CS 434 Implementation Assignment 3 Report

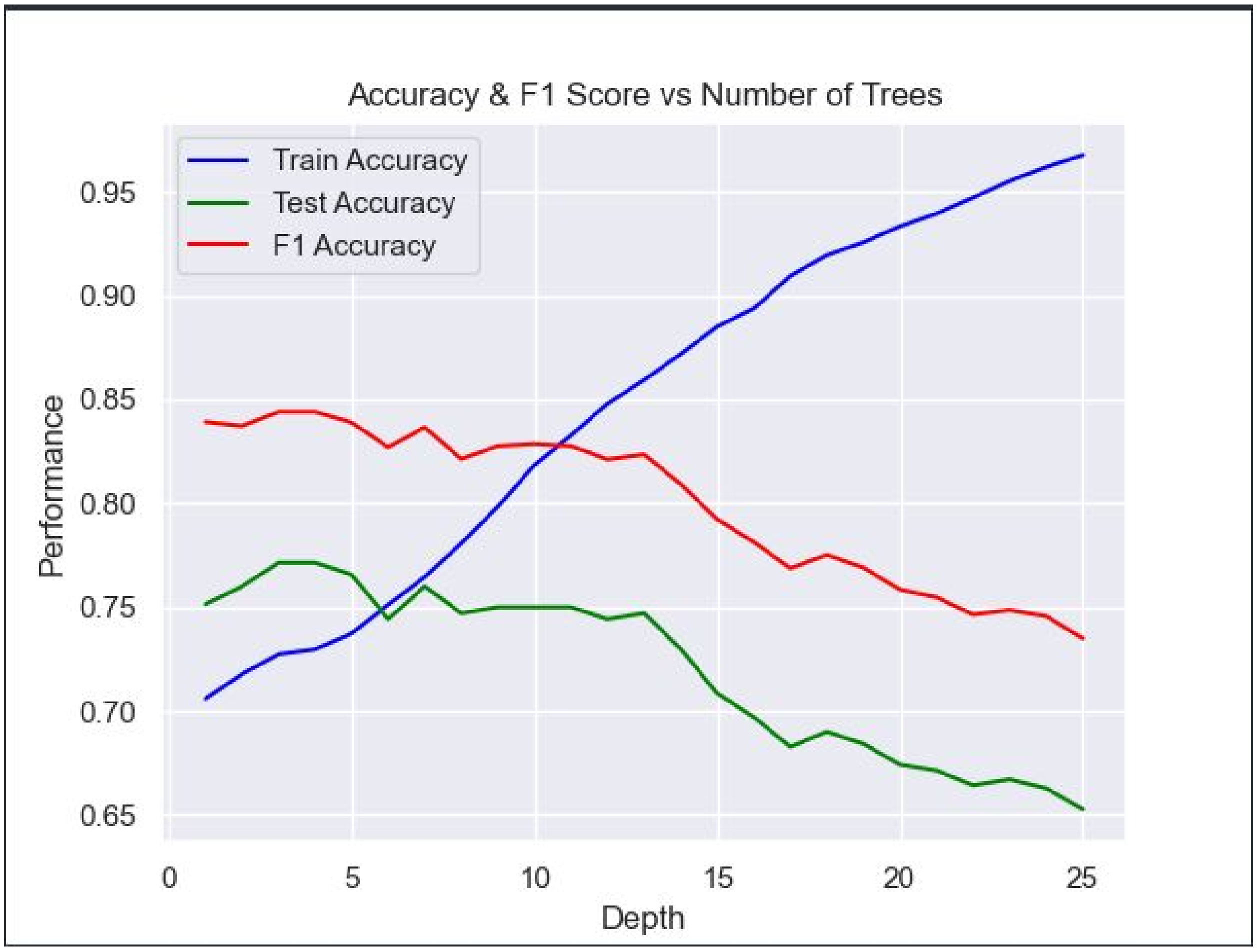
Group 46 - Race Stewart, Joshua Diedrich, Grayland Lunn

Introduction

The purpose of this assignment was to classify US counties using a set of discrete features related to each county. To do this, we used various decision tree and ensemble methods including Decision Tree, Random Forest, and AdaBoost. The report entails any accompanying plots from the data and a description of our results.

Part 1

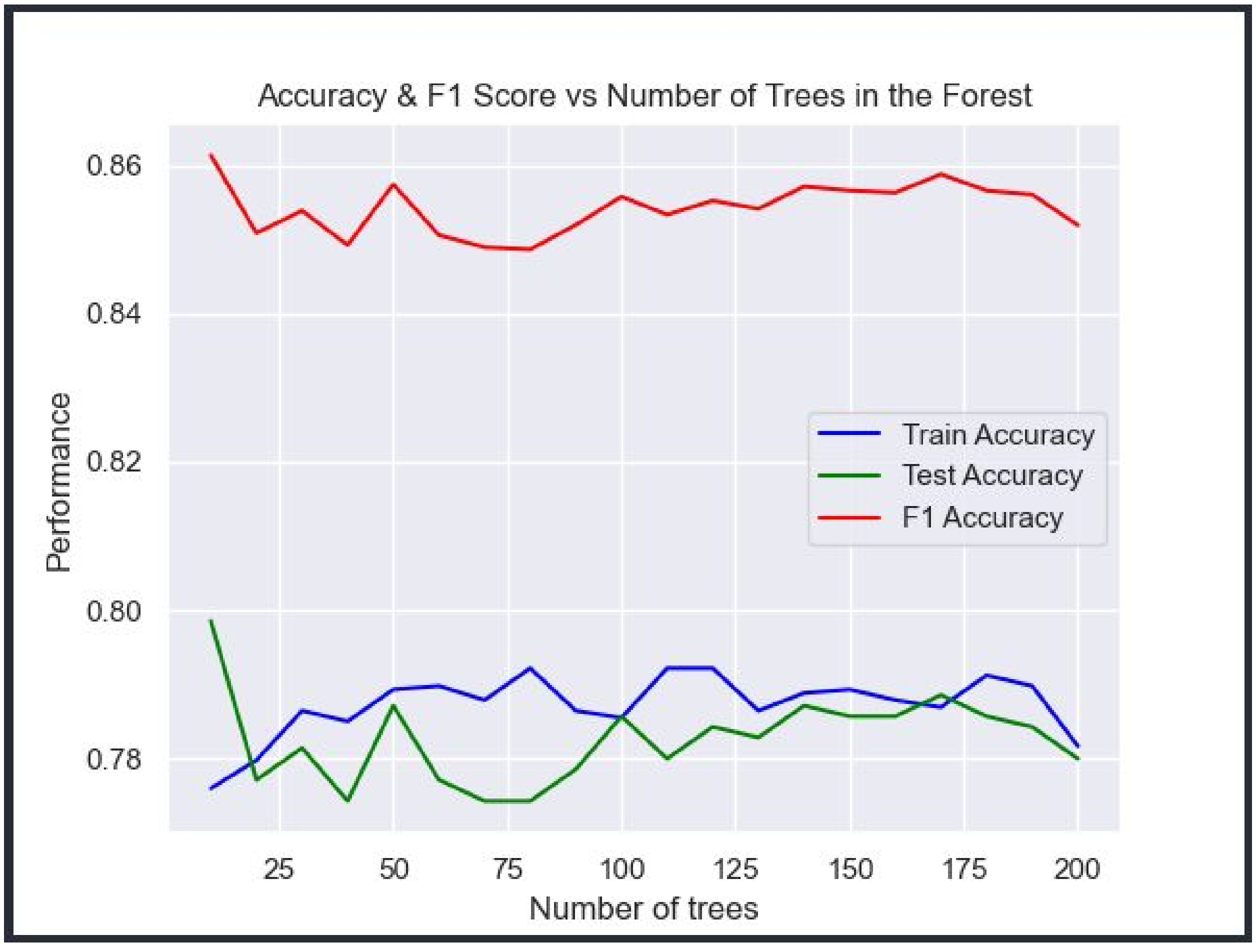
D.



E. According to our decision_tree_various_depths function, 8 is the depth that gives the best validation accuracy. It might be worth noting that both train and test accuracies continue to rise at a very slow rate after that point, though.

F. The most important feature for making a prediction according to the same function is feature 8 (White alone, percent, 2014). The threshold to split on for this feature is 5, though I'm not entirely sure that's what the question is asking. This is the feature that gets the highest gain for the top node in our decision tree.

Part 2

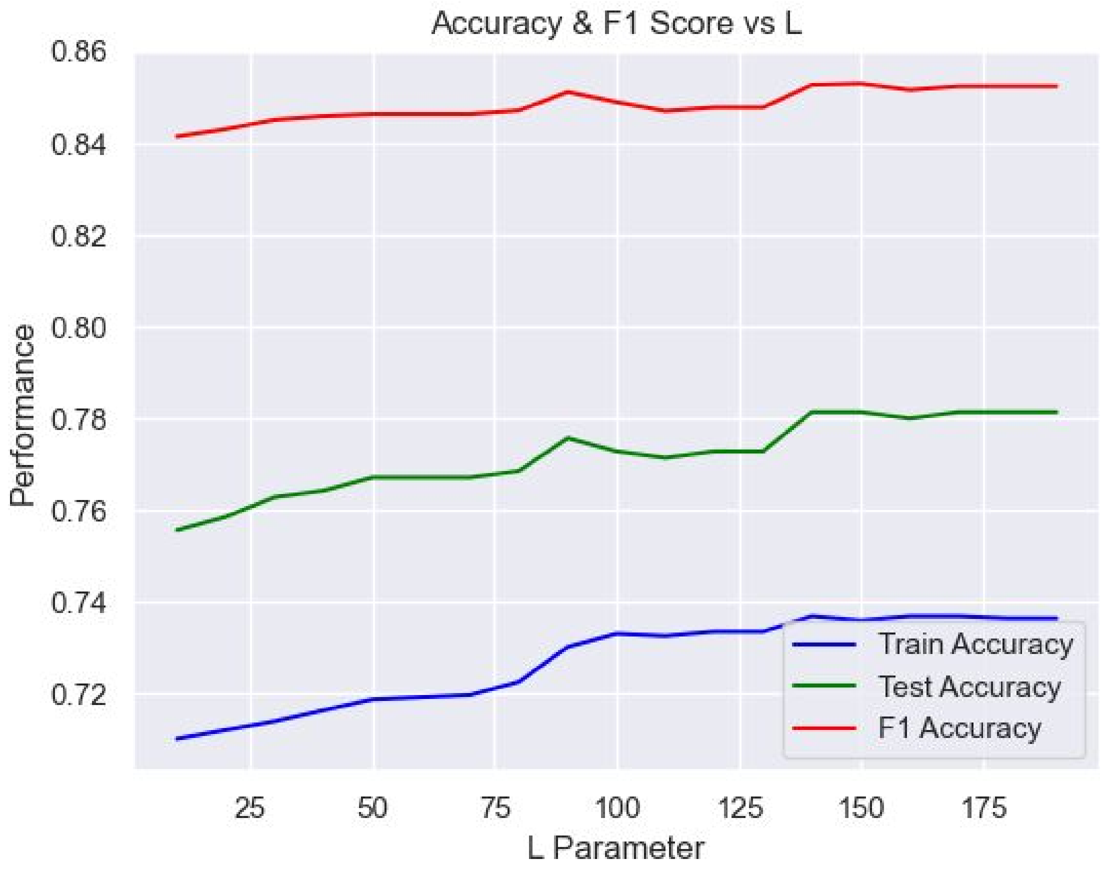
B. 

C. Training and testing performance remain relatively constant regardless of the number of trees. There is some inherent randomness, but nothing to conclude any significant effects from the number of trees.

D. There seems to be a climb as we go from 1 max_feature to 2 to 5, but it basically plateaus after that. This suggests that there is an optimal number of max features somewhere around the 20-40 mark.

F. Based on our results, we don't think we can determine that randomness affects the performance in any certain way. The accuracies remain relatively the same.

Part 3



F. As you can see, increasing the L Parameter provides an increase in training and testing accuracy, as well as the F1 accuracy. This increase seems to level out around an L parameter of 200. This is not a surprising result, as you should expect that increasing the number of base classifiers would yield better results.

Part 4

B. Decision trees will often have issues learning from imbalanced datasets. When we choose a split point in our tree, we want to separate our samples into two groups with a high amount of separation. If the majority class dominates both of these groups, we will probably see good separation for both groups. This means our majority class will make our splits look more pure than they should. One way we could combat this with our decision trees is by adding class weights. When evaluating a split point, we would factor in the weight for each class. If you apply a slightly smaller weight to majority classes, and/or a larger weight to minority classes, you might see that the results would improve. Another technique we could use here would be to oversample the minority or undersample the majority. There are arguments for both, but both will in turn give you a more balanced dataset, which might also improve testing results. The drawback here is that our data can appear to have a higher variance when undersampling, and a lower variance when oversampling. Most of these involve pre-processing our data. But to better verify our answers are correct we could have used an ROC curve to see our diagnostic ability of our binary classifier. Any of these methods are worth trying out on our dataset, and are likely to yield interesting results.