本次報告所用的資料分別是 iris, wdbc, glass, wine

有無用 attribute bagging 的時間和精準度差別:

Iris(4 attribute):

```
The Time without Attribute Begging :  12.208340644836426
The CorrectRate without Attribute Begging :  0.9375555555555547
```

```
The Time with Attribute Begging :  6.3773276805877686
The CorrectRate with Attribute Begging :  0.9411111111111097
```

Wbdc(31 attribute):

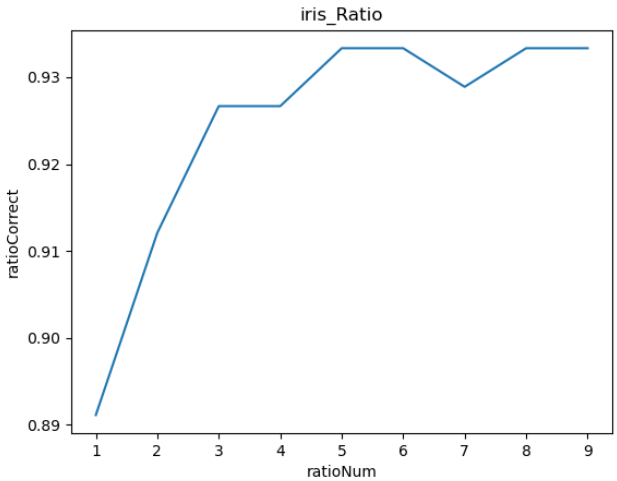
```
The Time with Attribute Begging :  151.05429983139038
The CorrectRate with Attribute Begging :  0.9228070175438589
```

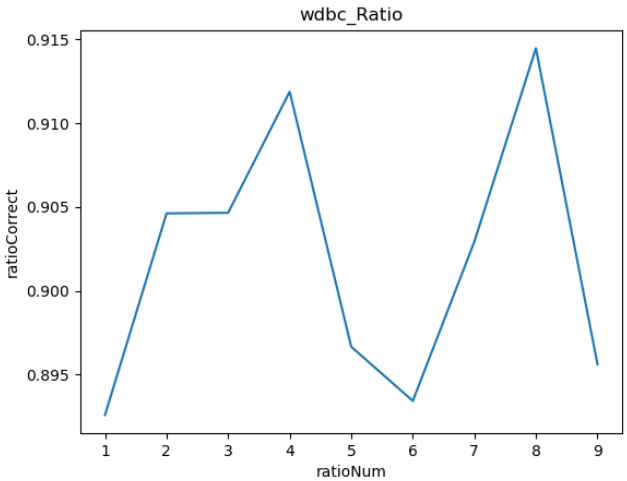
```
The Time with Attribute Begging :  878.6200737953186
The CorrectRate with Attribute Begging :  0.9249122807017539
```

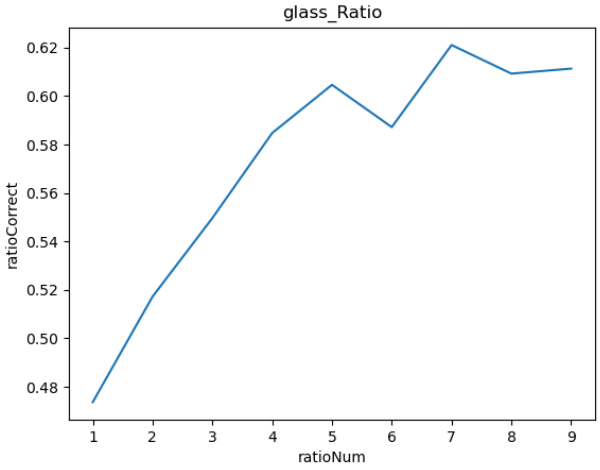
在 attribute 多的情況下，wbdc 的速度有著非常明顯的加速，但精準度基本上卻沒什麼差別，因此之後的實驗皆會採用 bagging。

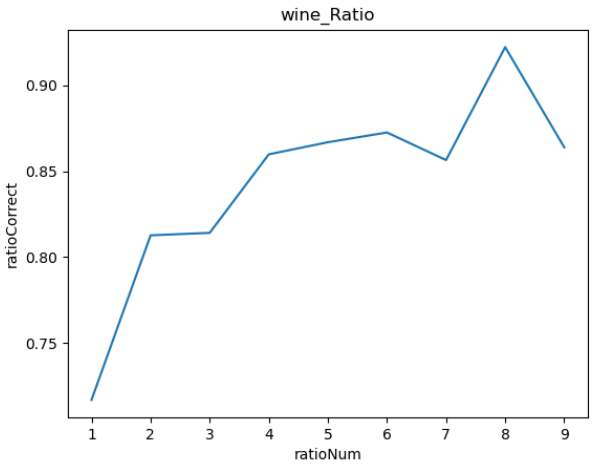
Relative sizes of the training and validation subsets:

※Random forest tree number=20



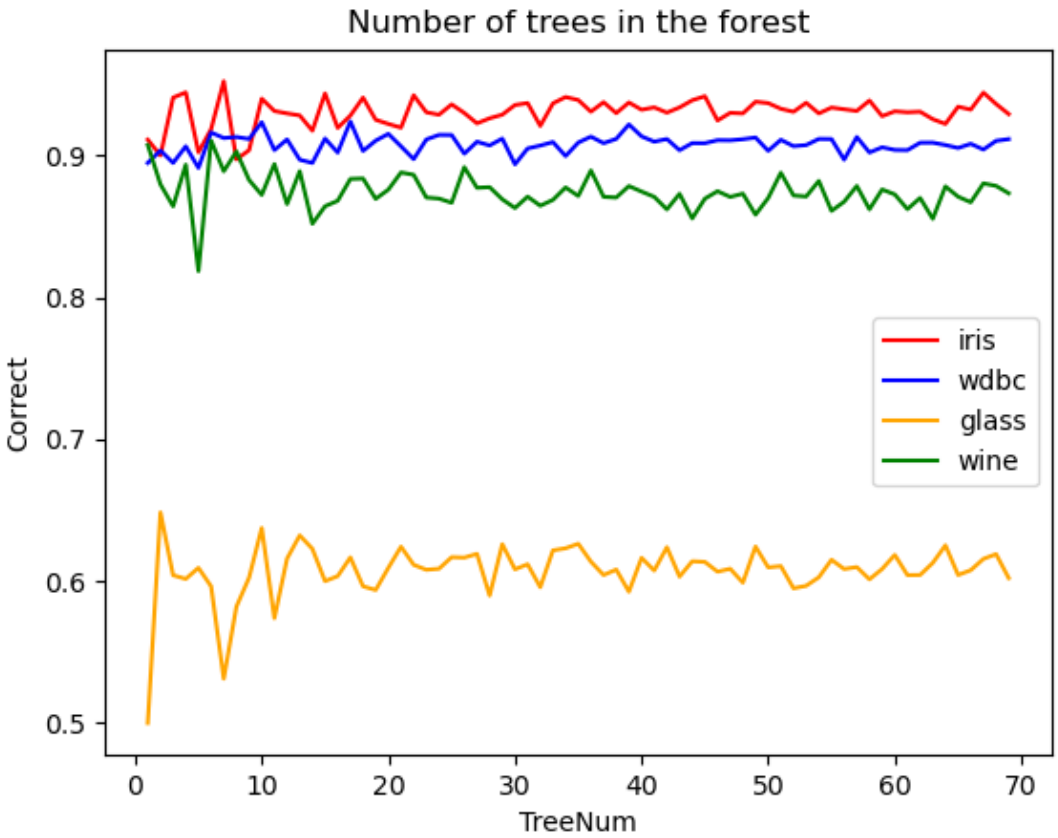






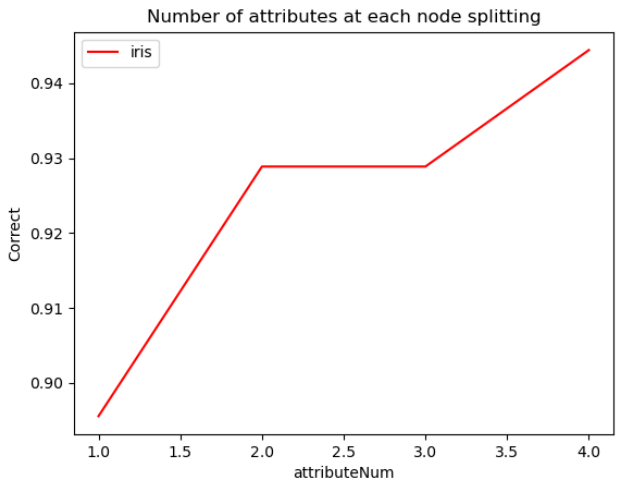
從兩種不同的 data 可以看得出來，當 Relative sizes of the training and validation subsets 大概到 7:3 的時候準度是最高的，而之後隨著比例愈高，準度很不穩定的原因我覺得是 validation subset 的數量太少了，很容易就 overfitting。之後測試的比例皆為 7:3。

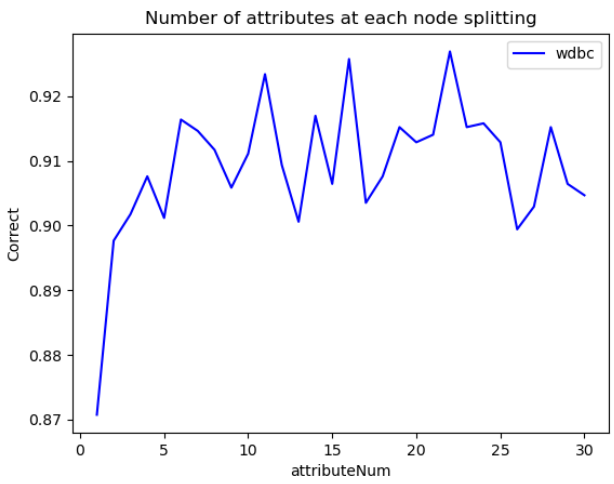
Number of trees in the forest:

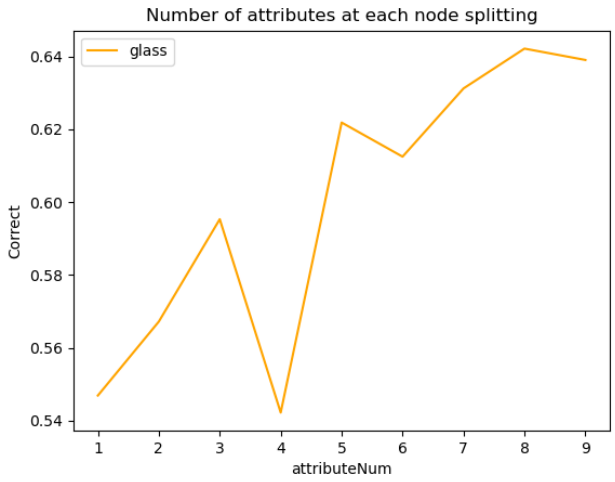


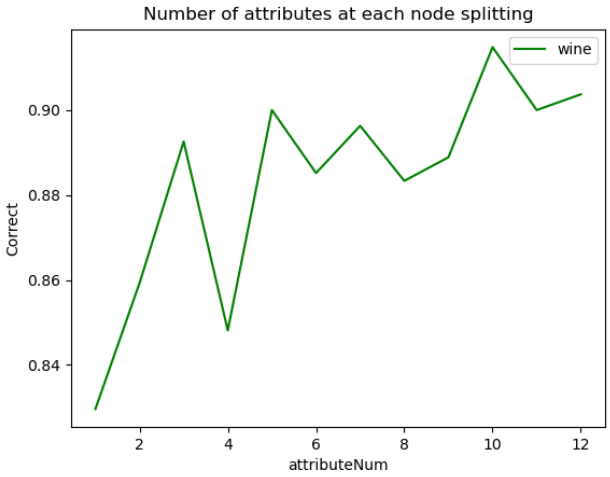
可以看出，隨著森林的數目越多，準確度越趨近於穩定。準確度沒有隨著樹的數量而提高，我覺得是因為 data 的數目太少了，所以即使樹再多，抽到同一批 data 的機率也蠻高的。因為 TreeNum 在這裡的影響不大所以下一個實驗就使用 TreeNum=10。

Number of attributes at each node splitting:



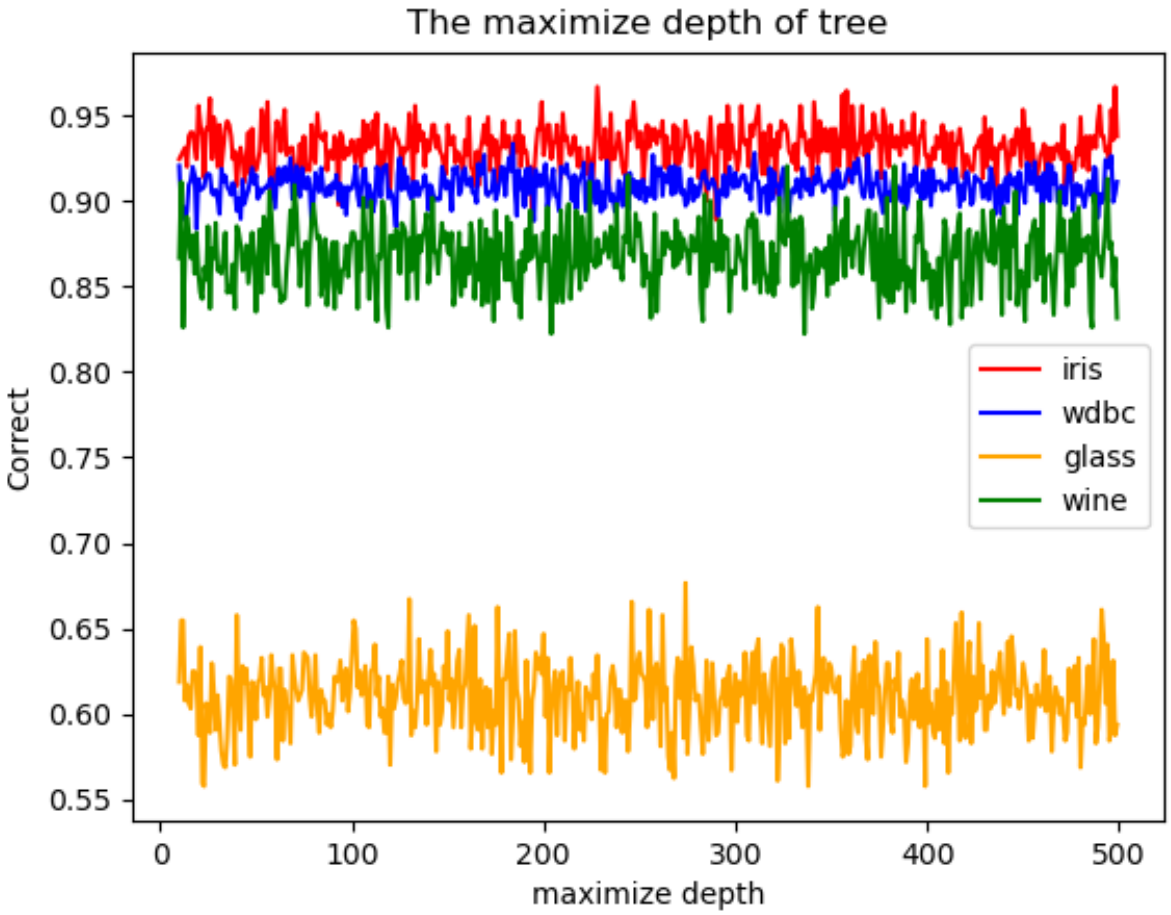






在直覺上，attribute 所採用的數量愈多應該會越準，而實驗出來的結果也在我的預期內，雖然每一個 data 幾乎都在採用所有的 attribute 時精準度都有稍微下降，但大致上來看，attribute 的確採用的越多準度越高。但即使如此，這所付出的時間代價太大了，從第一個實驗就可以知道，採用根號的數量和全部的數量，準度其實不會差太大，然而時間卻是平方倍，所以在準度與效率的取捨上，取根號還是最優的方法。

Methods that limit a tree's size: upper bound on the tree's depth:

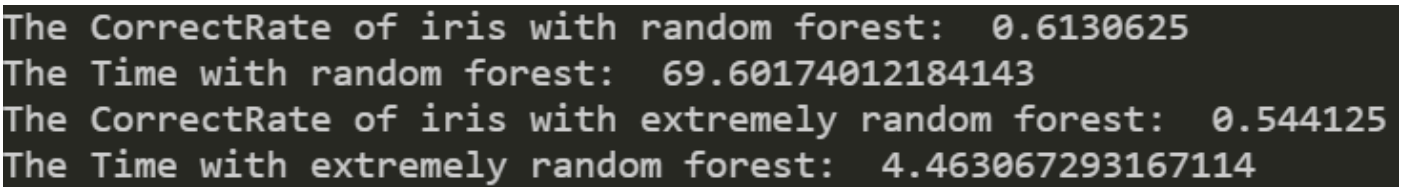


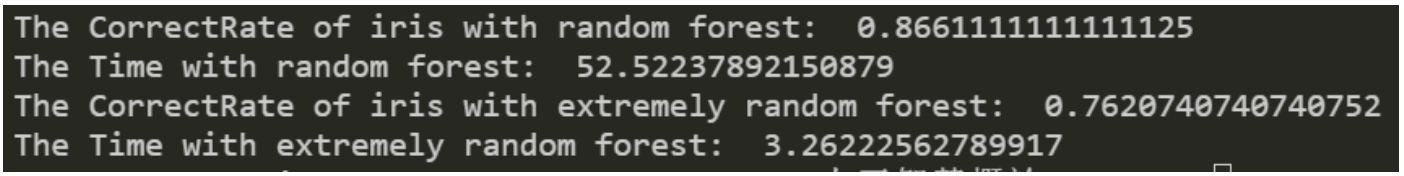
做出來的實驗結果跟自己預期的差距蠻大的...，本來想說深度越淺時，準確度越低，結果實驗顯示是差不多的，我想了一會兒，後來我試試看那些資料通常深度會到多深，結果發現幾乎都在 15 以內，所以最大深度的測試測到 500 是完全沒意義的...(跑了半小時以上)

Extremely random forest:

```
The CorrectRate of iris with random forest:  0.9322222222222231
The Time with random forest:  30.30561637878418
The CorrectRate of iris with extremely random forest:  0.8894666666666677
The Time with extremely random forest:  2.460723638534546
```

```
The CorrectRate of iris with random forest:  0.9072514619883083
The Time with random forest:  97.30204892158508
The CorrectRate of iris with extremely random forest:  0.8592397660818738
The Time with extremely random forest:  13.627638578414917
```





由上而下依序是 iris、wdbc、glass、wine 的結果。

結果顯示，extremely random forest 的精準度並沒有比 random forest 差太多，反而在時間上是 random forest 的好幾倍(畢竟省下了算 gini 的時間)，所以在取捨上，如果不是精準度要求很高，我覺得選擇 extremely random forest 的方法也是一個不錯的選擇。

Summary:

經過了上面 6 個實驗，大概可以得出結論，就是 forest 的 tree number 不需要太高大概是資料量的 10%就好了，而如果是追求運算效率的可以選擇 extremely random forest，而如果是追求準度則可以選擇 random forest，還有 train data : validation data=7:3 最好，最後選擇在 attribute 的數量上還是建議根號就好。

這次的作業雖然是我寫過最多行的作業，但也是我覺得最有趣的(除了 group project)，我想是因為寫起來沒有遇到太多困難，像之前幾次的作業，光是 Debug 的時間就比寫 Code 的時間還要來的多了…，這學期下來學到了很多人工智慧的東西，我覺得比起讀那些理論，實際做過一次印象才深刻，讓理論走出課本讓這門課變得十分有趣!

附錄:

我的 code:

https://github.com/KaivinC/AI_HW4