Lab 3 Report

C4.5

Based on the fact that we used C4.5 we were able to reuse our implementation from the last lab. In order to use this classifier in order to determine accuracy of the results, we needed to determine the base split values based on the weights. Once these values are found, the tree essentially works itself out.

At this time we are still working on our implementation for C4.5 but can assume that the values will be very simple to both Random Forest. We expect the iris dataset to be the most accurate due to the small size closely followed by the seeds dataset. This being said, we expect the agaricus-lepiota dataset to be the least accurate as the dataset is huge. Based on C4.5 and our knowledge of the classifier, we can assume that the larger the dataset, the less accurate it is.

Random Forest

Similar to the classifiers in the last lab, the Random Forest classifier uses C4.5 in the implementation in order to create the individual decision trees. Although C4.5 is not the required tree algorithm, for the purpose of this lab, we have used it. In order to achieve this, we build all possible trees that can be created by the algorithm which results in us having a “forest.” The results vary for each dataset but due to the bagging and default values that we have set for the trees, we have a uniform set of trees that are no deeper than 10. The results from this are as follows.

For the iris dataset, we took the base values that were given such as the min size and the minimum depth. This provided us with the accurate data that was required and once C4.5 was called on the data, we were able to determine the forest for the iris dataset. This forest resulted in a mean accuracy of 96% for approximately 5 trees. Five trees for this dataset seemed to be the sweet spot.

For the seeds dataset, we took the same base values that we had in the last dataset with a random seed value of 2. This allowed the data to reset and no values to carry over from the last time the program was run. Once this was done, the program could be run on the seeds dataset which resulted in us being able to determine that the most accurate run of the random forest algorithm is when there are 10 trees. While there is 10 trees, the best mean accuracy of 90% is found.

Finally for the agaricus-lepiota dataset, we used the same process as the previous datasets. Just like the seeds dataset, we had to use a seed value of 2 to reset the values and make sure nothing carried over. This resulted in us being able to determine that the most accurate forest level 10 trees as the number of possible trees is quite large due to the sheer number of data in the set. Ten trees with this dataset resulted in us having a mean accuracy of 85%. This is considerably lower than the other sets due to the sheer size of it.

KNN

Unlike both C4.5 and Random Forests, K Nearest Neighbors required a much more high level approach. In order to run this method, all that is needed is to do ‘python knn.py’ and make sure that the dataset is in the correct spot. This should allow the user to observe all the knn outputs even though it may take a while. Our implementation of the method outputs both the training set, the test set, as well as the accuracy of each of the predictions. This allows us to find the most accurate knn out of the bunch. What follows is the results for each dataset.

For the iris dataset, we started with a base split value of 0.67. From there, we both increased and decreased the values until we found a value that was extremely close to 100%. Starting at 0.67 resulted in out first output being very close to the desired value at 93.3% and increasing the value increased the accuracy. The optimal split value that we found for this dataset was 0.70 which was at 98%. This value produces the most constant results and the most accurate results for the nearest neighbors.

For the seeds dataset, the algorithm required a little bit of tuning that resulted in the creation of several new methods including a new distance and reader method. This allows us to be able to accurately compare the neighbors which resulted in a less pleasant result than the last one. Similar to the iris dataset, we started out with a split of 0.75 which resulted in an accuracy of 84.9% which gives us the most accurate nearest neighbor set since the dataset was extremely large.

Finally for the agaricus-lepiota dataset, the algorithm again required a little bit of tweaking so the calculation would work correctly. This meant changing all of the alphabetic values to numeric values so that the distance function would work. The result that we found caused the program to take quite a while to run. Based on this, we were unable to run a lot of values and only find a generic value that we found was best. This value was found to be 0.65 which resulted in an accuracy of 52%. This is not the most ideal value that we could have but due to time constraints, this was the best that we could do.

If the output of KNN would like to be viewed, we have included it in our zip file under the name of knn.csv. This file is rather large but includes the thought process of the program and all of its output. Each dataset is separated into its own section within the csv and is separated from the previous via a line of # signs.

Overall

Based on the algorithms and methods that were run within this lab, we were able to derive many different trees from sets of data. Based on these trees, we were able to determine quite a bit of data and information from them. This was found by using the hyper-parameters of

folds for random forests, and a split value for K Nearest Neighbors. As for C4.5, we did not use any hyper-parameters as we dont run the method multiple times as the tree is just created once.

Once these hyper-parameters were used, we were able to use the K-Fold cross validation method in order to determine whether or not the method was accurate or not. This resulted in us being quite confident in the results that we got. Once this was discovered, we were able to determine the best and most accurate values for each of the classification techniques. What follows is the results of each of the methods in a table.

Classifier	Iris.data	seeds_dataset.csv	agaricus-lepiota.data.csv
C4.5	TBD	TBD	TBD
Random Forests	96%	90%	85%
KNN	98%	84.9%	52%

Based on our results, we can see that our findings are right in line with the ideas that we have behind these classifiers overall. KNN is in fact the “lazy” classifier while Random Forests appears to be the most accurate. C4.5 is also a good choice but does not give the whole picture or all possible options. Therefore, the best classifier to use if you are looking for the most accurate is Random Forests and the worst one was KNN.

This lab was very eye opening on how each of the different classifiers really worked and that not all machine learning classifiers are equal.