Kevin Eibeck

861227549

Late days used for this assignment = 0

Total late days used = 0

Assignment 2

Change line 5 in the code to the correct filepath

# Question 0: Getting real data

Approach: I use the readlines method followed by the split method to parse the data. If the data doesn’t contain a question mark it is added, otherwise it is replaced with the average of that column. Then I use the numpy delete function to create the data point x feature matrix. As well as indexing to create the label vector. Both the matrix and vector are global variables so they don’t need to be passed into as parameters.

# Question 1: k-Nearest Neighbor Classifier

Approach:

1. Using indexing I partition the data into an 80/20 split of training and test data points. I then pass these into the knn\_classifier function.
2. Inside the knn\_classifier I first iterate through every data point in x\_test, for each datapoint I find the k closest neighbors using the distance function function.
3. Then I put the highest occurring label as the corresponding y\_test for the x\_test data point.
4. After the knn\_classifier returns knn\_test, I iterate through it. Counting the number of times y\_test equals knn\_test, the number of times knn\_test correctly predicted malignant cancer, and the number of times knn\_test correctly predicted benign cancer.
5. Finally I print out the counts divided by their respective denominators to find accuracy, sensitivity, and specificity.

Results:

Accuracy: 0.985714285714

Sensitivity: 0.942857142857

Specificity: 1.0

# Question 2: Evaluation

## Cross Validation

Approach:

1. Inside of a for loop I used the numpy delete method as well as the splice function to partition the data set into the training and test sets.
2. Then I call the knn\_classifier function. Then I compare the results with y\_test. I append the accuracy, sensitivity, and specificity results onto a list for each respective performance metric. Finally I add the mean and standard deviation for each metric to a the global variable called knn\_results

## Evaluating k-NN

Approach:

1. I created a grapher function which would call the cross validation method each time it was needed.
2. Inside the grapher function is a triple nested for loop. The first being for each metric, so that it can create a subplot for the errorbar graph. The second for loop is for p = 1 and p = 2, so that they have different keys on the graph. The final for loop is for k = 1:10 neighbors.
3. Inside the k for loop is a call to a helper function which returns a tuple of the mean and standard deviation for the corresponding metric, p, and k. The mean and standard deviation are appended to their corresponding list. Which are used later for the graph
4. The helper function checks if the crossValidation function has been called for the parameters passed in (k,p) and either calls the crossValidation function or doesn’t. Then returns the tuple
5. For the evaluation, their is a vector called optimal which has a length of 20. Optimal stores the sum of the mean of each metric for the corresponding p (which is the 10’s place) and k (which is the 1’s place).
6. Outside of the nested for loop is an if statement which takes the sum of the first half of optimal and compares it to the second half. The higher one is the better p.
7. There is another for loop that compares the sum of each k with p =1 and 2, against each other and placing the highest yield and index into the variable called best\_neighbor. Finally giving you the k that gives the best yield.
8. There is another for loop which compares each number in optimal to give you which combination gave the best yield.

Results:

Without randomizing the data:

The best p is 2

The best k is 6





