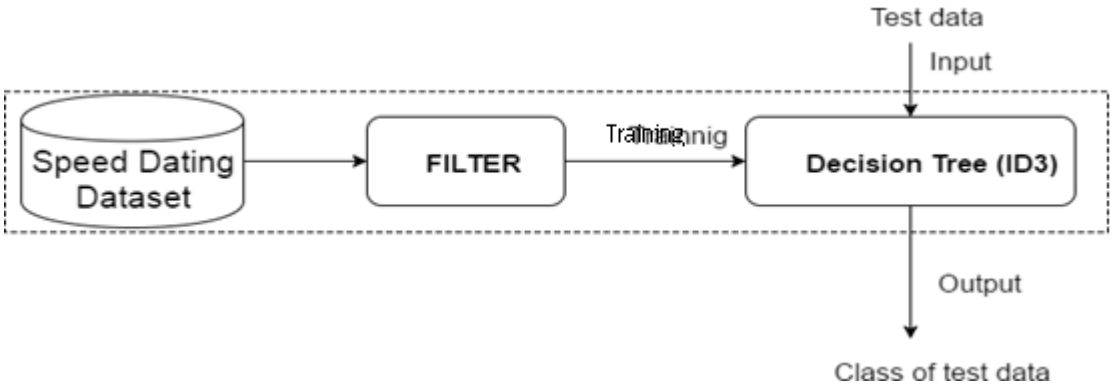
本 Project 設計了一些分類規則，他們對應了 absolutely rules。主要以狗來做實驗，定義了 k 個 features(長毛/短毛、單色/多色、主要顏色、立耳/垂耳、體型)，其中 k 為變數。通過這些規則產生一些資料，我們把產生的資料量數量定義為 M，然後實作 Decision Tree 和使用 weka 的 decision tree 於 M，以此得到一個分類模型。最後把模型產生的 rules 和 absolutely rules 作比較。

一、Decision tree classifier

- **系統架構**

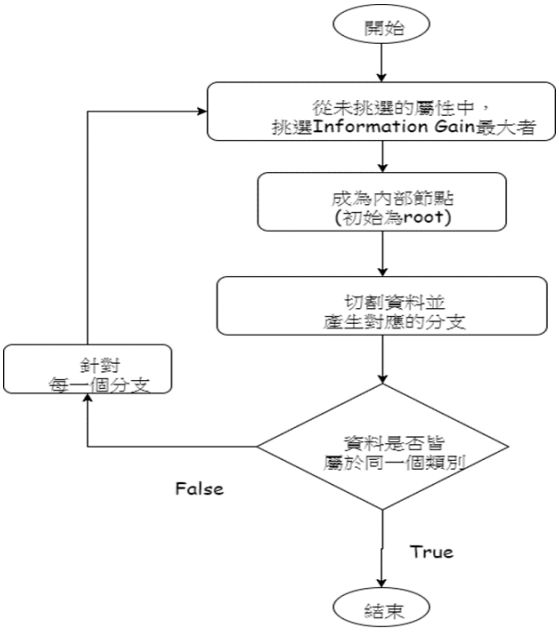


圖一、系統架構圖

圖一為系統架構圖，我們使用根據規則產生的資料集當作此 Project 的訓練資料，由於可以考慮的屬性太多，這裡把 k 值設為 5，採用五個屬性，再將過濾後的資料分成建模資料和 right rules。

再來用資料裡面的一部份充當 absolutely rules。系統可分為兩個階段: 1.**訓練階段** 2.**比較階段**，訓練階段先 input 訓練資料，讓演算法進行訓練與建立好決策樹，然後比較階段把 rules 拿去做比較。

- **實作細節**



圖二、建立決策樹之流程圖

訓練階段建立決策樹之流程如圖所示，由三個步驟不斷遞迴執行，流程詳細說明如下：

(1) 挑選屬性

從未挑選的屬性中，挑選分類效果最好的屬性來進行切割與分類，由於實作的是 ID3 的版本，所以使用**資訊獲利**（**Information Gain**）來當作分類效果的準則，當某屬性資訊獲利結果越大，代表使用該屬性分類資料後凌亂程度越低，分類效果越好，所以挑選**資訊獲利最大的屬性**來當作分類屬性，假設屬性 A 可將資料集合 S 分割成 v 個不同的子集合$\{S_1, S_2, \dots, S_v\}$，則屬性 A 的資訊獲利計算方式如下：

$$\text{Gain(S, A)} = \text{Entropy(S)} - \sum_{j=1}^{v} \frac{|S_j|}{|S|}) \text{Entropy(S)}$$

其中亂度(Entropy)為資訊量的凌亂程度指標，當亂度計算結果越高，代表資訊凌亂程度越高，其定義如下：

$$\text{Entropy(S)} = -\sum_{j=1}^{v} p(j|t) \log p(j|t)$$

$p(j|t)$ is the relative frequency of class j at node t.

(2) 切割資料

根據上一個階段所挑選的屬性，將資料進行分類，並產生對應的分支。

(3) 檢查是否達到終止條件

針對每一個分支檢查，若分支內的資料皆屬於同一個類別，則停止遞迴，否則針對每一個分支持續上述兩個步驟。

二、實驗結果與效能分析

- **開發環境**

過濾後的資料集：

資料數量	8378筆
屬性數量	27個

實驗環境：

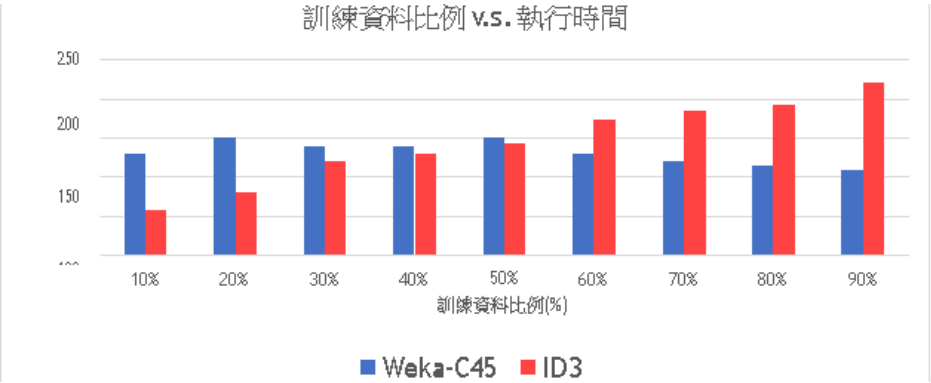
實作語言	Java
CPU	intel-i5-3570 3.4GHz
Memory	8GB
OS	windows 10x64

實驗內容：

- 評估變數：feature 數量 k, 資料量 M
- 比較演算法：Decision Tree(ID3) v.s. weka

實驗

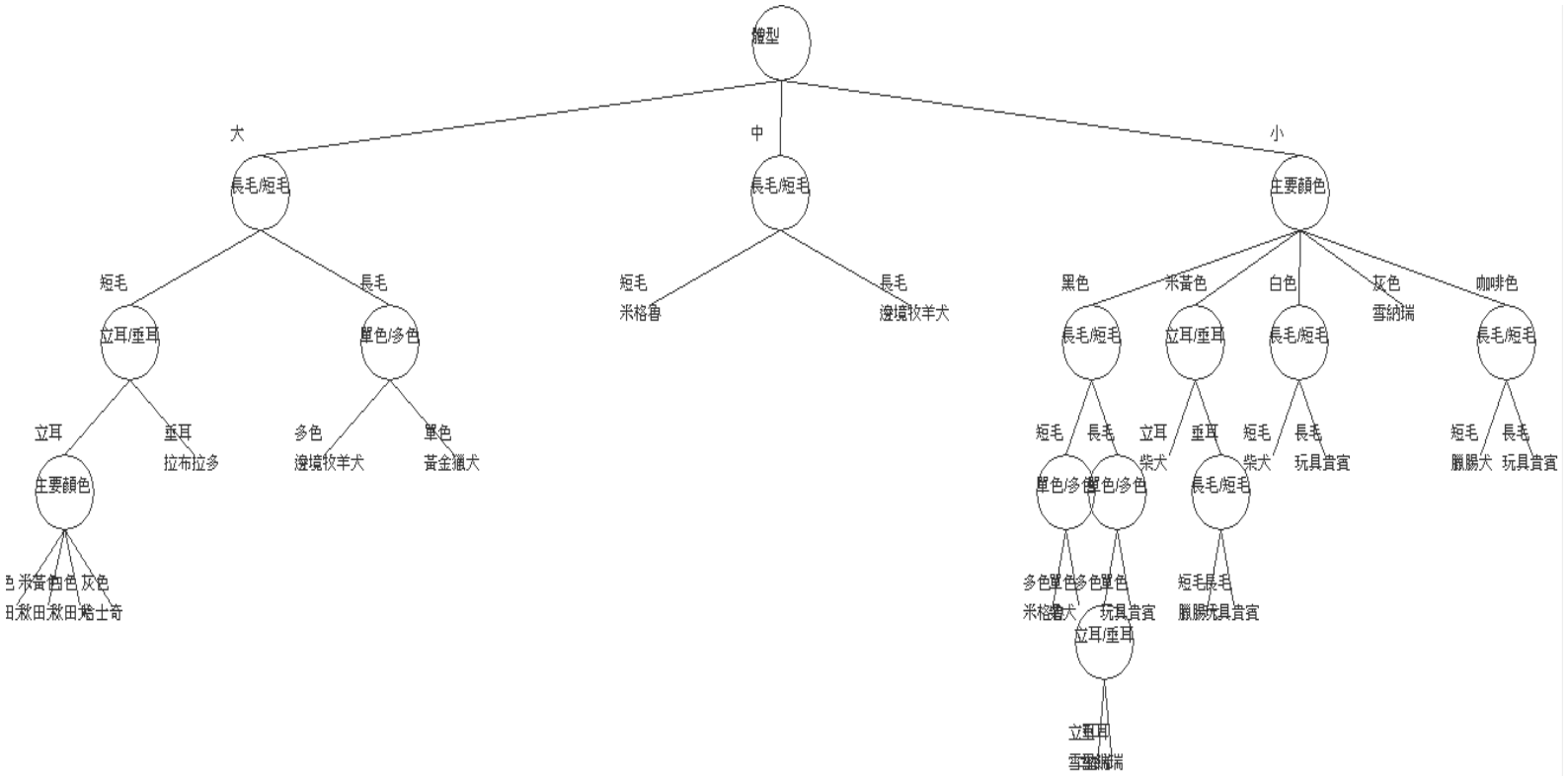
在此實驗調整件數訓練資料占整體資料之比例，比較我們實作的決策樹(ID3 版)與 weka 內的決策樹(C45 版)的執行時間比較，實驗結果如圖三所示。



圖三、執行時間比較

由實驗結果可見，weka 的決策樹的執行效率較不受訓練資料的數量影響，實作的決策樹受訓練資料比例影響的程度較大，原因在於我建構決策樹的實作方法上，沒有做結構上的優化，使用了較多的迴圈來計算亂度與資訊獲利，所以當 訓練的資料量一旦變大，就需要花較多的時間在對資料重複計算，而 weka 所實作的版本可能有做效能上的優化，因此影響的程度較低。

當 k = 5，M = 100 時，跑出來的結果如圖四、圖五所示：



圖四、Decision tree 結果

長毛/短毛	單色/多色	主要顏色	立耳/垂耳	體型	output
短毛	單色	米黃色	立耳	小	柴犬
短毛	單色	米黃色	立耳	小	柴犬
短毛	單色	白色	立耳	小	柴犬
短毛	單色	米黃色	立耳	小	柴犬
短毛	單色	米黃色	立耳	小	柴犬
短毛	單色	黑色	立耳	小	柴犬
短毛	多色	黑色	垂耳	小	米格魯
短毛	多色	黑色	垂耳	中	米格魯
短毛	多色	黑色	垂耳	小	米格魯
短毛	多色	黑色	垂耳	中	米格魯
短毛	多色	黑色	垂耳	中	米格魯
長毛	單色	米黃色	垂耳	大	黃金獵犬
長毛	單色	米黃色	垂耳	大	黃金獵犬
長毛	單色	米黃色	垂耳	大	黃金獵犬
長毛	單色	咖啡色	垂耳	大	黃金獵犬
長毛	單色	咖啡色	垂耳	大	黃金獵犬
長毛	單色	米黃色	垂耳	大	黃金獵犬
長毛	單色	米黃色	垂耳	大	黃金獵犬
長毛	單色	米黃色	垂耳	大	黃金獵犬
長毛	單色	咖啡色	垂耳	大	黃金獵犬

長毛/短毛	單色/多色	主要顏色	立耳/垂耳	體型	output
短毛	多色	黑色	立耳	大	哈士奇
短毛	多色	咖啡色	立耳	大	哈士奇
短毛	多色	咖啡色	立耳	大	哈士奇
長毛	單色	白色	垂耳	小	玩具貴賓
短毛	多色	黑色	垂耳	中	米格魯
短毛	單色	米黃色	垂耳	大	拉布拉多
長毛	單色	米黃色	垂耳	小	玩具貴賓
短毛	單色	米黃色	垂耳	大	拉布拉多
短毛	多色	米黃色	立耳	大	秋田犬
長毛	單色	咖啡色	垂耳	小	玩具貴賓
長毛	單色	黑色	垂耳	小	玩具貴賓
短毛	多色	黑色	垂耳	中	米格魯
短毛	單色	黑色	立耳	小	柴犬
短毛	多色	米黃色	立耳	大	秋田犬
短毛	多色	灰色	立耳	大	哈士奇
短毛	單色	米黃色	立耳	小	柴犬
短毛	單色	米黃色	立耳	小	柴犬
短毛	多色	黑色	立耳	大	哈士奇
短毛	單色	咖啡色	垂耳	小	臘腸犬
短毛	單色	咖啡色	垂耳	大	拉布拉多

圖五、資料規則結果

如圖五，左半邊為建樹的資料規則，右半邊為原有的規則。我們發現 absolutely rules 產生的資料，建立模型後產生的規則會有很大變化。相似率只有百分之八十左右，分析可得 k 越小，更容易出錯誤，但若將 k 逼近於 1，那得到的結果就沒有任何意義了，k 太大就會出現很多不必要的因素，導致決策樹效率降低。一隻狗的重要屬性應該是 10 個以內，像 data 4 中的數據屬性會有二十多個，其中有些就是無關緊要的了。我覺得對於 k 來說，曲線應該是趨近於常態分佈，這樣得出的規則做比較相似度最高。而數據量 M 要盡可能的大，這樣才會更加的準確。

四、結論

- **自己寫的 Decision Tree**
  - 缺點：
    1. 準確度較低
    2. 執行速度受訓練資料數量影響大
    3. 無法有效處理連續型資料
  - 改善方法：
    1. 使用 C45 版本中 Gain Ratio 來取代 ID3 的 Information Gain，並實作剪枝技術，避免 C45 版本產生 Overfitting 的問題。
    2. 在實作後重構並優化對資料的處理方法，透過資料結構與演算法減少讀取與重複計算的問題，提升執行速度。
- **心得：**

決定自己寫個簡單的決策樹是因為題目給予很大的自由，決策樹的實作較簡單且找出來的規則也較直覺，準確度也高，而影響決策樹效率的關鍵在於透過什麼樣的方法來找出分類效果較好的屬性，在這點上就產生了許多種版本的決策樹，我選擇了 ID3 的版本來實作。對於自己給定規則再

去生成資料去構建模型，再把模型做出來的規則與原規則去比較，我始終沒有明白其中正真的含義並且也不懂這麼做到底是為了什麼。不過我的決策樹遠不如 weka，可能我沒有專注在設計演算法的部分，而專注在設計資料集的部分而忽略了。