Decision Tree

**Baseline results:**

Confusion matrix:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Predicted Labels | | | | | | | | | | | |
| Real Labels |  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **0** | 144 | 0 | 4 | 2 | 2 | 3 | 2 | 0 | 4 | 1 |
| **1** | 0 | 155 | 6 | 4 | 3 | 2 | 3 | 3 | 3 | 2 |
| **2** | 1 | 5 | 110 | 7 | 2 | 5 | 6 | 7 | 6 | 9 |
| **3** | 2 | 2 | 9 | 119 | 2 | 11 | 4 | 1 | 13 | 3 |
| **4** | 0 | 2 | 6 | 2 | 123 | 3 | 3 | 2 | 5 | 16 |
| **5** | 3 | 1 | 4 | 9 | 3 | 99 | 6 | 2 | 7 | 7 |
| **6** | 2 | 0 | 7 | 3 | 6 | 2 | 137 | 2 | 6 | 0 |
| **7** | 0 | 0 | 4 | 1 | 4 | 0 | 2 | 152 | 2 | 6 |
| **8** | 2 | 4 | 4 | 13 | 4 | 5 | 3 | 5 | 96 | 14 |
| **9** | 2 | 1 | 1 | 3 | 12 | 6 | 3 | 5 | 0 | 127 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Label** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **Precision** | 0.923077 | 0.911765 | 0.709677 | 0.730061 | 0.763975 | 0.727941 | 0.810651 | 0.849162 | 0.676056 | 0.686486 |
| **Recall** | 0.888889 | 0.856354 | 0.696203 | 0.716867 | 0.759259 | 0.702128 | 0.830303 | 0.888889 | 0.64 | 0.79375 |

Overall accuracy = 76.2%

**Experiments:**

**Variation on the baseline:**

I tested changing the values for the minimum number of samples required for a split and the minimum number of samples required to form a leaf. First I tried changing each value by itself, and both the recall and precision went down drastically for both. I also tried a decision tree with both values changed, and still all performance measures went down.

Then I tested changing the max depth of the tree. Here, changing the max depth actually resulted in interesting differences. Using a max depth of 20 gave better results for some labels and worse results for others. However, overall it was slightly worse than no max depth. I then messed around with other similar values for max depth and found that a max depth of 18 was generally the best. However, the changes were small overall when comparing the overall accuracy.

**Max Depth = 18**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Predicted Labels | | | | | | | | | | | |
| Real Labels |  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **0** | 140 | 0 | 7 | 2 | 2 | 3 | 3 | 0 | 4 | 1 |
| **1** | 0 | 158 | 6 | 4 | 3 | 2 | 3 | 3 | 0 | 2 |
| **2** | 1 | 4 | 117 | 7 | 1 | 3 | 13 | 6 | 3 | 3 |
| **3** | 2 | 2 | 11 | 121 | 0 | 12 | 2 | 0 | 11 | 5 |
| **4** | 1 | 1 | 4 | 0 | 122 | 2 | 4 | 5 | 5 | 18 |
| **5** | 8 | 1 | 4 | 7 | 1 | 101 | 6 | 1 | 5 | 7 |
| **6** | 2 | 0 | 8 | 1 | 6 | 2 | 136 | 1 | 8 | 1 |
| **7** | 0 | 1 | 5 | 2 | 2 | 0 | 3 | 149 | 2 | 7 |
| **8** | 2 | 4 | 7 | 10 | 3 | 4 | 8 | 5 | 94 | 13 |
| **9** | 1 | 2 | 3 | 3 | 6 | 4 | 4 | 7 | 2 | 128 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Label** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **Precision** | 0.89172 | 0.913295 | 0.680233 | 0.770701 | 0.835616 | 0.759398 | 0.747253 | 0.841808 | 0.701493 | 0.691892 |
| **Recall** | 0.864198 | 0.872928 | 0.740506 | 0.728916 | 0.753086 | 0.716312 | 0.824242 | 0.871345 | 0.626667 | 0.8 |

Since I was unable to find any variation that consistently improved the baseline model, I decided to continue using the unaltered decision tree.

**Hand-engineered features:**

The first feature I tested was simply taking the average of all pictures in an image. Since taking the average divides each total by the same number of pixels, the total of the pixels values was used instead as it was equivalent to using the average but used less operations. However, the pixel sum feature did not make any major improvement on the model, and instead had slightly worse results. Another similar feature I attempted was using the number of 0 pixels as a feature. This feature would correspond to the amount of empty space in an image, which I thought would help differentiate between numbers like 8 which takes up a lot of pixel space, and 1 which takes up much less. However, this too did not significantly improve results.

The next feature I tested was the pixel total for each row, which is the same as the pixel average for each row. This feature worked very well, with an approximate increase in overall accuracy of about 2%. While the precision and recall scores for some labels did fall, it was only by a small amount compared to the increase in the scores of other labels. I then also tested a fourth feature of the pixel total for each column, which also increased the performance scores but in to a much less degree than the row total. My theory on why this happened is because there is more variation between numbers vertically than horizontally for most numbers.

Next I tried dividing each image into 4 by 4 blocks and taking the pixel total of each block as a feature. This did result in a slight increase in the performance metrics, but not by much. I then tried the same feature but with 2 by 2 blocks, 7 by 7 blocks (dividing the image evenly into 16 blocks) and 14 by 14 blocks(cutting the image in fourths). 14 by 14 blocks performed much worse, and 2 by 2 performed about as well as 4 by 4. However, 7 by 7 blocks actually did improve results by a slight margin, around 0.5%.

**Results**:

The final decision tree using all the best methods I tested was one that used no pruning, no maximum depth, and had hand-engineered features of average row pixels, average column pixels, and average pixels of each 7 by 7 block. This model obtained an overall accuracy of about 78.5% when testing on the testing data set. Below are the results obtained with this tree.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Predicted Labels | | | | | | | | | | | | | | | | | | | | | |
| Real Labels |  | | **0** | | **1** | | **2** | | **3** | | **4** | | **5** | | **6** | | **7** | | **8** | **9** | |
| **0** | | 137 | | 1 | | 9 | | 2 | | 2 | | 2 | | 3 | | 0 | | 5 | 1 | |
| **1** | | 0 | | 160 | | 5 | | 1 | | 4 | | 3 | | 2 | | 4 | | 1 | 1 | |
| **2** | | 2 | | 6 | | 111 | | 5 | | 2 | | 5 | | 7 | | 6 | | 8 | 6 | |
| **3** | | 1 | | 2 | | 9 | | 119 | | 1 | | 13 | | 3 | | 1 | | 11 | 6 | |
| **4** | | 1 | | 1 | | 6 | | 3 | | 129 | | 1 | | 2 | | 3 | | 5 | 11 | |
| **5** | | 4 | | 1 | | 4 | | 9 | | 2 | | 101 | | 6 | | 4 | | 4 | 6 | |
| **6** | | 2 | | 1 | | 9 | | 2 | | 8 | | 2 | | 136 | | 0 | | 3 | 2 | |
| **7** | | 0 | | 0 | | 5 | | 2 | | 3 | | 0 | | 1 | | 151 | | 4 | 5 | |
| **8** | | 1 | | 3 | | 7 | | 11 | | 2 | | 5 | | 6 | | 5 | | 98 | 12 | |
| **9** | | 0 | | 1 | | 3 | | 4 | | 10 | | 6 | | 3 | | 5 | | 1 | 127 | |
| **Label** | | **0** | | **1** | | **2** | | **3** | | **4** | | **5** | | **6** | | **7** | | **8** | | | **9** | | |
| **Precision** | | 0.925676 | | 0.909091 | | 0.660714 | | 0.753165 | | 0.791411 | | 0.731884 | | 0.804734 | | 0.843575 | | 0.7 | | | 0.717514 | | |
| **Recall** | | 0.845679 | | 0.883978 | | 0.702532 | | 0.716867 | | 0.796296 | | 0.716312 | | 0.824242 | | 0.883041 | | 0.653333 | | | 0.79375 | | |

Overall accuracy = 78.5%

**Miss predicted images for decision trees:**



Image 1: Actual = 6, Guessed = 5



Image 2: Actual = 2, Guessed = 7



Image 3: Actual = 0, Guessed = 5