COM 307
Uyen Tran

PROJECT 3

In this project, I went through 6 steps in my code:

1. Find the split point and split feature
2. Build a tree
3. Build a forest
4. Predict using a tree
5. Predict using a forest
6. Compute the accuracy

I'll go into more detail about each step below.

Step 1:

- I use the Gini index to find the split point
- The split point is computed through 4 steps:
    1) Compute the Gini impurity of the original data. The Gini impurity is calculated using this formula:

> Consider a dataset D that contains samples from k classes. The probability of samples belonging to class i at a given node can be denoted as p_i. Then the Gini Impurity of D is defined as:
>
> $$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

    2) Compute the Gini index of each feature. The feature of the smallest Gini index is selected
    3) For that feature, I calculated the Gini index for each data point and select the one with the smallest Gini index.
    This is the formula that I used to calculate the Gini index in steps 2 and 3:

> If a data set D is split on an attribute A into two subsets D_1 and D_2 with sizes n_1 and n_2, respectively, the Gini Impurity can be defined as:
>
> $$Gini_A(D) = \frac{n_1}{n} Gini(D_1) + \frac{n_2}{n} Gini(D_2)$$

Step 2:

- So I build a tree based on the feature and split point I got from step 1. In this step, I first build a tree using recursion. The function is much shorter. However, I didn't know how to trace back and predict (in step 3) using this tree (which is a preorder traversal tree).

Therefore, I change the code and build the tree using a queue, which will return a level order traversal.

- For the number of features m, I did some research and figured out that the popular m is the square root of the total number of features, which is sqrt(34) ~ 6
- It's interesting that although I gave the tree 6 features to use, it only uses around 4 features and 1 of 4 features is used substantially more than the other features. This suggests what we've learned in class that a decision tree would use relevant features and eliminate irrelevant ones.

Step 3:

- I randomly select 6 features from 34 features and there's no repeated feature in a tree. However, there might be repeated features in multiple trees. For example, I use features = [1,**5**,12,33,**8**,23] for one tree and use features = [27,32,21,**8,5**,19] for another tree. This random nature suggests that the more tree I use, the more subjective the result would be.
- To build a forest, I just need to run step 2 n times, with n being the number of trees

Step 4:

- Predict using a tree. This part is tricky because at first, I didn't know how to predict now that I have a tree. What I did is using a level order traversal. For each split point, if the test data is smaller than the split point, then it goes to the (i*2 + 1)-indexed group in the traversal. Else, it goes to the (i*2 + 2)-indexed group in the traversal.

Step 5:

- Predicting using the forest is just aggregating all the results I get from the tree and then picking the majority. However, this part is sometimes interesting because the number of "good" and "bad" turns out to be equal at one time or another. To fix this, I build a forest with an odd number of trees.

Results:

- For testing and training, I use 80% of the data to train and 20% to test
- My model does a mild job with accuracy ranging from 55% - 82% (5 iterations) with 10 trees and I stop the tree when the total data in the len(df) reaches 50. I tried to play around with this number and plug in different numbers. As expected, the results get better when I tried 20 instead of 50 and it keeps getting better with 10. The accuracy with 10 trees and m = 10 is ranging from 70% - 81% (5 iterations).
- This time, I tried to use more trees. I increased the number of trees from 10 to 21. It performs better, with accuracy ranging from 65% - 90% (5 iterations)