Programming Project #2

Chris Wells

# Introduction

The purpose of this project is to implement a nonparametric k-nearest neighbor (KNN) algorithm to perform classification and regression for each of the six sample datasets.

After pre-processing I will first tune the models by finding the best-performing k and variance (for regression) values. This will be performed by pulling 20% of the examples from the dataset to use as a “test set” for models with different parameter values and selecting the best-performing model for training.

For k-nearest neighbor classification I will predict the class based on a simple majority vote, but in regression I will be using a Gaussian radial basis function (RBF) kernel to determine the distance of the closest neighbor and predicting the value based on the distance and the neighbor’s value. The predicted value will be labelled ‘correct’ if it is within a certain error threshold of the target.

Since this nonparametric model is more sophisticated than the Null Model predictor used in the previous programming assignment, I expect accuracy scores for each of the datasets to improve in this assignment. I also expect that the accuracy of the final trained model will be comparable to the model’s performance against the validation dataset. Finally, I expect that the more complex datasets (more features) will experience more of a drop-off in test accuracy given their increased risk of overfitting.

# Algorithm

I implemented the KNN algorithm as a Python class with the k-value and variance as class attributes and prediction methods for both classification and regression. Given that the algorithm requires finding the nearest neighbor for each point in the test set the code used to calculate distance must be optimized in order to avoid taking an unacceptably long time with the larger datasets.

## Distance Calculation

For each of the test points I treated the training set as an MxN matrix, with M features and N data points. I then subtracted the test point from each of the N rows of the training set and calculated the Euclidean distance for each row, producing an array of N length with each item corresponding the distance of a training set point to the test set point. I only needed to use Euclidean distance because in pre-processing I had converted categorical and ordinal features to numerical via one-hot encoding and ordinal encoding using code from Project 1.

## Predicting Values

After sorting the points in the training set by distance in ascending order, I then simply extracted the first k points in the list to find the k nearest neighbors. For classification I predicted values by simply counting the most heavily populated class among these k values, but for regression problems I applied an RBF kernel to predict a numerical value.

For each of the nearest neighbors I applied the RBF equation to find a value corresponding to the distance of the neighbor to the test value:

The K value would be 1 if they were in the same location, and approach 0 the further away they are. I selected the neighbor with the highest K value and multiplied it by the neighbor’s target to get an estimate for the test value.

## Scoring and Thresholds

To score classification was simply a matter of recording how many classes were accurately predicted, but for regression I subtracted the predicted value from the actual value and recorded the prediction as ‘correct’ if it was within a certain threshold. I gave this threshold a different value for the different regression data sets below:

* For Abalone the target values are integers over a small range relative to the other datasets (1 through 29). I set the threshold value to 1 label predictions correct only if they matched this integer value. Given the low threshold I expect the accuracy for this dataset’s predictions to be lower than the other two.
* Forest Fires and Machine have much wider ranges of values, so instead of restricting the threshold to a constant value I decided to base the threshold on the standard deviation of the target values. Given a normal distribution if I divide the standard deviation by 4 and use this as the threshold, predictions will be labelled as ‘correct’ when they are extremely close to the target compared to other values.

The calculated threshold values are below:

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Std Dev** | **Threshold** |
| Forest Fires | 63 | 16 |
| Machines | 154 | 38 |

## Hyperparameter Tuning

In order to tune the hyperparameters k and variance, I needed to first decide what range of values to test. Since I was using 5-fold cross validation to check against the validation set 5 different values for each could be tested. For the sake of performance I did not check all 25 combinations of k and variance values – instead I tuned k while using a ‘midrange’ variance value of 5 and then tuned the variance for the best k value.

The K value should be odd in order to break ties and I wanted to make sure that the best performing model had found a ‘sweet spot’, neither selecting the lowest k value nor the highest for all the datasets which could indicate that I needed to use a lower or higher range of values. After some experimentation I settled on k = 5, 10, 15, 20, and 25 as suitable values. Performance values are listed below:

|  |  |  |
| --- | --- | --- |
| **Dataset** | **K** | **Accuracy** |
| **Abalone** | 5 | 37.25% |
|  | 10 | 36.29% |
|  | **15** | **39.16%** |
|  | 20 | 36.29% |
|  | 25 | 37.60% |
| **Breast Cancer** | **5** | **92.86%** |
|  | 10 | 92.86% |
|  | 15 | 93.57% |
|  | 20 | 92.14% |
|  | 25 | 92.14% |
| **Car** | 5 | 83.53% |
|  | **10** | **86.99%** |
|  | 15 | 80.92% |
|  | 20 | 80.06% |
|  | 25 | 79.48% |
| **Forest Fires** | **5** | **88.35%** |
|  | 10 | 87.38% |
|  | 15 | 86.41% |
|  | 20 | 86.41% |
|  | 25 | 87.38% |
| **House Votes** | 5 | 89.66% |
|  | 10 | 88.51% |
|  | 15 | 89.66% |
|  | **20** | **91.95%** |
|  | 25 | 90.80% |
| **Machine** | **5** | **45.24%** |
|  | 10 | 42.86% |
|  | 15 | 45.24% |
|  | 20 | 45.24% |
|  | 25 | 45.24% |

For calculating the variance, I first looked at examples of the RBF curve for different values (Sreenivasa, 2020) and based on this decided that variances between 1 and 10 gave an acceptable range. Split among 5 values this meant variances of 1, 3, 5, 7, and 9. Performance values are listed below:

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Variance** | **Accuracy** |
| **Abalone** | 1 | 37.01% |
|  | 3 | 36.29% |
|  | **5** | **39.16%** |
|  | 7 | 36.29% |
|  | 9 | 37.60% |
| **Forest Fires** | **1** | **90.29%** |
|  | 3 | 88.35% |
|  | 5 | 86.41% |
|  | 7 | 84.47% |
|  | 9 | 87.38% |
| **Machine** | 1 | 45.24% |
|  | 3 | 42.86% |
|  | 5 | 45.24% |
|  | 7 | 47.62% |
|  | **9** | **52.38%** |

# Results

## Pre-Processing

After pre-processing each dataset possessed the following dimensions:

|  |  |  |
| --- | --- | --- |
| **Dataset** | **# Features** | **# Rows** |
| Abalone | 10 | 4,177 |
| Breast Cancer | 9 | 699 |
| Car | 6 | 1,728 |
| Forest Fires | 12 | 517 |
| House Votes | 48 | 435 |
| Machine | 37 | 209 |

## KNN Accuracy

After completing setup and hyperparameter tuning, I ran the algorithm on all the data sets using 5-fold cross-validation. Results are shown below compared to the accuracy scores on the validation set above and the results from Project 1’s Null Model:

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Tuning Accuracy** | **KNN Accuracy** | **Project 1 Accuracy** |
| Abalone | 39.16% | 37.01% ± 1.74% | 16.48% ± 0.00% |
| Breast Cancer | 92.86% | 95.71% ± 1.04% | 65.47% ± 0.00% |
| Car | 86.99% | 83.57% ± 1.05% | 70.04% ± 0.00% |
| Forest Fires | 90.29% | 87.69% ± 3.24% | MSE 3,745.54 ± 5,792.85 |
| House Votes | 91.95% | 93.38% ± 3.71% | 61.50% ± 0.00% |
| Machine | 52.38% | 52.76% ± 7.43% | MSE 20,900.77 ± 11,904.80 |

## Edited KNN Accuracy

Since out of the classification sets, Car had the lowest accuracy I tried running the Edited KNN algorithm would potentially edit out ‘noisy’ values and provide better generalization. Removing nearest neighbors when their classes differed, almost half the training set’s rows were edited out. Re-running Car with the new training set gave:

|  |  |  |
| --- | --- | --- |
| **Dataset** | **KNN Accuracy** | **Edited KNN Accuracy** |
| Car | 83.57% ± 1.05% | 63.87% ± 1.64% |

Since the accuracy of the algorithm with the edited KNN set decreased, it looks like the removed points were not noise but important to understanding the underlying problem.

# Discussion

Across the board, the KNN accuracy proved to be greater than the basic Null Model used in project 1 which returned the most common class or the average regression target.

## Abalone

The accuracy was greatly increased for this dataset, especially given the small threshold value that I am using to determine success.

## Breast Cancer and Car

## The accuracy increased for both of these datasets, though the result wasn’t as dramatic considering their relatively high accuracies for Project 1. I believe the relatively low number of features in both datasets contributed to the high and consistent results since this is less vulnerable to overfitting.

## Forest Fire and Machine

## Since I was using MSE to measure accuracy in Project 1 this isn’t a straight comparison, but given the one-quarter standard deviation threshold used to measure a ‘correct’ prediction for both, it is apparent that the error on the projected area and performance is MUCH smaller and more consistent than calculated by the Null Model.

## Given the much larger number of features in the Machine Dataset and markedly worse performance, I suspect that the model may have overfitted the data in this case.

## House Votes

## This dataset saw the biggest gain in classification accuracy compared to Project 1 which is not surprising given the many vote features that are correlated by party given partisan polarization in US politics over the previous decade (DeSilver, 2022). What surprised me was the relatively high standard deviation that was not present in the other classification sets, possibly due to this dataset not working as well with my stratified partition method. In addition, the data does not appear to have been overfitted which was a worry for me considering the large number of features.

# Conclusion

In this project the K-Nearest Neighbor algorithm has been successfully created and run for the datasets. After pre-processing, the model hyperparameters were ‘tuned’ to give the best performance for each dataset, then trained and tested using 5-fold cross-validation.

The results appear to have been nothing but beneficial, with notable accuracy increases across all the datasets. The tuned models gave similar performance on the validation and test datasets, although number of features was not necessarily correlated with accuracy.

# Works Cited

DeSilver, D. (2022, March 10). *The polarization in today’s Congress has roots that go back decades*. Retrieved from Pew Research Center: https://www.pewresearch.org/fact-tank/2022/03/10/the-polarization-in-todays-congress-has-roots-that-go-back-decades/

Sreenivasa, S. (2020, October 12). *Radial Basis Function (RBF) Kernel: The Go-To Kernel*. Retrieved from Towards Data Science: https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a