一、Introduction

使用 Data set：Adult data set

目的：使用 Decision tree、ANN、K-means 演算法分析 data set，盡可能地提高正確率，並在過程中探討所採取的策略對結果的影響原因。

原始參數：age , workclass , fnlwgt , education , education-num , marital-status , occupation , relationship , race , sex , capital-gain , capital-loss , hours-per-week , native-country , salary 共 14+1 項。

二、Experiment

⇨ 原始資料

Multilayer Perceptron：82.8%

Decision tree - J48：86.2%

⇨ 根據 Gain Ratio Attribute Eval 刪除價值較低的參數，剩餘 6+1 參數

Multilayer Perceptron：84.2% (上升 1.4%)

Decision tree - J48：85.8% (下降 0.4%)

⇨ 發現 capital-gain 中含有 99999 的資料，假設其為 dirty data，刪除

Multilayer Perceptron：84.6% (上升 0.4%)

Decision tree - J48：85.7% (下降 0.1% 不如預期)

⇨ 類神經網路 – 改變 Hidden layer 的 node 數 (6+1 參數預設 9 node)

7 node：84.6% 11node：84.9%

類神經網路 – 兩層 hidden layer

第一層 9node 第二層 4node：84.85% (差距微小)

⇨ 決策樹 – 改變 minNUMobj (葉子上最少要有多少資料量)

：輸入 2~10 的差別並不大，都在 85.6%徘徊

Leaves：49 Size：88

決策樹 – 再改變 Binary Splits 為 True

minNUMobj 為 2：85.81% minNUMobj 為 4：85.78%

Leaves：41 Size：81

決策樹 – 再改變 unpruned 為 True

minNUMobj 為 2 和 4：85.6%

Leaves：82 Size：163

最終決定之 Model：

演算法：Decision tree –J48

參數：education-num , marital-status , relationship , sex ,
capital-gain , capital-loss , salary 共 6+1 項

Preprocess：刪掉 capital-gain 參數為 99999 的資料

Binary Splits：True

minNUMobj：4

unpruned：False

其餘為預設值

準確率：85.8%　Leaves：41　　tree size：81

Time taken to build model：0.6 Seconds 左右

ROOT 為：Marital-status

```
=== Summary ===

Correctly Classified Instances       27802               85.8033 %
Incorrectly Classified Instances      4600               14.1967 %

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.951     0.442      0.874     0.951     0.911      0.862    <=50K
                 0.558     0.049      0.781     0.558     0.651      0.862    >50K
Weighted Avg.    0.858     0.349      0.852     0.858     0.849      0.862

=== Confusion Matrix ===

     a     b   <-- classified as
 23516  1204 |     a =  <=50K
  3396  4286 |     b =  >50K
```

三、Discussion

(一) Multilayer Perceptron 類神經網路

1. 資料量大時ANN十分耗時，為了改善它十分耗時這件事，最直觀的方法就是刪減Data set，但顯然不可能刪除一筆一筆的資料，因此轉為減少參數，使用Select attributes中GainRatioAttributeEval演算法，發現有不少資料的重要性偏低，經過刪減測試、再刪減測試，最後選出六種較為重要的參數，不只運算時間大大減少（ 半小時內可做完 ），也讓正確率微幅上升2%左右，表示在資料中這六項參數的 “W_{ij}” 比較高，價值也較高。

2. 嘗試改變感知元的數量 (原本為 9 個點)
    - ⇨ 減少感知元數量 3 4 5 6 7 8 個，最後的結果都與原先 9 個點差距微小 (1%以內)，但減少感知元明顯可以加快運算速度，而速度是類神經網路的缺點，因此是個不錯的選擇。
    - ⇨ 增加感知元數量至 10 11 個，但同樣發現準確率並沒有明顯的提升，仍然在 84%~85%之間徘徊，但運算時間增加，推測感知元 9 個為兼顧速度與準確度的中間值。

3. 不論用多少個感知元，有個共通點是，一定會有一個 NODE 對 output 的影響力最大 (W_{ij} , W_{jk} 較大)，而影響這個 NODE 最大的參數必為 capital-gain，完全超越其他參數，有接近十倍的差距，因此可知 capital-gain 在整個資料當中有相當重要的地位。

(二)Simple K-means

1. 分群結果發現所有 >=50K 的分群中，capital-gain 平均值都大於 2000 甚至到 4000。(capital-gain 平均值為 1000 左右)

(三) Decision – tree - J48

1. 準確率最高的 ROOT 為 capital-gain，當作 ROOT 時，得到過史上最高正確率 90.3% ，但 Tree size 為 2147，Leaves 1074 比 6 參數的 MODLE 多很多 (使用原始 Data、Use training set、binarySplits 為 True、unpruned 為 True、minNumObj 為 4)，推測原因為 capital-gain 在資料中具有最大的參考價值，當作 ROOT 可將資料分得更好，而二分法和 unpruned 讓 Tree 變得更大更長，分類的條件更多，雖說讓成功率上升不少，但有 overfitting 的嫌疑。

2. 為避免決策樹過大，刪除價值低之參數降為 6 參數，此時正確率下降 2%，但在可接受範圍內。在 Preprocess 中發現 capital-gain 有極大值 99999，推論其為雜訊，嘗試將所有 99999 的資料都刪除，結果不如預期，但仍使正確率微幅上升 0.5%左右，推測原因為決策樹分枝時是以大於小於區分，並不是將數字本身拿來計算，因此雖然 99999 比絕大部分的資料還要大很多，但並不影響 Decision tree 的運作，但在 K-means 可能就會造成影響。

3. 嘗試將 J48 中的參數 minNumObj 調高，但發現超過 100 時準確率逐漸下降，到 5000 時已經低於 80%，6000 時到達 76%，呈幾何下降，推測為葉子上的資料量仍太多，產生還沒分完就結束的情形，所以正確率下降。

4. 將 unpruned 設為 Ture 原本預計正確率會下降且 leaves 和 size 會暴增，但結果並不如預期，正確率幾乎無變動，Leaves 和 size 只增加一倍，推測原因為參數降至 6 項，造成原本的 Tree 本身就不大，裁剪效果沒有原始 DATA 的好 (原始 DATA 有無裁剪相差 10 倍左右，成功率可上升 1%~2%)。

5. 最終 Model 是以 marital-status 為 ROOT，並非參考價值最高的 capital-gain，推測原因是 marital-status 參數中總共有 14844 人是 Married-civ-spouse，而總資料量為 32402，再加上 Model 有二分法的限制，marital-status 可以較為剛好的將所有資料大致分為兩半，因此系統選擇 marital-status 當作 ROOT。

四、小結

1. 刪除一些資料價值較低的參數，可以做出更精簡的 Tree 更有利於分析，且可以大幅減少計算時間，更有效率。
2. 找出 DATA 中少數的 dirty data (例如有"?"的資料)將其刪除，若這種資料過多，可以考慮選擇對雜訊免疫力較高的演算法，如 Decision tree。
1. Hidden layer 的 node 數，可以設定為 input layer 的一半左右。
2. 一層的 hidden layer 已經足夠，設置太多不必要的層會降低運算速度。