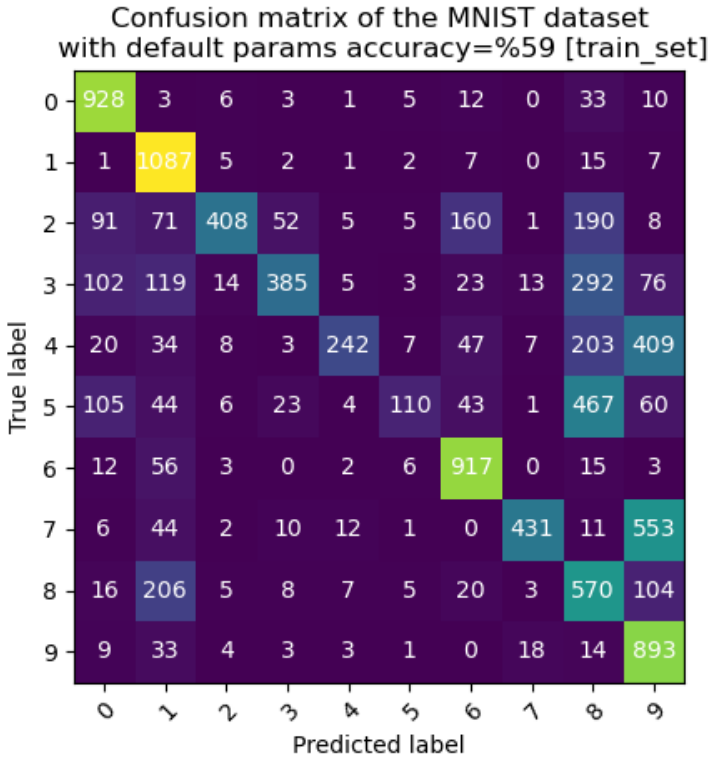
i. Var_smoothing => Portion of the largest variance of all features that is added to variances for calculation stability

3. Steps

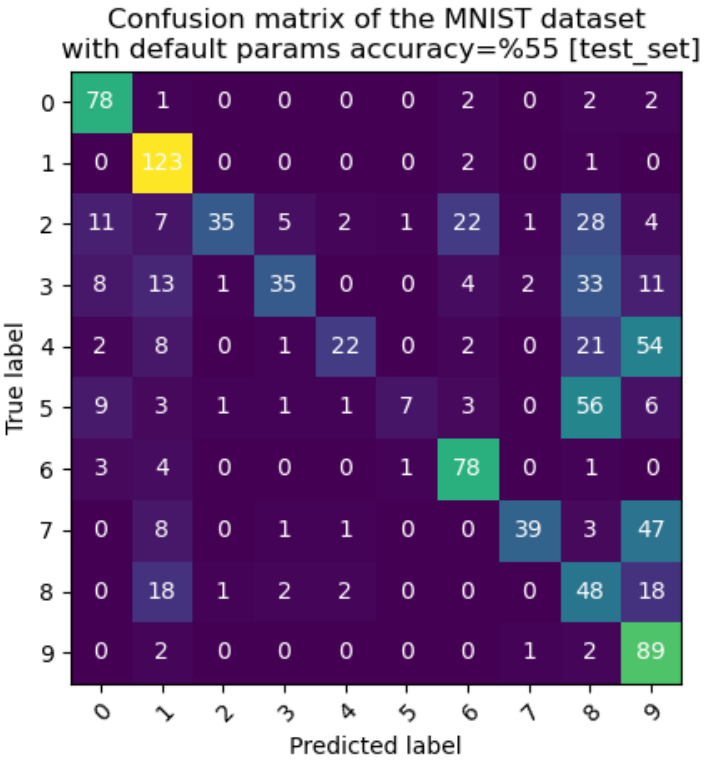
i. Calculate the prior probability for given class labels
ii. Find Likelihood probability with each attribute for each class
iii. Put these values in Bayes Formula and calculate posterior probability
iv. See which class has a higher probability, given the input belongs to the higher probability class

4. Results

i. Confusion matrix of train and test dataset with default parameters

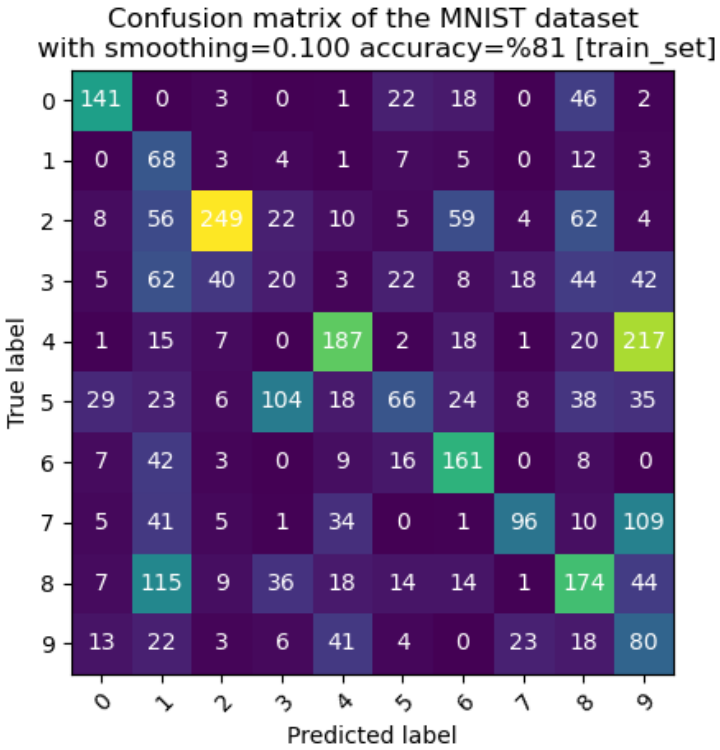


1.

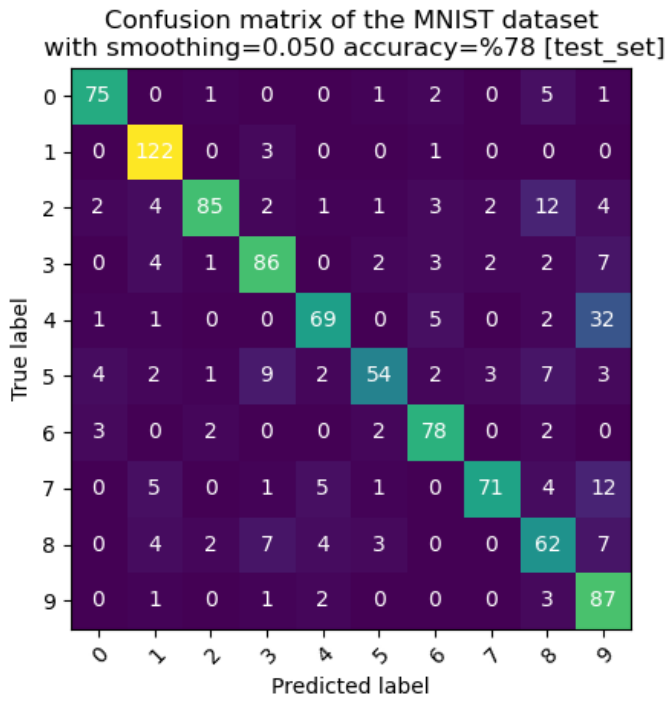


2.

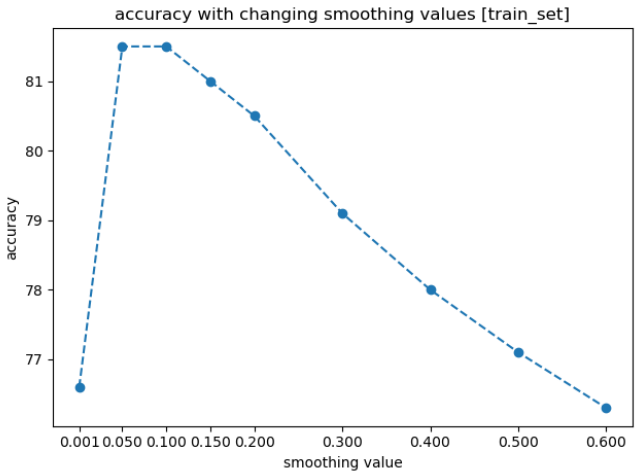
ii. Confusion matrix of train and test dataset with tuned parameters



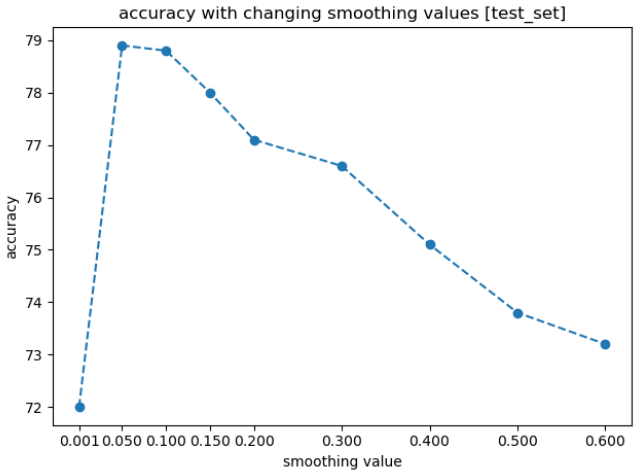
1.



2.



3.



4.

5. Parameter Effect Discussion

i. With Gaussian Naïve Bayes Classifier only hyperparameter that was easily tune able was smoothing value. With low smoothing value i.e. 0.001 the images are not standardized enough, and they are prone to noise. With high smoothing value i.e. 0.6 the images are too similar which is not enough to distinguish the classes. The sweet smoothing value range is between 0.05 to 0.1.

3. Decision Tree Classifier

1. Description

i. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

2. Parameters

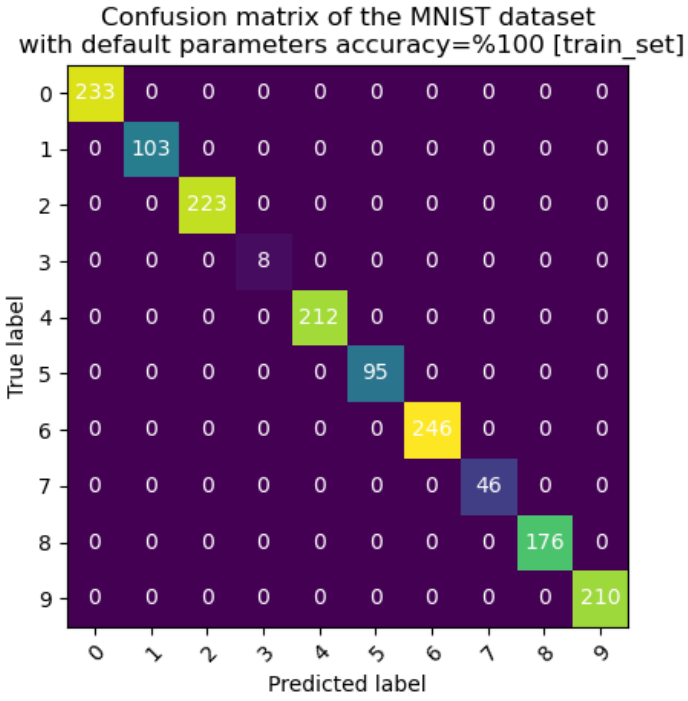
i. Criterion => gini, entropy (gini for gini impurity, entropy for information gain)
ii. Splitter => best, random (best chooses best best split, random chooses best random split)
iii. Max_depth => any int value (The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.)
iv. Min_samples_split => any int or float (The minimum number of samples required to split an internal node)
v. Min_samples_leaf => any int or float (The minimum number of samples required to be at a leaf node)

3. Steps

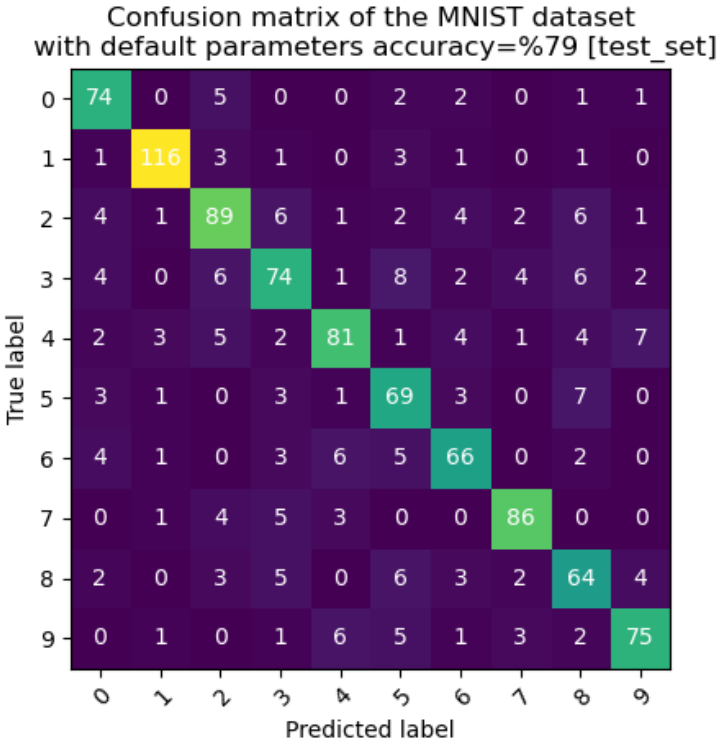
i. Determine the Decision Column
ii. Calculating Entropy for the classes
iii. Calculate Entropy for Other Attributes After Split
iv. Calculating Information Gain for Each Split
v. Perform the First Split
vi. Complete the Decision Tree

4. Results

i. Confusion matrix of train and test dataset with default parameters

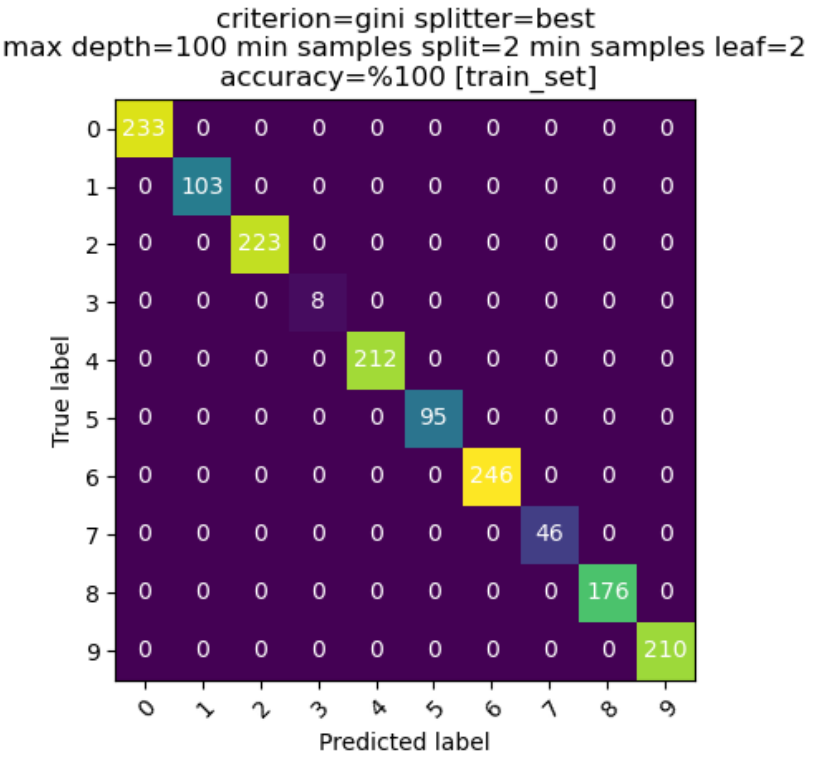


1.

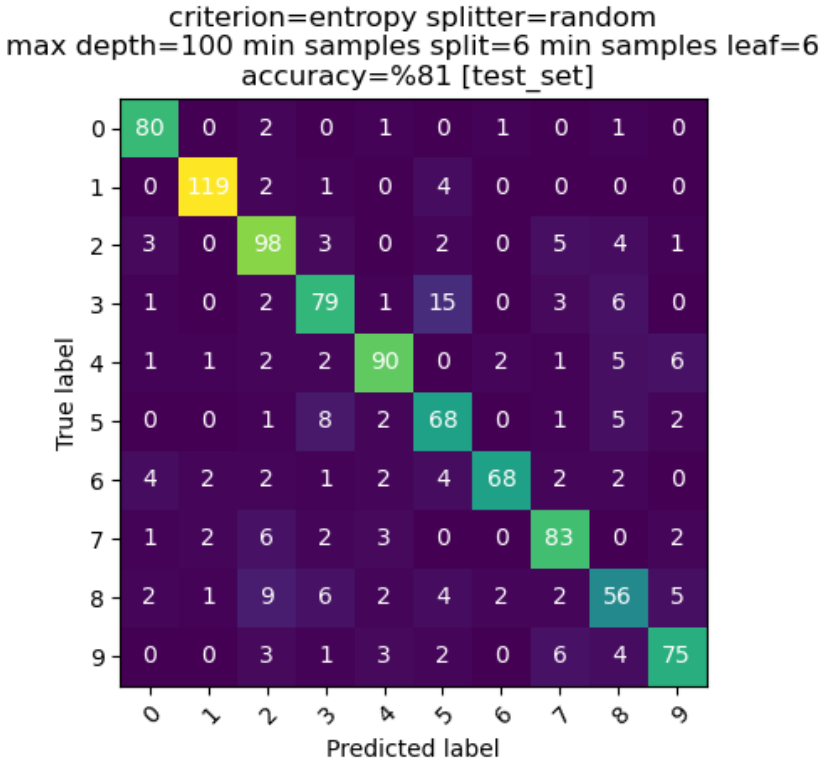


2.

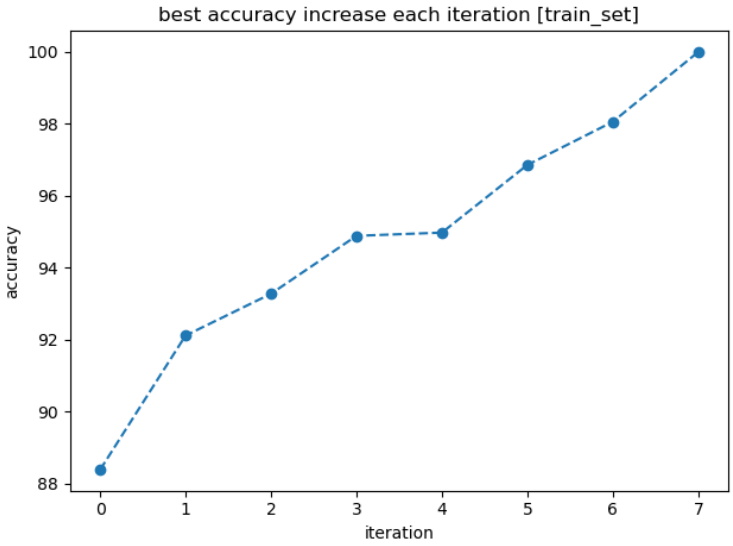
ii. Confusion matrix of train and test dataset with tuned parameters



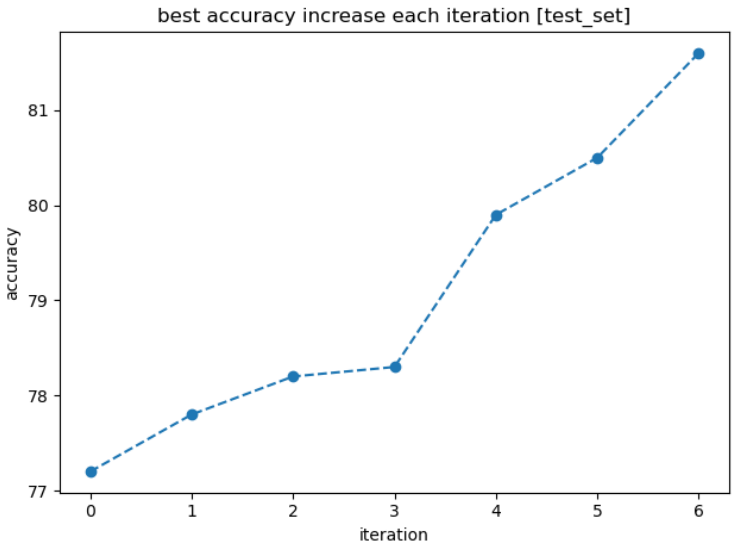
1.



2.



3.



4.

5. Parameter Effect Discussion

i. Starting with the quality of the split Gini tries to make the decision tree impure as possible and as expected gave the best training value accuracy however did not collect enough information about the decision tree hence entropy gave better results on test set with accuracy reaching %81. The best splitter gave the most accurate result on train set but again with the same principle best splitter did not explore the decision tree as much as the random split. Max depth should be big as possible as seen in both train and test results; max depth limitation should not be set. Min samples leaf and split should also be tuned. These parameters gave the best result on train set with value 2 but on the test, set did not perform best, with both being at value 6. So, in conclusion although lower sampling requiring creates a more distinct tree, because the sample sizes being small, they are prone to outliers in the data. In this case the sweet value for both values were 6.