Nam Nguyen
SID: 861152982
netID: nnguy072
Late days used: 0
Total late days used so far: 0

Project 2 Report

Question 0

In this part of the project, we were tasked to load in cancer research data. The data had some missing points and I found that the best and probably easiest way to handle this was to simply not include the data points that had missing data. Other things we probably could have done was take the average or do a linear regression (or some kind of regression) to fill in the data. I believe with around 699 data points and only 18 missing it won't make that big of a deal on the points. I think we still have a fair enough sample size to accurately do the calculation and project.

Question 1

This part we had to implement k-nearest-neighbors. To do so I made three functions. One is the main function that calls it called knn_classifier and the others were helper functions. My first helper function was a distance function that was ripped from project 1, the other helper function finds the majority by counting whether the nearest neighbors were a 2 (benign) or a 4 (malignant). The main function knn_classifier calculates the distances between a test point and the training training data. I made a hashtable/dictionary using the key/value pair index/distance. I sorted the distances in ascending order because I want the k closest neighbors. Then I take their index and find the labels from y_train. Then I call the majority function that predicts what the label for that test point might be. This repeats for all the test data points. Below are some pictures of the results

```
ucrwpa-4-1-10-25-98-218:K-Nearest-Neighbors nam$ python classifier.py
What do?
1. K-Nearest Neighbors
2. K-Fold Cross-Validation
1
Enter Lp P value: 2
Enter K value: 3
Doing knn classifier...
[2 2 2 2 4 2 2 4 4 4 4 2 2 4 2 2 2 2 2 2 4 4 2 2 2 4 2 4 2 4 4 4 2 4 2 2 2
 2 2 2 2 2 2 4 4 4 2 2 4 2 2 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 4 2 2
 4 2 2 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 2 2 4 4 2 2 2 2 2 2 2 2 2 4 4 4 2 2
 2 2 2 2 2 2 2 2 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4]
```

```
ucrwpa-4-1-10-25-98-218:K-Nearest-Neighbors nam$ python classifier.py
What do?
1. K-Nearest Neighbors
2. K-Fold Cross-Validation
1
Enter Lp P value: 1
Enter K value: 3
Doing knn classifier...
[2 2 2 2 4 2 2 4 4 4 4 2 2 4 2 2 2 2 2 2 4 4 2 2 2 4 2 4 2 4 4 4 2 4 2 2 2
 2 2 2 2 2 2 4 4 4 2 2 4 2 2 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 4 2 2
 4 2 2 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 2 2 4 4 2 2 2 2 2 2 2 2 2 4 4 4 2 2
 2 2 2 2 2 2 2 2 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4]
```

```
ucrwpa-4-1-10-25-98-218:K-Nearest-Neighbors nam$ python classifier.py
What do?
1. K-Nearest Neighbors
2. K-Fold Cross-Validation
1
Enter Lp P value: 2
Enter K value: 1
Doing knn classifier...
[2 2 2 2 4 2 2 4 4 4 4 2 2 4 2 2 2 2 2 2 4 4 2 2 2 4 2 4 2 4 4 4 2 4 2 2 2
 2 2 2 2 2 2 4 4 4 2 2 4 2 2 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 4 2 2
 4 2 2 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 2 2 4 4 2 2 2 2 2 2 2 2 2 4 4 4 2 2
 2 2 2 2 2 2 2 2 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4]
```

I notice that they're all pretty much similar.

Question 2

In this part, we want to do knn with cross validation. Before anything I shuffle the data using numpy's shuffle function. Then I split the data into k equal sections. Then I iterate through each section using that as the test set and the rest as the training set. I pretty much do knn for each test data point. After I calculate some metrics such as accuracy, sensitivity, and specificity using the formulas from the slides. And finally I print out the data with an error bar (I used standard deviation for the error bars). Below are the results…

Accuracy/Sensitivity/Specificity for P = 2, K = 1

Accuracy/Sensitivity/Specificity for P = 1, K = 1