Data Classifying with Python

Homework 3

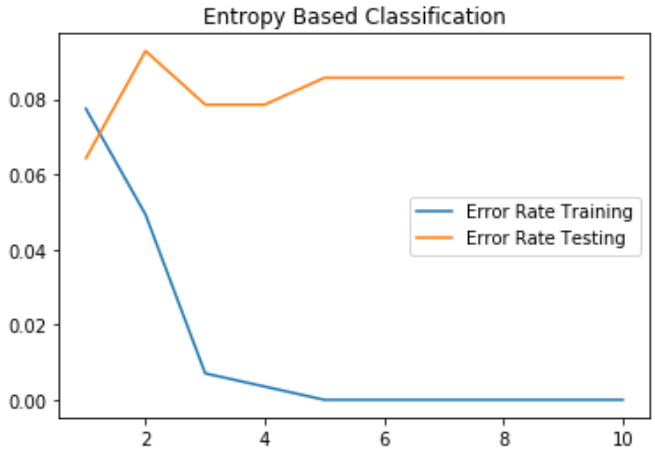
Jeffrey Bruggeman
Fundamentals of Data Science

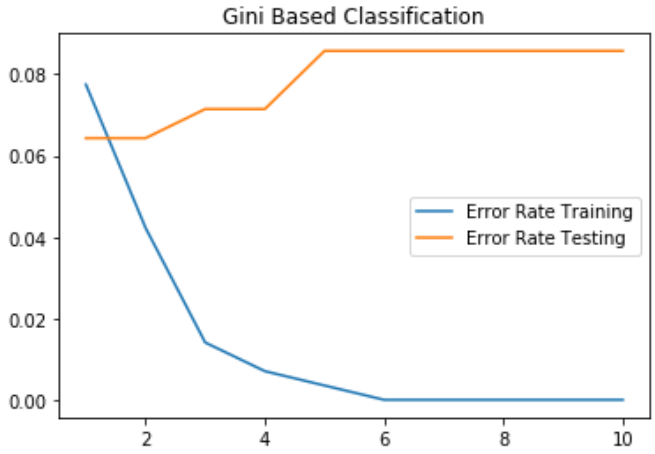
I. Task 1

The dataset that the classifications will be completed upon is data relating to whether a cancer is malignant or benign. The data associated with the classifier data are physical measurements recorded as floating-point numbers. The physical properties are multiple descriptive features of the cancer's texture, area, convex point, etc. There are thirty-one attributes in this dataset, one being the classifier and the rest are float measurements. The dataset also is composed of 424 records, 212 of which are benign data and the other 212 are malignant data, making the dataset balanced.

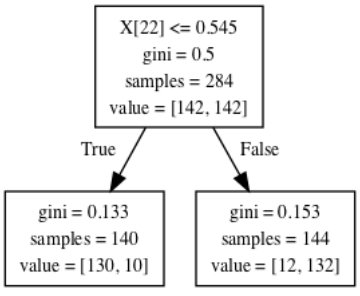
II. Task 2

Task 2 had us splitting the dataset into training and testing data frames, building decision trees, then using the built trees to classify the testing data. The accuracy at which the tree model would classify the testing data was recorded along with the training data accuracy. After testing at many different random states when dividing up the dataset, the gini based tree models were able to classify with more accuracy, specifically at ranges 1-4. After resetting back to random state 0, the simple gini decision tree with 1 height had only a 6.4% error rate, so I went with that for my best tree. The preference for the Gini model comes from how entropy and Gini are calculated- they are both used to calculate information gain, but Gini and the CART ID3 algorithm does not use a logarithmic calculation. So there is a computational bias, but for small sets for these there won't be much of a difference between the two. The charts produced from the calculations are displayed here, showing that for random state 0 there is very little difference between gini impurity and the entropy model based on this dataset.



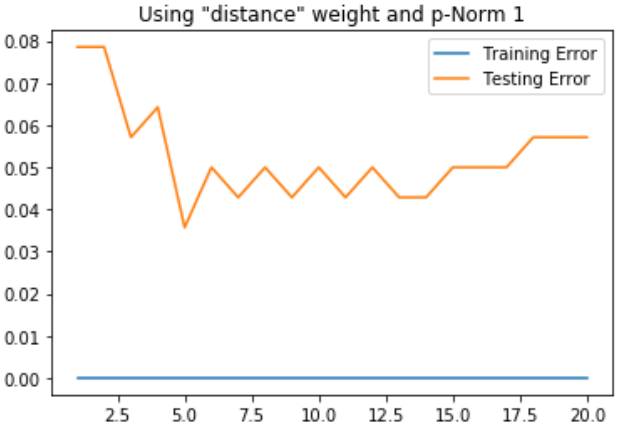


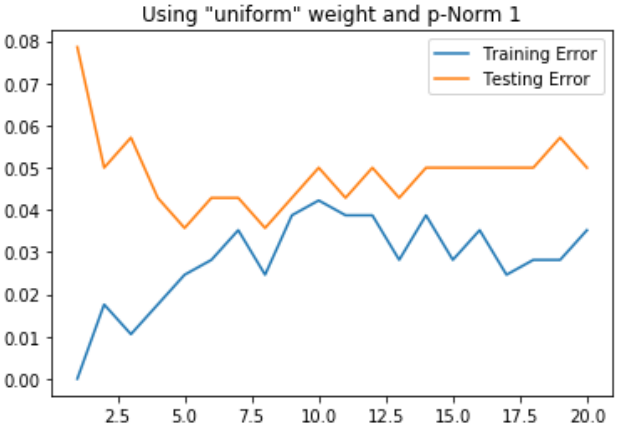
Ultimately, the best tree that was generated was visualized using graphviz to look like this.

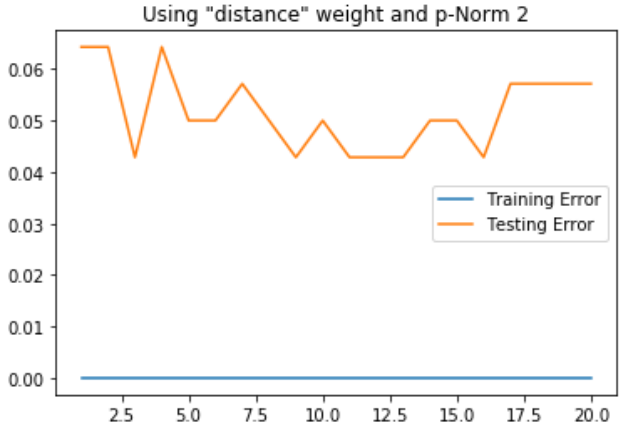


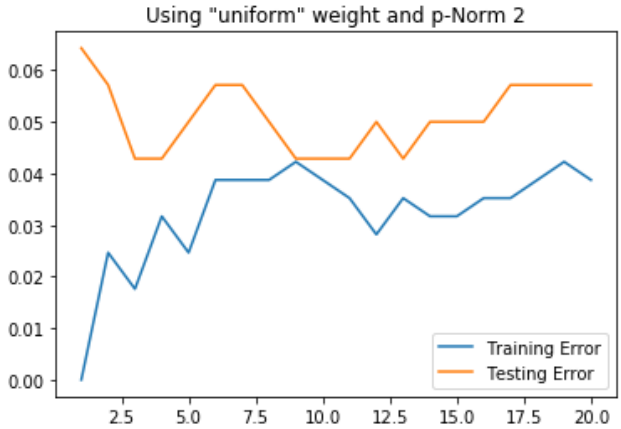
III. TASK 3

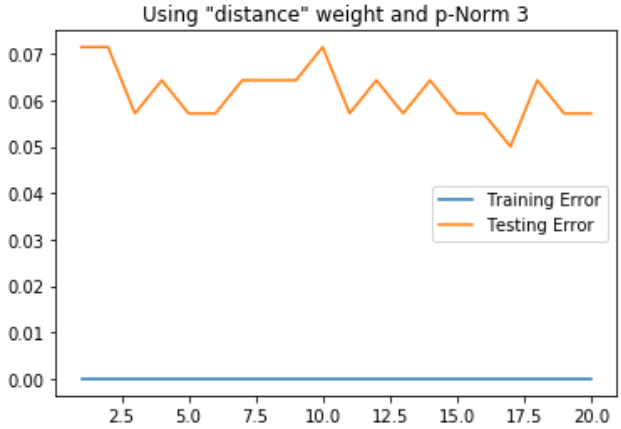
For Task 3, we had to create knn classifier models using the same dataset we used previously for the decision trees. To explore the actual rankings of this data, I decided I would test out 6 different combinations of 20 generated knn classifiers, for a total of 120 generated models. Because this is a small dataset, this was a trivial amount to calculate, but for larger datasets completing this many calculations could be taxing. From the models I generated, the best model ended up being a knn classifier using uniform selection, Manhattan distance calculation, and 5 nearest neighbors with a testing error rate of 3.5%. This model was tied in terms of error rate with a model that used distance selection, Manhattan distance, and 5 neighbors, but I still chose uniform selection because using distance selection requires each distance to be weighted which is an additional calculation per neighbor. The additional accuracy of using distance is noteworthy, however.

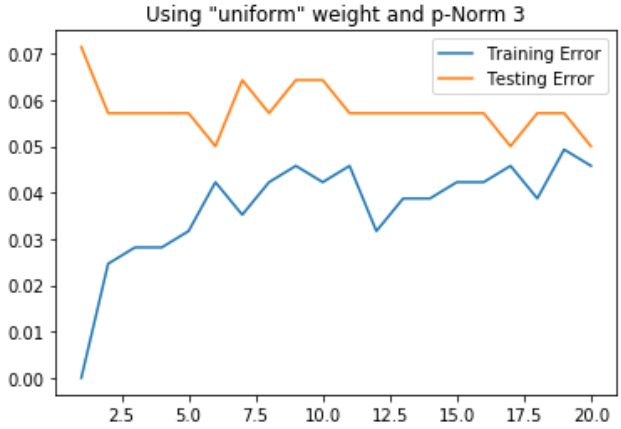












IV. TASK 4

Task 4 consisted of generating a list of impurity values for each attribute calculated by generating 10,000 random forest trees and averaging the impurity reduction from using each attribute. The resulting values can be used to inform much more concise decision tree model generation. The information generated is as follows:

```
 1) concave points3      0.127353
 2) perimeter3           0.127296
 3) concave points1      0.107300
 4) area3                0.105350
 5) radius3              0.100885
 6) concavity3           0.063737
 7) concavity1           0.062399
 8) perimeter1           0.043745
 9) area2                0.032031
10) area1                0.029723
11) radius1              0.028588
12) texture3             0.018028
13) compactness3         0.017129
14) smoothness3          0.015681
15) compactness1         0.015546
16) radius2              0.012630
17) texture1             0.011996
18) perimeter2           0.011851
19) concavity2           0.010380
20) smoothness1          0.009146
21) symmetry3            0.007336
22) concave points2      0.006484
23) fractal dimension2   0.006390
24) fractal dimension3   0.005456
25) compactness2         0.005221
26) texture2             0.004036
27) symmetry1            0.003839
28) smoothness2          0.003484
29) fractal dimension1   0.003482
```

The robustness that comes from generating 10,000 instances and averaging them together would typically create a sturdy model that would be much better than a few simply generated decision trees or knn classifiers, so this method will likely create a superior classifier. A way of implementing knn or decision tree models at a similarly effective rate would likely to use k-fold cross validation on the data before creating models.

V. TASK 5

For Task 5 we were given some freedom to create our own models and generate data to see if we could come up with more effective models than what was generated previously. Initially I started using the unmodified data originally found in CancerData, I simply mapped the class attribute and ran knn (with the previously most effective settings: uniform and Manhattan distance, but iterating through 10 neighbor values) and decision trees (entropy) for the data. After that, I clamped the data and ran the same 2 model classifiers. Both of the knn classifiers from the previous two tests gave me a 2.9% testing error rate at 5 neighbors, which would be the best model I created in this project. My best guess for the reason behind this increase in accuracy is because the normalization of the data in the initial step might have changed some majorly important measurements when putting all distances on the same scale. Effectively, normalizing data is saying that the distance 0 to 1mm is the same as 0 to 1km if those happen to be your mins and maxes. This change can reduce the importance of several measurements, even though the scale is kept. The decision tree models best error rate was 6.4% on testing, which is similar to previous tests. Next, I used the two best attributes from the previous random forest generated information to make a small decision tree, which produced the best tree I had created so far with only a 5% error rate at a depth of 2. For the next model I used the knn classifier on the previous random forest data to see if it would also increase the accuracy. Likely, the reduction in data made the set too sparse to find a better model than was previously generated- the best result here was 6.4% testing error at 7 neighbors. Finally, I created a scatter point matrix of the clamped data set to inspect if there was anything obvious that I could use to generate new fields. Using this I saw a few sets of information that seemed like good candidates for knn, so I normalized and multiplied those fields together to create a dataframe with only 3 fields in it- the mapped class and 2 generated attributes. Using knn on this data I was able get a testing error rate of 5.7% with 6 neighbors, which is a pretty good dimensionality reduction, similar to the reduction with the random forest informed decision tree. Additional testing I would have liked to run, given more time, would have been attempting to run several classifiers in tandem with confusion matrices for additional testing, as the nature of cancer malignance classification is quite serious and as a scientist we would want to reduce false negatives. Also, I would want to figure out how to

use k-fold cross validation or perhaps something as simple as iterating through random states for data splitting, then find the best circumstances for model creation over many iterations to build the most robust model possible. Finally, I want to study more about attribute generation to see if there are any general rules to follow or data characteristics to look out for to help further reduce dimensionality.

[1] "Python Documentation," *Python 3.7.1rc1 documentation*. [Online]. Available: https://docs.python.org/3/contents.html. [Accessed: 24-Oct-2018].

[2] "Pandas Documentation," *Pandas documentation*. [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/index.html [Accessed: 24-Oct-2018].

[3] "Scikit-learn Documentation," *Scikit-learn documentation*. [Online]. Available: http://scikit-learn.org/stable/documentation.html [Accessed: 24-Oct-2018].

[4] "Matplotlib Documentation," *matplotlib documentation*. [Online]. Available: https://matplotlib.org/contents.html [Accessed: 24-Oct-2018].