While using the k-nearest-neighbors algorithm, the value of k has a great impact on the classification of a specific example within a dataset. Assuming k=1, the algorithm is quite simple and classifies the example the same way as the example closest to it in the training set. An increase in k will therefore increase the number of neighbors involved with classification. Assuming k=5, if the 5 nearest neighbors to the testing example are blue, red, red, blue, and red, then the algorithm will declare the testing example as red because the frequency of red to blue is 3 to 2. Intuitively, using a low k value such as 1 will expose your model to things such as class label noise. Generally, the error rate will be high for low k, then decrease to a minimum and begin to increase once again as k gets too large. This can be shown here:

Line chart

Description automatically generated

In project #3, we’ve been tasked with showing how changing the value of k changes the classification of our data using 3 different sets from the UCI repository. We can do this pretty simply by showing how the error rate changes along with k within each of these sets. This, however, is limited and somewhat uninteresting, so we will have to add more to the discussion later.

Let’s begin with the first dataset, titled Wine. This dataset involves 30 variables to determine whether or not a breast mass is malignant or benign. Here is the head of the data for the first few columns:Table

Description automatically generated

One of the main issues with the base data here is that a few of these columns are magnitudes larger than the rest. Fortunately, this can be rectified easily by using a normalization method. By taking (X-MIN)/(MAX-MIN), the data can be normalized between 0 and 1 as shown here:

Table

Description automatically generated

Now, we are ready to run the k-nearest-neighbors algorithm and plot out the error rate for each specific k value. We can split the data into a training set and a testing set which will be used to test our model. The breakdown we’ve chosen here is 70% training/30% testing. Here is the plot of error rates:

Chart

Description automatically generated

The results found in this graph do in fact mimic the expectation that the error rates would be high for the low k values, followed by a minimum and a subsequent increase as k gets larger. It seems as though the optimal k value here would be around 11. It begs the question, how can we show that the lower values for k (=1,2,3,4,etc.) provide a larger error rate than our optimal k, which we have found to be 11? Unfortunately, it isn’t that simple, and it is impossible to provide a classification space while using 30 different variables, as we cannot internalize a 30-dimensional graph to plot all of our examples. So, for the sake of argument, we have split the data into only 2 columns to show exactly how this algorithm works. We will use the variables ‘radius\_mean’ and ‘compatness\_mean’ for this argument.Chart, scatter chart

Description automatically generated

In the figure to the left, we have plotted the radius mean on the x-axis and the compactness mean on the y-axis. The blue color here represents the malignant cases and the red represents the benign cases. After running the knn algorithm for k=1, this is the classification space that it returns. This is to say that any testing example that falls in the red area will be classified as benign and for the blue, malignant. One interesting thing to note in the case of k=1 is we can see a few red points that appear within the blue area that likely would be under the scope of class label noise. However, with only 1 nearest neighbor, there is no solution to that problem. It is likely that any example in that red blob in the middle of the sea of blue would be misclassified as red.Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

Above we have provided the same graphic but with k=5 and k=11 (optimal k from our whole set analysis). It is pretty easy to see that increasing the k value here has helped to eliminate the noise involved with our first plot.