Introduction/Background

In this Assignment we programmed two algorithms to predict the value of a number in a picture. We also made a decision tree for a small data set using the Interactive Dichotomizer 3 algorithm. For both the training and testing data, the two algorithms used pictures of numbers that were translated into vectors, whose coordinates correspond to grayscale pixel values. The first algorithm, called K Nearest Neighbors, took a vector from the testing data set as input, and determined which vectors in the training data were closest to it. The value that was most frequent among the k nearest neighbors was determined to be the value of the inputted vector. The second algorithm, called K Means Clustering, picked k random points in the training set as cluster centers and assigned all other points in the training set to the cluster of the point it was closest to. Then new cluster centers were calculated based on the average position of the points in a cluster, and once again the points were re-assigned to the closest cluster centers. This process was repeated until center points stopped moving, or until 50 iterations has passed.

Results for K Nearest Neighbors

I ran the algorithm on the data set for k values 1, 3, 5, and 7. Here are the results:

k value	Error rate	Accuracy
1	309/10000	96.91%
3	295/10000	97.05%
5	312/10000	96.88%
7	306/10000	96.94%

According to these tests, the best value of k is 3. Its confusion matrix is the following:

```
>>> display_confusion_matrix()
    |    0     1    2    3    4    5    6    7    8    9 <- detected values
0   |  974     1    1    0    0    1    2    1    0    0
1   |    0  1133    2    0    0    0    0    0    0    0
2   |   10     9  996    2    0    0    0   13    2    0
3   |    0     2    4  976    1   13    1    7    3    3
4   |    1     6    0    0  950    0    4    2    0   19
5   |    6     1    0   11    2  859    5    1    3    4
6   |    5     3    0    0    3    3  944    0    0    0
7   |    0    21    5    0    1    0    0  991    0   10
8   |    8     2    4   16    8   11    3    4  914    4
9   |    4     5    2    8    9    2    1    8    2  968
^inputs
```

We see that sevens were often mistaken for ones. This is probably because of how similar ones and sevens look to each other. Sometimes even being indistinguishable to the human eye. Oddly enough, ones were never mistaken for sevens. Fours were also often mistaken for nines and eight were mistaken for threes. Both of these can also be explained by the visual similarities between the said numbers, making the potential for overlapping pixel values high.

We see that the nearest neighbor is not always the correct answer, however the most frequent among a large group of nearest neighbors is not necessarily the most accurate method either. An avenue worth exploring would be to assign weights to the nearest neighbors. For example, the nearest neighbor could have value 10, the second nearest could have value 7, the third nearest could have value 5, etc. Many weighing methods could be explored to find the optimal one. The goal of this method would be to give priority to more similar vectors, while allowing potential errors to be overruled by a large amount of near neighbors that predict a different value.

Results for K Means Clustering

I ran the algorithm on the data set for k values 3, 6, 10, and 20. Here are the results:

k value	Sum of Squared Errors	Average Distance From Center	Purity
3	4446955202	1725	0.22894
6	4324374238	1635	0.32646
10	4272901345	1579	0.37535
20	4970413578	1484	0.41464

The Sum of Squared Errors is an internal metric that gave us a number that relates to the distance between a vector and the center point of the cluster it is in. For some reason this number went down as the number of clusters increased. This could mean that the algorithm doesn't work well for large amounts of clusters, or that I wrote a function that calculated it incorrectly.

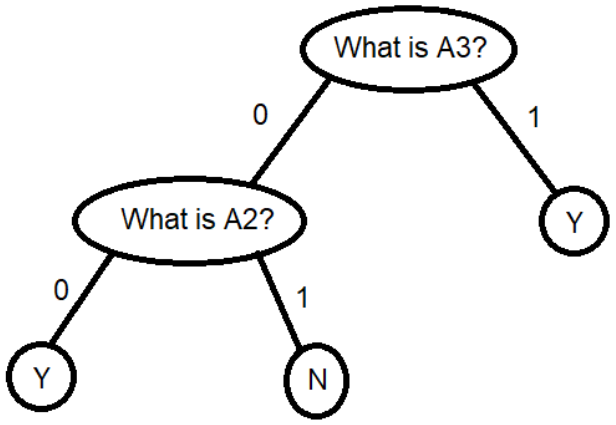
The Average Distance From Centers is also an internal metric that calculated how far every point is from the center of its cluster and takes the average value for every point's distance. As expected, we see that this metric went down as we increased the amount of clusters. This is because the more clusters there were, the smaller the distance between its center and all its points.

Purity is an external metric that described the variety within each cluster. Specifically, clusters that were dominated by one value got a high score, and those that were diverse got a low score.

If we want to use K Means Average to predict number values in pictures, it is important the data can be organized into clusters, whose vectors are of the same value. Diverse clusters make for indistinguishable number vectors. That is why I believe that purity is the most useful metric to test on this algorithm before determining if it should be used to predict number values.

Results for the decision tree

Using the ID3 algorithm on the data set in the assignment produced the following decision tree:



The calculations used to discover it are on the next page.

I do not think it would be feasible to use a decision tree model for the digit OCR task from Problem 1. There are not enough good ‘yes or no’ questions to ask about the vectors to determine the number they represent. I thought about making an algorithm that asks something like: “Is the value of this coordinate greater than 200?”, but this kind of algorithm would need the numbers of the same value to be very similar to one another, and would completely fail on numbers that are not centered properly. Also, there would be so many questions to ask, that it would either take too long to make the tree, or would require a lot of human intervention. Over all, there seem to be easier and faster ways to solve the task.

A different image classification task that would be well-suited to a decision tree be one that turns each number into a graph with edges and vertices along the written numbers. It could see if a number contains a closed off space and immediately determine that it must be a 0, 4, 6, or 9, and if it has two closed off spaces than it must be an 8. It could look at how many horizontal, vertical, and diagonal lines a number has, and in what relation to each other they are, to narrow it down even further.

```
Is A1 equal to 1?
A1 = 1: Y on 3 out of 4 | A1 = 0: Y on 1 out of 2
Entropy = 4/6 * (-3/4 * log(3/4) - 1/4 * log(1/4)) + 2/6 * (-1/2 * log(1/2) - 1/2 * log(1/2))
        = 4/6 * 0.811278 + 2/6 * 1
        = 0.874185

Is A2 equal to 1?
A2 = 1: Y on 2 out of 4 | A2 = 0: Y on 2 out of 2
Entropy = 4/6 * (-2/4 * log(2/4) - 2/4 * log(2/4)) + 2/6 * (-2/2 * log(2/2) - 0/2 * log(0/2))
        = 4/6 * 1 + 2/6 * 0
        = 0.666667

Is A3 equal to 1?
A3 = 1: Y on 3 out of 3 | A3 = 0: Y on 1 out of 3
Entropy = 3/6 * (-3/3 * log(3/3) - 0/3 * log(0/3)) + 3/6 * (-1/3 * log(1/3) - 2/3 * log(2/3))
        = 3/6 * 0 + 3/6 * 0.918296
        = 0.459148


Pick question A3
A3 = 1: Leaf Node
A3 = 0:

Is A1 equal to 1?
A1 = 1: Y on 1 out of 2 | A1 = 0: Y on 0 out of 1
Entropy = 2/3 * (-1/2 * log(1/2) - 1/2 * log(1/2)) + 1/3 * (-0/1 * log(0/1) - 1/1 * log(1/1))
        = 2/3 * 1 + 1/3 * 0
        = 0.666667

Is A2 equal to 1?
A2 = 1: Y on 0 out of 2 | A2 = 0: Y on 1 out of 1
Entropy = 2/3 * (-0/2 * log(0/2) - 2/2 * log(2/2)) + 1/3 * (-1/1 * log(1/1) - 0/1 * log(0/1))
        = 2/3 * 0 + 1/3 * 0
        = 0


Pick question A2
A2 = 1: Leaf Node
A2 = 0: Leaf Node
```