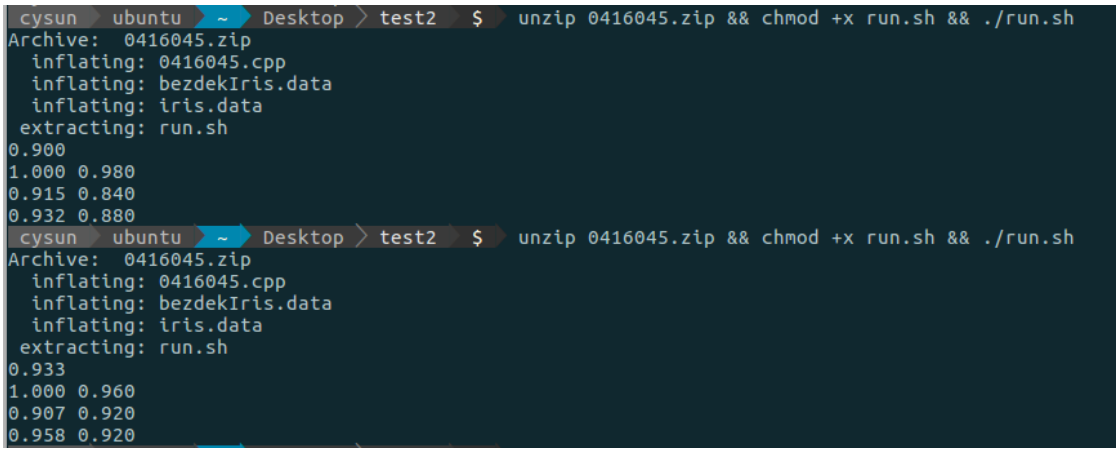
Machine Learning - assignment #1

0416045 孫嘉妤

程式運行結果

這次作業的程式用 150 筆的 Iris 資料，以 ID3 algorithm 建出分辨 Iris 品種的決策樹，並以 5 等份的 K-fold cross validation 驗證決策樹準確率。下圖為在我自己電腦執行時的截圖。



經過數次反覆執行，我觀察到的預測結果概述如下：

- Total accuracy
  - 大部份落在 0.90 ~ 0.935 之間，有時會掉到 0.86 ~ 0.90。
- Iris Setosa
  - 基本上是正確率最高的品種。Precision 經常為 1，Recall 也幾乎在 0.95 以上。
  - 我觀察資料發現，這樣的結果是因為 Setosa 的花瓣寬度（petal width）明顯較其他兩種小許多。
- Iris Versicolour
  - Precision 大部分在 0.9 ~ 0.95，偶爾也會出現 0.9 以下的數字。
  - Recall 表現較差，集中在 0.8 ~ 0.9，偶爾高於 0.9。
- Iris Virginica
  - Precison 通常略高於 Versicolour，以大於 0.95 居多。
  - Recall 較 Precison 低，大多在 0.85 ~ 0.95。

程式語言

> C++

從讀取資料、ID3 algorithm 實作到 K-fold cross validation，整份作業皆使用 c++。其中 include 的都是很普遍的 header，在不同版本的 C++ 下執行，結果應不會有太大差異。

使用程式庫

code 中使用的所有 header 如下：

```
// I/O
#include <iostream>
#include <iomanip>

// open file (read data)
#include <fstream>

// 字串處理
#include <string>
#include <cstring>
#include <sstream>

#include <vector>

// 數學計算
#include <algorithm>
#include <cmath>

// shuffle 用
#include <ctime>        // std::time
#include <cstdlib>      // std::srand
```

開發環境

在撰寫程式及除錯時主要使用 Visual Studio 2015，並在 windows 10 上運行。程式部份完成後才移至能執行 Shell Scripts 的 unbuntu 16.04 LTS 測試。此外也有在系上工作站測試，以上三者的執行結果並無明顯差異。

code 重點部份解說

shuffle

```
srand(unsigned(std::time(0)));
random_shuffle(iris.begin(), iris.end());
```

由於直接使用 c++ 內建的 random 函數無法達到真正的 random 效果，因此用 time 作為 seed 在每次執行時 shuffle 資料。

均分 K-fold

先區分出三種品種的 Iris，再將三類各自分成 5 份，例如 fold[0] = setosa[0] + Versicolour[0] + Virginica[0]；fold[1] = setosa[1] + Versicolour[1] + Virginica[1]。如此可以避免 fold 有極端的品種比例。

選取 Continuous attribute 的 feature

```
vector<double> feature(vector<Iris> iris, int a_type)
```

1. 將 iris 以指定 attribute 排序
2. 從排列後的 iris[0] 開始到最後項，每次 loop 比較 class 和先前是否相同
3. 若 class 改變便記下改變時的 attribute 值。
    - 這邊有特別處理的一點是由於單一特徵數字相同的 iris 算還不少，因此會避免記錄到重覆的 boundry。

4. 取所有相鄰兩 boundry 的中間值作為該 attribute 分類的 threshold。

以 ID3 建立 decision tree

```
Node* build_decision_tree(vector<Iris> iris,
vector<vector<double> > threshold)
```

1. 確認目前的 iris 是否已經都是同種類或 iris 數為零
    - 如果是，return 並記錄下 decision tree 的這個 node 是哪個品種
    - 若 iris 數為零，return 並記錄該 node 為 empty
2. 尋找 information gain 為最大之 decision tree 分支法
    - 先計算目前 entropy
    - 分別計算用四種特徵分類後之 remainder
    - 求出最大的 information gain
3. 以 IG 最大之特徵將目前的 iris 分成許多 part
    - 對於新分裂的 part，遞迴地重覆 build_decision_tree(part[i])
4. 記錄目前 node 是以何種特徵分支
5. 將各 part 間的 threshold 記錄在 node 中

在這個實作中，建立的 decision tree 類似下圖（圖來自課本），為非二元的 tree。每個 node 的分支數量視 feature 而定。