COSC 425 Project 3
Ryan Pauly
University of Tennessee Knoxville

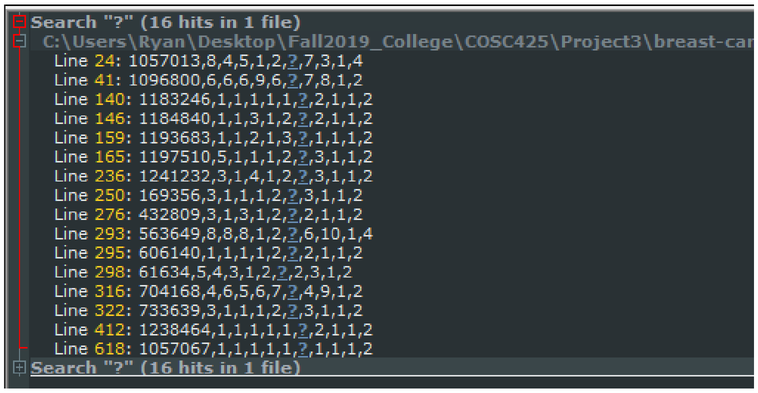
Project Description

The goal of project 3 is to accurately make classification predictions based off cell information relating to breast cancer. With a classifier noted as a numeric 2 and 4 for benign and malignant respectively. Ideally, the predictions should be reluctant to choose benign as this could potentially lead to a false negative and thus potentially falsely informing a patient that he or she does not have breast cancer. Concurrently, the predictions for this type of dataset are critical and need to perform as optimally as possible as to ensure patient health and safety is not mislead by faulty predictions. Part 1 of this project attempts to predict the classifier with the k nearest neighbors' algorithm, which takes advantage of the Euclidean distance formula to essentially compare a row of cell data to other cell data and their classifiers and using the closest proximity other cell data to determine its most likely classifier value, benign or malignant. Part 2 takes a higher-level approach utilizing a decision tree algorithm which uses impurity measure algorithms to determine the most optimal way to categorize data based off a conditional and thus make more accurate conditionals based off more pure splits.

Pre-processing Steps

The data file 'breast-cancer-wisconsin.data' is a comma separated file with the following features: ID number, clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, mitoses, and finally the classifier. Excluding ID number and the classifier features, all features are discrete with values of 1-10. As mentioned previously, the classifiers are either a 2 or a 4 for benign and malignant respectively.

Overall, the data is very clean as is, but there are a few issues to note. First and foremost, in the bare nuclei column, there were 16 instances where a "?" is put in place rather than a discrete value of 1-10. Rather than fill this information with a potentially incorrect value, any row which had this "?" in place was removed from the data entirely. At first, this was easily unnoticeable, but after attempting to read in the file there were some errors which gave away that some inapplicable values were entered. With a quick search through the data in Notepad++, this was discovered:



```
Search "?" (16 hits in 1 file)
  C:\Users\Ryan\Desktop\Fall2019_College\COSC425\Project3\breast-car
    Line 24: 1057013,8,4,5,1,2,?,7,3,1,4
    Line 41: 1096800,6,6,6,9,6,?,7,8,1,2
    Line 140: 1183246,1,1,1,1,1,?,2,1,1,2
    Line 146: 1184840,1,1,3,1,2,?,2,1,1,2
    Line 159: 1193683,1,1,2,1,3,?,1,1,1,2
    Line 165: 1197510,5,1,1,1,2,?,3,1,1,2
    Line 236: 1241232,3,1,4,1,2,?,3,1,1,2
    Line 250: 169356,3,1,1,1,2,?,3,1,1,2
    Line 276: 432809,3,1,3,1,2,?,2,1,1,2
    Line 293: 563649,8,8,8,1,2,?,6,10,1,4
    Line 295: 606140,1,1,1,1,2,?,2,1,1,2
    Line 298: 61634,5,4,3,1,2,?,2,3,1,2
    Line 316: 704168,4,6,5,6,7,?,4,9,1,2
    Line 322: 733639,3,1,1,1,2,?,3,1,1,2
    Line 412: 1238464,1,1,1,1,1,?,2,1,1,2
    Line 618: 1057067,1,1,1,1,1,?,1,1,1,2
Search "?" (16 hits in 1 file)
```

Figure 1. Notepad++ file with CTRL+F search to find "?" values in the data file.

The other issue is less severe and easily avoidable depending on how the data is used. This was the ID number, the first feature of the data. The ID number feature has no application or significance on the cell data.

Once the data is cleaned, the data is then shuffled at random and partitioned in percentage amounts. Training is the first 70%, then validation is the next 10%, and the remaining 20% of the shuffled data is placed in testing. A 70/10/20 partition is a standard setup for training, validation, and testing respectively.

Description

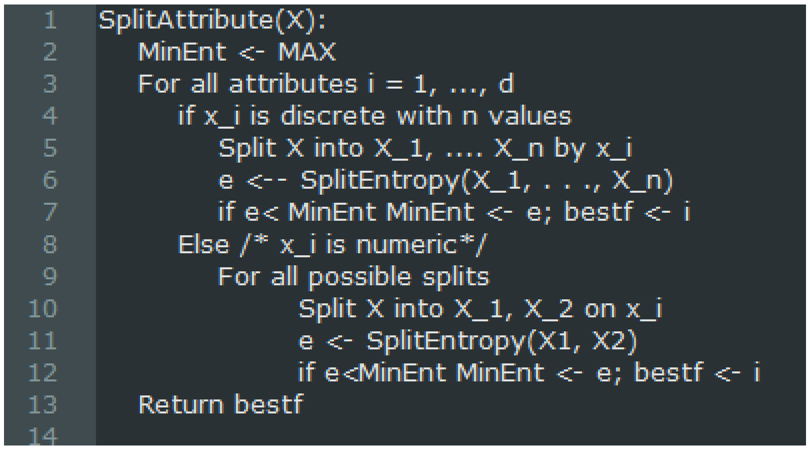
Part 1 of the project utilizes a k-nearest-neighbors algorithm approach to this classifier prediction problem. In the k-nearest-neighbors algorithm, a Euclidean distance is calculated between features of a row exhaustively against all training rows and then these Euclidean distances are sorted in ascending (smallest to largest: i.e., shorter distances are at the beginning and further distances at the end of the sorted list) order, and finally the predictions are made based off first k elements to use from the sorted distances. Then, from those first k elements and their predictions, find the most common classifier and that becomes the final prediction for this row of data.

An essential component to the k-nearest-neighbors algorithm is the Euclidean distance formula, or simply the distance formula in the k-nearest-neighbors algorithm. This equation is also used to calculate the minimal intercluster distance (distance between points in different clusters), and the maximal intracluster distance (distance of distinct points within a cluster). The Euclidean distance formula is very straightforward, it simply calculates the distance between two 2-dimensional points. The 2-dimensional Euclidean distance formula can be described by the following equation:

$$distance(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2} \tag{1}$$

In equation 1, p = (p1, p2) and q = (q1, q2). Equation 1 is used to calculate the distance for each feature against the training rows and totaled for the entire row of features and returned as the distance. Again, these distances along with their respective indices are saved and sorted based off the distance, where the closest distances to the row is at the beginning and the furthest distance is at the end of the sorted list. Then, pick the first k distances and use their respective indices to find the classifier value of that specific training row and save it as one of the k predictions. Of those predictions, the most common classifier is then chosen to be the final prediction for the initial row of test or validation data passed in. The most common classifier is found by calculating the mode, which is best explained in this project as the maximum of the number of each classifier count.

Part 2 of the project uses the decision tree classification algorithm to predict the classifier given cell data. For the decision tree, the data is split apart into a left and right group based on a conditional value, where the left is simply less than the conditional and the right is greater than or equal to the conditional. The conditional is found by exhaustively checking each individual item from each feature and row at a time. Based off the conditional, any item within the feature that is less than the conditional has its entire row put into the left group and any item within the feature that is greater than or equal to the conditional has its entire row put into the right group. With the two groups, an impurity measure is then calculated, where 0 is a completely pure split and 1 is a completely impure split. Once the impurity measure is found, it is compared to see if it is purer than the previously found impurity from its feature and conditional. The process then continues until the feature and conditional with the best impurity measure (0 being ideal) is found. This process can best be described with pseudo code:



```
SplitAttribute(X):
   MinEnt <- MAX
   For all attributes i = 1, ..., d
      if x_i is discrete with n values
         Split X into X_1, .... X_n by x_i
         e <-- SplitEntropy(X_1, . . ., X_n)
         if e< MinEnt MinEnt <- e; bestf <- i
      Else /* x_i is numeric*/
         For all possible splits
               Split X into X_1, X_2 on x_i
               e <- SplitEntropy(X1, X2)
               if e<MinEnt MinEnt <- e; bestf <- i
   Return bestf
```

Figure 2. Psuedo Code for splitting data for the decision tree using an Entropy impurity measure.

In figure 2, the impurity is calculated with the Entropy impurity measure equation, but this is just one way to calculate impurity. For a two-class problem (2 and 4) such as this, p can represent one classifier fraction and 1-p is the other classifier. Entropy is defined with the following equation:

$$\phi(p, 1-p) = -P \cdot \log_2(p) - (1-P) \cdot \log_2(1-P) \quad (2)$$

The next impurity equation is the Gini index, and it is defined as the following:

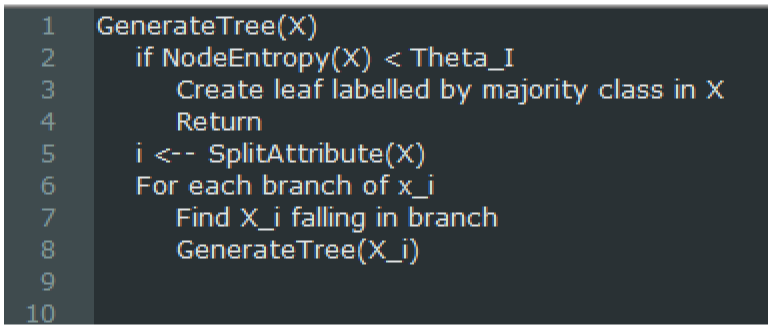
$$\phi(p, 1-p) = 2p(1-p) \quad (3)$$

Finally, the last impurity measure is the misclassification error which is simply the following:

$$\phi(p, 1-p) = 1 - max(p, 1-p) \quad (4)$$

Equations 2, 3, and 4 are three ways to calculate the impurity and each are used in project 3 part 2.

Once the best conditional value to split the data is found from the purest impurity value, the split data then must be put into the decision tree as a node for its contents to be split and used to make predictions. The following pseudo code creates and builds the decision tree:

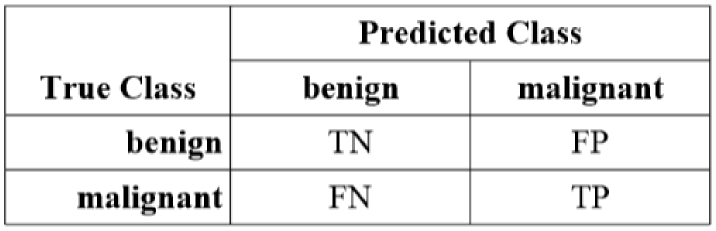


```
GenerateTree(X)
   if NodeEntropy(X) < Theta_I
      Create leaf labelled by majority class in X
      Return
   i <-- SplitAttribute(X)
   For each branch of x_i
      Find X_i falling in branch
      GenerateTree(X_i)
```

Figure 3. Decision Tree construction for classification in pseudo code.

In figure 3, the tree is generated with recursion, building down the left or right branch until a leaf node is created. For prediction, the tree would then be accessed and used to compare to the data traversing the tree, calculating its predictions based off the classification that the leaf nodes predict based off the most common classifier present in its leaf rows. Therefore, a pure split is desirable as the likelihood of the leaf's prediction being accurate increases the purer it is.

Both parts of project 3 have several metric deliverables. With respect to the benign and malignant results and the actual results of the two classifiers: TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives, this all is best represented in a confusion matrix as defined as the following:



	Predicted Class	
True Class	**benign**	**malignant**
benign	TN	FP
malignant	FN	TP

Figure 4. Confusion Matrix from the project 3 pdf.

Next the metrics that are needed for each prediction of the project: First the accuracy equation, which is very straightforward is defined below:

$$\text{Accuracy} = (\text{TN} + \text{TP}) / (\text{TN} + \text{TP} + \text{FN} + \text{FP}) \tag{5}$$

The next equation is the true positive rate, another straightforward equation:

$$TPR = TP/(TP + FN) \tag{6}$$

Further, the next equation is the positive predictive value:

$$PPV = TP/(TP + FP) \tag{7}$$

Then, the true negative rate, another simple equation:

$$TNR = TN/(TN + FP) \tag{8}$$

Finally, the $F_1\ Score$, which is the harmonic mean, a little more complicated than the previous four equations and it is defined as:

$$F_1\ Score = 2 * PPV * TPR/(PPV + TPR) \tag{9}$$

Finally, we wish to compare how standardizing the data compares to non-standardized data. To standardize the dataset, we perform a calculation which scales the variables down significantly. The goal with standardization is to scale the data down to a mean which equals 0 and a standard deviation equal to 1. To achieve this, we use the z-normalization formula:

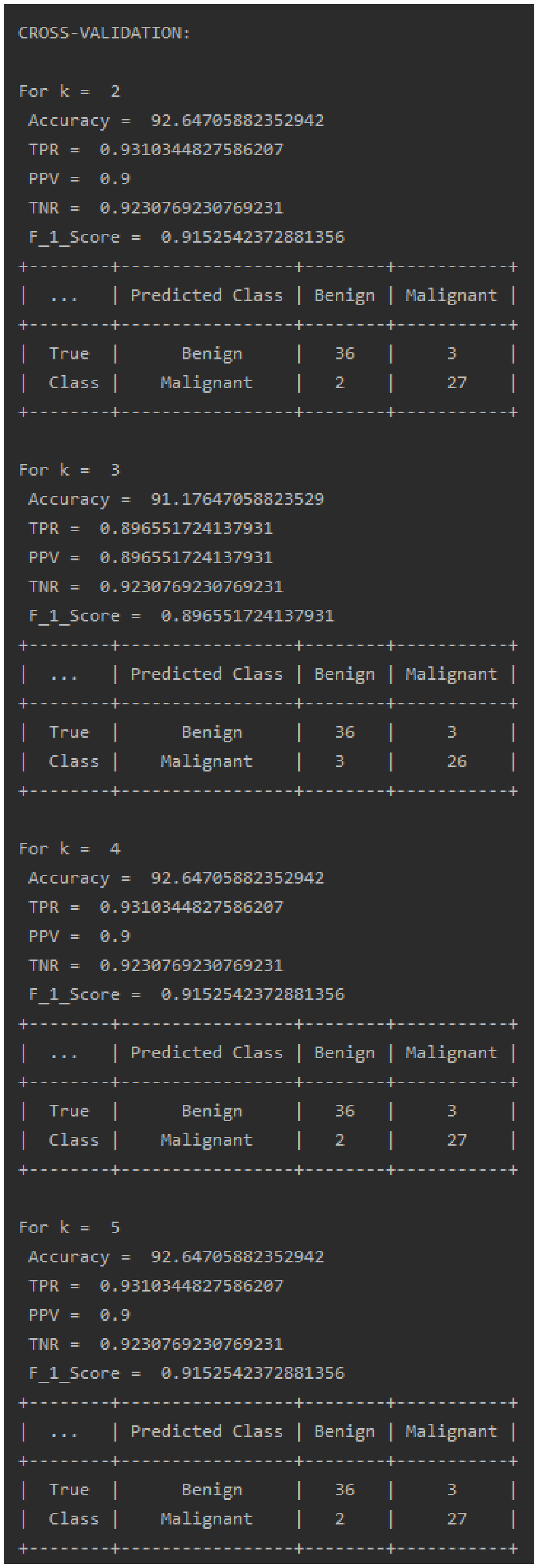
$$z = \frac{X - \mu}{\sigma} \tag{10}$$

X represents the value we wish to standardize, and Z is the result of the standardization. We subtract from X the mean of X or μ, and lastly, we divide it by the standard deviation of X or σ. Note, the data standardization only takes place on the k-nearest-neighbor algorithm.

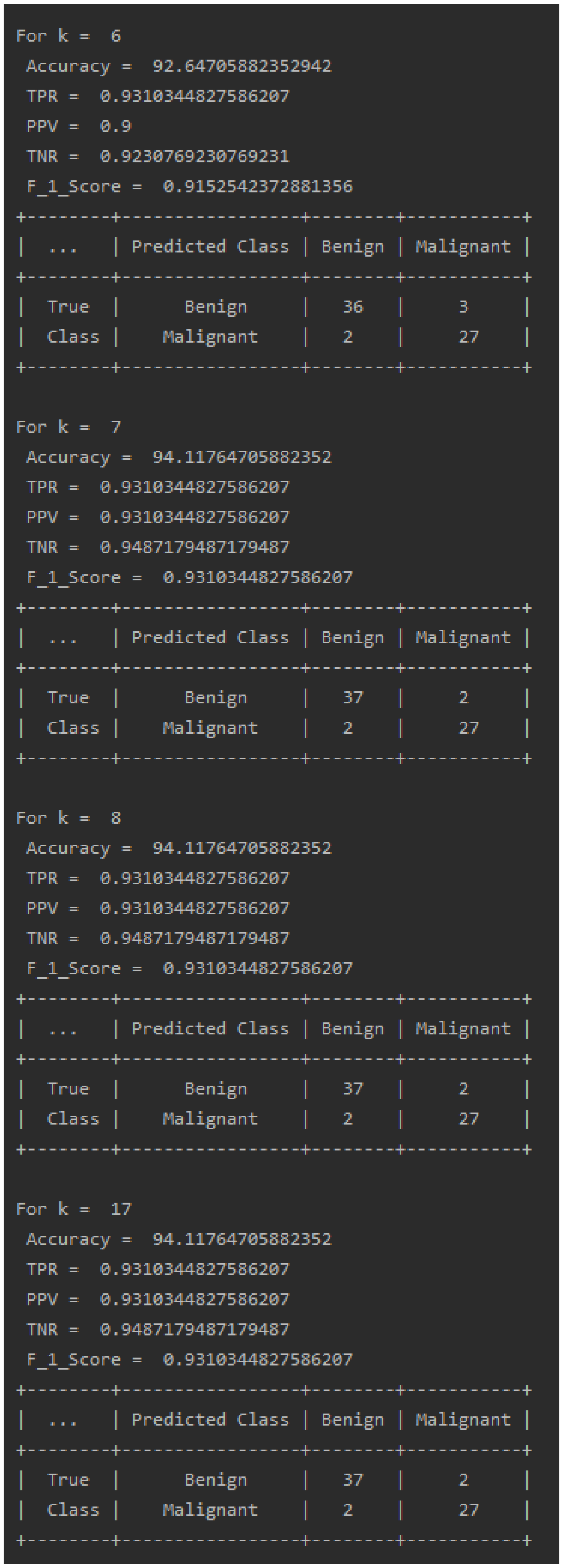
Analysis

Part 1

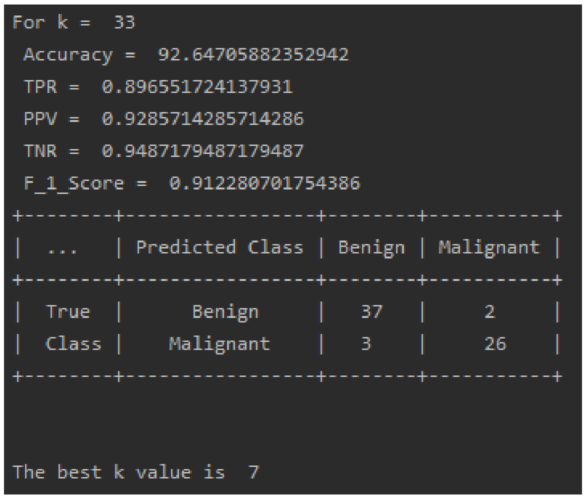
Firstly, the k-nearest-neighbors algorithm of part 1, specifically the cross validation of the un-normalized data. This uses the training data and the validation data to find the most optimal k value for the k-nearest-neighbors algorithm, cross validation. The results of the cross validation on the non-standardized data are on the next page. Each k value is passed in and tested with the training and validation data, printing the required metrics. Accuracy is then compared against all iterations, where the maximum accuracy is saved along with the k value which produced the maximum accuracy.



```
CROSS-VALIDATION:

For k =  2
 Accuracy =  92.64705882352942
 TPR =  0.9310344827586207
 PPV =  0.9
 TNR =  0.9230769230769231
 F_1_Score =  0.9152542372881356
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   36   |     3     |
|  Class |    Malignant   |   2    |     27    |
+--------+----------------+--------+-----------+

For k =  3
 Accuracy =  91.17647058823529
 TPR =  0.896551724137931
 PPV =  0.896551724137931
 TNR =  0.9230769230769231
 F_1_Score =  0.896551724137931
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   36   |     3     |
|  Class |    Malignant   |   3    |     26    |
+--------+----------------+--------+-----------+

For k =  4
 Accuracy =  92.64705882352942
 TPR =  0.9310344827586207
 PPV =  0.9
 TNR =  0.9230769230769231
 F_1_Score =  0.9152542372881356
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   36   |     3     |
|  Class |    Malignant   |   2    |     27    |
+--------+----------------+--------+-----------+

For k =  5
 Accuracy =  92.64705882352942
 TPR =  0.9310344827586207
 PPV =  0.9
 TNR =  0.9230769230769231
 F_1_Score =  0.9152542372881356
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   36   |     3     |
|  Class |    Malignant   |   2    |     27    |
+--------+----------------+--------+-----------+
```

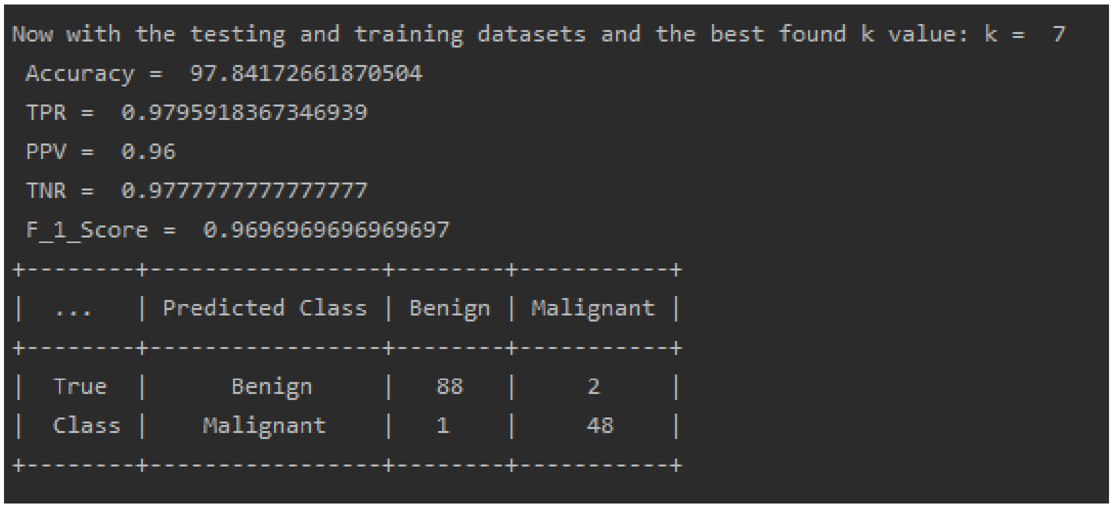


```
For k =  6
 Accuracy =  92.64705882352942
 TPR =  0.9310344827586207
 PPV =  0.9
 TNR =  0.9230769230769231
 F_1_Score =  0.9152542372881356
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   36   |     3     |
|  Class |    Malignant   |   2    |     27    |
+--------+----------------+--------+-----------+

For k =  7
 Accuracy =  94.11764705882352
 TPR =  0.9310344827586207
 PPV =  0.9310344827586207
 TNR =  0.9487179487179487
 F_1_Score =  0.9310344827586207
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   37   |     2     |
|  Class |    Malignant   |   2    |     27    |
+--------+----------------+--------+-----------+

For k =  8
 Accuracy =  94.11764705882352
 TPR =  0.9310344827586207
 PPV =  0.9310344827586207
 TNR =  0.9487179487179487
 F_1_Score =  0.9310344827586207
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   37   |     2     |
|  Class |    Malignant   |   2    |     27    |
+--------+----------------+--------+-----------+

For k =  17
 Accuracy =  94.11764705882352
 TPR =  0.9310344827586207
 PPV =  0.9310344827586207
 TNR =  0.9487179487179487
 F_1_Score =  0.9310344827586207
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   37   |     2     |
|  Class |    Malignant   |   2    |     27    |
+--------+----------------+--------+-----------+
```



```
For k =  33
 Accuracy =  92.64705882352942
 TPR =  0.896551724137931
 PPV =  0.9285714285714286
 TNR =  0.9487179487179487
 F_1_Score =  0.912280701754386
+--------+-----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+-----------------+--------+-----------+
|  True  |      Benign     |   37   |     2     |
|  Class |    Malignant    |   3    |     26    |
+--------+-----------------+--------+-----------+



The best k value is  7
```

With the results of the cross-validation, there seems to be a point of diminishing returns where a k value that is too low can be detrimental to accuracy, but a k that is too large can also worsen the accuracy because of the introduction of noise from considering the classifiers of items from distances much further away. Once the cross-validation is completed the best k value is determined to be 7 with an accuracy of about 94, then with a k of 7 for the final run and the training and testing data sets put in the following outcome is received:

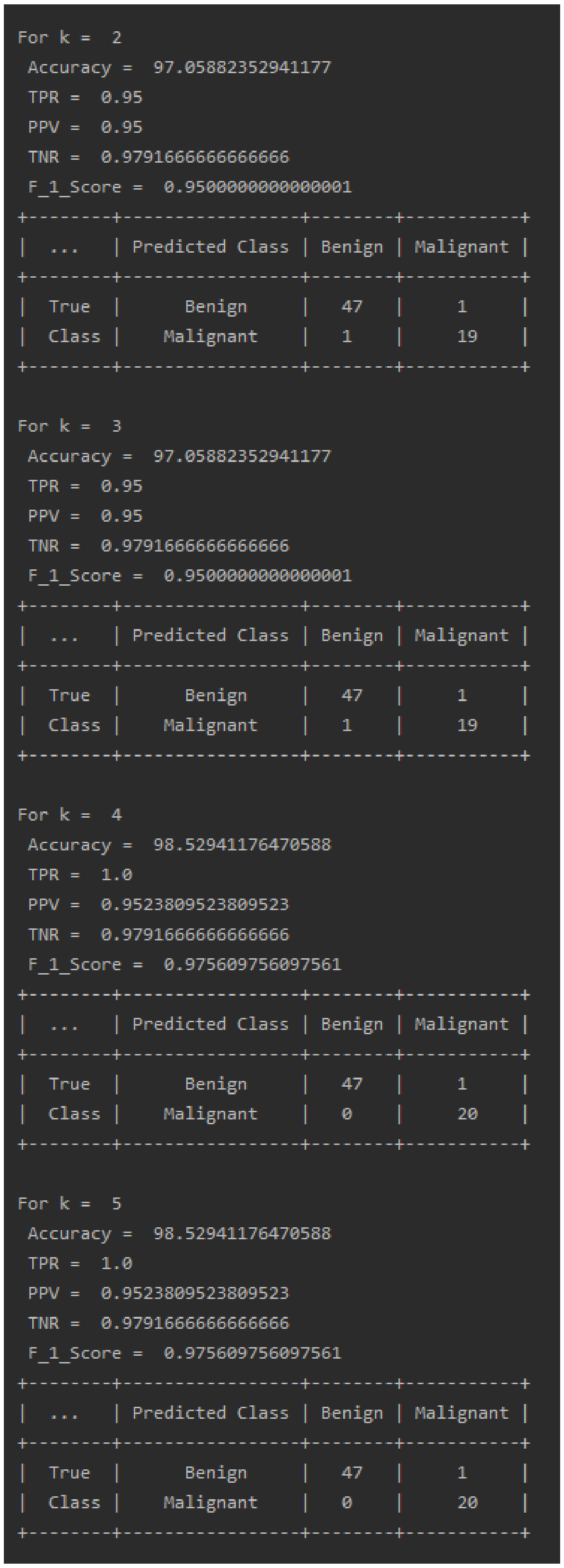
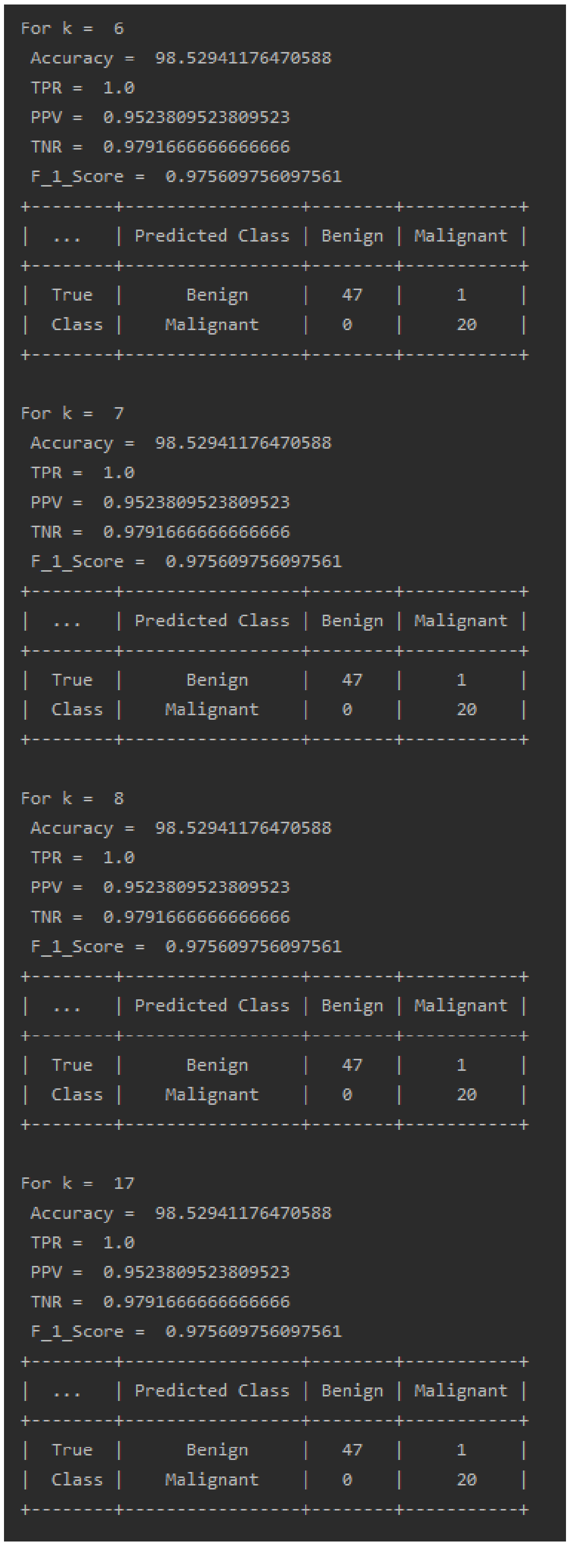


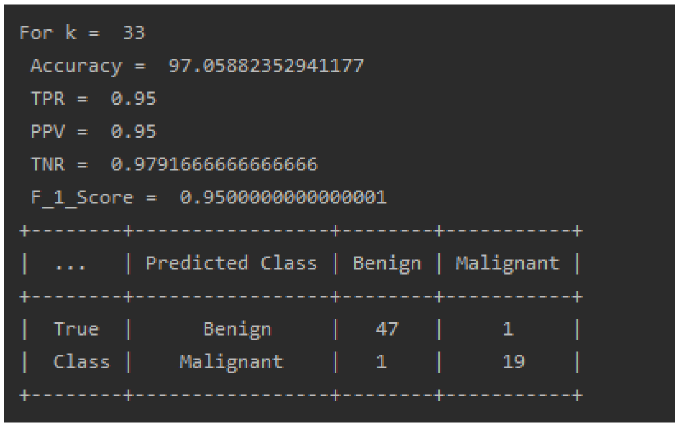
```
Now with the testing and training datasets and the best found k value: k =  7
 Accuracy =  97.84172661870504
 TPR =  0.9795918367346939
 PPV =  0.96
 TNR =  0.9777777777777777
 F_1_Score =  0.9696969696969697
+--------+-----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+-----------------+--------+-----------+
|  True  |      Benign     |   88   |     2     |
|  Class |    Malignant    |   1    |     48    |
+--------+-----------------+--------+-----------+
```

Figure 5.Metrics of the cross-validated optimal k value in use with the testing and training for the knn algorithm.

Thus, the final metric report with the k value found through cross-validation returned an accuracy of roughly 97.84, a respectable value. There was only one false negative case as well, which is better than the number of false negative values returned in all cross-validation metrics.

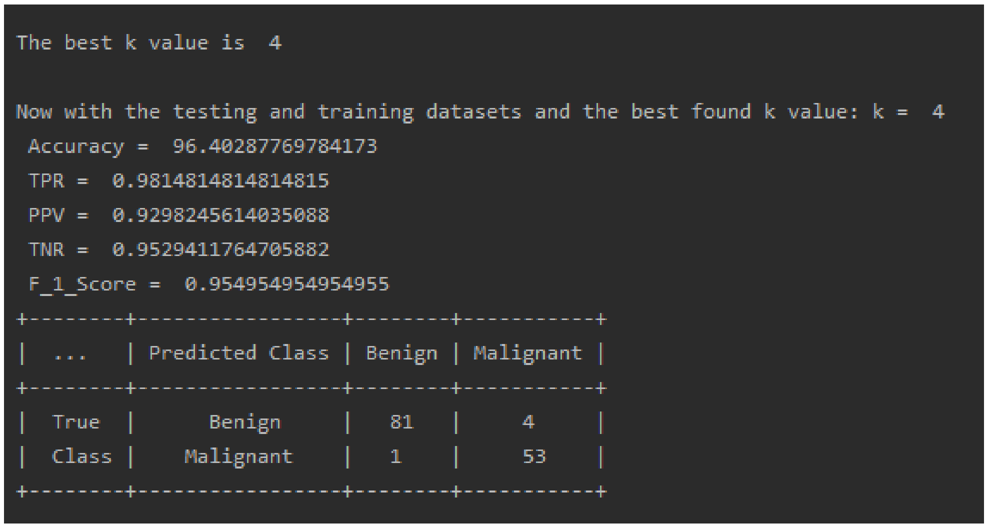
Next, the standardization of the k-nearest-neighbors algorithm. My assumption is that this would change very little, as the data for the cell size is already discrete and standardized between 1 and 10, so the standardization would have minimal change on the outcomes. However, overall, during the cross-validation phase, the accuracies seemed to be much higher than the non-standardized results shown previously. The results of the standardized cross-validation are the following:


```
For k =  2
 Accuracy =  97.05882352941177
 TPR =  0.95
 PPV =  0.95
 TNR =  0.9791666666666666
 F_1_Score =  0.9500000000000001
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   47   |     1     |
|  Class |    Malignant   |   1    |     19    |
+--------+----------------+--------+-----------+

For k =  3
 Accuracy =  97.05882352941177
 TPR =  0.95
 PPV =  0.95
 TNR =  0.9791666666666666
 F_1_Score =  0.9500000000000001
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   47   |     1     |
|  Class |    Malignant   |   1    |     19    |
+--------+----------------+--------+-----------+

For k =  4
 Accuracy =  98.52941176470588
 TPR =  1.0
 PPV =  0.9523809523809523
 TNR =  0.9791666666666666
 F_1_Score =  0.975609756097561
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   47   |     1     |
|  Class |    Malignant   |   0    |     20    |
+--------+----------------+--------+-----------+

For k =  5
 Accuracy =  98.52941176470588
 TPR =  1.0
 PPV =  0.9523809523809523
 TNR =  0.9791666666666666
 F_1_Score =  0.975609756097561
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   47   |     1     |
|  Class |    Malignant   |   0    |     20    |
+--------+----------------+--------+-----------+
```

```
For k =  6
 Accuracy =  98.52941176470588
 TPR =  1.0
 PPV =  0.9523809523809523
 TNR =  0.9791666666666666
 F_1_Score =  0.975609756097561
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   47   |     1     |
|  Class |    Malignant   |   0    |     20    |
+--------+----------------+--------+-----------+

For k =  7
 Accuracy =  98.52941176470588
 TPR =  1.0
 PPV =  0.9523809523809523
 TNR =  0.9791666666666666
 F_1_Score =  0.975609756097561
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   47   |     1     |
|  Class |    Malignant   |   0    |     20    |
+--------+----------------+--------+-----------+

For k =  8
 Accuracy =  98.52941176470588
 TPR =  1.0
 PPV =  0.9523809523809523
 TNR =  0.9791666666666666
 F_1_Score =  0.975609756097561
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   47   |     1     |
|  Class |    Malignant   |   0    |     20    |
+--------+----------------+--------+-----------+

For k =  17
 Accuracy =  98.52941176470588
 TPR =  1.0
 PPV =  0.9523809523809523
 TNR =  0.9791666666666666
 F_1_Score =  0.975609756097561
+--------+----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+----------------+--------+-----------+
|  True  |     Benign     |   47   |     1     |
|  Class |    Malignant   |   0    |     20    |
+--------+----------------+--------+-----------+
```



```
For k =  33
 Accuracy =  97.05882352941177
 TPR =  0.95
 PPV =  0.95
 TNR =  0.9791666666666666
 F_1_Score =  0.9500000000000001
+--------+-----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+-----------------+--------+-----------+
|  True  |      Benign     |   47   |     1     |
|  Class |    Malignant    |   1    |     19    |
+--------+-----------------+--------+-----------+
```

It is possible that the shuffling of the data could have caused the non-standardized cross-validation metrics to be less, but after multiple runs of the program the lower accuracies for non-standardized were consistent. On average the cross-validation results for standardized cell data had a much higher accuracy. Moreover, the false negative count for cross-validated results were also lower, a good thing for the patient that this could potentially help.



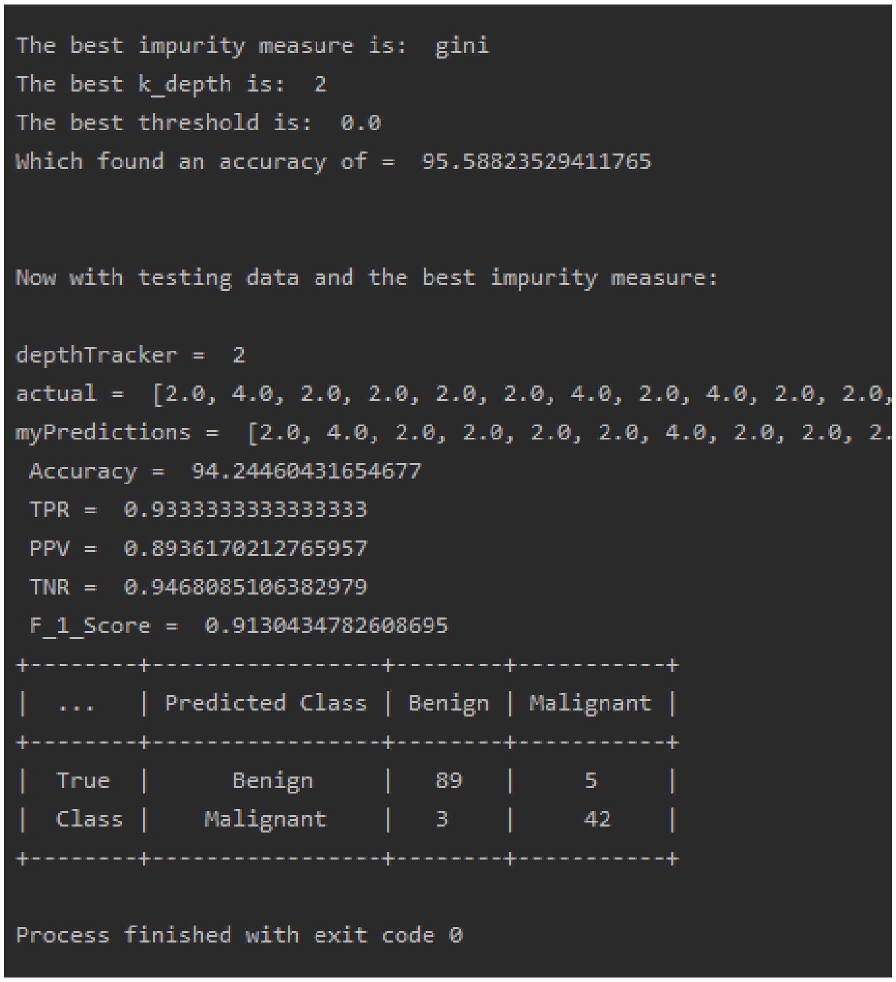
```
The best k value is  4

Now with the testing and training datasets and the best found k value: k =  4
 Accuracy =  96.40287769784173
 TPR =  0.9814814814814815
 PPV =  0.9298245614035088
 TNR =  0.9529411764705882
 F_1_Score =  0.954954954954955
+--------+-----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+-----------------+--------+-----------+
|  True  |      Benign     |   81   |     4     |
|  Class |    Malignant    |   1    |     53    |
+--------+-----------------+--------+-----------+
```

Figure 6. Metrics using the cross-validated k value on the standardized training and testing datasets.

More interestingly, the actual metrics using the cross-validated k value, has an accuracy that is lower than the non-standardized data.

Part 2

To find the best accuracy cross-validation can be performed to determine the best impurity measure, the best maximum k depth, and the impurity threshold. This is performed with three for-loops which evaluate each potential option exhaustively. As one can expect, this takes a long time to perform, but this is most likely due to this approach to cross-validation. Of course, the training data is passed in to create the tree and the validation data is used to evaluate the tree's performance. Once the best impurity measure, best maximum k depth, and best impurity threshold are found, the tree is then constructed and evaluated with the training and testing data sets respectively, joined by the best impurity measure option, the best maximum k depth, and best impurity threshold which yields the following final metrics:



```
The best impurity measure is:  gini
The best k_depth is:  2
The best threshold is:  0.0
Which found an accuracy of =  95.58823529411765


Now with testing data and the best impurity measure:

depthTracker =  2
actual =  [2.0, 4.0, 2.0, 2.0, 2.0, 2.0, 4.0, 2.0, 4.0, 2.0, 2.0,
myPredictions =  [2.0, 4.0, 2.0, 2.0, 2.0, 2.0, 4.0, 2.0, 2.0, 2.
 Accuracy =  94.24460431654677
 TPR =  0.9333333333333333
 PPV =  0.8936170212765957
 TNR =  0.9468085106382979
 F_1_Score =  0.9130434782608695
+--------+-----------------+--------+-----------+
|  ...   | Predicted Class | Benign | Malignant |
+--------+-----------------+--------+-----------+
|  True  |      Benign     |   89   |     5     |
|  Class |    Malignant    |   3    |     42    |
+--------+-----------------+--------+-----------+

Process finished with exit code 0
```

Figure 7. The results of the exhaustive cross-validation and the metric results with the best found parameters and the testing data.

The results are strong, but slightly worse than those found in Part 1 of project 3. This of course may be potentially due to the shuffling of data but could also be due to mistakes in how any of the impurity measures were calculated. An accuracy of 94% is not terrible, but a higher accuracy would be more favorable, especially with critical classification related to diagnosis of breast cancer. There were 3 false negatives reported in the confusion matrix, something that hopefully would be 0.

One of the advantages to decision trees is that because of the conditional comparisons it does not need data standardization/normalization, and again the data in a way is already standardized to some degree as it is discrete and within the bounds of 1-10.

Discussion

This project focuses on accurately predicting whether cells are benign or malignant with respect to breast cancer. Predictions such as these are sensitive, and thus it is paramount that a high accuracy is received in order to not falsely diagnose or falsely misdiagnose, specifically false negatives being the absolute worst possible outcome of a prediction in this instance.

The data was very simple and easy to read in, requiring minimal cleanup to take advantage of it. Locations where inapplicable values were entered, were completely removed as to not potentially inject false information, though I suspect replacing these inapplicable values with an average of the feature would have changed little with only 16 rows being affected by this data entry issue.

The k-nearest-neighbor algorithm predicts a classifier based off the distances of other classifiers from the item that needs to be predicted, where the other items closest to the item to predict are the ones that typically have more accurate classifiers to base the prediction on. These results could be standardized though it made little change to the overall outcome.

To build the classification decision tree, the (classification and regression tree) CART algorithm is essentially the setup of the tree, a tree which divides and conquers data by determining a split via a conditional value which categorizes the data into a left and right group, where left is less than the conditional and right is greater than or equal to the conditional. The conditional is determined through an exhaustive check on each possible value as a conditional which gets the best impurity where 0 is the purest split possible and thus the most desirable impurity value for a conditional to create. Recursively the tree is built starting at the root which splits all the training in half with the most optimal split, and the following children nodes split the already "halved" data and so on until either a threshold, a max depth is reached, or a row amount is within a certain boundary as to not make child nodes when no data to split is remaining. Then, depending on the most common classifier within a leaf node assuming the leaf is impure, is chosen (either a 2 or a 4). The training data is used to build the decision tree, and then either the validation or testing is used to evaluate the tree by following respective paths of the tree based on the contents of the validation or testing data rows passed in. Data standardization was unnecessary as the decision tree is a predictor, and therefore would have little to no impact on performance.

Overall, the project was very informative, especially the decision tree classifier. It was incredibly difficult to create and given more time to tune and perfect the decision tree, I imagine its results would surpass that of the k-nearest-neighbors algorithm.

Sources

Alpaydin, Ethem. *Introduction to Machine Learning*. third ed., MIT Press, 2014.

"Confusion Matrix." *Wikipedia*, Wikimedia Foundation, 22 Oct. 2019, en.wikipedia.org/wiki/Confusion_matrix. Accessed 4 Nov. 2019. https://en.wikipedia.org/wiki/Confusion_matrix.

"Decision Tree Learning." *Wikipedia*, Wikimedia Foundation, 12 June 2019, en.wikipedia.org/wiki/Decision_tree_learning. Accessed 4 Nov. 2019. https://en.wikipedia.org/wiki/Decision_tree_learning.

"Euclidean Distance." *Wikipedia*, Wikimedia Foundation, 5 Sept. 2019, https://en.wikipedia.org/wiki/Euclidean_distance.

"Mode (Statistics)." *Wikipedia*, Wikimedia Foundation, 10 Oct. 2019, en.wikipedia.org/wiki/Mode_(statistics). Accessed 4 Nov. 2019. https://en.wikipedia.org/wiki/Mode_(statistics).

"K-Nearest Neighbors Algorithm." *Wikipedia*, Wikimedia Foundation, 19 Mar. 2019, en.wikipedia.org/wiki/K-nearest_neighbors_algorithm. 4 Nov. 2019, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.