# 支援決策服務之即時預測引擎架構設計

吳政瑋[1] 林柏豪[2] 黎曲峯[3]傅瑞曦[3]陳鏡宇[3] 王秉豐[3] 曾新穆[1]

[1]交通大學資訊工程學系

[2]成功大學資訊工程學系

[3]財團法人資訊工業策進會智慧網通系統研究所

{silvemoonfox, john24wg}@gmail.com, {chufengli, rayfu, chingyuchen, pfwang}@iii.org.tw, vtseng@cs.nctu.edu.tw

## 摘要

隨著物聯網(Internet of Things)技術及大數據(Big Data)的應用興起，如何利用決策支援系統(Decision Support System)有效率地分析大數據，改進並加速使用者制定決策是一門重要的研究議題。先前的研究大多著重於非記憶體內計算的支援決策系統，因而無法即時且有效率地從大數據中萃取有用的資訊。為了解決上述問題，本研究提出一套支援決策服務的即時預測引擎架構，在分散式環境及記憶體內計算的架構下(如：Apache Spark)，利用整體學習(Ensemble)，融合多種預測模型的預測結果，做出最終決策之服務機制，達到快速判斷與即時預測之功能。

**關鍵詞**：大數據應用、決策支援系統、Apache Spark、整體學習

## Abstract

Developing a *decision support system* to efficiently and effectively analyze *big data* is an important research issue in many applications. However, most of existing studies did not integrate decision support system with in-memory computing technology and use different types of prediction models to effectively make prediction. Therefore, they could not provide users with accurate prediction results in real time. In view of this, this paper proposes a novel framework for real-time prediction engine of decision support services, which allows to make prediction in real-time with high scalability. We develop a novel distributed and in-memory based prediction module, which uses in-memory computing technology (e.g., Apache Spark) to enhance the overall performance of the system. Thus, the proposed framework is able to handle big data with high scalability and provide more accurate prediction to users.

**Keywords**: Big data applications, decision support system, Apache Spark、ensemble

## 1. 前言

隨著智能手機(Smart Phone)、感知裝置(Sensing Devices)、物聯網(Internet of Things)、社群媒體(Social Media)及開放資料(Open Data)的盛行[13][14]，各種類型的大數據(Big Data)[3][4][7][17]正以驚人的速度呈倍數性成長。依據國際研究暨顧問機構 Gartner[28] 的定義，大數據為「大量、快速累積、具有多樣性的資訊資產，需要新的處理技術以提升決策品質、發掘問題、最佳化流程」。

由於大數據隱含潛在、未知且有用的資訊，如何設計一套支援決策系統(Decision Support System)有效率且即時地分析大數據，幫助決策者有效率地制定決策，已儼然成為一個重要的研究議題[3][4][6][7][17] [30][31]。然而，由於大數據具備三項主要特性: (1)大量(Volume)、(2)快速(Velocity)及(3)多樣性(Variety)[3][4][7][17]，因而導致傳統支援決策系統無法在單機環境中執行。因此，我們需將平行運算(Parallel Computing)[25][26]的概念整合至欲開發的決策系統中。

然而，結合平行運算發展一套即時支援決策系統可能會遭遇以下幾個問題：(1)所開發的方法能將主要問題分解成數個獨立的子工作，交由數台電腦分別執行。若分解的不恰當，系統容易發生負載不平衡之問題(Load Balancing Problem)，進而降低預測模組的執行效率。(2)當數台電腦平行處理時，系統需要考慮同步性問題(Synchronization Problem)以降低電腦間傳遞訊息的成本(Communication Overheads)。若無法妥善處理同步性問題，不僅可能造成系統執行效率低落，甚至導致不正確的運算結果。(3)系統需具備容錯能力(Fault Tolerance Capability)與錯誤回復能力(Fault Recovery Capability)。當某些電腦發生故障時，系統仍能正確地進行預測工作而不會導致錯誤結果。(4)系統需具備可延展性(Scalability)。當資料量增加時，系統可藉由增加電腦提升處理效率與品質。

記憶體內計算(In-Memory Computing)技術[26]是近年來最熱門的大數據分析技術之一，其主要概念是將傳統在應用伺服程式(Application Server)、資料庫伺服程式(Database Server)、儲存設備間的資料交換與運算的過程，改為在記憶內完成。其主要的特色包括: (1)將資料存放在記憶體中以加快資料處理速度；(2)透過壓縮技術減少資料量；(3)減少資料的移動，僅搬移運算後的結果，而非搬移資料去運算。Apache Spark [26]為近年來最被推崇的雲端計算開發系統之一，有別於硬碟式 MapReduce 架構的 Hadoop [25]，Apache Spark 整合記憶體內計算技術有效率地改善傳統 Hadoop 無法有效利用記憶體處理資料之缺點。在 Sort Benchmark Competition (資料排序基準競賽)中，Spark 以不到三十分鐘的執行速度，排序多達 100 Terabytes 的資料量，打破了由 Hadoop 保有 72 分鐘的世界記錄。

Apache Spark 的主要特點包括: (1)支援 Scala、Java 和 Python 等三種高階程式語言，以方便不同領域之程式設計師開發應用。(2)實驗結果顯示 Apache Spark 憑藉其特有的 RDD (Resilient Distributed Dataset) 記憶體內計算技術，解決傳統 Hadoop MapReduce 架構在迭代式演算法中所遇到的效能瓶頸，提供優於 Hadoop MapReduce 運算架構約十到一百倍的執行效能。(3) 與許多大數據分析系統如: Hadoop HDFS、S3、Storm 等系統的存儲介面相容，具備豐富的擴充模組如:以 SQL 指令操作資料的 SparkSQL、豐富的機器學習(Machine Learning)函式庫 MlLib 以及即時串流處理技術 Spark Stream 等。隨著記憶體價格的逐漸下降，In-Memory Computing 技術的採用已經越來越向主流靠近。

雖然目前已有相關研究致力於記憶體內計算技術的機器學習演算法[5][8][18][20][22][24]與資料探勘(Data Mining)技術[1][12][15][16][19][20][22][23][27]，然而這些方法並未針對平行運算架構的即時支援決策系統而設計。有鑑於此，本研究提出一套以 Apache Spark 為基礎的支援決策服務系統架構，其結合平行式架構及記憶體內計算技術，利用整體學習(Ensemble)技術[2][9][10][11][15][21][23]，融合多種預測/分類模型的預測結果，做出最終決策之服務機制，達到快速判斷與即時預測之功能。實驗結果顯示本研究所提出的系統具有優良的延展性(Scalability)及相當快速的執行效率。

## 2.相關研究

此章節介紹關於平行運算系統(Parallel Computing System)[3]的相關研究，包括: (1)共享記憶體式(Shared Memory-based) 與 (2)無共享式(Shared-nothing)系統。

### 2.1 共享記憶體式平行運算系統

共享記憶體式平行運算系統之主要特色是該系統允許多個處理單元(Processing Unit)同步存取一個共享的記憶體空間，例如: 含有數千個小型高效率核心的 GPU(Graphics Processing Unit)多核心系統。由於該系統所共享的記憶體空間是固定的，當資料量超過固定記憶體空間所能處理之範圍時, 該系統會因記憶體容量不足而無法執行。換言之，此類型的系統通常延展性不佳，較不適用於處理大數據。

### 2.2 無共享式平行運算系統

無共享式系統又稱為分散式計算系統，該系統的主要特色是在不共享記憶體的情況下，允許多個處理單元藉由傳遞訊息完成一件工作。這種類型的系統通常較具備延展性。MPI (Message Passing Interface)無共享式架構便是一個典型的例子。然而，MPI 只適用於低階程式語言(例如: C 和 Fortran)的平台，並不適用於高階程式語言(例如: Java 和 R)的平台與應用。有鑒於此，Apache Foundation 開發了一套以 Java 高階程式語言為基礎的的開放原始碼(Open Source)，稱為 Hadoop[25]，用以快速平行處理大數據。

Hadoop 包含兩個主要的核心技術: HDFS (Hadoop Distributed File System) 與 MapReduce Software。HDFS 是一個可靠的分散式檔案系統(Reliable Distributed File System)，而 MapReduce Software 則是用來平行處理大數據的技術。一個 MapReduce 的程序包含兩個步驟:Map Stage 與 Reduce Stage。在 Map Stage，每一個 Mapper 處理一小段資料並將這些資料換成數個鍵-值(Key-Value Pairs)。在 Reduce Stage，Reducer 則負責彙整 Mapper Stage 所產生的 Key-Value Pairs，用以求得結果。Hadoop 的應用相當地廣泛，可用來處理資料的加減乘除運算、矩陣相乘/除、排序、計數等問題。雖然 Hadoop 可能適用於大數據分析中的某些應用，但該系統在處理資料時並未使用 記憶體內計算技術，導致效率不佳。

## 3.所提出的方法

本研究所提出的即時支援決策系統其架構如圖 1 所示，共包括五個主要的模組: (1)使用者介面模組(User Interface Module)、(2)模型管理模組(Model Management Module)、(3)資料管理模組(Data Management Module)、(4)分散及記憶體內計算預測模組(Distributed and In-memory based Prediction Module)、(5) 決策服務機制模組(Decision Service Mechanism Module)。以下詳細敘述各模組之主要功能及特色。

### 3.1 使用者介面模組

此模組提供使用者介面，使用者可以使用模型管理模組及資料管理模組分別對決策模型及要預測的資料進行管理、操作及存儲。此外，使用者亦可透過此介面設定相關參數。設定完相關參數後，即可使用分散及記憶體內計算預測模組進行即時決策。

### 3.2 模型管理模組

由於各類型的決策/預測模型被廣泛地運用於不同領域中，且模型往往隨著使用者的需求而新增或刪除，如此才能確保支援決策系統的效能穩定。然而，傳統方法對於模型的管理卻還是停留在以人工作業的方式進行管理。此模組可方便使用者管理不同預測模型並縮短佈署預測模型所需之時間。該模組之主要功能包括:

(1) 依照使用者需求，方便使用者挑選決策模型。

(2) 方便使用者監測模型效能，汰換效率差的模型及新增效率高的模型，節省人力成本並提昇系統的整體執行效能。

(3) 紀錄模型生命週期，幫助使用者監督模型上線到下線的時間，進而協助使用者判斷何時需調整模型或重建模型。

(4) 依照使用者需求，方便使用者挑選 Ensemble 機制。

### 3.3 資料管理模組

透過資料管理模組，使用者可以輕易地存儲及管理要預測的資料，並可透過此模組將要預測的資料匯入至分散及記憶體內計算預測模組進行即時決策。

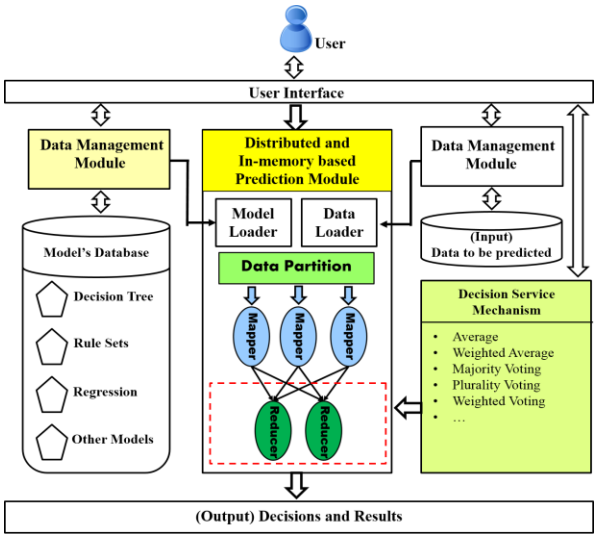


圖 1、所提出之即時支援決策系統架構圖

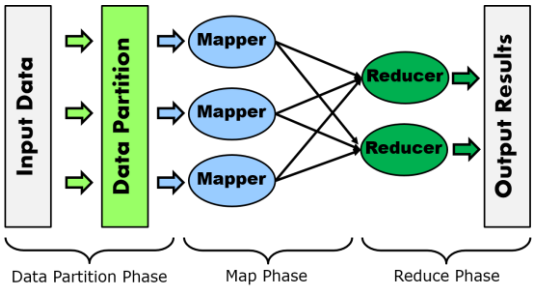


圖 2、分散及記憶體內計算預測模組架構圖

### 3.4 分散及記憶體內計算預測模組

由於先前方法的架構(如: Apache Hadoop)在使用模型進行預測時，大多將中繼計算結果儲存於硬碟上，所以當資料量大時，容易導致系統花過多的時間於 I/O 上，進而降低系統的整體執行效能。

有鑑於此，本研究提出一個創新的分散及記憶體內計算預測模組架構，其可依據 Machine Capacity 在如 Apache Spark 之記憶體內計算架構下，以分散式計算方式結合數個預測模型之預測結果，以平行處理大量資料並達到快速判斷與即時預測。在圖 1 中，Data Loader 以及 Model Loader 模組之主要功能將使用者選定的決策模型及欲進行預測的資料匯入記憶體供分散及記憶體內計算預測模組進行後續處理。

本研究所提出的分散及記憶體內計算預測模組，其架構如圖 2 所示，主要可分為三個 Phase，分別為: (1) Data Partition Phase、(2) Map Phase 以及 (3) Reduce Phase。以下分別敘述各 Phase 之主要目的及其處理過程。

(1) **Data Partition Phase:** 此 Phase 的主要目的乃是將要預測的資料切割成數個互不重疊的 Partition，並將 Partition 指派到不同的 Mapper 以利後續 Map-Reduce 之處理。

(2) **Map Phase:** 此 Phase 的輸入資料為 Data Partition Phase 所產生的 Partition，也就是一組要預測的資料集合。令 $M^* = \{M_1, M_2, \ldots, M_k\}$ 為使用者所選擇的 $k = |M^*|$ 個預測模型。而每一個 Mapper 會將 Partition 中每筆資料 $D_j$ 複製 $k$ 次，並對第 $i$ 次 $(1 \le i \le n)$ 複製所產生的的資料 $D_{ji}$ 呼叫 $PF(D_{ji}, M_i)$ 預測函數(*Prediction Function*)，此函數計算出模型 $M_i$ 對於 $D_{ji}$ 之預測結果 $R_{ji}$。接著，Mapper 會將這些結果整理成 Key-Value 的形式 $\langle D_{ji}, (M_i, R_{ji})\rangle$，並將這些 Key-Value Pairs 輸入至 Reducer 中。

(3) **Reduce Phase:** 在 Reduce Phase 中，Reducer 會收集擁有相同 Key 的 Key-Value Pairs $\{\langle D_{j1}, (M_1, R_{j1})\rangle, \langle D_{j2}, (M_2, R_{j2})\rangle, \ldots, \langle D_{jn}, (M_n, R_{jk})\rangle\}$，其中 $D_j = D_{j1} = \ldots = D_{jk}$。每一筆 Key-Value Pair 代表模型 $M_i$ 對於資料 $D_j$ 的預測結果。接著 Reducer 會依照本研究所提出的決策服務機制模組架構將擁有相同 Key 的 Value 匯整成最終的預測結果。

分散及記憶體內計算預測模組之主要優點及特色如下:

(1) 整合記憶體內計算技術[26]，將資料存放在記憶體中，以加快資料處理速度，有效率地改善傳統方法無法有效使用記憶體處理資料之效能瓶頸。

(2) 沿襲 Apache Spark [26]之各項優點，如: (a)降低負載平衡之問題(Load Balancing Problem)，進而提升支援決策系統之整體效率; (b)解決同步性問題，降低電腦間傳遞訊息的成本; (c)具備容錯能力與錯誤回復能力; (d)具備可延展性(Scalability), 系統可藉由增加電腦提升處理效率與品質。

(3) 支援三種高階程式語言 Scala，Java 和 Python，以方便不同領域之程式設計師開發應用。

(4) 具備豐富的擴充模組並與其它 Echo-system 具備高度相容性，如: Spark Streaming、MLlib、Spark SSQL、Hive、Hadoop MapReduce[12]、及 Storm 等系統。

### 3.5 決策服務機制模組

決策服務機制模組之主要目的乃利用 Ensemble 整合方法[2][9][10][11][15][23]，融合多種預測模型的預測結果，以記憶體內計算方式，提升支援決策系統的預測能力並做出最終決策之服務機制。Ensemble 整合方法的主要概念如下所述: 令 $M^* = \langle h_1, h_2, h_3, \ldots, h_T\rangle$ 為預測/決策模型所組成的集合，其中 $h_i$ $(1 \le h \le T)$ 為第 $i$ 組模型。假設查詢為 $x$，$h_i(x)$ 為模型 $h_i$ 所預測的結果，$H(x)$ 為融合 $h_1$、$h_2$、$h_3$、...、$h_T$ 的決策結果。

常見的 Ensemble 整合方法包括: (1)平均法 (Average) 、 (2) 加權平均法 (Weighted Average)、(3)過半數投票法(Majority Voting)、(4)多數投票法(Plurality Voting)與(5)加權投票法(Weighted Voting)，其中平均法與加權平均法適用於數值型目標屬性，而過半數投票法、多數投票法與加權投票法適用於類別型目標屬性。其中，令 $h_i^r(x)$ 為第 $i$ 組模型預測資料 $x$ 屬於目標屬性 $c_r$ 的情況，過半數投票法的計算方式是以超過半數的模型所預測的結果為最終預測結果，其計算公式為:

$$H(x) = c_r, \text{ 若 } \sum_{i=1}^{T} h_i^r(x) > \frac{1}{2}\sum_{k=1}^{s}\sum_{i=1}^{T} h_i^s(x) \text{ 。}$$

若不符合上述條件，則系統回覆 Rejection，表示無法預測 $x$ 所屬的類別。

換言之，若採用過半數投票法，在 Reduce Phase 中，模型 $M_i$ 對第 $j$ 筆資料 $D_j$ 的預測結果為 $R_{ji}$，令 $x$ 等於 $D_j$ 且 $h_i(x)$ 等於 $R_{ji}$，代入 $H(x)$ 中，即可求出 $D_j$ 的經過 Ensemble 後的預測結果。雖然其它 Ensemble 方法也可適用於本研究所開發的系統中，我們目前僅有實作平均法。在未來的研究中，我們會持續將其它 Ensemble 的方法整合至所開發的系統中。

## 4. 實驗分析

本章節以實驗驗證所提方法之執行效率。由於先前的研究已證實 Ensemble 整合方法的準確率優於單一預測模型，故本研究僅針對所提方法的執行效率進行實驗分析。實驗所採用的資料集為模擬資料，其包含 780 個條件屬性以及 1 個二元型的目標屬性。實驗所使用的硬體為 1 台 Master (RAM 32G，CPU Intel i7) 和 6 台 Slave (RAM 8G， CPU Intel Celron)，所提出的方法由 Java 實作(版本為 Spark 1.5.2)，共使用 8 個節點(Node)。所採用的預測模型為 MiLib 所提供的決策樹(Decision Tree)。

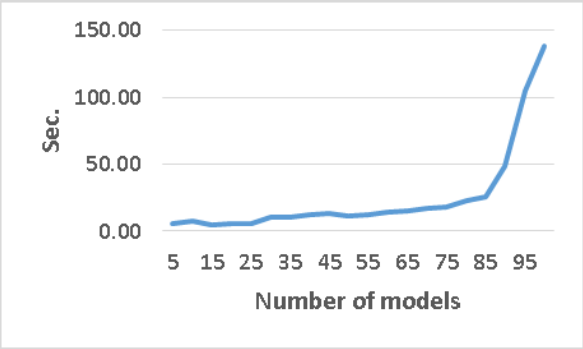


圖 3、所提出的方法在不同模型數量時的執行速度

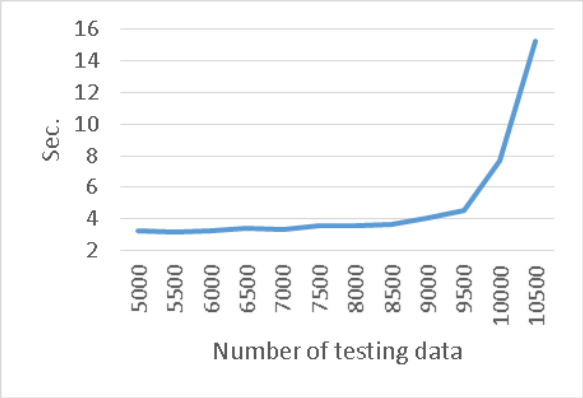


圖 4、所提出的方法在不同資料筆數時的執行速度

在第一組實驗中，我們測試所提出的方法在不同模型數量的執行速度。此實驗所採用的測試資料(Testing Data)筆數固定為 1,000 筆，模型的數量以 5 為間隔，由 5 變動到 100，例如 5、10、…、100。圖 3 顯示本組實驗的實驗結果。從圖 3 我們可看出本研究所提出的方法具備良好的執行效率。舉例說明，當模型數量為 100 時，由於要進行預測的資料共有 1,000 筆，一共要進行 100 × 1,000 = 100,000 次預測。面對如此龐大的資料量，本研究所提出的方法其執行速度為 138 秒。換言之，平均一筆資料的執行時間只需 0.0138 秒。由於所提出的方法乃基於記憶體內計算技術，並利用 MapReduce 平行化技術處理要進行預測的資料，因此當模型數量增加時，所提的方法仍然能夠非常有效率。

在第二組實驗中，我們測試所提出的方法在不同資料筆數時的執行速度。此實驗所採用的模型數量固定為 10，資料筆數以 500 為單位，由 5,000 變動到 10,000，例如 5000、5400、…、10,000。圖 4 顯示本組實驗的實驗結果。從圖 4 我們可看出本研究所提出的方法具備良好的執行效率。舉例說明，當資料筆數為 10,000 時，所提出的方法只花費將近 16 秒。因此，我們所提的方法具備相當良好的延展性。

## 5. 結論

本研究所發展的「支援決策服務之即時預測引擎架構設計」乃針對大數據環境之特性而設計，其主要目的乃結合記憶體內計算技術及平行計算架構，提出一套支援決策服務的即時預測引擎架構，以達到即時性、準確性與可擴展性之預測功能以支援決策服務。實驗結果顯示本研究所提出的方法具備優良執行效率及延展性。

## Acknowledgement

這篇論文是財團法人資訊工業策進會研究成果的一部份。我們在此感謝資策會支持這個計劃的研究。

## 參考文獻

[1] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” in Proc. of Int'l Conf. on Very Large Data Bases, pp. 487-499, 1994.

[2] L. Breiman, “Bagging Predictors,” Machine Learning, Vol. 24, No. 2, pp. 123-140, 1996.

[3] A. Bifet, “Mining big data: current status, and forecast to the future,” in Proc. of ACM SIGKDD Explorations, Vol.14, Issue 2, 2012.

[4] A. Bifet, “Mining big data in real time,” Informatica Vol. 37, pp. 15-20, 2013

[5] L. Breiman, J. H. Friedman , R. A. Olshen and C. J. Stone, “Classification and Regression Trees,” Wadsworth, Belmont, 1984.

[6] A. Bifet, H. Kirkby, G. R. Pfahringer, B, “MOA: Massive Online Analysis,” Journal of Machine Learning Research, 2010. Available at: http://moa.cms.waikato.ac.nz/

[7] J. J. Berman, “Principle of Big Data: Preparing, Sharing, and Analyzing Complex Information,” Published by Morgan Kaufmann.

[8] T. M. Cover and P. E. Hart, “Nearest

Neighbor Pattern Classification," IEEE Trans. Inform. Theory, Vol. 1-13, pp. 21-27, 1967.

[9] T. G. Dietterich, "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization," Machine Learning, pp. 1-22, 1999.

[10] B. Efron and R. Tibshirani, "An Introduction to The Bootstrap," London: Chapman and Hall, 1993.

[11] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, "From Data Mining to Knowledge Discovery in Databases," AI Magazine, pp. 37-54, 1996.

[12] B. Guo et al, "Opportunistic IoT: Exploring the harmonious interaction between human and the internet of things," Journal of Network and Computer Applications, Vol 36, Issue 6, pp. 1531-1539, 2013.

[13] K. Hwang, G. C. Fox and J. J. Dongarra, "Distributed and Cloud Computing – From Parallel Processing to the Internet of Things," Published by Morgan Kaufmann.

[14] J. Han, M. Kamber and J. Pei, "Data Mining: Concepts and Techniques, 3rd Edition," Published by Morgan Kaufmann.

[15] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," in Proc. of the ACM SIGMOD Int'l Conf. on Management of Data, pp. 1-12, 2000.

[16] E. Letouzé, "Big Data for Development: Opportunities & Challenges," 2011.

[17] P. Langler, W. Iba and K. Thompson, "An Analysis of Bayesian Classifiers," AAAI Press, pp. 223-228, 1992.

[18] J. R. Quinlan, "Induction of Decision Trees," Machine Learning, Vol 1, pp. 81-106, 1986.

[19] J. R Quinlan, "C4.5 Programs for Machine Learning," Morgan Kaufmann Publishers, San Mateo, CA., 1993.

[20] J.R. Quinlan, "Bagging, Boosting and C4.5," 1996.

[21] J. H. Sang, S. S. Venkatesh, E. A. Conant, P. H. Arger, C. M. Sehgal, "Comparative Analysis of Logistic Regression and Artificial Neural Network for Computer-Aided Diagnosis of Breast Masses," 2005.

[22] P. Tan, M. Steinbach and V. Kumar, "Introduction to Data Mining," Published by Addison-Wesley.

[23] I. H. Witten and E. Frank, "Data Mining: Practical Machine Learning Tools and Techniques. 2nd Edition," Morgan Kaufmann, San Francisco, 2005.

[24] Apache Hadoop. Available at: http://hadoop.apache.org

[25] Apache Spark. Available at: https://spark.apache.org/

[26] Gartner Incorporation. Available at: http://www.gartner.com/technology/home.jsp

[27] International Data Corporation (IDC). Available at: http://www.idc.com.tw/

[28] SAMOA. Available at: http://samoa-project.net

[29] Vowpal Wabbit. Available at: http://hunch.net/~vw/