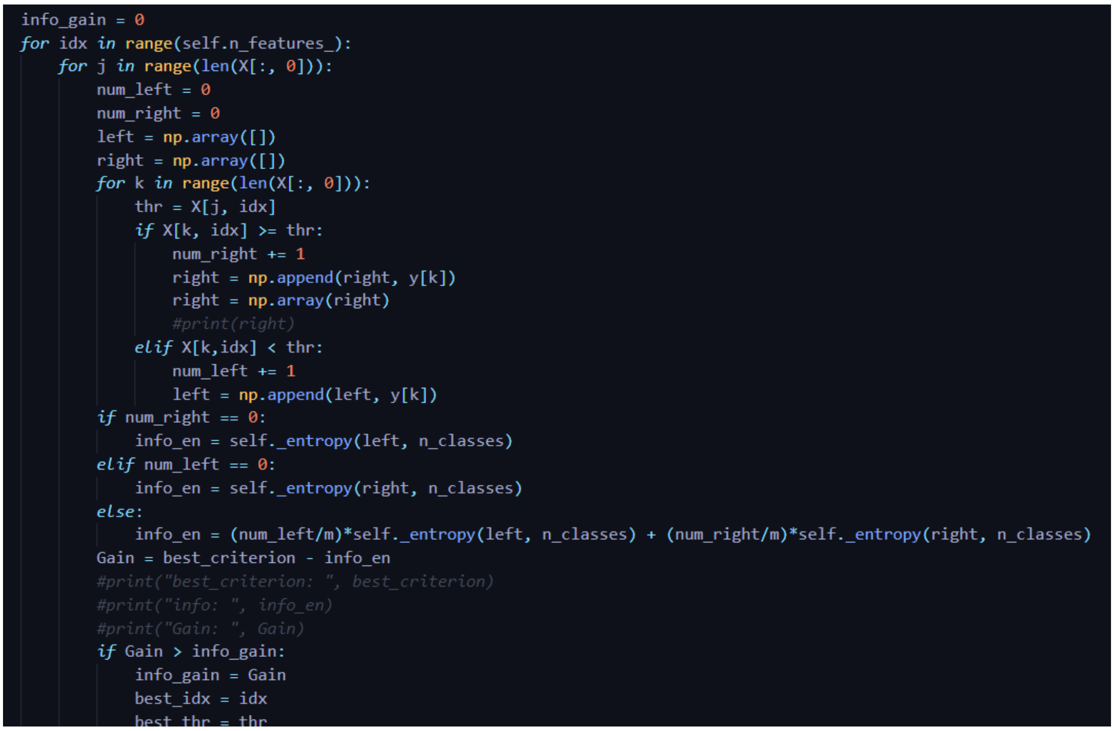
Reprot 108502561

1. Function

    1. _feature_split(): find best idx and thr so that we can spilt data the right way.

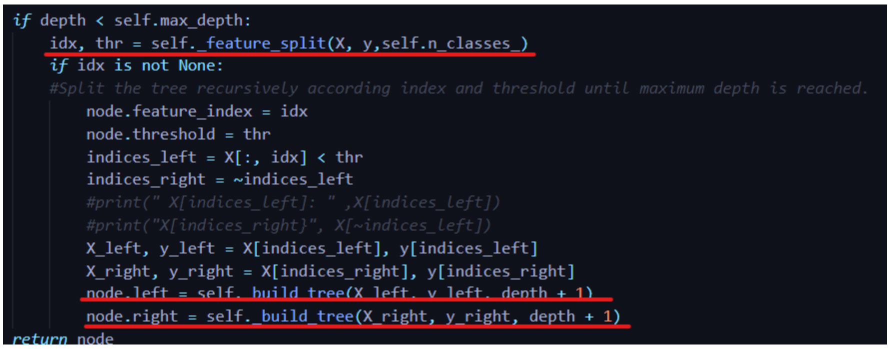
       I used three for loop here.

       here should be a better way, but my brain stop working



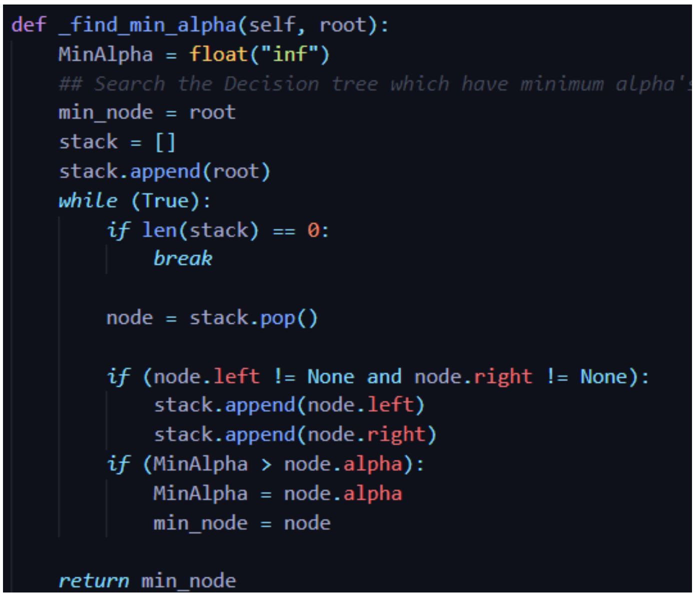
```
info_gain = 0
for idx in range(self.n_features_):
    for j in range(len(X[:, 0])):
        num_left = 0
        num_right = 0
        left = np.array([])
        right = np.array([])
        for k in range(len(X[:, 0])):
            thr = X[j, idx]
            if X[k, idx] >= thr:
                num_right += 1
                right = np.append(right, y[k])
                right = np.array(right)
                #print(right)
            elif X[k,idx] < thr:
                num_left += 1
                left = np.append(left, y[k])
        if num_right == 0:
            info_en = self._entropy(left, n_classes)
        elif num_left == 0:
            info_en = self._entropy(right, n_classes)
        else:
            info_en = (num_left/m)*self._entropy(left, n_classes) + (num_right/m)*self._entropy(right, n_classes)
        Gain = best_criterion - info_en
        #print("best_criterion: ", best_criterion)
        #print("info: ", info_en)
        #print("Gain: ", Gain)
        if Gain > info_gain:
            info_gain = Gain
            best_idx = idx
            best_thr = thr
```

    2. _build_tree(): we recursively call _build_tree() in this function, and with the help of _feature_split(), we can build a decision tree



```
if depth < self.max_depth:
    idx, thr = self._feature_split(X, y,self.n_classes_)
    if idx is not None:
    #Split the tree recursively according index and threshold until maximum depth is reached.
        node.feature_index = idx
        node.threshold = thr
        indices_left = X[:, idx] < thr
        indices_right = ~indices_left
        #print(" X[indices_left]: " ,X[indices_left])
        #print("X[indices_right}", X[~indices_left])
        X_left, y_left = X[indices_left], y[indices_left]
        X_right, y_right = X[indices_right], y[indices_right]
        node.left = self._build_tree(X_left, y_left, depth + 1)
        node.right = self._build_tree(X_right, y_right, depth + 1)
return node
```

3. _find_min_alpha(): As its name, This function would find min alpha which will be used to decide which node to cut.



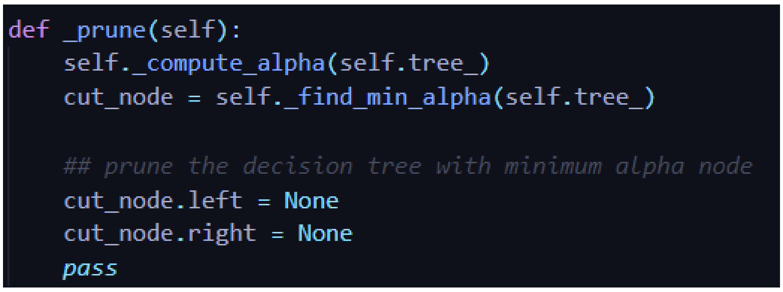
```
def _find_min_alpha(self, root):
    MinAlpha = float("inf")
    ## Search the Decision tree which have minimum alpha's
    min_node = root
    stack = []
    stack.append(root)
    while (True):
        if len(stack) == 0:
            break

        node = stack.pop()

        if (node.left != None and node.right != None):
            stack.append(node.left)
            stack.append(node.right)
        if (MinAlpha > node.alpha):
            MinAlpha = node.alpha
            min_node = node

    return min_node
```

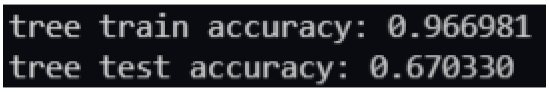
4. _prune(): prune node.



```
def _prune(self):
    self._compute_alpha(self.tree_)
    cut_node = self._find_min_alpha(self.tree_)

    ## prune the decision tree with minimum alpha node
    cut_node.left = None
    cut_node.right = None
    pass
```

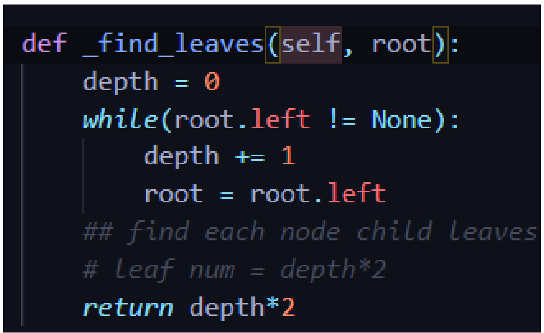
2. Decision tree before post pruning accuracy



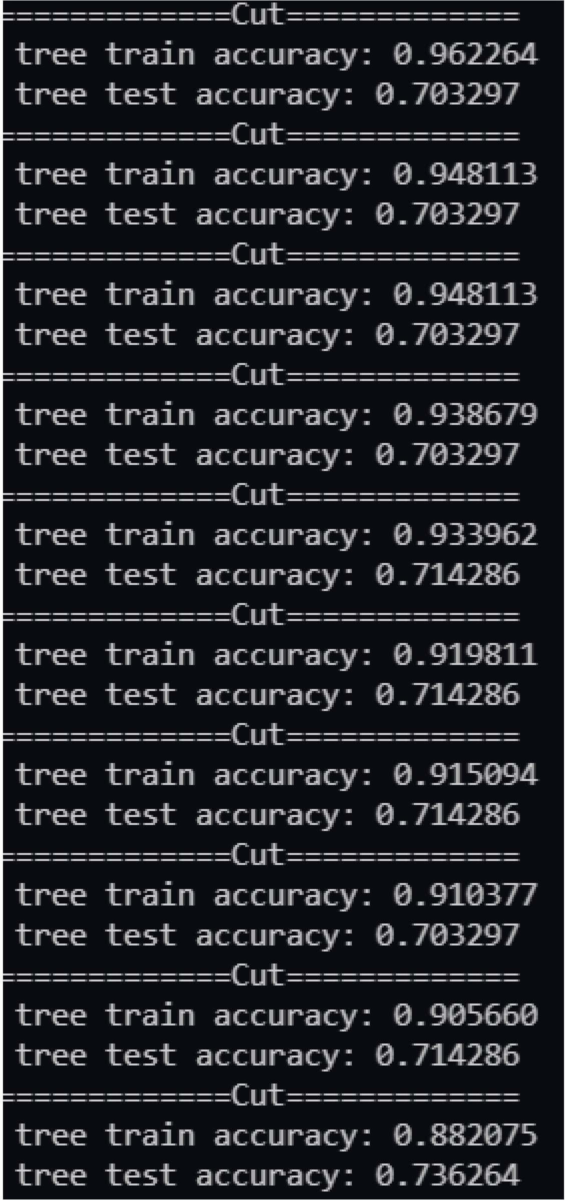
```
tree train accuracy: 0.966981
tree test accuracy: 0.670330
```

3. Decision tree after post pruning accuracy

If we use root.left here, we will get

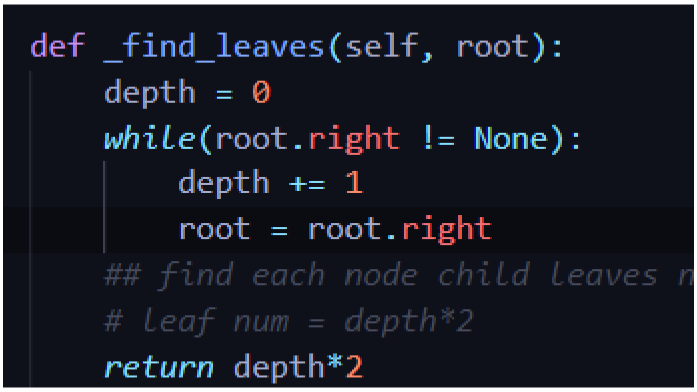


```
def _find_leaves(self, root):
    depth = 0
    while(root.left != None):
        depth += 1
        root = root.left
    ## find each node child leaves
    # leaf num = depth*2
    return depth*2
```

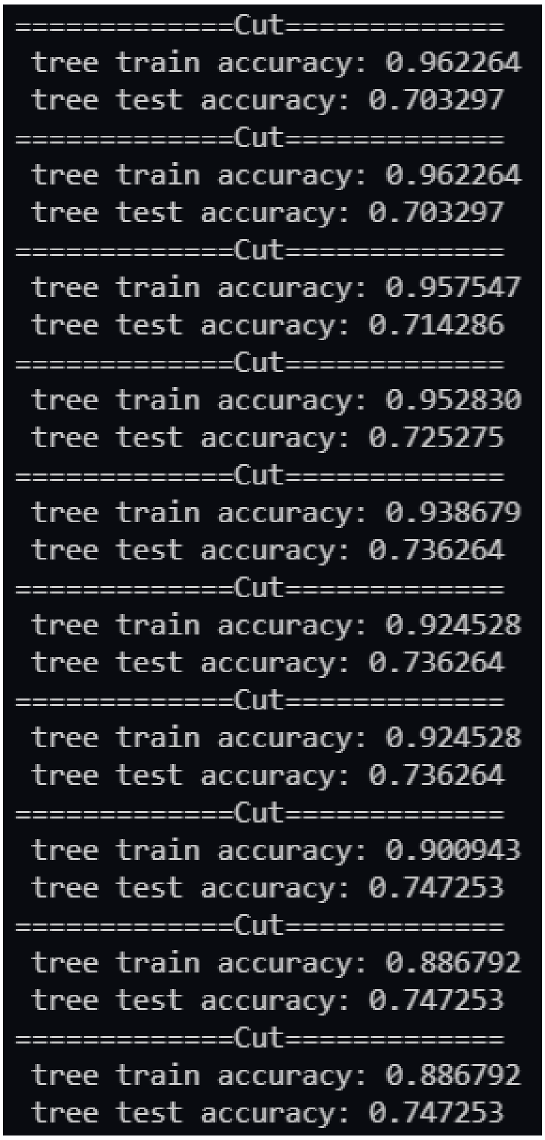


```
=============Cut=============
tree train accuracy: 0.962264
tree test accuracy: 0.703297
=============Cut=============
tree train accuracy: 0.948113
tree test accuracy: 0.703297
=============Cut=============
tree train accuracy: 0.948113
tree test accuracy: 0.703297
=============Cut=============
tree train accuracy: 0.938679
tree test accuracy: 0.703297
=============Cut=============
tree train accuracy: 0.933962
tree test accuracy: 0.714286
=============Cut=============
tree train accuracy: 0.919811
tree test accuracy: 0.714286
=============Cut=============
tree train accuracy: 0.915094
tree test accuracy: 0.714286
=============Cut=============
tree train accuracy: 0.910377
tree test accuracy: 0.703297
=============Cut=============
tree train accuracy: 0.905660
tree test accuracy: 0.714286
=============Cut=============
tree train accuracy: 0.882075
tree test accuracy: 0.736264
```

But if we use root.right here, we will get



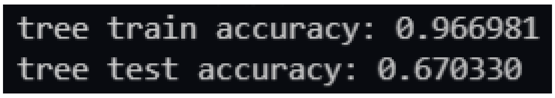
```python
def _find_leaves(self, root):
    depth = 0
    while(root.right != None):
        depth += 1
        root = root.right
    ## find each node child leaves n
    # leaf num = depth*2
    return depth*2
```



```
============Cut============
 tree train accuracy: 0.962264
 tree test accuracy: 0.703297
============Cut============
 tree train accuracy: 0.962264
 tree test accuracy: 0.703297
============Cut============
 tree train accuracy: 0.957547
 tree test accuracy: 0.714286
============Cut============
 tree train accuracy: 0.952830
 tree test accuracy: 0.725275
============Cut============
 tree train accuracy: 0.938679
 tree test accuracy: 0.736264
============Cut============
 tree train accuracy: 0.924528
 tree test accuracy: 0.736264
============Cut============
 tree train accuracy: 0.924528
 tree test accuracy: 0.736264
============Cut============
 tree train accuracy: 0.900943
 tree test accuracy: 0.747253
============Cut============
 tree train accuracy: 0.886792
 tree test accuracy: 0.747253
============Cut============
 tree train accuracy: 0.886792
 tree test accuracy: 0.747253
```

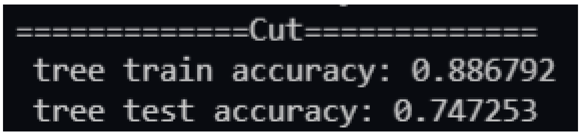
4. The effect of different parameters

Max_depth = 8



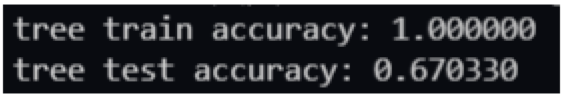
```
tree train accuracy: 0.966981
tree test accuracy: 0.670330
```

After 10 cut



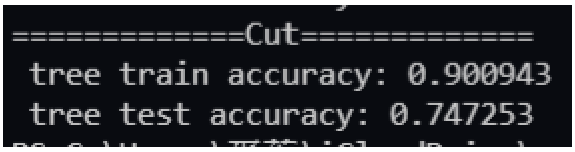
```
=============Cut=============
 tree train accuracy: 0.886792
 tree test accuracy: 0.747253
```

Max_depth = 16



```
tree train accuracy: 1.000000
tree test accuracy: 0.670330
```

After 10 cut



```
=============Cut=============
 tree train accuracy: 0.900943
 tree test accuracy: 0.747253
```

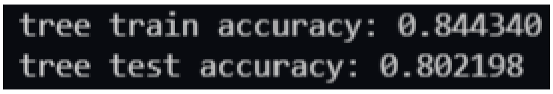
It seems like increase max_depth will increase train accuracy

But did not improve test accuracy

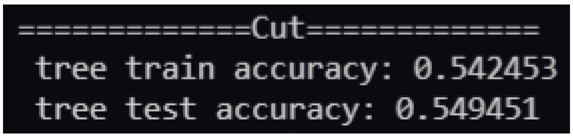
5.A brief discussion of the results(Ex: After prune tree, will the testing accuracy be better, if yes, why it would be better, if not, why it be worse?)

Will be better(if depth is enough), because prune can prevent overfitting

Depth = 4



```
tree train accuracy: 0.844340
tree test accuracy: 0.802198
```



```
=============Cut=============
 tree train accuracy: 0.542453
 tree test accuracy: 0.549451
```

When depth is too low

Prune didn’t help