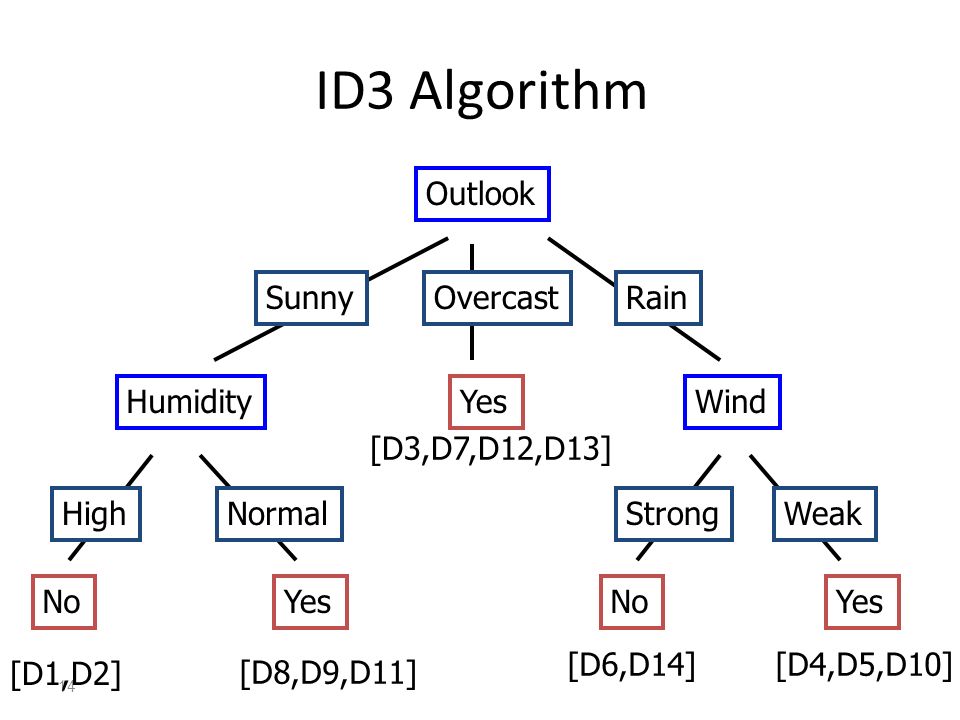
Introduction to Machine Learning Program assignment #1

Environment: Ubuntu 16.04.3 LTS

Using library: numpy、math、random

Language: Python 2.7.12

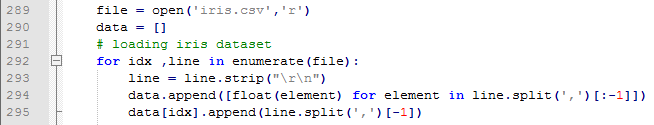
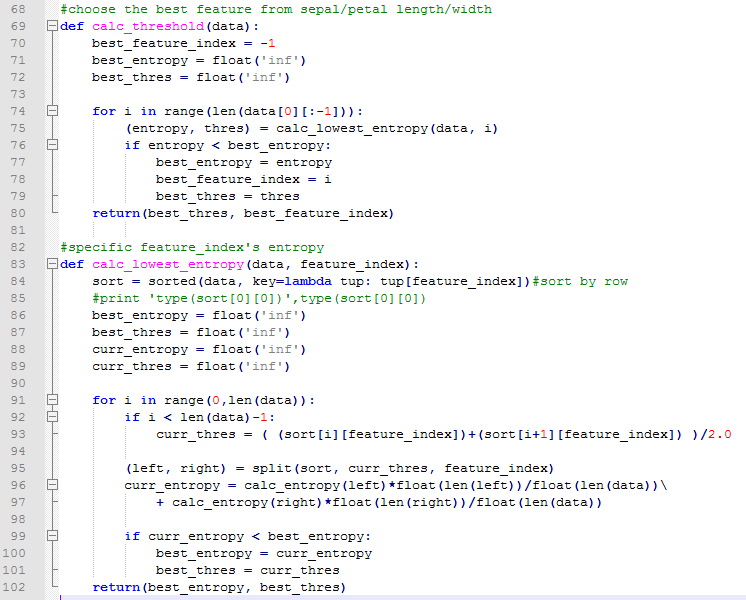
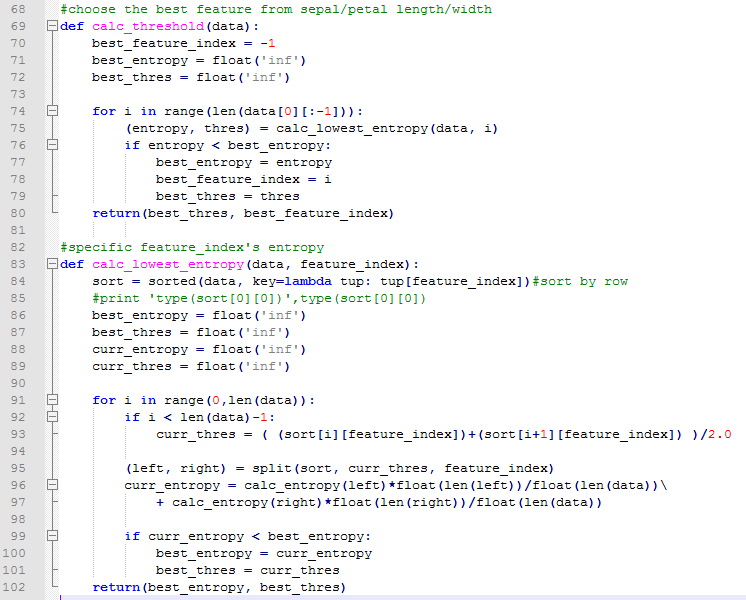
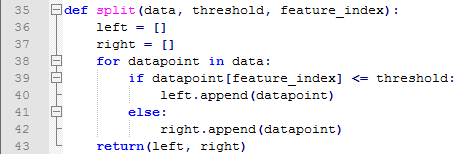
**ID3 Algorithm**



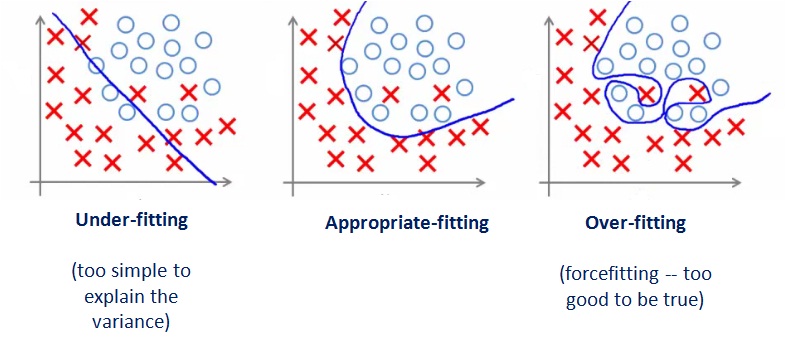
(圖片來源：http://mropengate.blogspot.tw/2015/06/ai-ch13-2-decision-tree.html)

Decision Tree可以用來分類，用information gain(越大越好)、entropy(亂度越小越好)選出最適合分類的feature當節點，把training data放上去計算，以分類、長分支，小於節點的接左子樹、大於節點的接右子樹，直到所有data都接上去，一棵Decision Tree便種好了！

**我的作法與使用的function**

1. (這次的作業我有參考github上Alex Yang先生的code─https://github.com/alxyang)
2. 用一個class─Node來定義節點  
   裡面有left、right、data、threshold\_indices、threshold、leaf、pure、label這些性質。  
   left、right代表那個節點有沒有左右子樹，因為大家一開始都是獨立的點，所以預設None。  
   leaf即樹的最末枝節點，pure代表該節點以下的資料都是同一種花種，一開始皆預設為true，因為是獨立的資料點，還沒接到樹上，自己當然跟自己一樣。
3.   
   main裡的Line289-295一開始先把資料用list的形式吃進來，因為資料是按照花種(Iris-setosa→Iris-versicolor→Iris-virginica)排好的，所以我們先使用shuffle打亂順序(要include random)，以免樹都是往某一個方向長。
4. 接著ID3 Algorithm內有用到find impure leaf、calculate threshold、spilt這些方法：  
   首先介紹find impure leaf (node)，這是一個recursive function，會從樹頂一直往樹的分支找，找出不純的地方，當左或右子樹分類不同時，便回傳那時候的節點(curr\_node)。  
   把curr\_node餵進calc\_threshold的function計算threshold值跟最好的feature\_index(得出結果是花瓣/花萼的長/寬之一)，再把threshold值跟feature\_index放到spilt的function以切割出資料的左右。最後把左、右子樹分別接上ID3\_Tree的左、右，我們就種完一棵大樹了！  
   以下將介紹calc\_threshold與spilt的function：
5. calc\_threshold(data)  
     
     
   這個function會選出並回傳(best\_thres, best\_feature\_index)。  
   Line82-102其中會call另外一個calc\_lowest\_entropy(data, feature\_index)的function，該function會先將data依照feature的值重新排序，每一個資料點跟資料點相加除以2當作threshold值做切割，我們便能分別計算左右子樹的entropy，從所有切割結果中挑一個entropy最小的，得出特定feature的(best\_entropy, best\_thres)並回傳進剛剛calc\_threshold的function內。  
   Line69-80用for loop跑四次，從各個feature的best entropy中再選出所有data裡最猛的，就能算出我們要的(best\_thres, best\_feature\_index)。
6. split(data, threshold, feature\_index)  
     
   這是用來切割左、右子樹的function，根據先前計算決定的feature\_index切好之後會黏到該datapoint的左、右子樹。

**Testing Result**

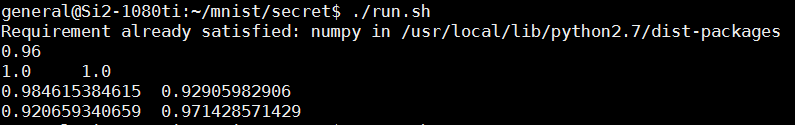


(圖片來源：http://mropengate.blogspot.tw/2015/06/ai-ch13-2-decision-tree.html)

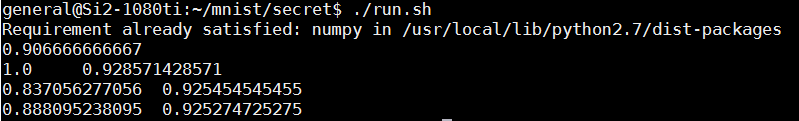
Iris dataset裡只有150筆資料，所以我們先將他打亂，再用K-fold validation切割出training set和testing set，testing的結果如下圖，90幾%的正確率已經非常高，若是追求100%之後可能會overfitting。  
然而，因為資料數實在太少，可以看出accuracy差異很大，每一次train的結果都不一樣，即使我test time設80次取平均，accuracy平均值最大仍可以相差到4%。

ID3 DecisionTree:

表現好時



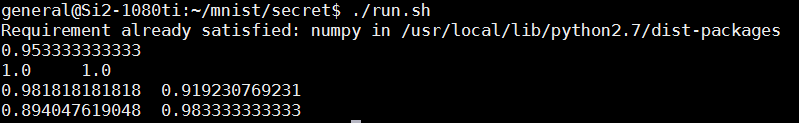
表現不好時



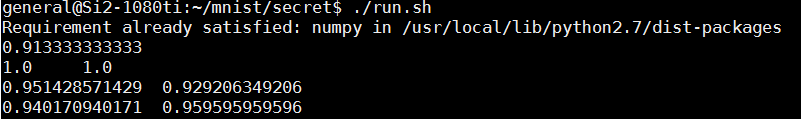
→可以看出總accuracy的誤差有5.4%之多

Random\_forest:

表現好時

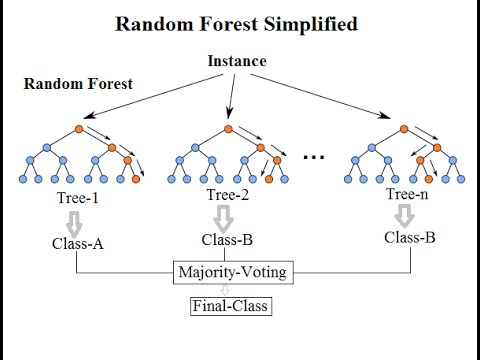


表現不好時



→可以看出總accuracy的誤差有4%之多

**Random\_Forest(bonus)**



(圖片來源：https://www.youtube.com/watch?v=ajTc5y3OqSQ)

用隨機的方式建構出一座森林，森林的每一棵樹都是由ID3 Algorithm種的，一樣用K\_fold分開training data和testing data，差異在於predict完結果之後，用voting的方式，森林裡的每一棵樹做投票，Iris-setosa、Iris-versicolor、Iris-virginica誰的票數多，就偵測那筆data是該花種。

**我的作法與使用的function**

和ID3\_DecisionTree幾乎一樣，但是Random\_Forest的Forest是由training set一次次打亂、每次取其中前40筆資料，餵進ID3種下一棵棵樹而組成。

而prediction的部分，則是要累加每一次預測的投票結果，根據樹們多數決來判斷testing datapoint的花種。

經過多次嘗試之後，我選accuracy相對較大較穩定的設定條件──森林裡有15棵樹，每棵樹40筆資料，黏成一座森林，即長度為15的list。

◎Predict\_and\_vote(datapoint, Forest)

Testing時把資料都丟到森林裡的每一棵樹裡面，跟著他們的分支走，走到底就是prediction的花種，再用array計算每一個label有幾票，票多的就是我們預測的結果。

