Econ 31703: Assignment 4

Due date: June 9, 2022 (optional)

Exercise 1

In this exercise, we will construct a (crude) version of a tree algorithm.

(a) Write a function that grows leaves. The function considers $(n-1)\cdot p$ partitions and computes the decrease in MSE for each partition. For example, suppose it is considering a partitioning $X_{ik} \leq X_{(m)k}$ with $X_{(m)k}$ being m-th smallest value in $\{X_{1k}, \cdots, X_{nk}\}$. The partition splits the sample into subsamples of size m and $n-m$. Then, the gain (decrease in SSE) is

$$\sum_{i=1}^{N} \left(Y_i - \bar{Y}\right)^2 - \sum_{i=1}^{N} \left(Y_i - \bar{Y}^{-}\mathbf{1}_{\{X_{jk} \leq X_{(m)k}\}} - \bar{Y}^{+}\mathbf{1}_{\{X_{ik} > X_{(m)k}\}}\right)^2$$

where $\bar{Y}^{-} = \frac{\sum_{j=1}^{N} Y_j \mathbf{1}_{\{X_{jk} \leq X_{(m)k}\}}}{\sum_{j=1}^{N} \mathbf{1}_{\{X_{jk} \leq X_{(m)k}\}}}$ and $\bar{Y}^{+} = \frac{\sum_{j=1}^{N} Y_j \mathbf{1}_{\{X_{jk} > X_{(m)k}\}}}{\sum_{j=1}^{N} \mathbf{1}_{\{X_{jk} > X_{(m)k}\}}}$. The function returns the best splitting criterion $\left(k^*, X_{(m^*)k^*}\right)$, the decrease in SSE, and the partition denoted with $(0,1)$ under the criterion.

```
tree.grow <- function(data){
## data is a n × (p+1) matrix,
## where the first column denotes y and the rest denotes x's.
## Loop through (n-1)×p partitions.
⋮
return((n+3)×1 vector)
}
```

(b) Using `tree.grow`, write a function that selects and splits a leaf, when given a tree structure.

To track the tree, the function is given following two matrices. The first matrix `split` tracks the splitting algorithm: the number of rows in the matrix denotes the number of splits. Leaf id identifies each leaf using the binary system: id for a new leaf is given as double the id of its decision leaf if it corresponds to being under the threshold and as double plus one otherwise. In the particular example of Table 1, the id being $5 = 2 \cdot 2 + 1 = 2 \cdot (2 \cdot 1 + 0) + 1$ means that

	leaf id	depth	k^*	threshold
1	1	0	1	1.53
2	2	1	2	-1.67
3	5	2	1	0.36

Table 1: Splitting algorithm (so far)

$X_{i1} \leq 1.53$ and $X_{i2} > -1.67$. The 'depth' indicates how many splits have been previously done to this leaf. Figure 1 shows the tree structure generated by the splitting algorithm.

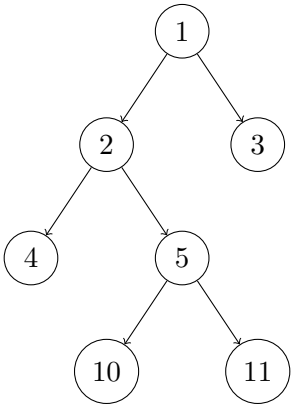

Figure 1: (Current) tree structure

Then, the second matrix `assign` tracks the current leaf assignment: the number of rows in this matrix should be n. Note that leaf ids in this matrix do not overlap with those in the first matrix: the leaf ids in the first matrix is where splitting happens and the leaf ids in the second matrix is the terminal nodes where splitting is yet to happen.

i	leaf id	depth
1	4	2
2	11	3
⋮	⋮	⋮
n	3	1

Table 2: (Current) leaf assignment

In addition to the tree structure, the function takes two stopping parameters as inputs: `min.size.for.split`, `max.depth`.

The function should work in the following way:

1. Using `assign`, compute the sizes of terminal leaves.

2. For leaves with depth less tha`max.depth` and size bigger than `min.size.for.split`, use `tree.grow` to find best splitting. If there is no such leaf, return some number or string, which will be later used in the wrapper function, to stop looping.

3. Find a leaf with the biggest decrease in SSE and update `split` and `assign`.

```
tree.update <- function(data,split,assign,min.size.for.split,max.depth){
## Suppose this is after S times of sample splitting.
## split is a S×4 matrix, dictating the splitting rules.
## assign is n×2 matrix, describing the leaf assignment.
## min.size.for.split and max.depth are positive integers.
## Loop through S leaves to find where ΔSSE is the biggest.
⋮
return(list(split=split_updated, assign=assign_updated))
}
```

(c) Using `tree.update`, write a function that grows the whole tree. Let the function return the final `split`, `assign`, and another matrix `leaves` whose columns are id, depth, size, and mean of the terminal leaves.

```
tree <- function(data, min.size.for.split=10, max.depth=10){
⋮
return(list(split=split_updated,
            assign=assign_updated,
            leaves=leaves))
}
```

For a dataset $(Y_i, X_i) = (0,1),(1,2),(1,3)(0,4),(1,5),(0,6),(1,7),(1,8),(0,9),(0,10)$, the tree algorithm wi`min.size.for.split`=`max.depth`=3 splits the sample into

$$(Y_i, X_i) = (0,1)\Big/(1,2),(1,3)\Big/(0,4),(1,5),(0,6),(1,7),(1,8)\Big/(0,9),(0,10),$$
$$\text{or } (Y_i, X_i) = (0,1)\Big/(1,2),(1,3),(0,4),(1,5),(0,6)\Big/(1,7),(1,8)\Big/(0,9),(0,10).$$

(d) Write a function that predicts for Y, using the outcome of `tree` function. Taking advantage of the way we constructed the leaf id, the function just has to move between each row of `split` and update the id accordingly.

```
tree.predict <- function(data, tree){
## data is a m×p matrix.
⋮
return(m×1 vector)
}
```

(e) Consider the following DGP: $(X_{i1}, X_{i2}, \varepsilon_i)^\intercal \overset{iid}{\sim} \mathcal{N}(\mathbf{0}, \mathbb{I}_3)$, for $i = 1, \cdots N$. Simulate a random sample of size $N = 1,000$. Generate $Y_i = 3 \cdot \min\{X_{i1}, X_{i2}\} + \varepsilon_i$. Run linear regression of Y on X_1, X_2 and then decision tree algorithm. Using tenfold cross-validation, compare the MSEs of the two estimation methods. You can either use the default values as given in the function or experiment over different values of your choice for `min.size.for.split`, `max.depth`. Interpret the result.

(f) Repeat **(e)** but now generate $Y_i = 3X_{i1} - 3X_{i2} + \varepsilon_i$. Interpret the result.