CS5228: Assignment 2

1. Introduction

In this assignment, you are going to implement a Decision Tree Regressor and a Gradient Boosting Regressor. The deadline for this project is 6pm on **29/Oct/2018**.

2. Decision Tree Regressor

In this part, you are going to implement a Least Squares Regressor Tree. We have provided the data and the test cases to you. You should use **Python 3.X** in your implementation.

CART (Classification and Regression Trees), a non-parametric statistical algorithm, is developed by Breiman, Friedman, Olshen, Stone in early 80's. CART can be used to predict or analyze both categorical (classification) and continuous or numerical (regression) data. The process of growing the Least Squares Regressor Tree by CART is summarized as follows:

Input: Training Dataset D;

Output: A regression decision tree $f(x)$.

In the input space of the training dataset, each region is recursively divided into two sub-regions and the output values on each sub-region are determined to construct a binary decision tree.

Steps:

1. Select the optimal splitting variable j and the splitting threshold s to solve

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

   Traverse the splitting variable j and scan the splitting threshold s for the fixed splitting variable j, and determine the pair (j,s) that minimizes the expression.

2. Split the region with the selected pair (j,s) and determine the corresponding output value:

$$\mathrm{R}_1(j,s)=\{x|x^{(j)}\le s\},\mathrm{R}_2(j,s)=\{x|x^{(j)}>s\}$$

$$\hat{c}_m=\frac{1}{N_m}\sum_{x_i\in R_m}y_i,\qquad x\in R_m,\qquad m=1,2$$

3. Repeat the above two steps considering each resulting region as a parent node until the maximum depth of the tree is obtained.

4. The input space is divided into M regions $\mathrm{R}_1,R_2,\dots,R_M$, then the decision tree is

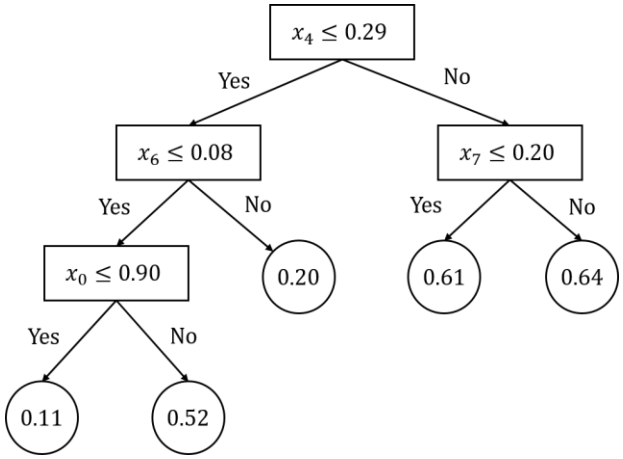
$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m)$$

Implementation details:

You are required to implement the function "`fit(self, X, y)`" and "`predict(self, X)`" in the file "DecisionTreeRegressor.py".

In the function "`fit(self, X, y)`", you should update the `self.node`, whose type is dictionary. There are four keys in `self.node`, which are "`splitting_variable`", "`splitting_threshold`", "`left`" and "`right`". The value of "`splitting_variable`" should be an integer number. The value of "`splitting_threshold`" should be a float number. The values of "`left`" and "`right`" should be either a float number or a dictionary.

For example,



the above decision tree can be represented by

```
Self.root = {"splitting_variable": 4,
             "splitting_threshold": 0.29,
             "left": {"splitting_variable": 6,
                      "splitting_threshold": 0.08,
                      "left": {"splitting_variable": 0,
                               "splitting_threshold": 0.90,
                               "left": 0.11,
                               "right": 0.52},
                      "right":0.20}
             "right": {"splitting_variable": 7,
                       "splitting_threshold": 0.20,
                       "left": 0.61,
                       "right": 0.64}
             }
```

3. Gradient Boosting Regressor

In this part, you are going to implement a Gradient Boosting Regressor based on the Least Squares Regressor Tree you implemented in part 2. We have provided the data and the test cases to you. You should use **Python 3.X** in your implementation.

In this task, the loss function is defined as

$$L(y, f(x)) = \frac{1}{2}(y - f(x))^2$$

Then, the negative gradient of $L(y, f(x))$ is

$$-\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} = y_i - f(x_i)$$

The algorithm of Gradient Boosting Algorithm is summarized as follows:

Input: Training dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}, x_i \in \mathcal{X} \in \mathbf{R}^n, y_i \in \mathcal{Y} \in \mathbf{R}^n$; Learning rate lr; Loss function $L(y, f(x))$.

Output: A regression decision tree $\hat{f}(x)$.

Steps:

1. Initialize $f_0(x) = argmin_c \sum_{i=1}^{N} L(y_i, c)$.
2. For $m = 1$ to M:
    a. For $i = 1,2, \dots, N$, compute the residual
    $$\gamma_{\text{im}} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$$
    b. Fit a regression tree to the targets γ_{im} resulting in terminal regions $R_{mj}, j = 1,2, \dots, J_m$.
    c. For $j = 1,2, \dots, J_m$, compute
    $$c_{mj} = lr \times argmin_c \sum_{x_i \in R_{mj}} L(y_i, f_{m-1}(x_i) + c)$$
    d. Update $\mathrm{f_m}(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \mathrm{c}_{mj} I(x \in R_{mj})$
3. The regression tree is $\hat{f}(x) = \mathrm{f_M}(x) = f_0(x) + \sum_{m=1}^{M} \sum_{j=1}^{J_m} \mathrm{c}_{mj} I(x \in R_{mj})$

Implementation details:

You are required to implement the function "`fit(self, X, y)`" and "`predict(self, X)`" in the file "GradientBoostingRegressor.py".

In the function "`fit(self, X, y)`", you should update the `self.estimators`, whose type is np array. Each element in this array is a regression tree object.

4. Library

It is okay for you to use some packages in python, you are okay to use some basic data structures and some basic functions which can help you implement your function.

5. Cooperation

It is okay for you to discuss implementation ideas with your classmates. However, you need to write the code yourself, plagiarism will not be tolerated in this module.

6. Error Reporting

If you spot any bugs for the template, and if the bug is minor, you are encouraged to modify it yourself and report in the forum to share it with your classmates.

7. Marking Criteria

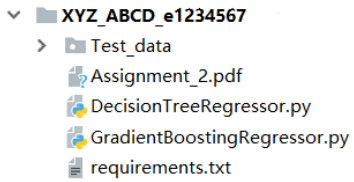
Accuracy is our first marking criteria for this project. We will generate several test cases to test the accuracy of your code. In addition, we will also consider the efficiency of your code, if the running time of your code is too slow, corresponding marks will be deducted from you.

8. Submission Format

When you submit your code, please zip your assignment folder and name your folder in the following format "YourNameInIVLE_YourUserIDInIVLE.zip" (You can check the displayed

information from the Class & Groups tab in the module CS5228 from IVLE). An example may be "XYZ_ABCD_e1234567.zip". In addition, you need to make sure that if we unzip your folder, the flies “DecisionTreeRegressor.py” and “GradientBoostingRegressor.py” reside in the "XYZ_ABCD_e1234567" folder.

Below is a sample submission format for your reference.



Please follow the submission format tightly, as we will apply this format to automatically mark your code. So, if you fail to obey this format, we will not be able to mark your code and corresponding marks will be deducted from you.