**Decision Tree Optimization using Pruning and Hyperparameter tuning**

We know that the decision trees are prone to overfitting. In this article I will be discussing how we can improve the prediction error for decision tree.

Overfitting of the decision trees to training data can be reduced by using pruning as well as tuning of hyperparameters.

Here am using the hyperparameter max\_depth of the tree and by pruning [ finding the cost complexity].

This is an experimental study using two data sets to compare the two approaches for understanding. This article helps in getting started for anyone who is new to machine learning and wants to use decision tree classifier using scikit learn for their modelling.

Decision trees are commonly used in machine learning because of their interpretability. The decision tree structure has a conditional flow structure which makes it easier to understand. In machine learning, a hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters are derived via training or the dataset. The hyperparameters learn from the data. We could use cross validation to set the best tuning hyperparameter. We find the training and testing errors at different max depths and selecting the one with minimum test error for fitting the final decision tree.

**Optimization by PRUNINIG:**

**Pruning a Decision tree refers to simplifying/compressing and optimizing it by removing parts of the tree that are not adding value in reducing the classification error and are redundant.**

While doing the experiment, tried the 2 **Classification Trees** with below datasets from [**UCI Machine Learning Repository**](https://archive.ics.uci.edu/ml/index.php):

1> [Heart disease dataset](https://archive.ics.uci.edu/ml/datasets/Heart+Disease) to predict whether or not a patient has heart disease.

**2>** [Wine quality dataset](https://archive.ics.uci.edu/ml/datasets/wine+quality) **reduced to two classes [Quality =5 & Quality=6].**

**Building preliminary classification tree:**

**By using scikit-learn DecisionTreeClassifier, built the complete tree and below are the accuracies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Training- Accuracy** | **Test-Accuracy** | **Max\_depth(by default)** |
| **Wine Quality** | 1 | 0.70 | 25 |
| **Heart Disease** | 1 | 0.76 | 11 |

**According to the scikit-learn documentation, if max\_depth is None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples. Max\_depth of the preliminary decision tree is got by accessing the max\_depth for the underlying Tree object.**

First, we try using the scikit-learn Cost Complexity pruning for fitting the optimum decision tree. **This is done by** using the scikit-learn Cost Complexity by finding the alpha to be used to fit the final Decision tree.

Pruning a Decision tree is all about finding the correct value of alpha which controls how much pruning must be done. One way is to get the alpha for minimum test error and use it for final decision tree.Hyperparameter alpha is the Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than ccp\_alpha will be chosen.

After scanning through the alphas, below listed for the one which result in minimum test error:

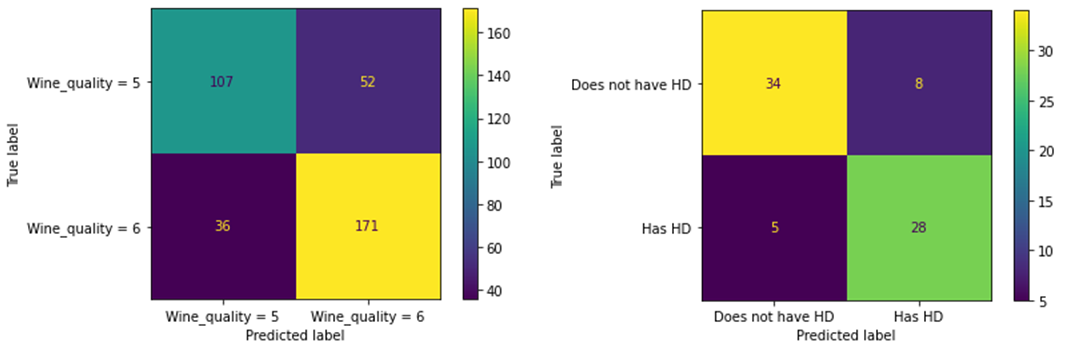
|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Training- Accuracy** | **Test-Accuracy** | **alpha** |
| **Wine Quality** | 0.910003 | 0.759563 | 0.000608 |
| **Heart Disease** | 0.851351 | 0.826666 | 0.016425 |

**The above table suggest one value for alpha, but another set of data might suggest another optimal value.**

**Hence, I have used k-fold cross validation to get the optimum alpha.**

**With cross validation we are trying to get the expected level of fitness of a model to the dataset that is independent of the data that were used to train the model.**

**Upon using the ideal value for alpha, we built the final decision trees and got the confusion matrices as below:**

****

Comparing with preliminary decision trees, we could see for Heart Disease dataset there is definite improvement (~6%) in accuracy and for Wine quality dataset at (~6%) improvement over the built preliminary tree.

**Optimization by max\_depth:**

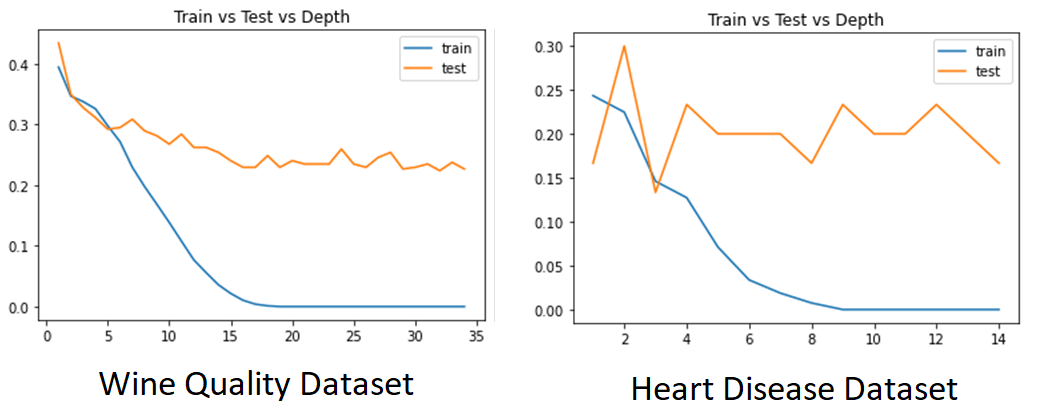
Also the other method used is to check the Training error vs Test error vs Depth of tree.

**Principle of Occam’s Razor states when two trees have similar classification error on the validation set pick the simpler one.**

**While experimenting with the two datasets, it was observed that the Decision trees started overfitting when we started to fit it at higher depths, test error started increasing as the depth of the trees increased after an optimum value.**

**K-fold cross validation is used while getting the training vs test error at different max\_depths using scikit learn decision tree classifier.**

**Below are the Graphs showing the behaviour for two datasets:**

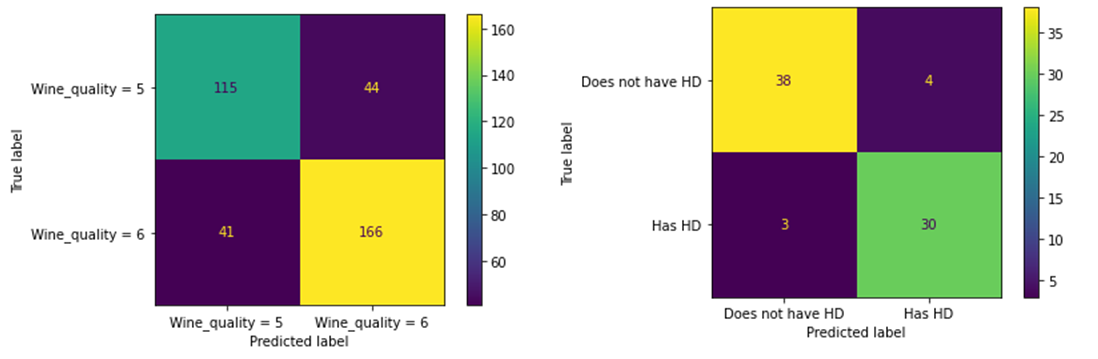
****

**With Wine quality dataset we could see the test error going bit high after max\_depth=15 and for Heart disease dataset at max\_depth=3.**

**We have fit the final decision trees using the max\_depth options giving the minimum test errors and below is the result:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Training- Accuracy** | **Test-Accuracy** | **Max\_depth** |
| **Wine Quality** | 0.97 | 0.77 | 15 |
| **Heart Disease** | 0.83 | 0.92 | 3 |

**Below shows the confusion matrix for the same:**

****

**From the two methods with the datasets used it is observed that optimization by finding the max\_depth at minimum test error proves to be the better one.**

**References :**

<https://www.coursera.org/projects/classification-trees-in-python>

<https://devdocs.io/scikit_learn/>