Moving Data Mining Tools toward a Business Intelligence System

Nittaya Kerdprasop, and Kittisak Kerdprasop

***Abstract*—**Data mining (DM) is the process of finding and extracting frequent patterns that can describe the data, or predict unknown or future values. These goals are achieved by using various learning algorithms. Each algorithm may produce a mining result completely different from the others. Some algorithms may find millions of patterns. It is thus the difficult job for data analysts to select appropriate models and interpret the discovered knowledge. In this paper, we describe a framework of an intelligent and complete data mining system called SUT-Miner. Our system is comprised of a full complement of major DM algorithms, pre-DM and post-DM functionalities. It is the post-DM packages that ease the DM deployment for business intelligence applications.

***Keywords*—**Business intelligence, data mining, functional programming, intelligent system.

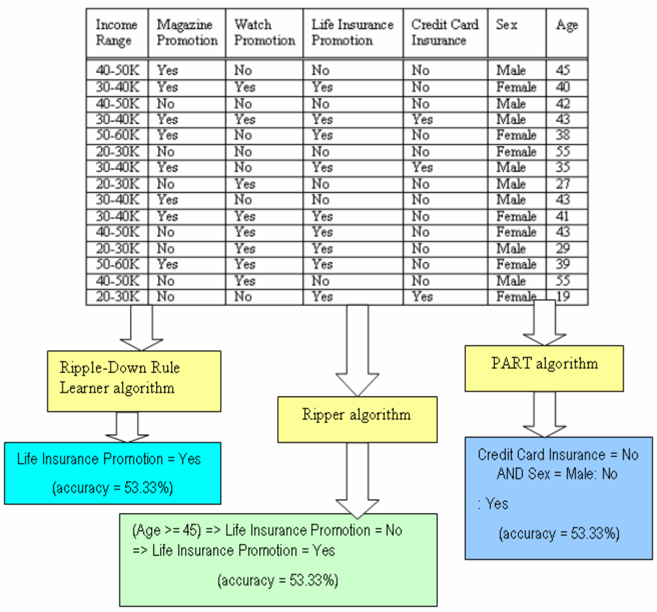
I. INTRODUCTION

DATA mining (DM) or *Knowledge Discovery in Databases* (KDD) has been defined [3] as the automatic discovery of previously unknown patterns or relationships in large and complex datasets. Most DM algorithms have been drawn from the areas of Statistics and Machine Learning adapted to induce knowledge from data contained within a database. The main objective of DM is to use the discovered knowledge for the purposes of explaining current behavior, predicting future outcomes, or providing support for business decision. The DM techniques used in business-oriented applications are also known as *Business Intelligence* (BI). BI is a general term to mean all processes, techniques, and tools that gather and analyze data for the purpose of supporting enterprise users to make better decisions [1], [7].

Despites its high claims and expectations, DM technology requires a highly trained professional to do an iterative, multi-step process of accessing and preparing data, choosing an appropriate algorithm to mine the data, analyzing the learned knowledge, and presenting nontrivial, valuable knowledge to executives or decision makers. Owing to advancement in the machine leaning research, mining can be done efficiently on a dataset of large size. A hindrance of DM employment as an automatic knowledge acquisition tool in BI is the part of post-mining evaluation to obtain only a relevance and valuable knowledge.

The difficulty of discovering and deploying new knowledge in the BI context is due to the lack of intelligent and complete DM system. Most DM packages are comprised of learning algorithms integrated into a visual environment. Such graphical environment is a useful facility for experienced data analysts or data miners, but it provides limited functionalities for a novice to interpret and evaluate significance of the mining results.

As an example, consider Fig. 1 that shows the three different mining results obtained from three rule-induction algorithms: Ripple-Down Rule Learner, Ripper (Repeated Incremental Pruning to Produce Error Reduction), PART. The dataset is taken from the credit card promotion database [9]. The mining objective is to learn a profile for individuals likely to take advantage of a life insurance promotion advertised along with their credit card statement. A learned profile can help the credit card company to send the promotion materials only to a select group of individuals who are likely to take advantage of a life insurance promotion.



Income Range	Magazine Promotion	Watch Promotion	Life Insurance Promotion	Credit Card Insurance	Sex	Age
40-50K	Yes	No	No	No	Male	45
30-40K	Yes	Yes	Yes	No	Female	40
40-50K	No	No	No	No	Male	42
30-40K	Yes	Yes	Yes	Yes	Male	43
50-60K	Yes	No	Yes	No	Female	38
20-30K	No	No	No	No	Female	55
30-40K	Yes	No	Yes	Yes	Male	35
20-30K	No	Yes	No	No	Male	27
30-40K	Yes	No	No	No	Male	43
30-40K	Yes	Yes	Yes	No	Female	41
40-50K	No	Yes	Yes	No	Female	43
20-30K	No	Yes	Yes	No	Male	29
50-60K	Yes	Yes	Yes	No	Female	39
40-50K	No	Yes	No	No	Male	55
20-30K	No	No	Yes	Yes	Female	19

Ripple-Down Rule Learner algorithm
Ripper algorithm
PART algorithm
Life Insurance Promotion = Yes (accuracy = 53.33%)
(Age >= 45) => Life Insurance Promotion = No
=> Life Insurance Promotion = Yes
(accuracy = 53.33%)
Credit Card Insurance = No AND Sex = Male: No
: Yes
(accuracy = 53.33%)

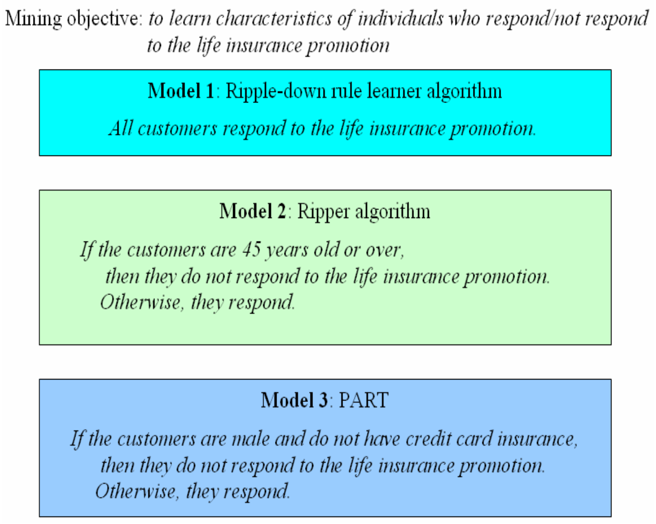
Fig. 1 Different mining results obtained from different algorithms

Manuscript received November 30, 2006. This work was supported in part by the research fund from Suranaree University of Technology and the grants from the National Research Council of Thailand, the Thailand Research Fund.

Nittaya Kerdprasop is with the School of Computer Engineering, Suranaree University of Technology, 111 University Avenue, Muang District, Nakhon Ratchasima 30000, Thailand (phone: +66-44-224432; fax: +66-44-224602; e-mail: nittaya@ sut.ac.th, nittaya.k@gmail.com).

Kittisak Kerdprasop is with the School of Computer Engineering, Suranaree University of Technology, 111 University Avenue, Muang District, Nakhon Ratchasima 30000, Thailand (e-mail: kerdpras@sut.ac.th).

The mining results (*i.e.* classification models) shown in fig. 1 obtained from the three runs on WEKA system [10], [11]. The three classification models show the same accuracy when tested with 10-fold cross validation technique, but all three models produce different knowledge. The knowledge conveyed by each model can be explained in Fig. 2.



Mining objective: *to learn characteristics of individuals who respond/not respond to the life insurance promotion*

Model 1: Ripple-down rule learner algorithm

All customers respond to the life insurance promotion.

Model 2: Ripper algorithm

If the customers are 45 years old or over,
then they do not respond to the life insurance promotion.
Otherwise, they respond.

Model 3: PART

If the customers are male and do not have credit card insurance,
then they do not respond to the life insurance promotion.
Otherwise, they respond.

Fig. 2 Interpretation on mining results

It can be seen that even though model 1 is as accurate as models 2 and 3, it provides less knowledge on predicting the response of future customers. It is, however, the burden of the data analyst to choose either model 2 or 3, or even to combine both models as new knowledge for the task of customer segmentation.

To choose and use the mining results are not a trivial task because the model-selection facilities provided by most DM packages are very rare or limited. It is, therefore, the objective of this research to design and implement a complete framework of a knowledge mining system called the *SUT-Miner* (it is named after the sponsor of this project, *i.e.* Suranaree University of Technology). We design the system to be both intelligent and complete with the full functionalities of pre-DM, DM, and post-DM. The system is an extension of the work presented in [5], [6].

This paper presents work in progress of the development of SUT-Miner system. The implementation is based on the functional paradigm using the Haskell language. The organization of this paper is as follows. Section 2 provides a brief explanation of DM tasks that are included in the design of our system. Section 3 sketches the overall architecture of the system. Section 4 explains the implementation and the experimental results showing the advantage of functional programming over the object-oriented programming. Section 5 concludes the paper and suggests an extension of the system towards an approximate and progressive scheme.

II. Data Mining Conceptual Model

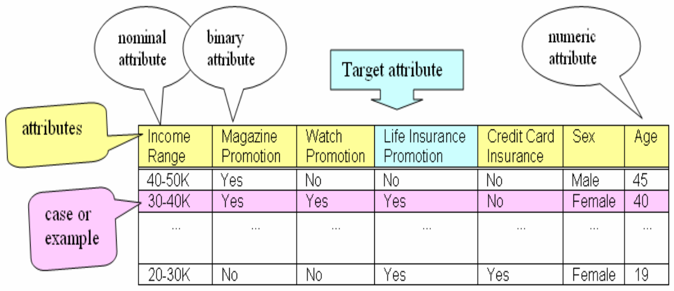
DM is about learning patterns. *Pattern* is an expression describing a subset of the data, *e.g.* $f(x) = 3x^2 + 3$ is a pattern induced from a given dataset {(0,3), (1,6), (2,15), (3,30)}, whereas the term *model* refers to a representation of the source generating the data, *e.g.* $f(x) = ax^2 + b$. However, in his paper we use the term pattern and model interchangeably.

According to [3], DM involves fitting models to, or determining patterns from, observed data. Primary goals of DM are prediction and description.

Prediction uses supervised learning technique to predict values of data using known values found from different data. DM tasks for prediction include classification, regression, time-series analysis.

Description focuses on employing unsupervised learning technique to find human-interpretable patterns describing the data. DM tasks for description are clustering, summarization, association, and sequence discovery.

To start building a DM methodology, it is necessary to set up a conceptual framework into which data and its structures might be classified. The terminology to denote structures of the dataset is summarized in Fig. 3.



Income Range	Magazine Promotion	Watch Promotion	Life Insurance Promotion	Credit Card Insurance	Sex	Age
40-50K	Yes	No	No	No	Male	45
30-40K	Yes	Yes	Yes	No	Female	40
...	...	...	...	...	...	...
20-30K	No	No	Yes	Yes	Female	19

Fig. 3 Summarization of a dataset terminology

We adopt the ontology of DM methodology proposed by [8] to abstract data and their relationships for guiding the DM method selection. The simple ontology presented in Fig. 4 provides a means to explicitly guide the novice data miners towards DM task selection. This guideline is data-driven in that the data types and structures are used as a basis for selecting appropriate DM method.

Our data mining system consists of three main parts: pre-DM, DM, and post-DM. The ontology presented in Fig. 4 is for a method selection in the DM part. We also need complete ontology for the parts of pre-DM and post-DM as well as the full specification of each DM task. These will be our future work.

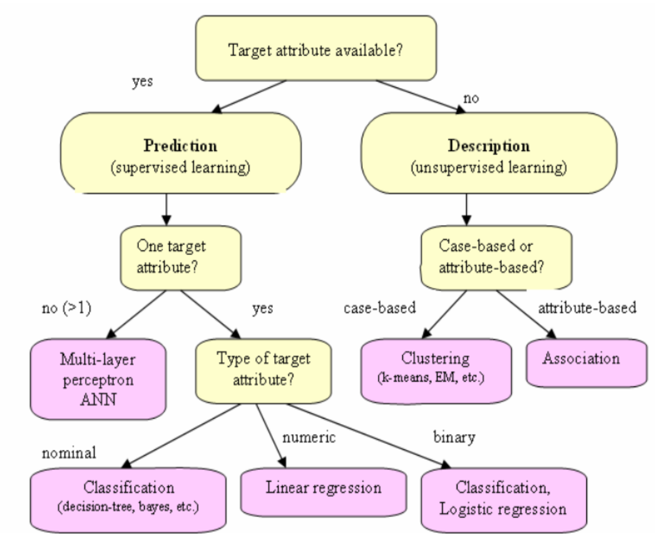


Fig. 4 A simple ontology for DM method selection

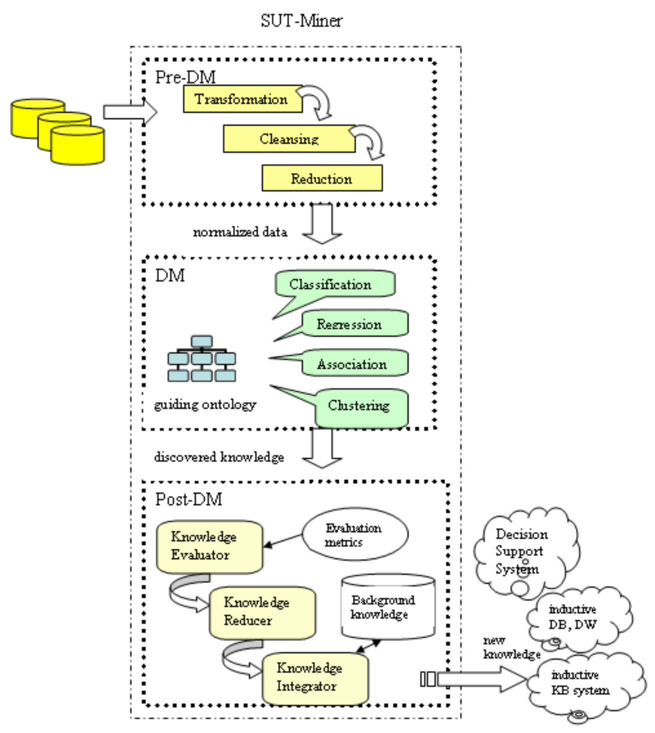


Fig. 5 The architecture of SUT-Miner system

III. An Overall Architecture of SUT-Miner

At a high level of our framework, we design the SUT-Miner system to be comprised of three main phases: pre-DM, DM, post-DM.

The pre-DM phase performs data preparation tasks such as to locate and access relevant data set(s), transform the data format, clean the data if there exists noise and missing values, reduce the data to a reasonable and sufficient size with only relevant attributes.

The DM phase performs mining tasks including classification, prediction, clustering, and association. The post-DM phase involves evaluation, based on corresponding measurement metrics, of the mining results. DM is an iterative process in that some parameters can be adjusted and then restart the whole process to produce a better result.

The post-DM phase is composed of knowledge evaluator, knowledge reducer, and knowledge integrator. These three components perform major functionalities aiming at a feasible knowledge deployment which is important for the applications in BI. The overall architecture of our SUT-Miner system is presented in Fig. 5.

IV. Rapid Prototyping with Haskell

The implementation of SUT-Miner system is mainly based on the functional programming paradigm using Haskell language [2], [4]. Functional languages (FL) offer a number of advantages over imperative languages (IL). FL can be used to express specifications of problems in a more concise form than IL. This results in the creation of program source codes that are shorter and easier to understand. The following example shows C versus Haskell codes to compute a list of fibonacci numbers starting with zero.

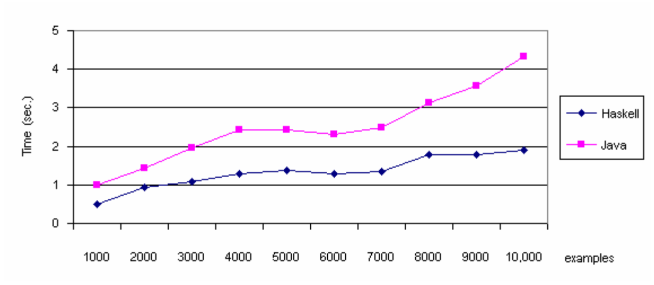
C-code

```
int * fib (int n)
  {  int a = 0, b = 1, i, temp;
     int * fibsequence;
     fibsequence = (int *) malloc ((sizeof int) *n);
     for (i = 0; i<n; i++)
        {  fibsequence[i] = a;
           temp = a + b;
           a = b;
           b = temp;
         }
     return fibsequence;
  }
```

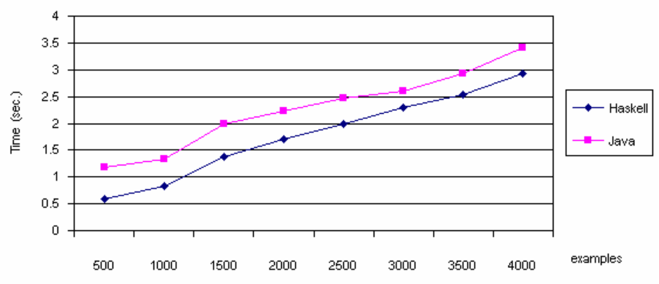
Haskell-code

```
fib :: [Int]
fib = 0: 1: [ a+b | (a, b) <- zip fib (tail fib) ]
```

Haskell is a pure FL having a polymorphic type system, *i.e.* a data type can take type variables as parameters. This feature provides a high level abstraction leading to generic programming. Haskell is also a lazy FL, *i.e.* a value is evaluated only when it is needed. This feature allows infinite structures, such as an infinite sequence of fibonacci numbers, to be defined. According to our experimentation, the speed of running Haskell program on a moderate-size dataset is quite impressive. The experimental results shown in Fig. 6 compare the running time of a Haskell program against a Java program for mining a linear regression model. The first experiment computes a regression model of two variables. The number of variables is increased to six in the second experiment. The experimentations are performs on a computer notebook with CPU speed 1.8 GHz and main memory 512 MB.



(a) mining regression model of two variables



(b) mining regression model of six variables

Fig. 6 Mining regression model with Haskell and Java

V. Conclusion and Future Work

We present work in progress on the development of the SUT-Miner, a complete data mining system. The system is complete in that the pre-DM and post-DM phases are also included in the DM process. Most DM packages contain only the DM modules, while some systems incorporate a pre-DM module as a data preparation phase.

According to our knowledge, a post-DM phase is omitted in most systems. Post-processing of DM is very essential to the success of DM utilization. This is due to the fact that discovered knowledge is sometimes voluminous and redundant. At present, knowledge evaluation and filtration have to be done by human experts. We thus design our system to include this knowledge processor as another major component of the mining system.

The implementation of the SUT-Miner system uses a Haskell functional language. The functional programming is a paradigm of our choice because of its advantages on modularity, conciseness, polymorphism, and formal specification which supports the proof of program correctness. We plan to extend our design to produce an approximate model by means of progressive mining. We currently investigate the feasibility of applying a Markov Chain Monte Carlo method in our approximate data mining scheme.

Appendix

The Haskell code to mine two-variable and six-variable regression models is provided here.

```
-- ----------------------------------------------------
  main = do
    hSetBuffering stdin LineBuffering
    ex <- readArff   --ex is example
    let exs = read ex::[[Float]]
    let n = length (head (read ex::[[Float]]))
    if (n==2)   then solution_2 exs
       else if (n==3)    then solution_3 exs
          else if (n==4) then solution_4 exs
             else if (n==5) then solution_5 exs
                else if (n==6) then solution_6 exs
  write x = do
    hdl <- openFile "matrix.txt" WriteMode
    hPutStr hdl x
    hClose hdl

  -- Funtion for Solutions
  -- Input/Output function

  multi :: [Float]->[Float]->[Float]
  my_length :: [[a]]->Float
  sumsq :: [Float]->[Float]

  my_length [] = 0
  my_length (x:xs) = 1 + my_length xs

  sumsq [] = []
  sumsq (x:xs) = [x*x] ++ sumsq xs

  multi [][]= []
  multi (x:xs)(y:ys) = [x*y] ++ multi (xs)(ys)

  -- s2_Solution for Exponential Regression
  --
  solution_2 ex = do
    let aa = 2
    let ab = s2_sum_x ex
    let ba = s2_sum_x ex
    let bb = s2_sum_x2 ex
    let ya = s2_sum_y ex
    let yb = s2_sum_xy ex

    let line_1 = "2\n"
    let line_2 = show aa ++ "," ++ show ab ++ "\n"
    let line_3 = show ba++","++ show bb ++"\n"
    let line_4 = "\n"
    let line_5 = show ya ++","++ show yb
    write (line_1++line_2++line_3++line_4++line_5)

  s2_sum_x :: [[Float]] -> Float
  s2_sum_y :: [[Float]] -> Float
  s2_sum_x2 :: [[Float]] -> Float
  s2_sum_multi :: [[Float]] -> Float
  s2_form_x :: [[Float]]->[Float]
  s2_form_y :: [[Float]]->[Float]

  s2_form_x [] = []
  s2_form_x [[a,b]] = [a]
  s2_form_x (x:xs) = s2_form_x [x] ++ s2_form_x xs

  s2_form_y [] = []
  s2_form_y [[a,b]] = [b]
  s2_form_y (y:ys) = s2_form_y [y] ++ s2_form_y ys
```

```
s2_sum_x [] = 0
s2_sum_x xs = sum(s2_form_x xs)

s2_sum_y [] = 0
s2_sum_y ys = sum(s2_form_y ys)

s2_sum_x2 [] = 0
s2_sum_x2 xs = sum(sumsq(s2_form_x xs))

s2_sum_xy []= 0
s2_sum_xy  s = sum(multi (s2_form_x s)(s2_form_y s))

s2_sum_multi [] = 0
s2_sum_multi s = sum(multi (s2_form_x s)(s2_form_y s))

-- s5_Solution for Exponential Regression
--
solution_6 ex = do
   let aa = 6
   let ab = s6_sum_x1i ex
   let ac = s6_sum_x2i ex
   let ad = s6_sum_x3i ex
   let ae = s6_sum_x4i ex
   let af = s6_sum_x5i ex

   let ba = s6_sum_x1i ex
   let bb = s6_sum_x1i2 ex
   let bc = s6_sum_x1x2 ex
   let bd = s6_sum_x1x3 ex
   let be = s6_sum_x1x4 ex
   let bf = s6_sum_x1x5 ex

   let ca = s6_sum_x2i ex
   let cb = s6_sum_x1x2 ex
   let cc = s6_sum_x2i2 ex
   let cd = s6_sum_x2x3 ex
   let ce = s6_sum_x2x4 ex
   let cf = s6_sum_x2x5 ex

   let da = s6_sum_x3i ex
   let db = s6_sum_x1x3 ex
   let dc = s6_sum_x2x3 ex
   let dd = s6_sum_x3i2 ex
   let de = s6_sum_x3x4 ex
   let df = s6_sum_x3x5 ex

   let ea = s6_sum_x4i ex
   let eb = s6_sum_x1x4 ex
   let ec = s6_sum_x2x4 ex
   let ed = s6_sum_x3x4 ex
   let ee = s6_sum_x4i2 ex
   let ef = s6_sum_x4x5 ex

   let fa = s6_sum_x5i ex
   let fb = s6_sum_x1x5 ex
   let fc = s6_sum_x2x5 ex
   let fd = s6_sum_x3x5 ex
   let fe = s6_sum_x4x5 ex
   let ff = s6_sum_x5i2 ex

   let ya = s6_sum_yi ex
   let yb = s6_sum_x1iyi ex
   let yc = s6_sum_x2iyi ex
   let yd = s6_sum_x3iyi ex
   let ye = s6_sum_x4iyi ex
   let yf = s6_sum_x5iyi ex

   let line_1 = "3\n"
   let line_2 = show aa++","++ show ab ++ ","++ show ac ++
            show ad ++ "," ++ show ae ++ "," ++ show af ++"\n"
   let line_3 = show ba++","++ show bb ++ ","++ show bc ++
            show bd ++ "," ++ show be ++ "," ++ show bf ++"\n"
   let line_4 = show ca++","++ show cb ++ ","++ show cc ++
            show cd ++ "," ++ show ce ++ "," ++ show cf ++"\n"
   let line_5 = show da++","++ show db ++ ","++ show dc ++
            show dd ++ "," ++ show de ++ "," ++ show df ++"\n"
   let line_6 = show ea++","++ show eb ++ ","++ show ec ++
            show ed ++ "," ++ show ee ++ "," ++ show ef ++"\n"
   let line_7 = show fa++","++ show fb ++ ","++ show fc ++
            show fd ++ "," ++ show fe ++ "," ++ show ff ++"\n"
   let line_8 = "\n"
   let line_9 = show ya++","++ show yb ++ ","++ show yc ++
            ","++ show yd ++ ","++ show ye ++ "," ++ show yf
   write (line_1++line_2++line_3++line_4++line_5++line_6++
            line_7++line_8++line_9)

s6_form_x1i [] = []
s6_form_x1i [[a,b,c,d,e,f]] = [a]
s6_form_x1i (x:xs) = s6_form_x1i [x] ++ s6_form_x1i xs

s6_form_x2i [] = []
s6_form_x2i [[a,b,c,d,e,f]] = [b]
s6_form_x2i (x:xs) = s6_form_x2i [x] ++ s6_form_x2i xs

s6_form_x3i [] = []
s6_form_x3i [[a,b,c,d,e,f]] = [c]
s6_form_x3i (x:xs) = s6_form_x3i [x] ++ s6_form_x3i xs

s6_form_x4i [] = []
s6_form_x4i [[a,b,c,d,e,f]] = [d]
s6_form_x4i (x:xs) = s6_form_x4i [x] ++ s6_form_x4i xs

s6_form_x5i [] = []
s6_form_x5i [[a,b,c,d,e,f]] = [e]
s6_form_x5i (x:xs) = s6_form_x5i [x] ++ s6_form_x5i xs

s6_form_yi [] = []
s6_form_yi [[a,b,c,d,e,f]] = [f]
s6_form_yi (x:xs) = s6_form_yi [x] ++ s6_form_yi xs

s6_sum_x1i [] = 0
s6_sum_x1i s = sum(s6_form_x1i s)

s6_sum_x2i [] = 0
s6_sum_x2i s = sum(s6_form_x2i s)

s6_sum_x3i [] = 0
s6_sum_x3i s = sum(s6_form_x3i s)

s6_sum_x4i [] = 0
s6_sum_x4i s = sum(s6_form_x4i s)

s6_sum_x5i [] = 0
s6_sum_x5i s = sum(s6_form_x5i s)

s6_sum_yi [] = 0
s6_sum_yi s = sum(s6_form_yi s)

s6_sum_x1i2 [] = 0
s6_sum_x1i2 s = sum(sumsq(s6_form_x1i s))

s6_sum_x2i2 [] = 0
s6_sum_x2i2 s = sum(sumsq(s6_form_x2i s))
```

```
s6_sum_x3i2 [] = 0
s6_sum_x3i2 s = sum(sumsq(s6_form_x3i s))

s6_sum_x4i2 [] = 0
s6_sum_x4i2 s = sum(sumsq(s6_form_x4i s))

s6_sum_x5i2 [] = 0
s6_sum_x5i2 s = sum(sumsq(s6_form_x5i s))

s6_sum_x1iyi []= 0
s6_sum_x1iyi s = sum(multi (s6_form_x1i s)(s6_form_yi s))

s6_sum_x2iyi []= 0
s6_sum_x2iyi s = sum(multi (s6_form_x2i s)(s6_form_yi s))

s6_sum_x3iyi []= 0
s6_sum_x3iyi s = sum(multi (s6_form_x3i s)(s6_form_yi s))

s6_sum_x4iyi []= 0
s6_sum_x4iyi s = sum(multi (s6_form_x4i s)(s6_form_yi s))

s6_sum_x5iyi []= 0
s6_sum_x5iyi s = sum(multi (s6_form_x5i s)(s6_form_yi s))

s6_sum_x1x2 [] = 0
s6_sum_x1x2 s = sum(multi (s6_form_x1i s)(s6_form_x2i s))

s6_sum_x1x3 [] = 0
s6_sum_x1x3 s = sum(multi (s6_form_x1i s)(s6_form_x3i s))

s6_sum_x1x4 [] = 0
s6_sum_x1x4 s = sum(multi (s6_form_x1i s)(s6_form_x4i s))

s6_sum_x1x5 [] = 0
s6_sum_x1x5 s = sum(multi (s6_form_x1i s)(s6_form_x5i s))

s6_sum_x2x3 [] = 0
s6_sum_x2x3 s = sum(multi (s6_form_x2i s)(s6_form_x3i s))

s6_sum_x2x4 [] = 0
s6_sum_x2x4 s = sum(multi (s6_form_x2i s)(s6_form_x4i s))

s6_sum_x2x5 [] = 0
s6_sum_x2x5 s = sum(multi (s6_form_x2i s)(s6_form_x5i s))

s6_sum_x3x4 [] = 0
s6_sum_x3x4 s = sum(multi (s6_form_x3i s)(s6_form_x4i s))

s6_sum_x3x5 [] = 0
s6_sum_x3x5 s = sum(multi (s6_form_x3i s)(s6_form_x5i s))

s6_sum_x4x5 [] = 0
s6_sum_x4x5 s = sum(multi (s6_form_x4i s)(s6_form_x5i s))
--
-- ----------------------------------------------------------------
```

ACKNOWLEDGMENT

This work was fully supported by research fund of Suranaree University of Technology granted to the Data Engineering and Knowledge Discovery (DEKD) Research Unit, in which Kittisak Kerdprasop is a director and Nittaya Kerdprasop is a member and researcher. The authors are also partly supported. by National Research Council of Thailand (NRCT) and the Thailand Research Fund (TRF). The first author would like to thank all the research assistants who participated in the SUT-Miner project, especially Prapanpong Nopsuwan who coded linear regression in Haskell and Java.

REFERENCES

[1] E. Awad and H. Ghaziri, *Knowledge Management*, Pearson Prentice Hall, 2004.
[2] R. Bird, *Introduction to Functional Programming using Haskell*, Prentice Hall, 1998.
[3] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery: An Overview," in *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1996.
[4] P. Hudak, J. Fasel, and J. Peterson, "A gentle introduction to Haskell," Yale University, *Technical Report Yale U/DCS/RR-901*, 1996.
[5] K. Kerdprasop and N. Kerdprasop, "Multi-agents in data filtering systems," in *Proc. 7th Int. Conf. on Software Engineering and Applications*, 2003, pp.471-475.
[6] N. Kerdprasop and K. Kerdprasop, "Enhancing the power of OLAP with knowledge discovery," in *Proc. 7th Int. Conf. on Software Engineering and Applications*, 2003, pp.43-47.
[7] M. Raisinghani (ed.), *Business Intelligence in the Digital Economy*, Idea Group Publishing, 2004.
[8] K. Rennolls, "An intelligent framework (O-SS-E) for data mining, knowledge discovery and business intelligence," in *Proc. 16th Int. Workshop on Database and Expert System Applications*, 2005, pp.715-719.
[9] R. Roiger and M. Geatz, *Data Mining: A Tutorial-Based Primer*, Addison Wesley, 2003.
[10] WEKA, available at http://www.cs.waikato.ac.nz/ml/weka
[11] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques* (2nd ed.), Morgan Kaufmann, 2005.

Nittaya Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. She received her B.S. from Mahidol University, Thailand, in 1985, M.S. in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, USA, in 1999. She is a member of ACM and IEEE Computer Society. Her research of interest includes Knowledge Discovery in Databases, AI, Logic Programming, Deductive and Active Databases.

Kittisak Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. He received his bachelor degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in computer science from Nova Southeastern University, USA., in 1999. His current research includes Data mining, Artificial Intelligence, Functional Programming, Computational Statistics.