CS165B: Intro. to Machine Learning, Fall 2020
Programming assignment # 3

Due: Wednesday 11:59pm, Nov. 25th

IMPORTANT NOTE You can discuss the problems with your TAs and instructor, but all answers and codes written down and turned in must be your own work.

Implement the **Classification and Regression Tree (CART)** algorithm and train a decision tree to do classification.

Data Set

The Red Wine Quality Data Set contains 1599 samples. Each sample has 11 features (attributes) and a quality score (QS) ranging from 0 to 10. To make it simple, you can treat all 11 features as continuous variables. Refer to this link for more details: https://archive.ics.uci.edu/ml/datasets/wine+quality. We label the samples into 3 different classes: "good/2" (QS>6), "normal/1" (QS=6), and "bad/0" (QS<6) based on their quality scores. Then, we partition the data set into a training set and a testing set including 1000 samples and 599 samples respectively. You are provided with the training set while the testing set is held back for grading use.

Skeleton code

A CART instance is able to take in a feature array `X` (a numpy array of size $(n, 11)$) and a label array `y` (a numpy array of size $(n,)$), train and store the trained tree by calling the `train` method. Then, it should be able to output label prediction of shape $(n,)$ given a feature array `X_test` by calling the `test` method. There must be an instance variable `self.tree` that stores the root node. You are allowed to add any other variables or class methods that you find useful.

```
class CART(object):
    def __init__(self, max_depth):
    def train(self, X, y):
    def test(self, X_test):
```

In addition to the methods above, we provide you with a class method called `visualize_tree` that prints out the stored decision tree recursively. It helps visualize your tree and to check if the split at a node is correct or not. You should use the provided `TreeNode` class which defines the data structure of the decision tree, i.e. every node in the decision tree should be an instance of the `TreeNode` class. Refer to the comments in the skeleton code for detailed descriptions.

Requirements

- Each tree node should have either zero or two children nodes (binary split). Each node should be split based on one and only one feature.
- Use **gini impurity** as the measure when making split decisions. You can repeat using the same feature if it keeps being the best choice among all others while developing the tree.
- Implement 5-**fold cross validation** and **grid search** to find the best `max_depth`. `max_depth` is the one and only one parameter to be searched for. This time, instead of implementing cross validation and grid search inside the model class, you are asked to implement them in the function named `GridSearchCV`. Do not train multiple trees in the `CART` model class. Refer to the skeleton code for more details.
- Everything should be **built from scratch**. You are not allowed to use any libraries that partially or fully develop the CART algorithm within, such as *scikit-learn*.
- (Optional) Try tree pruning/min impurity decrease/class weights/... to improve the performance.

Hints

How to make split on continuous features: A feature is continuous if its value can be any number within a range. Assume at a tree node, there are N training samples and the d-th feature forms a list $X^d = [x_1^d, x_2^d, x_3^d, ..., x_N^d]$. There may be some repetitive values, i.e. $x_i^d = x_j^d, i \neq j$. The split starts from a sorted list of unique values from X^d, for example denoted as $D = [D_1, D_2, ..., D_m]$. Your algorithm should start from setting the threshold as $\frac{D_1+D_2}{2}$, then $\frac{D_2+D_3}{2}$ and so on until $\frac{D_{m-1}+D_m}{2}$. The threshold value that generates the minimum gini impurity (or maximum gini gain) will be the best choice for split based on the d-th feature.