Project 2 - KNN Classification & Regression
Jonathan Klinger

Abstract:
The following project involved testing both KNN (K Nearest Neighbors) classification and regression on 2 datasets each. KNN classification was performed with three different approaches: 1) standard KNN; 2) edited NN (batch method); and 3) condensed NN. On both the Ecoli and Image Segmentation dataset, the best approach varied from run to run but each approach improved in performance with the application of the inverse distance kernel function. For Ecoli the optimal k-value was 6 and for Image Segmentation the optimal k-value was 3. Given these optimal k-values, I was able to acheive an average accuracy score of 90% for the Ecoli dataset and an average accuracy score of 89% on the Image Segmentation dataset using 5-fold cross validation. It is worth noting that for both datasets, some of the validation sets were able to acheive an accuracy of ~95%. For KNN regression, the best results were consistent and not dependent on kernel used in the model. For the Machines dataset, I acheived an R-squared score ~88% and an average deviation from actual value of ~39% at the optimal k-value of 2. For the Forest Fires dataset, the results were far from adequate, acheiving an R-squared of 0.01 and a mean absolute error of 1.15. There was no optimal k-value. After a small bump between k-values 1 to 2, both metrics stay pretty steady as k increases. This shows calculating the mean of the values closest to the query point is almost the same as taking the mean of the entire dataset (could also be using kernel function and weighted average instead of mean).

Problem Statement:
I decided to first try each method using the preprocessed data and the simple model. I used KNN classification for the Ecoli and Image Segmentation dataset and KNN regression for the Machines and Forest Fires dataset. My hypothesis at the start was that the simple models would not be suffecienct to adequately solve each problem. I expected that I would need to use some kernel function to weight the distances. Without the kernel function to weight the distances, I expected that the accuracy would fall sharply as we use a higher k-value. This would be due to the fact that the neighbors can start having extremely high distances from the query sample and with a uniform weighting they would be considered equally with the closer neighbors. I also suspected that the performance would would rise when using edited or condensed nearest neighbors for classification, as opposed to the more computationally complex normal version of KNN.

Implementation of Algorithms:
KNN classification is implemented as follows: data is fed into the predict method. The method gives you the option to use either of the 3 approaches to classification: 1) normal KNN; 2) edited NN; and 3) condensed NN. The features for each test sample are compared with every training sample by calculating the euclidean distance between the two. The euclidean distance between the top k neighbors are weighted by the distance between the test features and the k+1 neighbor. After that, you have the options to weight the distances further using an inverse distance kernel function. If this is chosen, weights are calculated using an inverse distance kernel function and the class that has the highest sum of weights is chosen as the predicted value. Otherwise, the mode of the k nearest neighbors is chosen and the weights are ignored.

KNN regression is implemented as follows: data is fed into the predict method. The features for each test sample are compared with every training sample by calculating the euclidean distance between the two. After that, you have the option to weight the distances using an inverse distance kernel function or a gaussian kernel function. If this option is chosen, the weights are calculated and the predicted value is equal to the sum of each weight multiplied by

the label divided by the sum of weights. Otherwise, the predicted value is computed by taking the mean of the nearest neighbors.

Experimental Approach:
For the KNN classifier, I computed my distances using the euclidean distance fucntion and returned the mode of the k nearest neighbors. The results seemed to vary a lot depending on the value of k. This was especially true for the Image Segmentation dataset. I noticed an issue with the scale of each feature for the Image Segmentation dataset, so I decided to normalize each of the features. This improved my performance on this dataset. Another thing I noticed about this dataset is that the average performance started to decrease once it reached k=4+ neighbors. I figured this had something to do with uniform weighting and the sparse distribution, so I decided to use an approach I found in a paper that weighted the distances and computed the weights for each neighbor using a gaussian kernel.[1] I then took the label that had the max weight from all the neighbors. This smoothed out and improved my performance when iterating over values for k for both datasets.

For the KNN regression I computed my distances using the euclidean distance function and returned the mean of the k nearest neighbors. I tried to implement the gaussian kernel to weight the distance metrics, but that didn't help my results. I assumed this had something to do with the distribution of the data, so I tried using a simpler form of weighting, inverse distance. This kernel also returned similar results. I found it interesting that the regression results were very similar with every kernel I tried (including the basic uniform weighting). This was odd in comparison to added lift that was provided by using the inverse distance function for classification. I did some digging and found a paper that discusses the consistency in knn regression results for all modes of convergence, assuming that the x variables have a distribution and the y variables are bounded.[2]

The approach I took to calculating the optimal k-value was to run the cross validation on 1-20 k-values and plotting the performance. For classification I used accuracy and for regression I used R-squared, Mean Absolute Error (MAE), and average % deviation from the actual. I also computed the mean for each performance metric and plotted that as well, which is where I gather my results from.

Results:

Image Segmentation dataset: for this dataset, taking the simple approach mentioned above, the normal KNN reached peak performance at k= 4 and 8 neighbors. For edited and codensed nearest neighbors, performance was slightly worse and steadily decreased as we increased the number of neighbors (as expected). I then normalized the features, because I saw a difference in the scale of each features range. This approach: 1) improved the accuracy of the same exact model, with a new optimal k-value of 3; 2) smoothed out the results for condensed and edited approaches across k; and 3) performance for condensed and edited increased to roughly the same as the normal KNN. The addition of an inverse distance kernel function improved our performance on all approaches, with a new optimal k-value of 4 and some validation test results reaching 90%+. Edited (optimal k=7), Condensed (optimal k=2), and Normal KNN had very similar performance on average, but the outperformer was typically the Normal KNN approach reaching 89% accuracy.

Ecoli dataset: for this dataset edited NN reached a peak performance at a k-value of 4 and 90% accuracy. These results just slightly outperformed the normal KNN. Condensed NN consistently underperformed the other approaches with a k-value of 3. Adding inverse distance kernel function for weighting the distances acheived similar results for the normal KNN and edited NN and slightly improved results for condensed NN.

Machines dataset: for the machines dataset the simple approach received peak performance at a k-value of 2 with an average R-squared score of 90% and average deviation of 40%. Some of our cross-validation tests reached an average deviation percentage on par with the author's performance (34%). I found that utilizing a gaussian kernel and an inverse distance kernel performed roughly similar to the normal approach for our 5-fold cross validation for this model.

Forest Fires dataset: this dataset was extremely hard to predict. Something I noticed right away is that there was very week correlations between our features and our predicted values, but out of all of them, month and DMC has the strongest predictive value. Using a simple KNN regression I received peak performance with an R-squared score of 0.01 and a mean absolute error of 1.15. After adding a gaussian and inverse distance kernel, the results were roughly the same. A low r-squared value shows that our features don't explain much of the variance in our dataset, which I beleive is evident by the low correlations and the massive imbalance in our dataset (~48% of y values are 0). There was no optimal k-value. After a small bump between k-values 1 to 2, both metrics stay pretty steady as k increases. I tested this out for k-values > 100 and the MAE and R-squared were roughly the same. This shows calculating the mean of the values closest to the query point is almost the same as taking the mean of the entire dataset. I also tried using some feature selection and isolating just the month and DMC features, but the results remained the same or decreased slightly.

Summary:
I was able to acheive very good results on both of the classification datasets and the machines dataset. The condensed and edit nearest neighbors approaches to filtering down the dataset seem to be very helpful, given the computation time on these small datasets and resulting performance mostly on par with the normal KNN method (aside from condensed on Ecoli). It was interesting to see how weighting the distance metrics gave the classification results a little bit of a bump but not the regression results. I determined through research that there are proofs of strong universal consistency in estimates for all modes of convergence for KNN regression.

Sources:
Weighted k-Nearest-Neighbor Techniques and Ordinal Classification by Schliep Hechenbichler
On the Strong Universal Consistency of Nearest Neighbor Regression Function Estimates by Luc Devroye, László Györfi, A. Krzyzak, and Gábor Lugosi