決策樹形式知識之線上預測系統架構

An On-Line Decision Tree-Based Predictive System Architecture

馬芳資

龍華科技大學資訊管理系講師

Fang-Tz Ma

Lecturer, Department of Information Management
Lunghwa University of Science and Technology
Email: ponymar@mis.nccu.edu.tw

林我聰

國立政治大學資訊管理學系教授

Woo-Tsong Lin

Professor, Department of Management Information Systems
National Chengchi University
E-mail: lin@mis.nccu.edu.tw

關鍵詞 (Keywords)：**決策樹**（Decision Tree）；**知識整合**（Knowledge Integration）；**線上預測系統架構**（On-Line Predictive System Architecture）

【摘要】

本研究提出一個決策樹形式知識的線上預測系統架構，其主要的目在於提供一個 Web-Based 的知識發掘(Knowledge Discovery, KD)及線上預測系統，而我們藉由使用這個系統可以進行歸納學習出決策樹形式的知識，並且在線上使用決策樹的知識來做分類和預測的工作。它的組成元件包含三個子系統：知識學習子系統、合併選擇決策樹子系統、線上預測子系統；三個儲存庫：決策樹知識法則庫、例子資料庫、和歷史知識法則庫；以及三個導入知識法則的介面：上傳例子集資料介面、輸入決策樹知識法則介面、及轉換決策樹 PMML(Predictive Model Markup Language)文件模組等。就整體系統運作流程而言，在知識學習方面，我們首先上傳例子集，接著使用知識學習子系統來發掘出知識，然後直接儲存於知識法則庫內。而在知識使用方面，我們可以利用線上預測子系統來存取知識法則庫內的知識以進行分類和預測的工作。在知識溝通方面，本系統提供一個轉換 PMML 格式文件的模組，方便導入其他採礦工具所歸納學習出之決策樹形式的知識。而在知識整合方面，本系統使用合併選擇決策樹子系統來合併多棵決策樹形式的知識而成一棵決策樹。運用這個子系統有助於維護決策樹法則知識庫內的知識，而讓決策樹形式的知識在保有簡單樹狀

結構下，進行知識法則的擴充，並且簡單樹狀結構有助於線上預測子系統對於系統預測結果之解釋和說明。有關後續研究方面，本研究擬實作此架構的元件，且對於合併決策樹方面，提出一些修剪策略來提昇決策樹之預測準確度，以及如何有效維護決策樹知識法則庫內的知識等課題。

【Abstract】

This paper presents an on-line decision tree-based predictive system architecture. The architecture contains nine components, including a database of the examples, a learning system of the decision trees, a knowledge base, a historical knowledge base, a maintaining interface of the decision trees, an interface to upload training and testing examples, a PMML (Predictive Model Markup Language) translator, an on-line predictive system, and a merging optional decision trees system. There are three channels to import knowledge in the architecture; the developers can upload the examples to the learning system to induce the decision tree, directly input the information of decision trees through the user interface, or import the decision trees in PMML format. In order to integrate the knowledge of the decision trees, we added the merging optional decision trees system into this architecture. The merging optional decision trees system can combine multiple decision trees into a single decision tree to integrate the knowledge of the trees. In the future research, we will implement this architecture as a real system in the web-based platform to do some empirical analyses. And in order to improve the performance of the merging decision trees, we will also develop some pruning strategies in the merging optional decision trees system.

緒論

隨著資訊科技的進步，網際網路的各項應用愈來愈廣，而且愈來愈深入每個人的日常生活之中。在資料採礦和知識發掘的領域裡，有很多學者將資料採礦的技術發展到 Web-Based 的環境，希望藉由此一標準平台，讓使用者能更容易去操作資料採礦工具來找出有用的知識，以及使用採礦工具所發掘出的知識來輔助制定決策。而有關這方面的文獻所探討的課題相當多，有些學者關心處理大量資料的採礦問題，是故採用平行及分散式的資料採礦架構，利用代理人(Agent)技術去處理資料及 XML(Extensible Markup Language)技術來定義文件格式；也有學者強調結合多個分散的採礦工具來發掘出知識。然而，對於知識的使用和整合方面，則較少著墨。知識發掘的目的在於發掘出有用的知識，然後應用於解決實際的問題，所以對於如何提供一個環境讓使用者能應用這些知識來解決問題，以及探討如何整合這些知識都是相當值得研究的課題。

因此，本研究提出一個決策樹形式知識的知識發掘及線上預測系統架構，而它是建置於 Web-Based 環境之中，其裡面包括有知識學習元件、知識儲存庫、知識整合元件、有助於知識流通的轉換知識成標準文件格式模組、以及線上使用知識來進行預測等單元。希望藉由這個系統，我們可以直接上傳例子集來歸納學習決策樹形式的知識或由介面模組來導入決策樹形式知識，而使用者也可以藉由瀏覽器使用知識法則庫內的知識來進行分類或預測的工作。

此外，在知識發掘領域裡，有很多學者提出不同的學習演算法，也有學者強調結合多個演算法來提昇系統預測的準確度。雖然每個學習演算法皆有其優缺點，藉由結合多個演算法可以互補有無，然而對於其發掘出的知識之累積並不是皆適用於各種組合策略，例如使用類神經網路所學習出的知識是一個黑箱(Black Box)，無法與其他形式之知識結合。決策樹演算法(如 ID3, C4.5)所推導出之知識，是以樹狀結構組織而成，有著清楚簡單的結構，讓人十分容易理解，也很符合人類進行決策之過程。雖然我們可以把決策樹形式的知識轉換成法則(Rules)而累積在一起，也有學者提出處理轉換後的法則之衝突(Conflict)和重覆的解決方法，但是這個轉換程序已破壞了原有決策樹之簡單結構的優點，失去了其知識的整體組成結構。所以我們希望在保有決策樹形式知識之簡單結構下，進行知識的整合，以累積經過複雜運算及耙時的知識發掘過程所產生出來的知識。因此在本系統架構內，有一個合併選擇決策樹的子系統，其主要的目的在於整合多棵決策樹的知識，也就是說，它的任務是達成知識整合的目的。

最後，簡述本篇文章的結構：第壹章介紹本研究的源起和目的；第貳章針對 Web-Based 平台之資料採礦及決策樹形式的知識整合文獻進行說明；第參章詳述本研究所提出之決策樹形式知識之線上預測系統架構；第肆章敘述合併決策樹形式知識之演算法運作流程；第伍章進行總結和說明後續研究方向。

文獻探討

本研究首先針對有關 Web-Based 的資料採礦和預測模式系統架構進行說明，接著針對本研究提出的決策樹形式知識之線上預測系統架構內之知識整合單元，說明有關決策樹形式的知識整合之文獻。

Web-Based 的資料採礦及預測模式系統

有關探討 Web-Based 平台的資料採礦和預測模式系統的主要課題皆集中於分散式的資料採礦及結合多個學習演算法，以及應用 XML 的技術於描述及定義預測模式的標準文件型態，還有一些學者結合數種最新技術來架構整個系統。

在強調平行及分散式的資料採礦作業方面，Kargupta, Stafford, 和 Hamzaoglu(1996)提出一個 PADMA(PArallel Data Mining Agents)系統，主要是應用於平行及分散式的資料採礦作業，他們使用代理人(Software Agents)技術存取及分析本地端的資料，且利用 Web-Based 介面來互動式地視覺化顯示資料(Data Visualization)。以及 Krishnaswamy, Zaslavsky, 和 Loke(2000)針對分散式資料採礦(Distributed Data Mining, DDM)提出一個混合式的架構(Hybrid Architectural Model)應用於電子商務上，他們是讓應用服務提供者(Application Service Provider, ASP)來提供分散式資料採礦的服務。這個架構結合主從式技術(Client-Server Technology)和行動代理人技術(Mobile Agent Technology)，其中有一個知識整合元件，它結合不同資料來源之資料採礦的結果，然後將此結果回覆給使用者。然而這樣的知識整合只是用來統合不同採礦工具的結果，無法達成知識整合和累積的功能。

接著，在應用 XML 技術方面，Krishnaswamy, Zaslavsky, 和 Loke(2001)提出一個支援 XML 技術的聯盟資料採礦服務(Federated Data Mining Services)系統，而它提供一個服務介面方便使用者存取資料採礦的工具來進行發掘出隱藏的知識。它包含數個分散的資料採礦系統及一位聯盟管理者(Federated Manager)，其中聯盟管理者負責維護資料採礦系統的資訊。而且他們提出一個 XML 文件型態定義(Document Type Definition, DTD)，稱為 DDMS-ML(Distributed Data Mining Systems-Markup Language)。它是用來描述每一個資料採礦系統的訊息，透過這個共通的文件格式，使得聯盟管理者可以很容易地取得各個資料採礦系統的資訊。

有關結合多項新的技術方面，Kurgan, Cios, 和 Trombley(2002)提出一個資料採礦工具箱架構(Data Mining Toolbox Architecture)應用於網際網路的環境。其中資料採礦工具箱會動態地搜尋有關特定採礦工作的資料採礦工具，然後將使用者的資料放入這些工具之中，而歸納學習出結果，然後再將這些結果彙整成一份最終的報告回覆給使用者。他們使用 XML、WSDL(Web Service Description Language)、SOAP(Simple Object Access Protocol)、HTTP、MIME、XML-RPC(Remote Procedure Call)、PMML(Predictive Model Markup Language)等技術，且使用 UDDI(Universal Description Discovery and Integration)的平台來存放其資料採礦工具，即以 Web Services 的方式，讓每一個資料採礦工具都是一個免費的 Web Services。這樣的架構有一些需考量的問題，首先是否有足夠的免費資料採礦工具可以使用，且它們的品質是否可靠，而且使用者是否能理解這些工具所輸出的結果。再者系統彙整後的報告是否有統合多個資料採礦工具的預測模式，並且建議使用者該如何去使用這些模式等資訊。最後使用者該如何去整合或累積這些資料採礦工具所發掘出的知識，這些都是我們在使用新技術時需要關心並考量的課題。

綜合上述的文獻，我們發現有一個重要的課題被忽略了，那就是知識的整合和使用。知識發掘是一項很重要的工作，然而即使用了最新的技術去發掘出很多的知識，但缺乏後續的知識整合、散播和使用的話，就無法發揮知識的效用，所以本研究提出一個 Web-Based 的決策樹形式知識的線上預測系統架構，此架構提供知識發掘和線上預測的功能，並且將決策樹形式知識的整合概念應用於此架構之中，期能累積知識發掘系統所歸納學習的知識。接著我們探討有關決策樹形式的知識整合之相關文獻。

決策樹形式之知識整合

有關結合多個方法的文獻相當多，本研究僅針對結合多個決策樹形式之知識的組合策略進行探討。這些組合策略包括投票表決方式、加權計分方式(加權投票法)、利用法則集(Rules Sets)來集結決策樹之知識法則、後設決策樹(Meta Decision Trees)、仲裁樹(Arbiter Tree)、合併樹(Combiner Tree)、以及合併決策樹(Merging Decision Trees)等方法進行說明如下：

$$C^*(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{i:Ci(x)=y} 1 \qquad \text{(被預測最多次的類別爲 y)} \qquad \text{(公式 1)}$$

投票表決方式(Majority Votes)

採用投票表決的方式來結合多棵決策樹的知識是最廣為使用的方法，Breiman(1994)提出的 Bagging 方法就是採用投票表決的方式來結合多個分類法則(Classifiers)(如公式 1)，而 Quinlan(1996)把 Bagging 的方法應用於他的 C4.5 演算法之中，用來結合多棵決策樹的知識法則。馬芳資(1994)引用群體決策的概念，把每一棵決策樹想成一個虛擬專家，利用投票表決來綜合五棵決策樹之知識法則，且以佔最多數之類別為系統預測的目標類別。在此每一棵決策樹之重要性相同，即每一棵決策樹的權重值皆為壹。然而以多數決來綜合最終的預測結果值，卻忽略了少數特殊樣態，並不是一個很好的方式，而且每一棵決策樹無論其預測準確度為何，其重要性皆相等，亦不適當。

加權計分方式(Weighted Votes)

加權計分的方式是給予每一棵決策樹不同的權重值，然後乘上其預測目標值而計算出總得分，而得分最高的類別為系統預測的目標類別。馬芳資(1994)在信用卡客戶信用預警研究之中，利用決策樹的平均測試命中率為樹的權重值，而乘上以純度為法則之權重值，再乘上預測目標值，然後加總計算出總分(如公式 2)。公式 2 中 n 代表結合決策樹的個數，而 i 代表第 i 棵決策樹，判定最終預測值是依據總分是否大於門檻值，若是則系統預測為信用良好的客戶，否則為信用不良之客戶。公式 3 為 AdaBoost 所採用的策略，它就是以分類法則 Ci 之誤判率 εi 除以(1-εi)的對數值為權重值，而以權重總和最大之預測值 y 為最終之預測結果值 C*(x)，換句話說，它是依據分類法則的誤判率大小，把誤判率較小的分類法則，設定較高的權重值(Bauer & Kohavi, 1999)。加權計分雖然依據決策樹的準確度給予其權重值，但仍無法避免忽略得分低之法則，而且這樣的判定無法給予使用者一個較客觀的判定理由。

利用法則集(Rules Sets)來集結決策樹之知識法則

Williams(1990)提出一個多重歸納學習(Multiple Induction Learning, MIL)的方法，係把多棵決策樹的知識轉成法則(Rules)，然後再把它們組合在一起。由於每一棵決策樹的每一條路徑皆可轉換成一條『若--則』(If-then)的法則，而路徑上的中間節點之判斷條件是以『且(And)』的方式來進行連結，末端葉節點的目標值(類別)則置於『則(then)』的後面，用來進行預測分類結果。Williams(1990)也提出一些解決法則間衝突(Rules Conflict)與重覆(Replicated Rules)的方法，並且提出一些修剪策略，讓決策樹的路徑轉換成的法則更為精簡及一般化(General)。

$$X = \sum_{i=1}^{n} A_i P_i O_i \qquad \text{(公式 2)}$$

X：表示總得分。
Ai：表示決策樹的權重值，以平均測試命中率爲此值，0<=Ai<=1。
Pi：表示法則的權重值，即以葉節點的純度爲此值，0<=Pi<=1。
Oi：表示系統的預測結果值，Oi 爲 1(表示信用良好客戶)或, -1(表示信用不良客戶)。

IF (X >= 門檻值) THEN 預測爲信用良好客戶
ELSE 預測爲信用不良客戶

註：測試命中率(Ai)＝（猜對個數）／測試總筆數
葉節點純度(Pi)＝（佔多數類別之個數）／（落在此葉節點之總筆數）

$$C^*(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{i:Ci(x)=y} \log \frac{\varepsilon i}{(1-\varepsilon i)} \qquad \text{(公式 3)}$$

Hall, Chawla, 和 Bowyer(1998)針對解決大量例子集的問題，以選取不相鄰(Disjoint)的例子集，且平行式地進行歸納學習出決策樹，然後再把決策樹轉換成法則集(Rule Sets)。他們引用 Williams(1990)提出之解決法則衝突和把相似的法則合成一條更為一般化的法則(General Rule)等方法，並且再加入其他學者提出之合併法則的方法。

後設決策樹(Meta Decision Trees)

Todorovski 和 Dzeroski(2000)提出後設決策樹(Meta Decision Trees, MDT)的方法來結合多個分類法則(Classifiers)。而 MDT 是改良 C4.5 演算法而成，它不同於 C4.5 演算法之處在於 MDT 的葉節點可以明確指出使用基底層的那一個演算法來進行預測，而非僅回覆預測結果值而已。在驗證 MDT 方法的實驗上，他們結合基底層(Base-Level)的五個演算法，包括 C4.5、LTree、CN2、k-NN、和 naïve Bayes 演算法等，且利用二十一個 UCI 資料集，實際跑出系統之預測準確度，並且與使用 C4.5 演算法之 Stacking 及使用 Boosting 和 Bagging 在結合多棵決策樹上進行比較，結果發現 MDT 的預測準確度皆優於這些結合的方法。

仲裁樹(Arbiter Tree)

Chan 和 Stolfo(1995)提出仲裁樹的方法。如圖 2-1 所示，首先利用分組的例子資料產生出多棵決策樹，然後再把這些案例資料去歸納學習出一棵仲裁樹。此外，他利用仲裁法則來綜合多棵決策樹之預測結果值，而仲裁法則的設定在一般情況是採用投票表決的方式。這種策略不同於投票表決之處，僅在於多了一棵仲裁樹，以及可以調整仲裁法則而已。

合併樹(Combiner Tree)

Chan 和 Stolfo(1995)提出第二種結合策略是利用合併樹(Combiner Tree)來綜合出最終之預測結果值。首先把多組的例子資料分別去歸納學習出多棵決策樹，做為基底層(Base-Level)的多個分類法則。接著再從例子資料中隨機抽出一組例子集並把它們放入基底層之分類法則去取得各個分類法則之預測值，然後再把這組例子集及其在基底層的預測值放入學習系統之中去產生一棵合併樹。系統在執行預測時，首先將測試例子放入基底層的決策樹而得出預測值，再把這些值與測試例子放入合併樹以綜合出系統最終的預測結果值(如圖 2-2)。

合併決策樹(Merging Decision Trees)

上述的這些組合方法的共同點就是無法保有原本決策樹簡單樹狀結構之優點，而此一部份探討的是把多棵決策樹合併成簡單的一棵決策樹之合併策略。Quinlan(1998)提出 MiniBoosting 演算法，係改良 Boosting 的方法而成，以結合最少棵(三棵)決策樹的形式來與其他方法進行比較。而他在文章中把三棵原始決策樹，藉由合併(Merge)的程序，結合成一棵簡單結構的決策樹。Quinlan(1998)採用兩兩合併的方法來結合三個分類法則 C_1、C_2、和 C_3，即首先把 C_2 合併到 C_1 之中，形成 $C_{1,2}$；然後再把 C_3 合併到 $C_{1,2}$ 之中，而最後形成一棵決策樹 $C_{1,2,3}$。

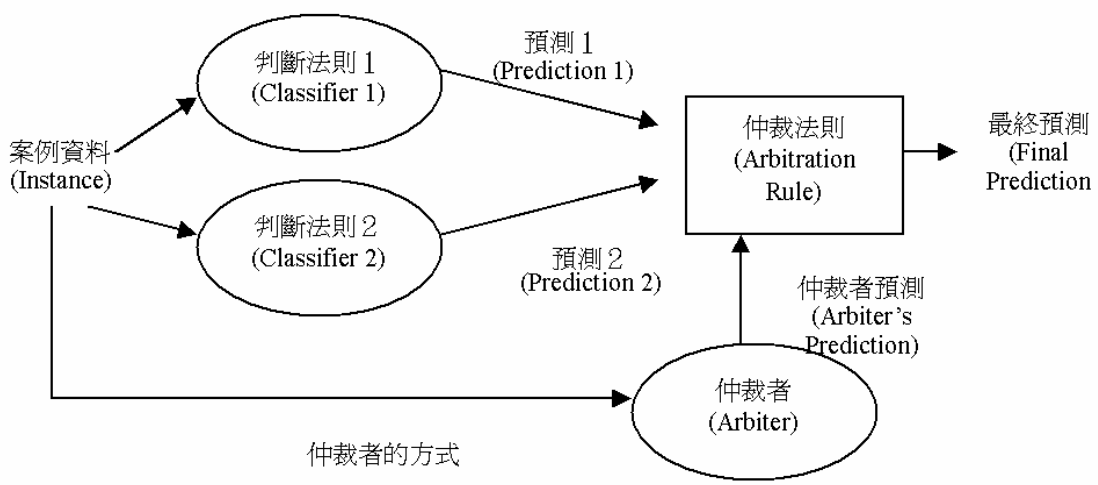

圖 2-1 結合兩個分類法則之仲裁者和合併者
資料來源：Chan & Stolfo (1995), "A comparative evaluation of voting and meta-learning on partitioned data." In Proc. Twelfth Intl.. Conf. Machine Learning.

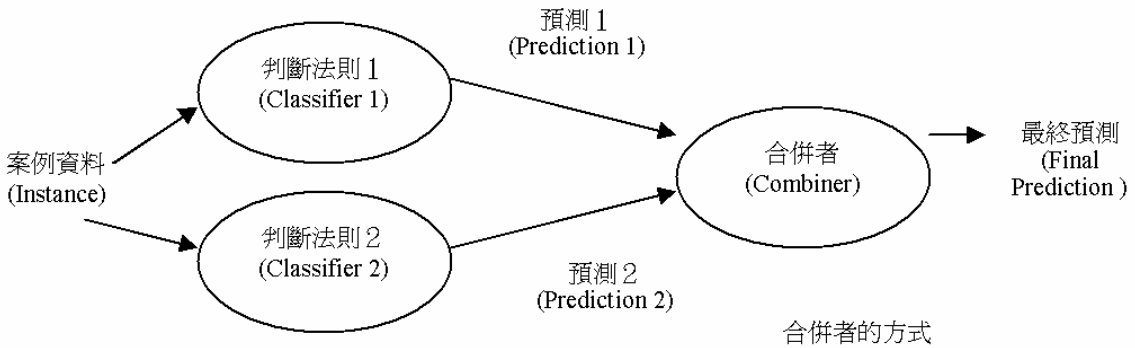

圖 2-2　結合兩個分類法則之仲裁者和合併者
資料來源：Chan & Stolfo (1995), "A comparative evaluation of voting and meta-learning on partitioned data." In Proc. Twelfth Intl.. Conf. Machine Learning.

Quinlan(1998)提出的合併演算法是把第二棵決策樹合併到第一棵決策樹的葉節點，例如合併 C1 和 C2 時，是用 C2 來取代 C1 的葉節點，然後再進行去除重覆和互相矛盾的分枝，如此完成兩棵決策樹的合併。然而這樣的做法會導致合併後的決策樹之節點數增加很多，且很多分枝下的葉節點內皆無任何例子資料落入此葉節點。為了解決這個問題，Quinlan(1998)把這些空的葉節點刪除，則造成合併決策樹之預測準確度驟降，它甚至比原始一棵決策樹的準確度還差。

有鑑於此，本研究的合併選擇決策樹演算法(Merging Optional Decision Trees, MODT)，主要是讓合併後的決策樹的節點數呈線性地成長，而且保留原始決策樹之知識法則，並且利用判定系統最終預測結果值的策略來提昇合併決策樹之預測準確度。而且本研究提出一個決策樹形式知識之線上預測系統架構，其中的知識整合工作就是運用合併選擇決策樹演算法來進行決策樹形式之知識整合，讓系統內的知識法則藉由此演算法的合併處理，能更完整及周延。接著，本研究將針對這個系統架構進行完整的說明。

決策樹形式知識之線上預測系統架構

本研究首先說明整體架構，接著說明各個組成元件的內部功能，然後說明本系統提供三種導入決策樹形式的知識法則的管道，最後說明系統運作流程。

整體系統架構

本研究提出一個決策樹形式知識之線上預測系統架構，而其主要的目的是提供一個 Web-Based 的決策樹形式知識之知識發掘及線上預測系統。它包括三個子系統，即決策樹知識學習子系統、合併選擇決策樹子系統、及線上預測子系統(Decision Trees Predictive System, DTPS)；三個資料庫，即例子資料庫、決策樹知識法則庫(Decision Trees Knowledge Base, DTKB)、和歷史知識法則庫；三個導入介面模組，即上傳例子集資料介面、輸入決策樹知識法則介面、和轉換 PMML 格式文件介面；以及二種使用系統的人員，即開發人員和一般使用者等(如圖 3-1)。

系統架構之組成元件

本系統架構之組成元件包括九個項目，即例子資料庫、上傳例子集資料介面、決策樹知識學習子系統、決策樹知識法則庫、歷史知識法則庫、決策樹知識法則維護介面、PMML 轉換模組、合併選擇決策樹子系統、以及線上預測子系統等。此外，在人員方面則有開發人員和一般使用者。

一、 開發人員：負責管理且維護整體系統架構的人。他負責導入決策樹形式的知識到知識法則庫內。在上傳例子集來歸納學習決策樹方面，開發人員需熟悉知識學習子系統之參數設定，而且他也負責執行合併決策樹形式知識及維護決策樹知識法則庫等兩項工作。

二、 使用者：是指一般社會大眾或公司員工，他們使用線上預測系統來進行實際預測。

三、 例子資料庫：例子資料庫是用來存放訓練例子集和測試例子集，以及與例子集有關之檔案，例如線索值資料型態和可能值個數及內容等等。它是一個歷史資料檔，且已經過前置資料處理，即去除雜訊和處理空缺值。

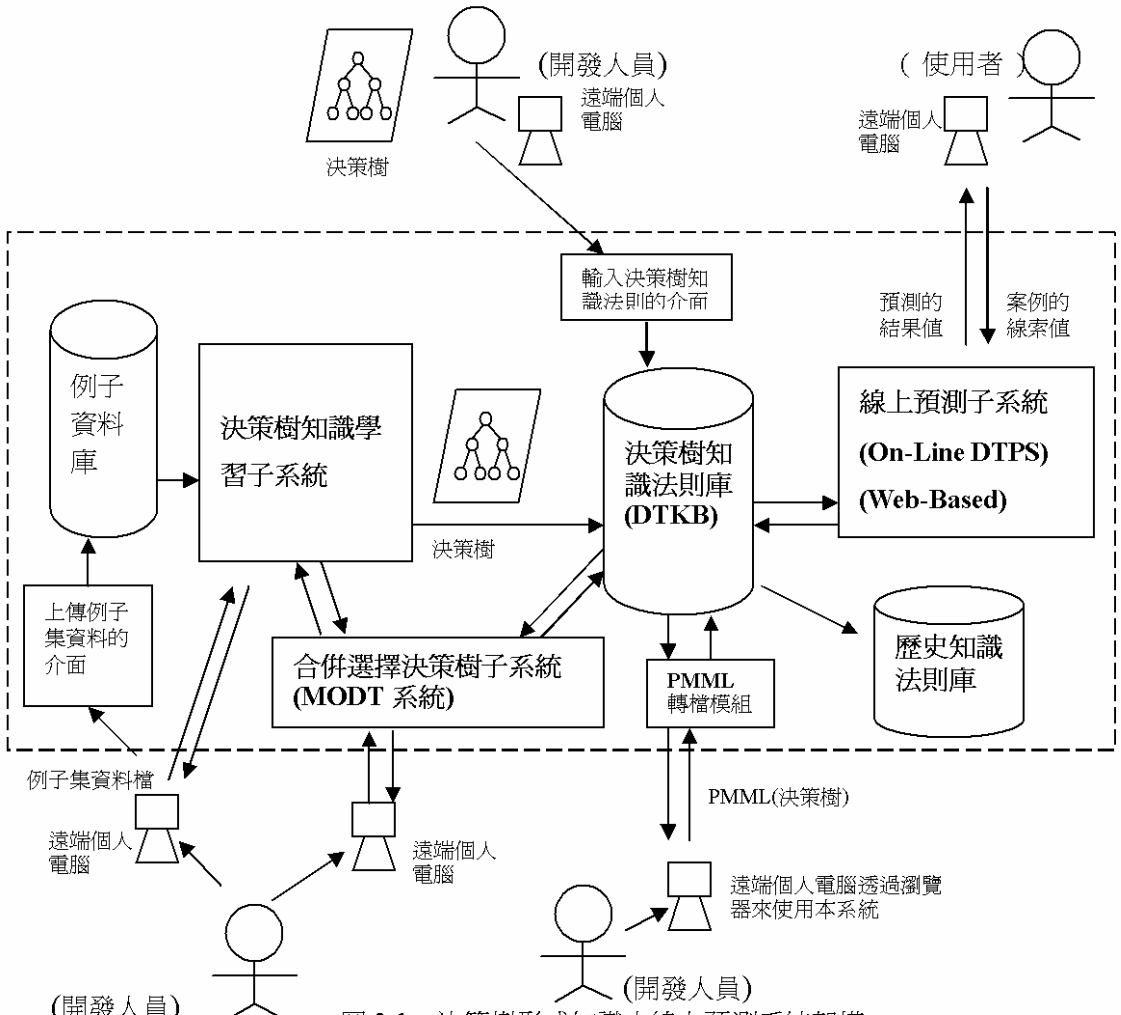

圖 3-1　決策樹形式知識之線上預測系統架構

四、　上傳例子集資料介面：開發人員可以透過這個介面來上傳欲使用決策樹知識學習子系統之例子資料檔。

五、　決策樹知識學習子系統：這是一個 Web-based 的決策樹知識學習系統，它是使用 Quinlan(1993) 提出的 C4.5 演算法來歸納推導出決策樹。它主要任務是讀入訓練例子集，透過學習演算法來歸納推導出決策樹形式的知識法則，然後直接轉入本系統架構之決策樹知識法則庫內。整個建立決策樹的處理是由伺服器端來執行，處理完成後會將決策樹及其驗證之結果輸出至開發人員的遠端個人電腦。

六、　決策樹知識法則庫：決策樹知識法則庫是採用關聯式資料庫技術來儲存決策樹形式之知識法則，它包括五個表格(Table)，其中第一個表格是用來儲存決策樹基本資料，即決策樹編號、決策樹名稱、節點總個數、葉節點個數、層級、各類別之法則數目、線索值個數、平均命中率、命中率標準差、信賴度、選擇節點個數等。第二個表格是用來儲存決策樹節點資料，內容包括決策樹編號、節點編號、向下分群線索值編號、分枝可能值、預測目標值、純度、父節點編號、子節點編號等等，並且以決策樹編號關聯到決策樹之基本資料表。而第三個表格是用來儲存線索值資料，內容包括決策樹編號、線索值編

號、線索值名稱、詢問使用者之提問問題、可能值個數、線索值型態等。第四個表格是用來儲存線索之可能值內容資料，內容包括決策樹編號、線索值編號、可能值。最後，第五個表格是用來儲存目標值內容和解釋資料，其內容包括決策樹編號、目標值、目標值的內容解釋。

七、 歷史知識法則庫：決策樹知識法則庫內之決策樹已完成合併後，即把此決策樹轉出至歷史知識法則庫內，所以歷史知識法則庫儲存整個決策樹的家族(Decision Trees’ Family)。

八、 輸入決策樹知識法則介面：這個介面主要是讓開發人員可以對決策樹知識法則庫進行資料維護，其包括新增、查詢、修改、和刪除(轉出)等資料操作功能。

九、 轉換 PMML 格式之模組：預測模式標記語言(PMML)是由 DMG(Data Mining Group)成員運用 XML 來描述資料採礦模式(Data Mining Models)所定義出來的一種標記語言，而它主要的目的是幫助不同的資料採礦工具能相互溝通(DMG, 2003)。PMML 提供描述幾種資料探礦工具的 XML 規格文件，例如決策樹、分類法則及關聯規則等。並且也可透過一些用 JAVA 寫的視覺化工具(Visualization Applet)，在網路上以圖形化的方式來顯示決策樹形式的預測模式。有鑑於此，本系統架構設計一個轉換模組來將內部之決策樹形式知識，轉換成 PMML 文件，以便於利用此項技術來呈現決策樹形式知識，及與外界資料採礦工具溝通。此外，開發人員亦可將現成的決策樹 PMML 文件，透過轉換模組，直接將此決策樹形式的知識轉入到決策樹知識法則庫。有關 PMML 2.0 之文件標籤及語法的資訊，請參考 http://www.dmg.org/pmmlspecs_v2。

十、 線上預測子系統：此系統主要是提供使用者一個線上使用決策樹知識法則的環境。使用者可以根據個人需求點選預測模式，然後以互動式的方式點選問題的答案，來進行系統預測。而且本系統架構是依據決策樹之走訪節點順序來提問及回答問題，所以使用者不需回答所有線索值的答案，而等到系統走訪到葉節點，就會回覆預測目標值，並且顯示其目標值之詳細解釋。對於使用者所走訪決策樹的路徑上的節點，其提問的問題和使用者所回覆的答案皆會顯示在螢幕上，讓使用者明瞭整個決策過程。

十一、 合併選擇決策樹子系統：合併選擇決策樹子系統主要是把兩棵決策樹合併成簡單的一棵決策樹，其做法是以兩兩合併的方法來結合決策樹形式的知識。本系統架構可運用於整合相同問題的決策樹形式的知識，而讓每一棵決策樹(或稱虛擬專家)藉由合併的機制來補充其知識法則，以提昇其預測的能力。有關合併選擇決策樹演算法的詳細說明，請參考『肆、合併決策樹形式知識』單元。

多重管道導入知識法則

本系統架構為了方便開發人員導入知識法則，所以提供三種導入的管道，包括直接鍵入、由 PMML 格式文件轉入、及由例子集歸納學習導入等，詳細說明如下：

直接鍵入

此即直接輸入決策樹基本資料及其內部節點資料和其他相關的資料。使用此介面的開發人員已經取得其他學習演算法所推導出的決策樹形式知識，而其中學習演算法可以是 ID3、C4、C4.5、CART、或其他可以產生決策樹形式知識之演算法。只要可以取得決策樹之相關資料，即可將此棵決策樹知識法則藉由此介面來鍵入到知識法則庫內。

由 PMML 格式文件轉入

本系統架構亦考慮到由外部知識學習系統來直接導入決策樹知識法則，也就是說，遵循共通標準來轉入 PMML 格式文件，同樣地我們也可將知識法則轉成 PMML 格式文件，與其它知識學習系統或預測系統做溝通，並且也可透過此項技術來視覺化呈現本系統架構之決策樹形式的知識。藉由這個轉換模組，系統可以快速且容易地導入外部之決策樹知識法則，提昇系統的彈性與擴充性。

提供例子集來歸納學習而轉入

倘若我們僅有一些歷史例子資料，而想運用這些資料來歸納出決策樹，然後利用此決策樹知識來進行預測，如此可用此一管道來進行學習與預測。這個管道是一套完整的由例子集來歸納學習並進行預測的流程，從上傳例子集檔案及設定學習參數和條件值，然後進行知識學習子系統，將歸納出的決策樹知識法則直接轉入到本系統架構的知識法則庫內。接著使用線上預測子系統來點選此一預測模式進行實際之預測。

系統運作流程

對於本系統之運作流程，我們以下列五個功能角度來進行說明，包括知識學習、儲存、整合、溝通和使用等。

知識學習

在知識學習(或稱知識發掘)方面，我們首先將已處理好的例子資料檔案，透過『上傳例子集資料的介面』，把檔案傳入到『例子資料庫』之中。接著開啟『決策樹知識學習子系統』的操作介面，設定訓練例子集和測試例子集的檔名、目標值的類別個數和可能值、線索值個數和其可能值、以及學習系統之參數設定等，然後按下開始學習，則系統自動會按照設定之條件值去進行歸納學習，待學習完成之後，系統會輸出決策樹知識法則的內容及其測試之結果值。依據這些輸出結果判定是否將此棵決策樹置入知識法則庫內，或者需要調整參數值來重新學習。倘若決定將此棵決策樹放入知識法則庫內，則點選置入知識法則庫之功能，然後設定此棵決策樹之編號及其相關資訊，則系統會把它直接轉入『決策樹知識法則庫』內。若需重新學習則回到『決策樹知識學習子系統』的操作介面，重新選定參數值來進行歸納學習(請參考圖 3-2)。

知識儲存

在知識儲存方面，主要是利用『決策樹知識法則庫』來儲存決策樹之知識法則，而它與其他元件的運作流程說明如下：我們可以利用『輸入決策樹知識法則的介面』來把現有之決策樹形式的知識直接鍵入到『決策樹知識法則庫』內，或者由上述的『決策樹知識學習子系統』來把它所歸納學習出之決策樹直接轉入到『決策樹知識法則庫』內，以及透過『PMML 轉換模組』把取得的決策樹的 PMML 格式文件轉入到『決策樹知識法則庫』內。此外，『線上預測子系統』可以擷取『決策樹知識法則庫』內的知識來進行預測或分類工作。『PMML 轉換模組』亦可將『決策樹知識法則庫』內的知識轉出成決策樹的 PMML 格式文件，以方便與外界知識發掘工具溝通。『合併選擇決策樹子系統』會到『決策樹知識法則庫』內讀取兩棵決策樹來進行合併，待合併完成後，再將合併後的決策樹存回『決策樹知識法則庫』之中，且建立合併後決策樹之相關資料，然後將原始兩棵決策樹轉出到『歷史知識法則庫』 (請參考圖 3-3)。

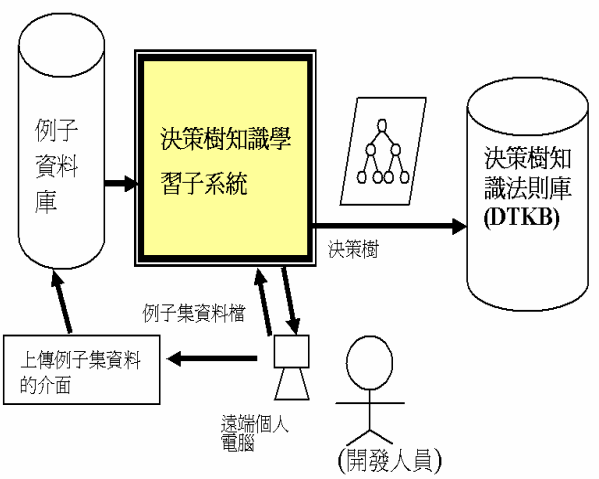

圖 3-2　知識學習之運作流程

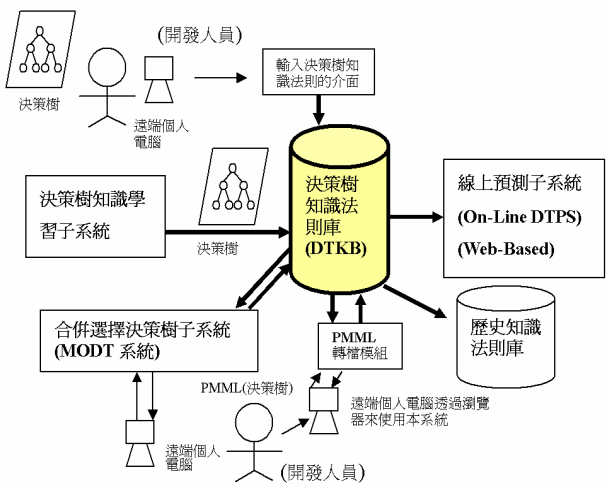

圖 3-3：知識儲存之運作流程

知識整合

在知識整合方面(請參考圖 3-4)，對於一個新的問題，我們可以先搜集其例子集，然後透過『決策樹知識學習子系統』來產生第一棵決策樹，接下來當有新的決策樹導入時，我們就可以利用『合併選擇決策樹子系統』將原有的決策樹與新的決策樹進行合併，待合併後，系統會將合併後的決策樹存回『決策樹知識法則庫』內，並且把原有的決策樹及那一棵新加入的決策樹轉出到『歷史知識法則庫』，如此完成一次的合併程序。以此類推，採用原有合併後決策樹結合新導入的決策樹的方式來進行決策樹形式的知識整合，所以同一問題在決策樹知識法則庫內僅有一棵合併後的決策樹，而整個問題的決策樹家族(Decision Trees ’ Family)則儲存在歷史知識法則庫內。

倘若一開始我們可以取得相同問題的多棵決策樹，則可以採用兩兩合併的方式來進行合併多棵決策樹，如此可以加速整個決策樹形式知識的整合，之後每當新增加一棵決策樹時，再將此新增的決策樹形式知識合併到原有之合併決策樹內。本系統使用合併選擇決策樹演算法來進行決策樹形式知識的整合，首先讀入兩棵原始決策樹，然後比對兩棵決策樹之所有節點，對於相同的節點進行例子數的加總，而父路徑相同而節點內容不同則利用一個選擇連結，將兩個節點合併成一個虛擬的選擇節點，詳細的內容說明請參考『肆、合併決策樹形式知識』單元。

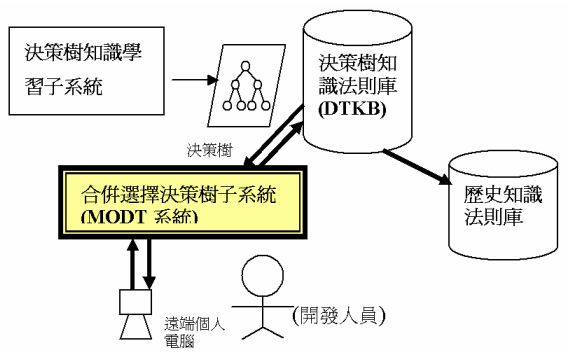

圖 3-4：知識整合之運作流程

知識溝通

在知識溝通方面，我們可以透過『PMML 轉換模組』將 PMML 格式文件轉入『決策樹知識法則庫』內，或將『決策樹知識法則庫』內的知識轉出成 PMML 格式文件。藉由標準化的 PMML 格式文件，我們可以很方便地導入其他學習系統所推導的決策樹 PMML 文件，甚至輸出系統之知識法則供其他預測系統使用，如此可達成知識法則之流通和共享。PMML 格式文件也很容易在瀏覽器上顯示樹形圖之預測模式，且每一個節點的資訊也可以很容易地被點選檢視(請參考圖 3-5)。

知識使用

在知識使用方面，使用者可以利用瀏覽器登入『線上預測子系統』網站，點選所欲執行之預測模式及其相關之參數設定，然後系統會到『決策樹知識法則庫』內讀取決策樹的知識法則。隨著走訪決策樹的節點來詢問問題且讓使用者點選答案，而這其間會存取『決策樹知識法則庫』內之線索資料表格及其可能值表格來取得問題和可能答案選項。直到走訪到葉節點時系統就會回覆系統預測目標值及其相關之解釋說明。使用者也可針對預測結果給予評分，或回覆實際之目標值，如此有助於獲得使用者的回饋，且可用來分析知識法則的準確度。此外，系統有一個 Q&A 的專區來提供常見問題資訊和取得使用者在操作系統、讀取預測解釋或其他關於系統運作等方面的意見(請參考圖 3-6)。

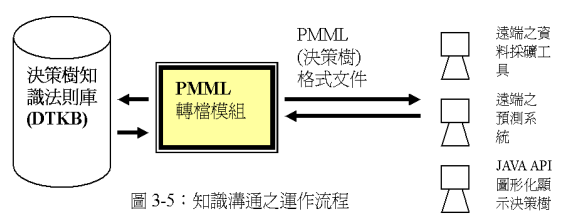

圖 3-5：知識溝通之運作流程

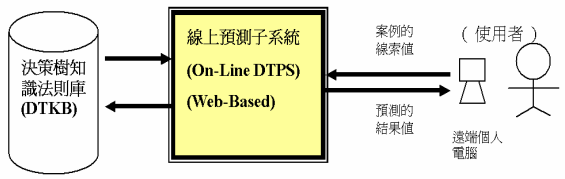

圖 3-6：知識使用之運作流程

合併決策樹形式知識

瞭解了本研究所提出之決策樹形式知識之線上預測系統的整體架構之後，我們可以清楚地看到決策樹知識法則庫的重要性，因為線上預測系統若要發揮其功能，必須要有完整、正確、及準確之知識法則。雖然本系統架構提供三種不同的管道來導入知識法則，但若缺乏有效的知識法則維護，恐怕無法發揮知識法則之功效。因此本研究使用合併選擇決策樹演算法來進行知識法則之整合，讓決策樹之法則能藉由結合其他決策樹之知識法則而更加周全和完整。而且合併多棵決策樹形式之知識，把原本多棵之決策樹的複雜知識結構合併成一棵簡單結構之決策樹亦有助於線上預測系統解釋預測結果。我們首先定義解說演算法所使用之符號定義，然後說明演算法之運作流程，最後說明合併多棵決策樹之運作策略。

演算法使用符號說明及名詞定義

為了能清楚說明合併選擇決策樹演算法，本研究針對使用的符號定義如下：

1. 目標值(Goal)： 或稱為類別(Class)，即系統欲進行分類或預測的結果值。在此我們以１和２來代表欲進行分類之兩個目標值。
2. 線索值(Cues)： A、B、和 C 代表例子(Instance)的線索值，即用來描述此例子資料的屬性值(Attributes)，而線索可以是分類或計量的變項。

3. 線索之可能值： a1 和 a2 代表線索 A 的可能值；b1 和 b2 代表線索 B 的可能值；c1 和 c2 代表線索 C 的可能值。
4. 根節點(Root Node)： 決策樹的第一個節點，是走訪決策樹的入口節點。
5. 中間節點(Internal Node)： 它有父節點也有子節點，而且會選取一個線索值往下分群，所以其內包含一個線索的條件判斷運算式。
6. 葉節點(Leaf Node)： 決策樹的末端節點，它沒有子節點。相同特徵的例子集就會落入同一個葉節點之中，且它是決策樹進行預測目標值的節點。
7. 選擇連結(Option Link)： 合併決策樹之擁有相同父路徑之節點，以此連結來合併這些具有相同父路徑之節點。
8. 決策樹編號： T1 代表編號 1 的原始決策樹，T2 代表編號 2 的原始決策樹。
9. 節點編號： [0]代表決策樹的第一個節點，也就是根節點(Root Node)。本研究是採用前序式走訪(Preorder Traversal)決策樹的節點，以流水號的方式進行節點編號，且以 0 為開始值而依序往下編號。
10. 合併決策樹編號： 合併決策樹 T(12)表示由決策樹 T1 和決策樹 T2 所合併後的決策樹；同樣地，T(12)3 表示先把決策樹 T1 和決策樹 T2 合併成決策樹 T(12)，然後再跟決策樹 T3 進行合併，最後合併成之決策樹 T(12)3，以此類推。
11. 強態法則(Strong Pattern Rules)：所謂強態法則(強態葉節點)之概念，即決策樹之每條由根節點到葉節點之路徑為一條法則，法則之好壞乃因葉節點之純度及落入此葉節點之例子數多寡來判定，純度愈高且例子數愈多，其法則之代表性較佳，即為強態法則。本研究設定葉節點的純度達到 85%且落於此葉節點的例子數在 10 筆以上者，其判斷法則視為強態法則，亦即其區別能力的確定性程度至少達到 85%。
12. 判定系統最終預測結果值之策略：在合併決策樹的結構中，每一條路徑若有選擇連結的節點，其就會多產生一個預測結果值。在合併兩棵決策樹的情形下，最多會出現兩個預測葉節點，而在綜合這兩個預測葉節點所採用的策略是以下列的法則來進行。首先判斷是否有強態葉節點，倘若沒有就以好壞客戶的例子數來判定最終預測值；若有就以落入強態葉節點之好壞客戶例子數來判定最終預測值；倘若兩個葉節點之例子數為零則以系統預設目標值為最終預測值。

IF (強態葉節點數目為零)
THEN IF (好客戶及壞客戶的例子數皆為零)
THEN 以預設目標值為預測結果值
ELSE 比較好客戶和壞客戶之例子數多寡以多者為預測結果值
ELSE 以落在強態葉節點之好壞客戶筆數之多寡為判定最終預測結果值

合併選擇決策樹演算法

本研究以三個步驟來說明整個合併選擇決策樹演算法之執行流程，包括讀入決策樹資訊並還原其樹狀資料結構，接著說明合併兩棵決策樹之運作流程，最後是驗證合併決策樹和輸出合併決策樹資訊及驗證的結果值(請參考圖 4-1)。其中第二個步驟合併決策樹程序說明如下：假設 n1 為決策樹 T1 的一個節點，而 n2 為決策樹 T2 的一個節點，程式首先由兩棵樹的根節點開始比對，比較兩個節點的選取線索和分枝是否相同，倘若相同則進行例子數加總，並繼續往子節點走訪比對；若兩個節點為葉節點且有相同的路徑，則合併兩個節點內容，加總例子數且重新計算各類別之例子數，以佔最多數之類別為此合併後葉節點之預測目標值，並且計算節點之純度。

若程式已比對到葉節點，則繼續往其兄節點走訪比對；若無兄節點則走往上層父節點，再尋找父節點之兄節點，待兩棵樹皆走回到其根節點，則表示已無可比對之節點，也就是說已比對完成兩棵樹的節點了。若在比對兩棵樹的節點發現兩個節點不相同時，則將 n1 節點的選擇連結指向 n2，且把 n2 的父節點和兄節點設為 n1 父節點和兄節點，也就是說把 n2 節點(包括其下之所有子節點)加入決策樹 T1 之中，成為 n1 的選擇關係的節點，由此可見，合併完成之決策樹的資訊儲存於決策樹 T1 之中，步驟 4 之輸出合併決策樹亦即輸出決策樹 T1 的資料結構，如此可節省儲存一棵新決策樹所需佔用之記憶空間。整個演算法之運作邏輯，本研究以文字及圖示說明如下：

假設：已知的決策樹 T1 和決策樹 T2
1．讀入決策樹 T1 和決策樹 T2 的資料且還原其樹狀資料結構。
2．進行合併決策樹程序：
2.1. 設定 n1 為 T1 的根節點，且設定 n2 為 T2 的根節點。
2.2. 比較 n1 和 n2 是否相等，若是則合併兩節點之例子數，到步驟 2.2.1；若不相等則建立兩者的 option 關係，到步驟 2.3。
2.2.1 判斷 n1 和 n2 是否為葉節點，若是重新計算其純度及設定其目標預測值，然後尋找下一個比對的節點，到步驟 2.4。
2.2.2 否則判斷 n1 和 n2 是否都有子節點，若是則設 n1 為其子節點，n2 為其子節點，到步驟 2.4。
2.3. 設定 n2 為 n1 的 option 節點，且設定 n2 的父節點為 n1 的父節點，n2 的兄節點為 n1 的兄節點，及 n1 的兄節點為 n2，然後尋找下一個比對的節點，到步驟 2.4
2.4. 假若所有節點都比對完畢，則停止，否則到步驟 2.2。
3．驗證合併後決策樹且輸出合併後決策樹及其驗證結果值。
結果：產生一棵合併後決策樹

圖 4-1：合併選擇決策樹演算法

一、 **首先讀入兩棵決策樹 T1 和 T2 之節點資訊，而它們是以前序走訪(Preorder Traversal 的方式記載決策樹的節點資訊，其中節點資料欄位內容包括節點編號、線索值編號、目標值、父節點編號、兄節點編號、子節點編號、選擇(Option)節點編號等。然後將此兩棵決策樹還原成樹狀資料結構，並建立好其間之指標連結關係，如圖 4-2 所示。決策樹 T1 和 T2 皆擁有六個節點，中括弧內標示著節點編號，而橢圓形表示節點，橢圓形內標記為 A 表示選取線索 A 進行往下分群，而分枝上標記為 a1，則表示同屬例子線索 A 的內容值為 a1 的例子皆落入此一分枝下的節點；若此節點無子節點則稱為葉節點，葉節點內之標記值 1 或 2 為系統預測目標值，也就是此預測葉節點之預測結果值。**

二、 **接著進行合併兩棵原始決策樹。如圖 4-3 所示，首先比對 T1 和 T2 的[0]節點，發現兩者同樣選取線索 A 來往下分群，且兩者皆有子節點，所以 T1 走往其[1]節點且 T2 亦走往其[1]節點。接著比對兩樹的[1]節點，發現兩者選取同樣的線索 B 來分群，所以 T1 和 T2 再走往它們的左邊子節點[2]。比對兩者皆為葉節點且預測目標值皆為 1，於是合併兩個葉節點之資訊儲存於 T1 之中。**

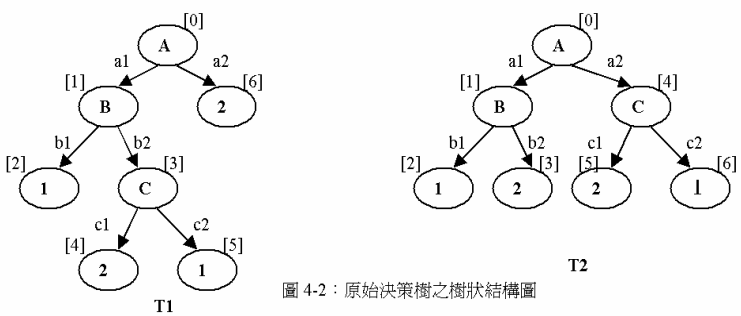

圖 4-2：原始決策樹之樹狀結構圖

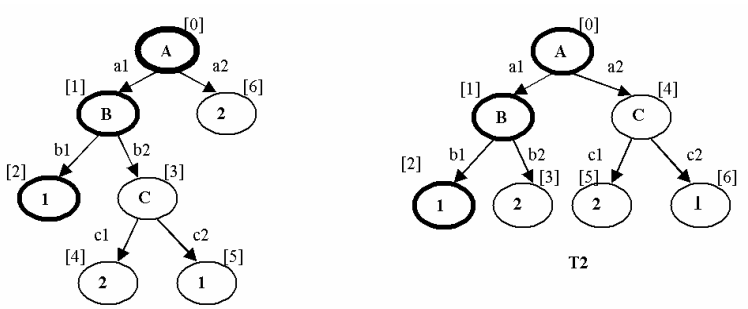

圖 4-3：第一條路徑的合併比對

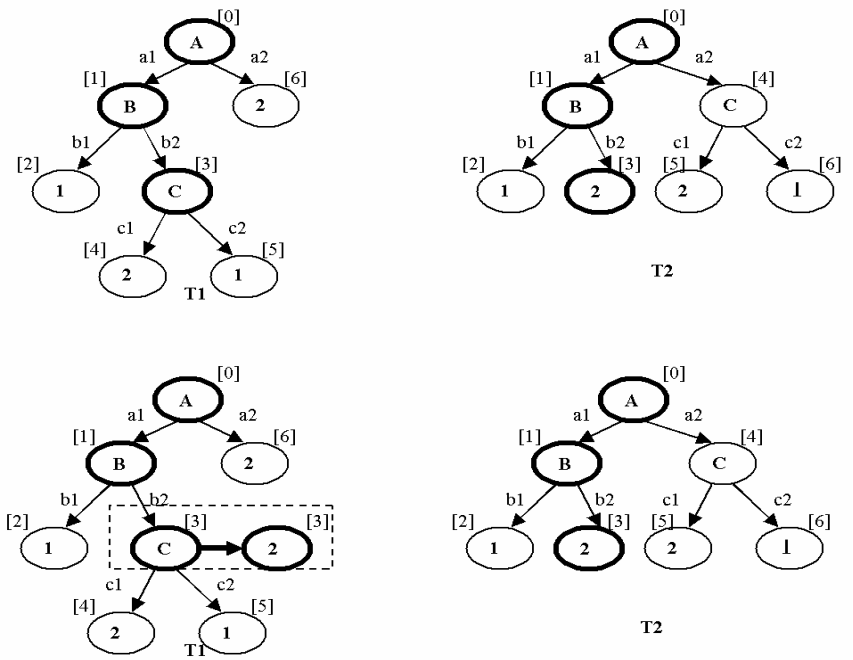

圖 4-4：第二條路徑的合併比對

合併完節點[2]的資訊之後，判斷 T1 和 T2 的節點[2]是否有兄節點，若有則走訪其兄節點，所以 T1 走往節點[3]且 T2 亦走往節點[3]（如圖 4-4）。比對兩個節點，發現 T1 的節點[3]為中間節點，而 T2 的節點[3]為葉節點，所以將 T2 的節點[3]合併到 T1，也就是把 T1 的節點[3]的選擇連結指向 T2 的節點[3]，且修改 T2 的節點[3]的父節點為 T1 的節點[1]。這個動作代表的是把 T2 的這個葉節點(法則)加入 T1 之中。

由於 T1 和 T2 的節點[3]無兄節點，所以走往他們父節點，也就是走往節點[1]，然後判斷節點[1]是否有兄節點，若有則走往其兄節點。T1 走到節點[6]，而 T2 走到其節點[4]，比對兩個節點是否相同，發現 T1 的節點[6]為葉節點，而 T2 的節點[4]為中間節點，所以兩者不同。於是將 T2 的節點[4]以下的子樹(Subtree)合併到 T1 之中，也就是把 T1 的節點[6]的選擇連結指向 T2 的節點[4]，且修改 T2 的節點[4]的父節點為 T1 的節點[0](如圖 4-5)。

合併完成後，繼續尋找下一個比對的節點。T1 的節點[6]已無兄節點則走往父節點[0]，而 T2 的節點[4]也無兄節點，是故走往其父節點[0]。比對兩個節點發現兩者皆為根節點[0]且已無兄節點，所以完成兩棵決策樹的合併比對工作，而合併完成的決策樹儲存於決策樹 T1 之中，即圖 4-5 中位於左下角的決策樹。

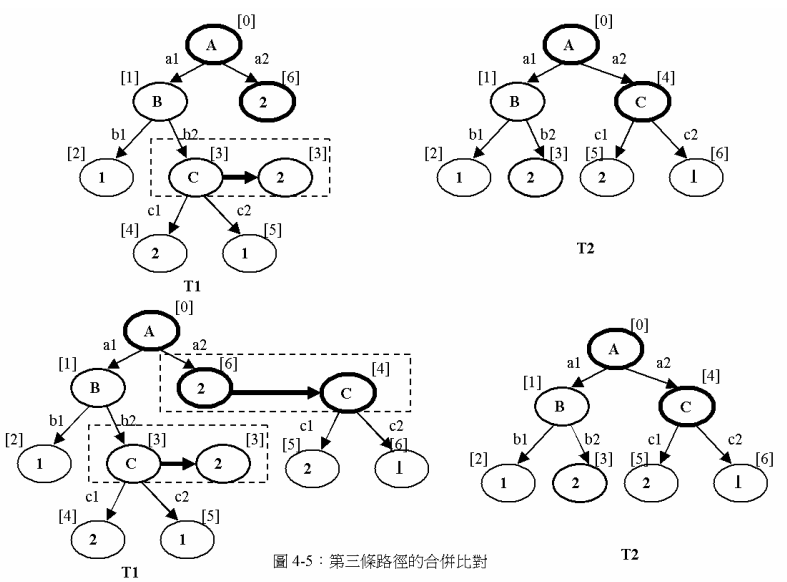

圖 4-5：第三條路徑的合併比對

合併後的決策樹之節點編號已混亂，所以必須重新調整決策樹節點之編號，本研究以前序走訪決策樹的方式來進行編號，完成後之決策樹如圖 4-6 所示。

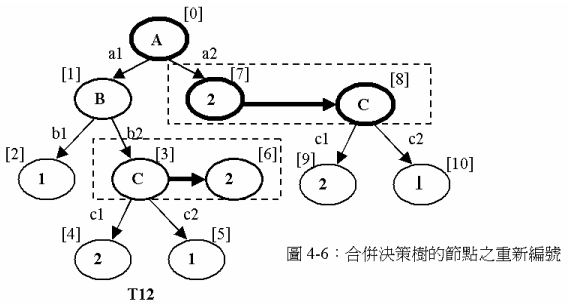

圖 4-6：合併決策樹的節點之重新編號

三、 合併完成後，則進行合併決策樹之準確度驗證。我們首先讀入測試資料檔，然後逐筆地進行測試工作並且計算正確命中之筆數，完成後輸出此合併決策樹之測試結果及其樹狀資料結構。在測試合併決策樹中，係依據測試資料之屬性值內容來走訪合併決策樹之節點，直到走到葉節點則進行系統之預測，然後比對系統預測結果值是否相等於測試例子之實際結果值，若相同則系統正確命中，反之則表示系統預測錯誤。合併決策樹不同於原始決策樹是存在有選擇關係的節點，所以在進行測試資料時系統在走到葉節點之後會回頭尋找是否有選擇關係的節點，倘若有則以此選擇關係之節點為起點繼續走訪到葉節點，直到路徑上的所有選擇關係的節點皆已走過為止。因此每一個選擇節點都會取得一個系統預測結果值，然後再以本研究所提出之合併預測結果值之策略來綜合這些葉節點之預測值，以進行系統對測試例子之預測，如此也顯示系統之綜合兩棵決策樹知識的能力。

多棵決策樹之合併方式

在合併多棵決策樹時，是採用兩兩合併的方式(如圖 4-7)。首先合併決策樹 T1 和 T2 為合併後決策樹 T(12)，以及執行合併決策樹 T3 和 T4 為 T(34)；然後再把 T(12)和 T(34)合併成一棵合併後決策樹 T(12)(34)。倘若欲再合併一棵決策樹 T5 時，則再把 T(12)(34)和 T5 合併成決策樹 T(12)(34)5，以此類推。本研究採用兩兩合併的目的在於可以平行地執行兩棵樹的合併，如此能增進合併多棵決策樹的執行效率。藉由合併多棵決策樹形式之知識，可以擴充決策樹(預測模式)的知識法則，達成知識整合與累積的目的。

為了驗證合併選擇決策樹演算法，本研究利用某一銀行之信用卡客戶信用資料，經過資料整理，刪除無效樣本及空缺值，共計可用樣本為 103,653 筆。以現狀碼等於零之正常流通卡為信用良好之客戶(簡稱好客戶)，而以現狀碼等於壹之強制停卡者為信用不良之客戶(簡稱壞客戶)。此例子集共有十七個線索值包括年齡、學歷、年收入、有無甲存、有無不動產、行業別、公司等級、職等、一般卡張數、金卡張數，帳款餘額、和六個月的繳款記錄等，其中年齡、年收入、及帳款餘額為數值性線索，其餘為非數值性線索。本研究採用馬芳資(1994)所提出之 IDC 4 系統來建立決策樹，此乃改良 Quinlan(1979)所提出之 ID3 演算法，運用資訊理論(Information Theory)中的熵函數(Entropy Function)來測度決策樹之節點不純度(Impurity)，做為選取線索值的標準。它可以處理連續性和分類性的變項，且可透過設定停止準則來避免過度學習(Over fitting)的問題。

整個實驗流程即先從原始資料庫中隨機抽出二千筆好客戶例子集及二千筆壞客戶例子集，然後各別再隨機分為六群，即五組的訓練例子集和一組測試例子集，且好壞客

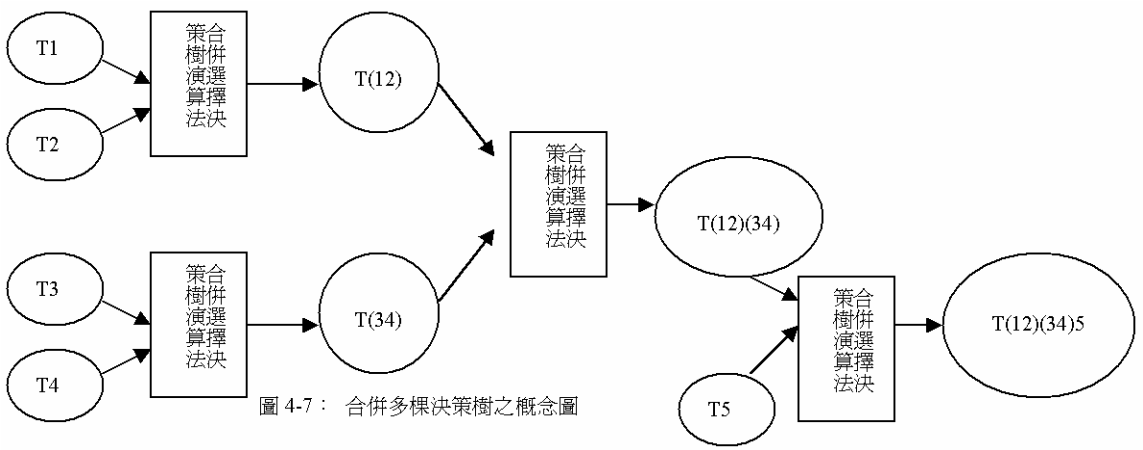

圖 4-7： 合併多棵決策樹之概念圖

戶資料比例相等。接著將五組訓練例子集放入 IDC 4系統中學習，以產生五棵原始決策樹；完成後，再利用合併選擇決策樹演算法進行兩兩一組的決策樹合併，然後再利用測試例子集，來進行原始決策樹及合併決策樹之驗證，測試完成後，分析比較驗證的結果，即測試例子中正確猜中的比率(猜對的筆數除以測試資料總筆數)。

此次實驗總共取得五棵原始決策樹及十棵合併決策樹，用一組測試例子驗證的結果為原始決策樹之平均準確度為 0.887966，標準差為 0.014834393；而合併決策樹的平均準確度為 0.901862463，標準差為 0.012475979。以平均值而言，合併決策樹的平均準確度略高於原始決策樹之平均準確度 1.4%。若進行個別準確度的比較，即合併決策樹(Tij)此原始決策樹(Ti, Tj)，則在二十次的比較之中，合併決策樹的準確度勝過原始決策樹的準確度的比例為 85%，而同時勝過兩棵原始決策樹的比例為 80%。

此外，對於多棵決策樹之合併次序是否影響合併後的結果，本研究進行三次的實驗，每次隨機抽取四千筆例子平均分為五組訓練例子集及一組測試例子集，其中好壞客戶比為１比１，然後再以兩兩合併的方式產生十組合併決策樹，而其中每一組有兩棵合併決策樹，即以決策樹 Ti 為基礎來併入決策樹 Tj 所形成的合併決策樹 Tij 和以決策樹 Tj 為基礎來併入決策樹 Ti 所形成的合併決策樹 Tji。在這三十組的實驗結果顯示不同的合併順序所產生的合併後決策樹之節點個數、選擇節點個數和葉節點個數皆相同，然而其預測準確度會有些微差距(如表 4-1 所示)，其誤差絕對值的平均數是 0.017442254，誤差平均值為-0.007151191，及誤差的標準差為 0.022215834(Sd)。

表 4-1 不同合併次序的合併後決策樹之驗證準確度及其差異

成對樣本(k)	Tij 的準確度	Tji 的準確度	差異 Dk
1	0.904011	0.889685	0.014327
2	0.906877	0.885387	0.02149
3	0.909742	0.896848	0.012894
4	0.912607	0.912607	0
5	0.888252	0.885387	0.002865
6	0.888252	0.901146	-0.01289
7	0.888252	0.916905	-0.02865
8	0.885387	0.901146	-0.01576
9	0.885387	0.916905	-0.03152
10	0.896848	0.912607	-0.01576

註：由於文章篇幅限制，僅列示前十筆資料。

我們更進一步利用統計假設檢定來進行驗證：假設 u1 是合併決策樹 Tij 之平均預測準確度，而 u2 是合併決策樹 Tji 之平均預測準確度，且假定每一對彼此相互獨立，而每一對內之 Tij 和 Tji 的準確度則是相依的。本研究所欲進行之雙尾 t 檢定如下所示：

H_0：(u1-u2)=0

H_1：(u1-u2)≠0

$C_1 = 0 + t_{\alpha/2} * S_d/\sqrt{n} = 0+2.756*0.022215834/\sqrt{30} = 0.01117844$

$C_2 = 0 - t_{\alpha/2} * S_d/\sqrt{n} = 0-2.756*0.022215834/\sqrt{30} = -0.01117844$

在顯著水準 α 值為 0.01 且自由度是 29 的情況下，$t_{\alpha/2}$ 等於 2.756，所以由公式算出上限 C_1 是 0.01117844 及下限 C_2 是-0.01117844，而樣本的平均值為-0.007151191 大於下限，所以接受 H_0，即表示兩者之誤差顯著等於零，也就是兩者的平均準確度是顯著無差異。

就合併演算法之處理邏輯而言，合併的順序是不會影響合併後的結果，然而在此實驗中造成合併後決策樹預測準確度不同的原因在於原始決策樹內有空的葉節點，也就是說沒有例子的葉節點，此時系統會以為基礎的決策樹之葉節點的預測值為合併後葉節點的預測值，因而造成合併決策樹會有不同的預測準確度。

再者，影響合併後決策樹的預測準確度之重要因素是判定最終預測結果值之法則，也就是說當有多條法則在預測某一案例時，其預測之結果值產生差異或預測之葉節點內之各類別例子數之比例相當時，該如何決定出一個最佳之預測結果值。在此，我們是以經驗法則來訂定最終判定之法則，且利用強態葉節點來加強預測之準確度，然而當合併樹的強態葉節點較少時，就無法有效提昇其預測準確度了，所以該如何訂定最佳之判定最終預測結果值是值得進一步探討的課題。

結論

本研究提出一個決策樹形式知識之線上預測系統架構，而它包含有知識學習、知識儲存、知識整合、知識溝通、和知識使用等五種功能。在知識學習方面，我們有一個運用 C4.5 演算法的決策樹知識學習子系統。在知識儲存方面，則利用關聯式資料庫技術所設計之決策樹知識法則庫。在知識整合方面，採用合併選擇決策樹子系統來進行整合多棵決策樹形式的知識。其主要是使用合併選擇決策樹演算法，以兩兩合併的方式來結合多棵決策樹形式的知識，而其運作方式是依循原始決策樹

的樹狀結構，進行比對兩棵原始決策樹，比對相同的節點則進行資料合併，而比對不同的節點則用一個選擇連結來結合，直到所有節點比對完成則結合成一棵簡單結構的決策樹。運用這個演算法有助於維護決策樹法則知識庫內的知識，讓決策樹形式的知識在保有簡單樹狀結構下，進行法則知識之擴充。

在知識溝通方面，則以 XML 為基礎所定義出來之 PMML 文件格式做為與外界溝通的介面。本系統架構提供一個轉換 PMML 文件格式的模組，透過它可以讓知識法則庫內的決策樹形式之預測模式能與其他採礦工具溝通，並有助於視覺化呈現預測模式的內容。最後，在知識使用方面，本系統架構之線上預測子系統可以讓使用者透過瀏覽器，登入預測子系統之網頁，點選所欲使用之預測模式，且瀏覽預測模式之相關文件說明，然後以互動式回答系統走訪決策樹預測模式時所提問之問題，最後得到系統給予之預測結果值及其解釋說明。本系統架構採用以使用者熟悉之 Web 介面做為整個系統架構的平台，主要是讓使用者在進行各單元時，較無學習障礙，如此有助於知識法則之產生、累積和使用。

企業組織可以運用此套系統架構將其日常營運所累積之作業交易記錄，透過前置資料處理後，上傳至知識學習子系統去歸納學習出決策樹形式的知識法則，直接儲存於知識法則庫內。然後再使用合併選擇決策樹子系統把相同問題之決策樹形式知識進行整合，接著就可以讓企業內部人員使用線上預測子系統來取用這些知識法則以進行分類或預測的工作。這套系統可以幫助企業發掘日常營運交易記錄內之隱藏的重要知識法則，並且可以儲存及累積這些知識法則，然後透過線上預測的功能來分享及使用這些知識法則，如此有助於企業內部之知識發掘、儲存、整合、和使用。總公司也可以藉由標準的決策樹 PMML 格式文件來分享其知識法則給各分公司來使用，或將其它採礦工具所發掘出的知識法則轉成 PMML 格式文件來導入到本系統之知識法則庫內，如此有助於企業內部之知識流通及分享。

對於後續之研究，擬針對本系統架構之各項單元進行實作，及探討如何增加本系統架構與外界採礦工具之整合能力。且在合併選擇決策樹演算法方面，擬提出一些修剪策略來提昇合併決策樹的準確度、以及探討如何有效地維護知識法則庫內之決策樹形式的知識。

參考文獻

馬芳資（民 83）。信用卡信用風險預警範例學習系統之研究。國立政治大學資訊管理系碩士論文。

Chan, P. K. & Stolfo, S.J. (1995). A Comparative Evaluation of Voting and Meta-learning on Partitioned Data. Proc. 12th Intl. Conf. on Machine Learning ICML-95, 1995. (http://citeseer.nj.nec.com/chan95comparative.html)

Chan, P. K. & Stolfo, S. J. (1995). Learning Arbiter and Combiner Trees from Partitioned Data for Scaling Machine Learning. Knowledge Discovery and Data Mining. (http://www.cs.columbia.edu/pdis_lab/papers/learning/kdd95/k.ps.Z)

Chan, P. K. & Stolfo, S. J. (1998). On the Accuracy of Meta-learning for Scalable Data Mining. Journal of Intelligent Integration of Information. (L. Kerschberg, Ed.). (http://citeseer.nj.nec.com/chan96accuracy.html)

DMG(2003). The Data Mining Group. http://www.dmg.org.

Freund, Y & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55, 119-139.

Gama, J. (1997). Probabilistic Linear Tree. Proc. 14th International Conference on Machine Learning (http://www.ncc.up.pt/~jgama/icml97.ps.gz)

Hall, L. O. & Chawla, N. & Bowyer, K. W. (1998). Combining Decision Trees Learned in Parallel. (http://www.csee.usf.edu/~chawla/kdd-98.ps)

Kargupta, H., Hamzaoglu, I., Stafford, B., Hanagandi, V., & Buescher, K.(1996). PADMA: Parallel Data Mining Agent for Scalable Text Classification. In Proceedings Conference on High Performance Computing '97 (pp.290-295). The Society for Computer Simulation International.

Kargupta, H., Hamzaoglu, I., Stafford, B., & Hamzaoglu, I. (1997). Web Based Parallel/Distributed Medical Data Mining Using Software Agents. (http://citeseer.nj.nec.com/381.html)

Krishnaswamy, S., Zaslavsky, A. & Loke, S.W.(2000). An Architecture to Support Distributed Data Mining Services in E-Commerce Environments. Proceedings of the Second International Workshop on Advanced Issues in E-Commerce and Web-Based Information Systems. San Jose, California, June 8-9, pp.238-246.

Krishnaswamy, S., Zaslavsky, A.,& Loke, S, W., (2001). Federated Data Mining Services and a Supporting XML Markup Language. Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34), Hawaii, USA, January 2001. In the "e-Services: Models and Methods for Design, Implementation and Delivery" mini-track of the "Decision Technologies for Management" track, IEEE Press, ISBN 0-7695-0981-9.

Kohavi, R. & Kunz, C. (1997). Option Decision Trees with Majority Votes. Machine Learning: Proceedings of the Fourteenth International Conference. (http://citeseer.nj.nec.com/kohavi97option.html)

Kurgan, L., Cios, K. J., & Trombley, M.,(2002). The WWW Based Data Mining Toolbox Architecture. (http://citeseer.nj.nec.com/kurgan02www.html)

PMML 2.0(2003) – Predictive Model Markup Language. http://www.dmg.org/pmmlspecs_v2.

Quinlan, J.R. (1993). C4.5: Programs for Machine Learning, San Mateo: Morgan Kaufmann.

Quinlan, J. R. (1996). Bagging, Boosting, and C4.5. In Proceedings Thirteenth National Conference on Artificial Intelligence, 725-730. AAAI Press.

Quinlan, J. R. (1998). MiniBoosting Decision Trees. Journal of Artificial Intelligence Research. (http://www.cse.unsw.EDU.AU/~quinlan/miniboost.ps).

Todorovski, L., & Dzeroski, S. (2000). Combining Classifiers with Meta Decision Trees. Machine Learning Journal, 50(3), 223-249; Mar 2003

Todorovski, L. & Dzeroski, S. (2000). Combining Multiple Models with Meta Decision Trees. In Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery. Springer, 2000, 54-64.

Williams, G. (1990). Induction and Combining Multiple Decision Trees. Ph.D. Dissertation, Australian National University, Canberra, Australia.

Zenko, B., Todorovski, L. & Dzeroski, S. (2001) A Comparison of Stacking With Meta Decision Trees to Other Combining Methods. ICDM Proceedings A of the Fourth International Multi-Conference Information Society IS' 2001, 144-147. Jozef Stefan Institue, Ljubljana, Slovenia, 2001 (http://ai.ijs.si/bernard/mdts/pub04.ps)