```
    class BinaryNode(object):
    def __init__(self):
        self.left_child = None #the left child of the node, another
     node
        self.right_child = None #the right child of the node,
    another node
        self.samples = None #a 1D integer array of training sample
    indices "in" the node
        self.feature_list = None #a 1D integer array of features we
     have not yet split upon on the branch
        self.feature_split = None #an integer, the feature index
    for which we split on at the node
        self.label_prediction = None #an integer, 1 for yes, 0 for
    no the label prediction our DT will output at the particular
    node

class RealNode(object):
    def __init__(self):
        self.left_child = None #the left child of the node, another
     node
        self.right_child = None #the right child of the node,
    another node
        self.samples = None #a 1D integer array of training sample
    indices "in" the node
        self.feature_list = None #a 3D array of triples (
    feature_index, feature_value, feature_sign) for which we have
    not yet split upon on the branch
        self.feature_split = None #an integer, the feature index
    f3or which we split on at the node
        self.feature_split_value = None #a double, the value for
    which we split the feature on
        self.feature_split_sign = None #a string, less or leq resp.
     corresponding to either "<" or "<=", for which we split the
    feature value with
        self.label_prediction = None #an integer, 1 for yes, 0 for
    no the label prediction our DT will output at the particular
    node
```

Listing 1: Python example

These classes will form the basis for our tree. Clearly, we need to denote left and right children of each node in the tree, but we also need to keep track of which samples are "in" each node during the model's training so that we can form output labels and see what is going on during the debugging process. Further, we need to make sure during training we do not split on the same feature twice so we will keep track of un-split features as well as which feature we choose to split on. These properties will all be None upon the return of $\mathrm{DT}_{train_binary}$ $save for feature_{s}plit as the rest are only needed during training. In addition to this, the two node clas$