第一大題:

1. \_feature\_split:

在此function裡，首先會求出此node還未被切割前的entropy也就是best\_criterion，然後會先分別對各個feature嘗試每種切法，每切完一次就會計算一次Gain(=best\_criterion-info)，以講義為例: 

然後記錄最大的Gain與此時的best\_idx和best\_thr，藉此找到最好的切法。

1. \_build\_tree:

在此function裡，首先會計算node裡最大的label數量，也就是correct\_label\_num，用總數量減去correct\_label\_num就會得到num\_errors，接下來便是將各種資料記錄在node上，之後會call \_feature\_split，取得best\_idx和best\_thr，再依這兩個數值去切隔成左右兩瓣，並重複把切出來的node放到\_build\_tree裡

1. \_find\_min\_alpha

顧名思義他的目標是找到最小的alpha，透過\_compute\_alpha裡的公式



用recursive的方法不停地將root.left 和 root.right放進\_find\_min\_alpha裡，並在\_find\_min\_alpha裡，紀錄在哪個node切掉，會得到最小的alpha，那這個node就是我們所要的可以損失最少正確性的切點。

1. \_prune

在這個function裡，會call \_find\_min\_alpha，去取得我們要切哪個node，並將這個node的左右node都設為None也就是切掉。

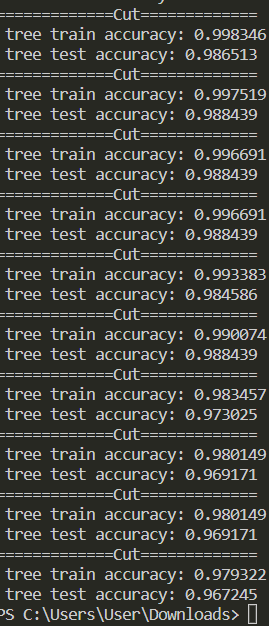
第二大題:

此為實驗結果:



第三大題:

此為實驗結果:

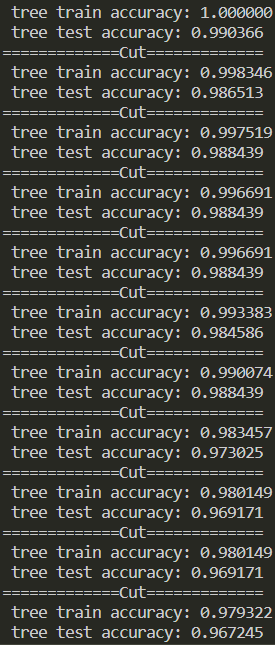


第四大題:

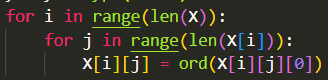
1. max\_depth: 這個變數決定樹可以建多深，若設定的太小會使data無法完全分類，也就會導致剛開始的tree train accuracy沒法為1，也會進而使之後的accuracy 都降低。
2. prune\_tree\_times: 裁減樹木的次數會決定切了幾個node，可以讓我們在accuracy 與 時間空間成本 做權衡。
3. test\_ratio: 可以決定要各拿多少%的資料來training 與testing
4. random\_state: 此為設定數據如何劃分的值，若值一樣便會得到一樣的劃分結果。

第五大題:

此為實驗結果:



由結果可發現:

1. 在還未post-pruning 前，tree train accuracy 為一，這是理想的結果，因為我就是依照這些training data 去訓練，理所當然tree train accuracy=1(經測試後MAX\_Depth至少要為14)，再來是tree test accuracy，我發現將X的value用以下方法處理可以得到較好的結果: 

且處理後對之後的運算比較沒後顧之憂，例如: np.bincount()的使用。

1. Tree train accuracy 經過每次的CUT呈現遞減的情況，這是合理的，因為你每prune一次就會損失一點accuracy，我們也能觀察到tree test accuracy 的變動，若prune完test accuracy 反而變高代表有overfitting 的情況產生。
2. 所以prune完，test accuracy 不一定會增高或減少，要取決於資料的分布。