AEON WILLIAMS, CS298F2020

K NEAREST NEIGHBOR

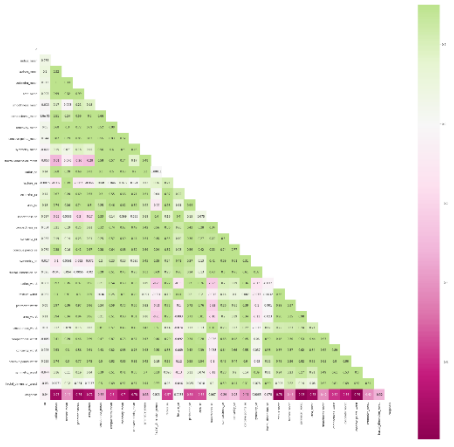
The Code

The model is given a train set and a sample from a test set, with the goal of predicting the class value of the test sample. The distance between each row of the training set and the test sample are calculated, then sorted from smallest to largest – smallest being the most similar, so the best option to choose as a neighbor. If we are doing "normal" KNN, the predicted value is the most frequent class value of the closest k number of neighbors. If we are doing distanced weighted KNN, the predicted value is the most frequent class value of the closest k number of neighbors, multiplied by the sum of 1/distance of each neighbor – this means that if the most frequent class value is from outlier neighbors that aren't actually *that* close, the other possible value is a more accurate prediction.

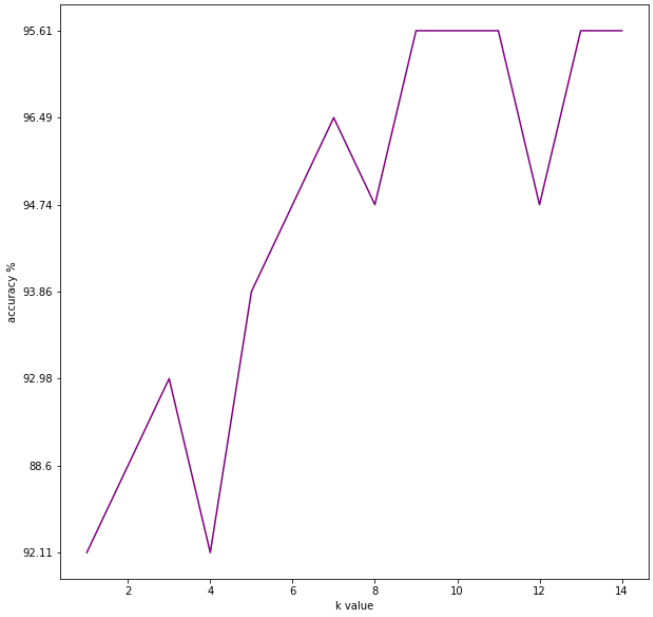
There are helper functions to measure the accuracy of a set of predictions, to normalize the data, and to run the model on an entire dataset, or a specified train and other set.

Data Gathering

Once the dataset is cleaned and normalized, the correlation matrix is displayed as a heatmap, to determine which features we want to use for the predictions. Chosen features are: - concave points_worst, radius_worst, compactness_worst, perimeter_mean, concavity_mean, compactness_mean, radius_se.



The dataset is split into 60% train, 20% validation and test. These sets are tested with a range of k values to determine the optimal value for final predictions. There is a bit of variance when evaluated multiple times, but a k value between 7-10 was consistently the most accurate.



Different splits of train and test were evaluated, with a train split of 0.6-0.8 being the most accurate.

When weighted and normal KNN were compared with the same parameters, there was less than 1% difference in accuracy, but normal KNN was consistently higher.

Final Predictions & Results

For final predictions, k was set to 9, and the split was 60% train 40% test., with very high accuracy results.

```
Accuracy: 93.54%  with k = 9

Confusion Matrix:
 [[145  26]
 [  3 275]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.98      0.85      0.91       171
         1.0       0.91      0.99      0.95       278

    accuracy                           0.94       449
   macro avg       0.95      0.92      0.93       449
weighted avg       0.94      0.94      0.93       449
```

The precision and recall percentages are always above 90%, along with the accuracy. The confusion matrix further shows how little false predictions there are, with most of them being false negatives.

To verify the validity of these results, to ensure they are not because of an overfit model, the model was ran on the Iris dataset, producing similarly high accuracy results.

```
Confusion Matrix:
 [[10  0  0]
 [ 0 13  1]
 [ 0  0  6]]

Classification Report:
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      0.93      0.96        14
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.98      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

Next Steps

Although the model is consistently very accurate, there is still a lot of room for improvement. This could take the form of:

- Analyzing accuracy with different combinations of chosen features
- Analyzing all possible permutations of feature choices, k value, and train/test split to find the absolute best options