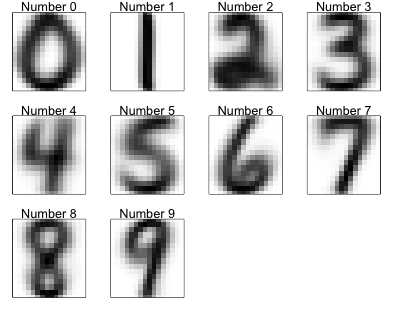
KNN Implementation Analysis

About

In the following report, we will be implementing a classifier that identifies the nine hand written digits (0-9) through statistical computing. We will be using the k-Nearest Neighbor (k-NN) method for pattern recognition and the cross-validation examination to determine error rates of our implementation. k-NN is a non-parametric method meaning that it makes no assumptions about the size of the data and does not require the data to fit the Gaussian distribution but rather relies on ranking of the data. Based on our classifier, the ranking is done by choosing the most frequent numbers surrounding the prediction point and identifies the recurring digit as the highest rank and therefore predicting the highest rank as the possible handwritten digit. The cross validation is an algorithm that uses a given data set to separate them into m folds choosing a validation set and m-1 training set(s) to estimate the error rates. The algorithm repeatedly uses the validation point to test on m-1 training set(s) until m trials are complete. Nevertheless, we will try to implement these methods and examine the results through statistical analysis and data visualization.

Analysis

In order to read the pixel file, I created a function called “read\_digits” with an argument that accepts a relative path to the file location. Since I’m expecting the digit folder to be all numeric, there was no need to change the data type, so I proceeded to making a data frame. Aside from that, let’s look what the handwritten digits from the data set actually looks. The below image displays the pixelated numbers from 0 to 9.

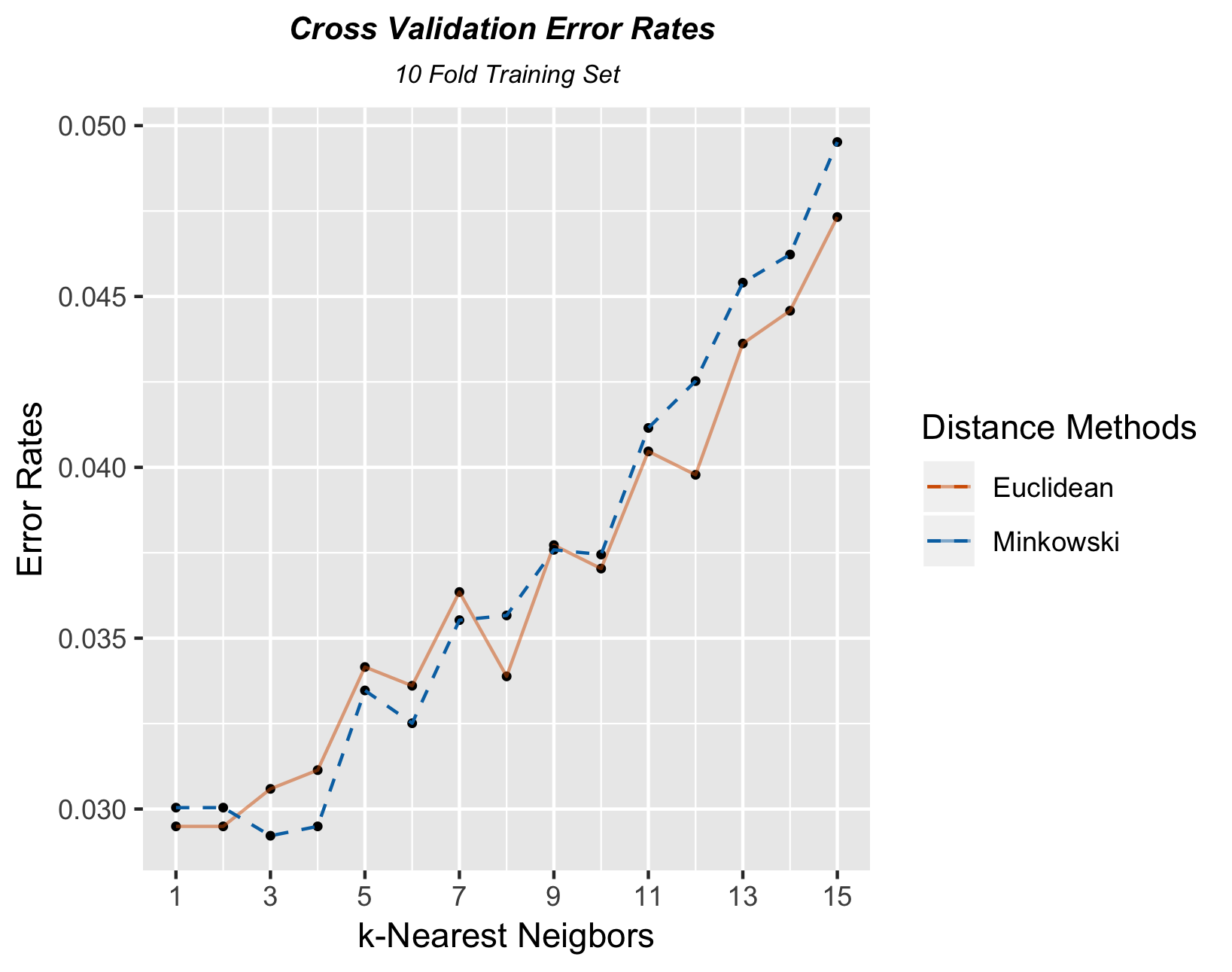


**Pixelated Numbers**

As you can see from the display above, majority of the numbers appears to be digitized, this is because I averaged all the pixels that belongs to the digits throughout the data set. Looking at the pixel values, I believe the most useful pixels are in columns 241 because it has the smallest variance .02 and less useful pixels in 230 with variance of .80. Considering that the pixels ranges from -1 to 1, it is not ideal to have the variance so close to the positive range of the pixels which suggest that column 230 might be responsible the blurry images.

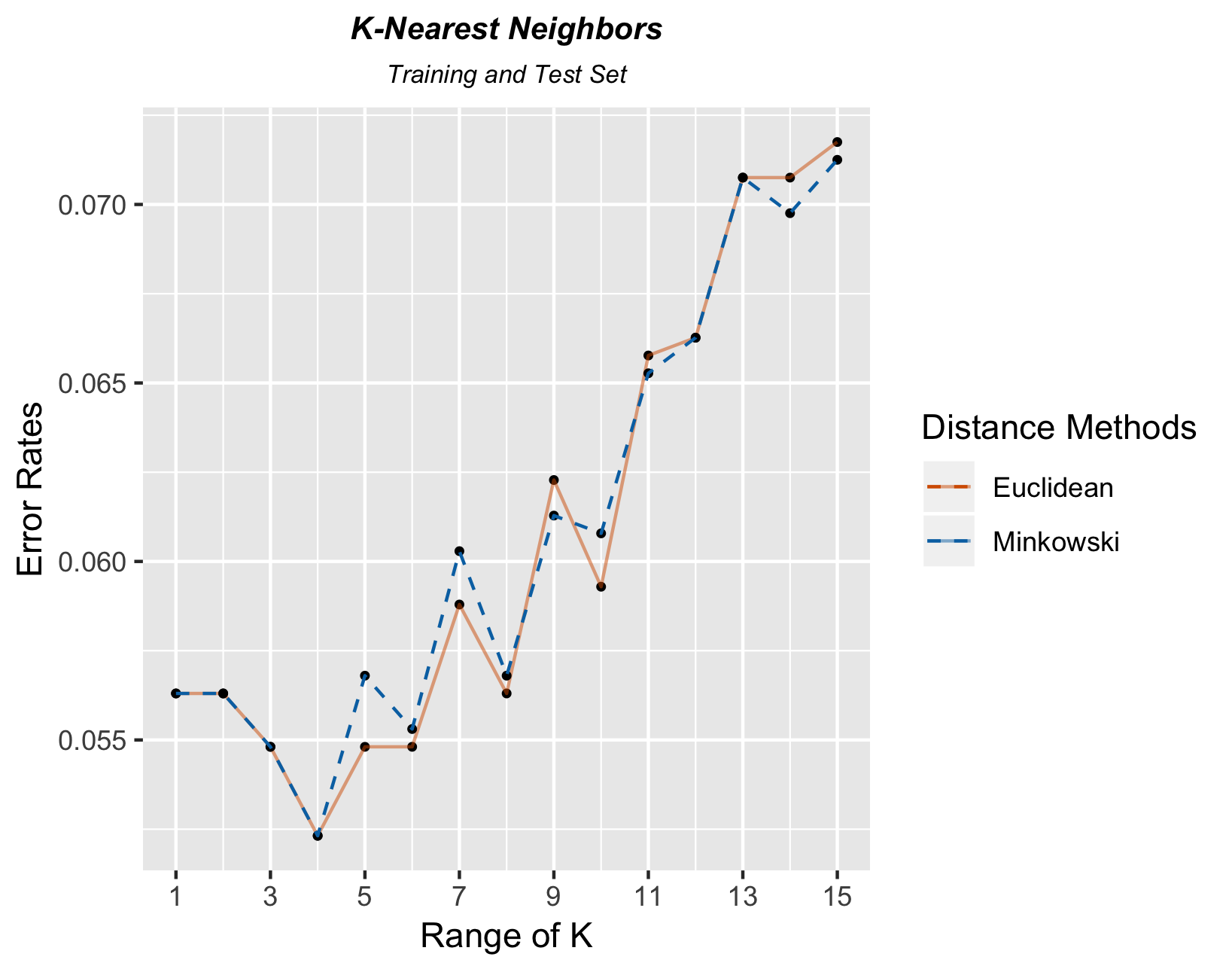
To perform the statistical method that I have mentioned earlier, I created a function called “predict\_knn” that takes in prediction and training points, distance metric, and k as arguments. This function calculates the distance from the prediction points to the training points and search for the nearest neighbors (k) based on the smallest distance and give a prediction number as output. Then I created another function called “cv\_error\_rates” to estimates the accuracy of my predictions. This function has similar arguments as “predict\_knn” and its objective is to randomize a training set and split the set into 10 parts and repeatedly train itself. Since calculating the distance repeatedly is time consuming, with each loop having O(n) time complexity, I had to adjust the distance calculation rather than computing it numerous times in “predict\_knn”. So, I modify the parameters of “predict\_knn” and calculate my distance in “cv\_error\_rates” and passing it to “predict\_knn”. This decreased the time that takes to call on the function, eventually only taking approximately one minute and thirty-seconds to achieve 10 folds cross validation. The draw back from this function is that it will not allow the user to choose the number of folds and that it deletes odd observations. For example, using the training set, there were 7291 observations and splitting it 10 times will result in the last observation being left out.

Nevertheless, after the rearrangement of both functions, I sampled 10 folds with k ranging from 1 to 15 for all possible distances to find the top two methods with the least error rates. I found the best methods with the lowest error rates to be the Euclidean and Minkowski distance with error rates ranging from 3 to 5 percent across the range of k. The below graph is an illustration of the error rates for the two distances.



As you can see there is an increasing trend for both distances as k increase. This means that when there are more neighbors around the prediction points, the predicting algorithm is more likely to choose the incorrect number. Note that since the error rate is low, the prediction accuracy is high. Nevertheless, I think the best k for both distances is 9 because this is where both the Euclidean and Minkowski distance intercept and that the error rate is not too low for high variance. If we were to increase k then kNN tends to choose majority classes and therefore making the model more bias from the Bias-Variance Tradeoff. So, I would not recommend increasing k for this particular cross validation.

Instead of using cross validation to calculate the error rates, I adjusted my kNN function, so it can estimate the error rates without the folds. Below is a graphical illustration that applies the kNN function on both the training and test set.



From the above illustration, you can see that they have the same increasing trend as the cross-validation graphs. However, the error rates from using kNN is much larger than that of cross-validation. This might be because only the training set were used for cross-validation. Nevertheless, both graphs show similar relationship such that as k increases, the likelihood for the algorithm to fail to predict increases. Recall that the best model was using either the Euclidean or Minkowski with k = 9, after analyzing the best fit model, the numbers that the algorithm tends to get wrong are 2, 4 and 8 with over 40 failed predictions for each number. While 0s and 1s are more likely to be predicted correctly with less than 5 miscounts. On another note, normally the kNN is supposed to look like a bowl shaped but both my graphs show an increasing trend instead. Because of the different shape, picking the best k was unorthodox which might not be the best choice for k.

Conclusion

Overall, we have covered the methodology for k-Nearest Neighbors and 10-fold cross-validation. During the examination, we find that calculating the distance repeatedly was not feasible in time and adjusted our functions, so it can better perform. We also compared the graphs of both kNN and cross-validation and found the relationship between two methods and choose the best model along with the best k for this particular data sets. Even though the idea behind kNN and cross validation might appear to be simple, statistical computing them was not an easy task.