Project #2

Mads Moldrheim

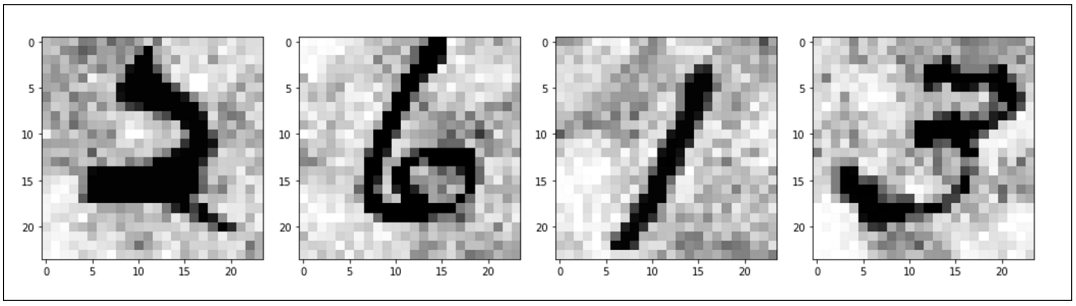
1 Introduction

1.1 Goal

The goal of this project is to produce a classifier that predicts labels of handwritten digits. This will be accomplished through a pipeline that will test various models and hyperparameters and select the best model.

1.2 Evaluation of the data set

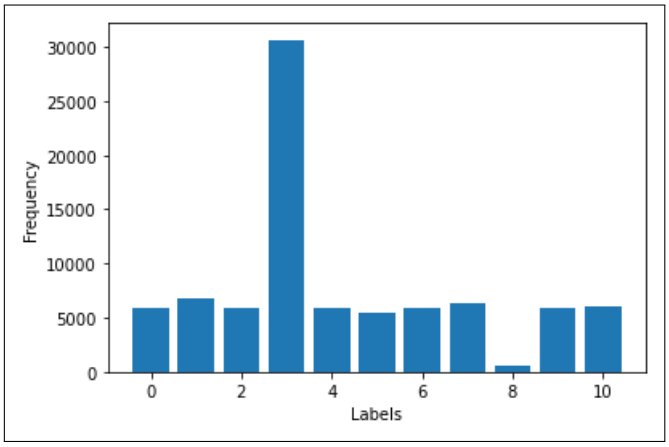
The data set is a matrix of 24x24 pixel grey-scale images. The images are handwritten digits from 0 to 10 and amount to a total of 85273 images. The shape of the matrix is (85273, 24, 24, 1) but will be reshaped into an (85273, 576) array. Figure 1 shows what the images from the data set looks like.



Figur 1: Four images from the data set

Figure1 shows that there is a lot of noise in the images. This is likely to affect the models.

The matrix of features is accompanied by an array of labels for the images. If we plot a histogram of the labels we get figure 2.



Figur 2: Histogram for the labels of the data set

Figure 2 shows that we have a heavily skewed data set, with a majority of 3's and a minority of 8's. This could be a problem, as 3's and 8's look alike. The model could be tempted to label all 8's as 3's and still get a high accuracy.

1.2.1 Balancing the data

To combat the skewed data, a function to balance the data has been implemented. The function takes the data set and identifies oversampled and undersampled labels. For oversampled data, the algorithm randomly removes data points. For undersampled data, the algorithm randomly duplicates data points from the original data set. The function will under- or oversample until the amount of data points for the relevant label matches the average number of data points for labels that are already balanced.

1.3 Performance measure

As it is equally important to correctly label all digits, the performance measure is average accuracy. The formula implemented is given as:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\text{number of correctly predicted labels of label i}}{\text{total number of label i}} \tag{1}$$

Where n is the number of unique labels. It will calculate the accuracy for each label individually and then average them. This way, all labels are equally weighted, and a model that disregards the minority label will be punished.

2 Models

The model families and their hyperparameters chosen for consideration are the following:

2.1 k-nearest neighbors

The module *KNeighborsClassifier* from *sklearn.neighbors* will be utilized to create a kNN classifier. Hyperparameters to vary are number of neighbors k and we can choose to weight the distance to each neighbor or not.

2.2 Decision tree

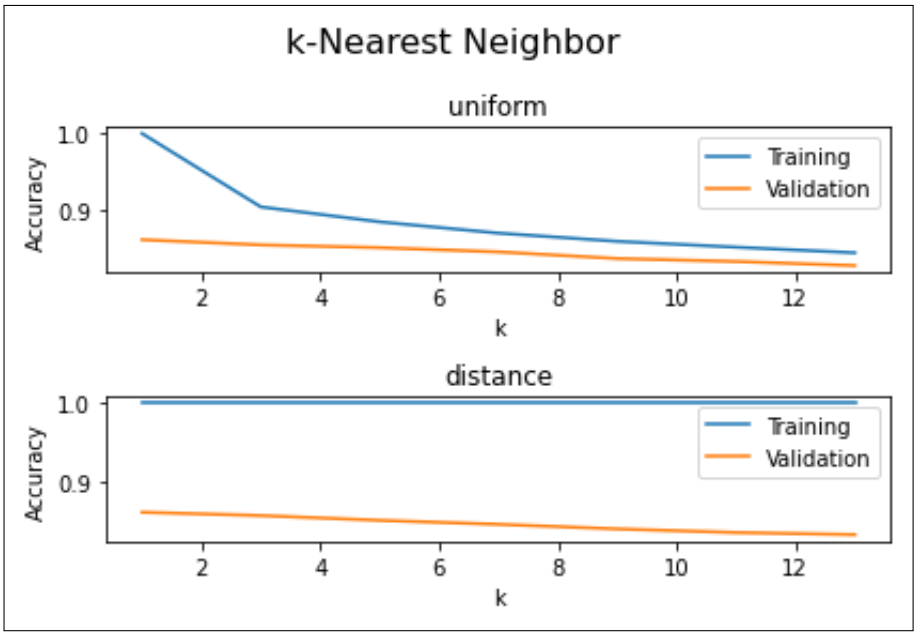
The module *DecisionTreeClassifier* from *sklearn.tree* will be utilized to create a decision tree classifier. Hyperparameters to vary are impurity measure and minimum amount of samples required to make a split.

2.3 Neural network

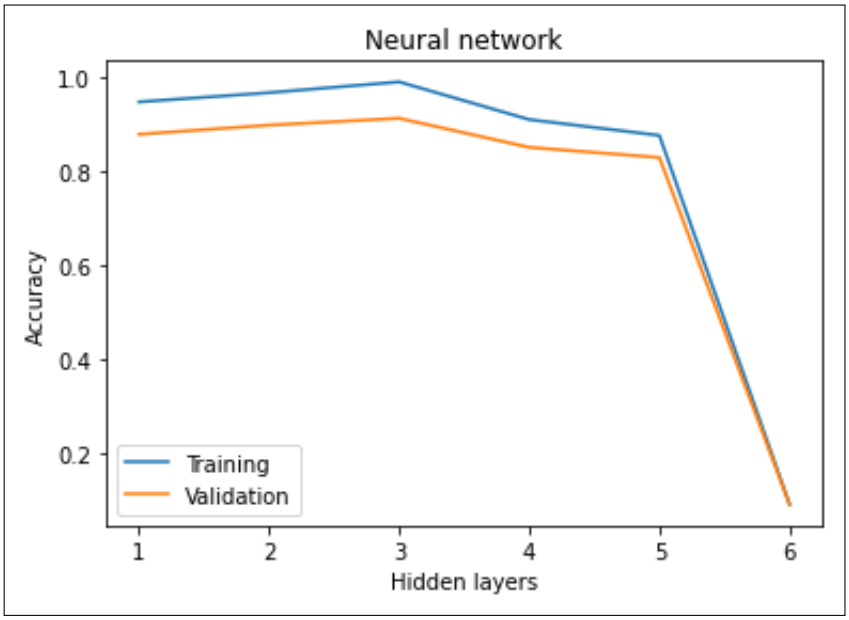
The module *MLPClassifier* from *sklearn.neural_network* will be utilized to create the neural network. Hyperparameters that will be varied are the depth of the neural network and number of nodes in each hidden layer. The number of hidden layers will vary, but for simplicity's sake a total amount of nodes is selected and evenly distributed across the hidden layers.

3 Model selection

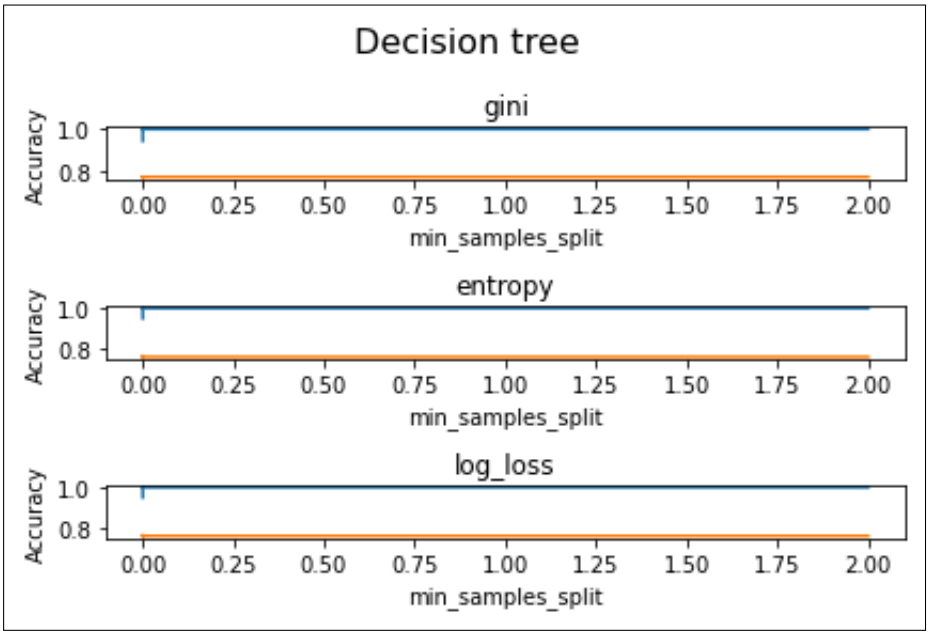
The final model is the model with the highest validation accuracy. All the accuracies of the models are checked and the final model is presented. Plots generated in the script for the three models and their accuracies are shown below in figure 3, 4 and 5.



Figur 3: Accuracies for k-nearest neighbor



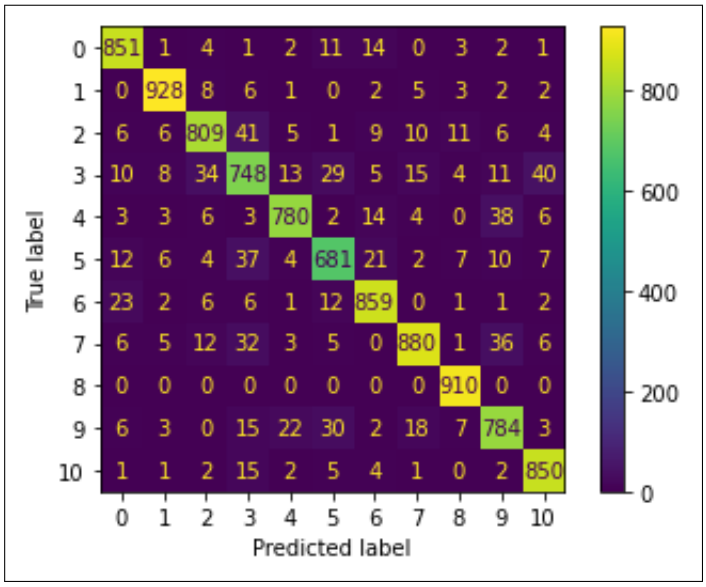
Figur 4: Accuracies for neural network



Figur 5: Accuracies for decision tree

The final selection of the model is printed in the console and is recited here:
The chosen model with the highest validation accuracy is the following:
Neural network classifier with hidden layers: 4 and nodes per layer: 64. Validation accuracy is 0.9124019900153765

model's accuracy on test data is 0.9162175406938345
Finally, the confusion matrix for the model on the test data is printed, shown here in figure 6.



Figur 6: Confusion matrix for the final model on test data

4 Discussion

The results show that the neural network was the best classifier for the task. The confusion matrix shows that there is not a skewed misclassification happening. The misclassifications are rather spread out.
The plots for the decision trees are rather surprising, as there is very little variance in the accuracy. This suggests that the chosen hyperparameters were not ideal.
There seems to be a bug in the plot for the neural network, as the graphics does not line up with the results.
An attempt to use a Support Vector Machine as one of the models was made. However, learning the model was very time consuming on the large data set, and was ultimately abandoned.