Josh Brown

Dr Duan

Applied ML

Feb 6, 2021

Project 1 Report

The data set our group chose was that of physical properties of kernels belonging to different varieties of wheat. We chose this because the dataset had a very good spread of data, and the classifications grouped quite nicely when put into the knn. Since we live in Ohio, we all can see farm fields out of our windows, and thought it was semi-interesting. The data was collected by “@databeats” on [Data.world](https://data.world/databeats/seeds), using soft X-ray technology, because it is less expensive and less destructive to the seeds.

Next, here is a statistical summary for the features of each class:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Class 1 | Area | Perim | Compactness | Length Kernel | Length Kernel | Asym Coef |
| Min | 11.23 | 12.63 | .8392 | 4.902 | 2.85 | .7651 |
| Max | 17.08 | 15.46 | .9183 | 6.053 | 3.683 | 6.685 |
| Median | 14.355 | 14.32 | 0.8805 | 5.534 | 3.2435 | 2.5455 |
| Mode | 14.11 | 14.21 | 0.8923 | 5.395 | 3.026 | 2.7 |
| Standard Dev | 1.21570357 | 0.576583067 | 0.016190929 | 0.231508029 | 0.177615541 | 1.173901286 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Class 2 | Area | Perim | Compactness | Length Kernel | Length Kernel | Asym Coef |
| min | 21.18 | 17.25 | 0.9108 | 6.675 | 4.033 | 6.682 |
| max | 15.38 | 14.66 | 0.8452 | 5.363 | 3.231 | 1.472 |
| median | 18.72 | 16.21 | 0.8826 | 6.1485 | 3.6935 | 3.6095 |
| Mode | 17.63 | 16.26 | 0.8763 | 6.285 | 3.594 | none |
| Standard Dev | 1.43949626 | 0.617417801 | 0.015500042 | 0.268191148 | 0.185539128 | 1.18186829 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Class 2 | Area | Perim | Compactness | Length Kernel | Length Kernel | Asym Coef |
| min | 13.37 | 13.95 | 0.8977 | 5.541 | 3.232 | 8.456 |
| max | 10.59 | 12.41 | 0.8081 | 4.899 | 2.63 | 1.661 |
| median | 11.835 | 13.25 | 0.84935 | 5.224 | 2.8345 | 4.839 |
| Mode | 12.7 | 13.05 | 0.8558 | 5.236 | 2.967 | none |
| Standard Dev | 0.723003584 | 0.340195565 | 0.021759634 | 0.1380152 | 0.147516069 | 1.336464928 |

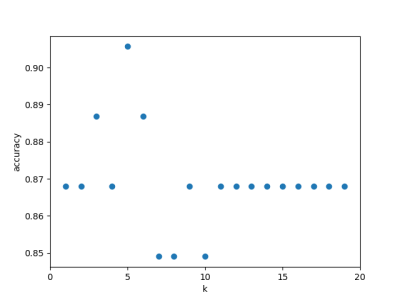
After running the data through the knn, using euclidean distance with 25% of data being test data and 5 neighbors, we had extremely accurate results. With this set of parameters, our network was able to accurately predict the seed over 90% of the time.

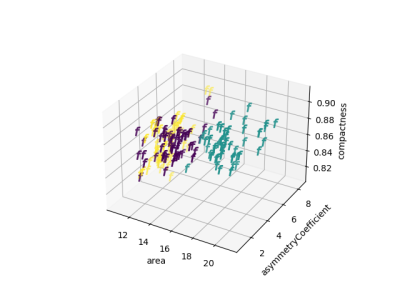
Here is a confusion matrix for the previous results:

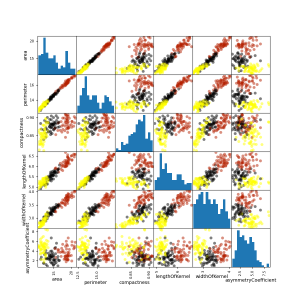
|  |  |  |  |
| --- | --- | --- | --- |
| Predicted \ Actual | Kama | Rosa | Canadian |
| Kama | 14 | 2 | 0 |
| Rosa | 1 | 19 | 0 |
| Canadian | 2 | 0 | 14 |

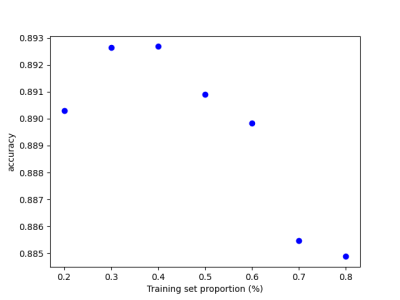
This was obtained by creating by individually testing each slice of the data frame, and appending the actual vs predicted to an array. By examining the array, one can create this matrix by counting the times the knn predicted incorrectly vs the total predictions for each category.

Our network used the euclidean distance metric. This is because we have a low amount of features per class. Euclidean distance becomes unreliable as your feature count increases, and since ours is low it makes sense to use euclidean. To obtain this metric we pass the Minkowski formula to the knn with p = 2. When p=2 the Minkowski metric becomes equal to the euclidean metric.

The K that gave the best performance was 5. This makes sense because each class contained 70 items. When testing the data, each class had ~55 trained points, therefore training with 5 neighbors is only 10% the total of the trained amount of data. By examining more than 10% of the trained data, we are almost guaranteed to get more incorrect predictions because the knn is looking at too many data points in comparison to the total trained. 

The accuracy depended on the training set size by about > 1%. This is because our data set in general is perfect for a knn. Each class uses the same features for measurement, and are very clustered and normalized, with few outliers. Therefore, even when we drastically change the training set proportion, our accuracy stays generally the same. An advantage to a big dataset is that there is more data to compare to, and therefore generally higher confidence when using knn. This comes at the tradeoff however, that larger data sets are slower to work with and take up more storage. Small data sets take up less storage and are faster, however because they are smaller, they will have less data and therefore be generally less reliable. When the training set is smaller, you can train the knn faster, and the knn is smaller in size, but unreliable. This is inverse from when the dataset is large.

Data leakage is when you test a piece of data that has already been used to train. Therefore the knn essentially already knows what it is because it has already seen it. Our knn does not have data leakage for two reasons, first all the data we used is unique. No two objects had the exact same data. Second, when splitting the data in the code, we are careful to not mix training and testing data. That is why testing data never gets used as training data, nor does it ever become recorded into the knn’s memory. 

Lastly, for our knn, we decided to weight each datapoint based on the inverse of the distance from the new point. This way, objects closer to the knn are weighted higher than ones farther away. This gives our knn a massive advantage when working with small training sets, because if it can only see 2 close neighbors but 3 very far ones, it will decide to side with the 2 close data points. This can be easily seen in the graph showing accuracy vs training set proportion.

In conclusion, our data set was an excellent tool for learning about and creating a knn. The data is well organized, and each class is distinguishable from one another, without having outliers. We were able to determine that the best mix of parameters were; 5 neighbors, euclidian, training size = .75, testing = .25, and weight based on distance.