2012 IEEE 27-th Convention of Electrical and Electronics Engineers in Israel

Proactive Data Mining Using Decision Trees

Haim Dahan and Oded Maimon
Dept. of Industrial Engineering
Tel-Aviv University
Tel Aviv, Israel

Shahar Cohen
Dept. of Industrial Engineering & Management
Shenkar College of Engineering and Design
Ramat Gan, Israel

Lior Rokach
Dept. of Information Systems Engineering
Ben-Gurion University of the Negev
Beer Sheva, Israel

***Abstract*— Most of the existing data mining algorithms are 'passive'. That is, they produce models which can describe patterns, but leave the decision on how to react to these patterns in the hands of the user. In contrast, in this work we describe a proactive approach to data mining, and describe an implementation of that approach, using decision trees. We show that the proactive role requires the algorithms to consider additional domain knowledge, which is exogenous to the training set. We also suggest a novel splitting criterion, termed *maximal-utility*, which is driven by the proactive agenda.**

***Index Terms*— Knowledge Discovery from Databases, Active Data Mining, Classification.**

I. Introduction

Data mining, the science of algorithmic analysis of data and pattern extraction, is common in support of organizational decision-making [2], [3], [5], [9], [10], [13]. In most of the cases, the patterns that are extracted from the input datasets are subjected to human evaluation, in order to decide on how to use them [2], [3], [5], [6]. Although widely applicable, these works leave an overly complex task in the hands of humans - they do not produce any explicit actions or suggestions [2], [3], [5], [6].

For example, the average customer churn rate, experienced by wireless operators is known to be around 2% per month [9]. Traditional data mining algorithms can receive churning data, extract churning patterns and provide the subsets of customers that are most likely to churn. These algorithms provide helpful answer to the question: “Who are the most likely customers to leave the operator?” However, in many cases the questions that are mostly important for the operator are actually more like: “How can we intervene, to reduce the churn rate of preferable customers?” “What are the potential benefits and costs of our means of intervention?” “Which subset of customers is most potentially beneficial for us to concentrate on?”

The data mining literature does includes a branch of active methods [7], [12]. However, this branch mainly focuses on post processing procedures. That is, the intervention with the input data is based on a given model, which is typically produced by a passive (non-proactive) algorithm. In these works, the passive models are blind and indifferent to the proactive agenda - embodied by the post-processing phase.

Another shortcoming of most of the proposed data mining algorithms is in the lack of consideration of specific domain knowledge. It is hard to believe that the same algorithm can do well on two different domains, without any modifications. Moreover, the consideration of domain knowledge is mandatory when pursuing the proactive agenda, since the training set by itself indicates nothing on what kinds of actions are possible (or practical).

Despite the relative maturity of the research on data mining, there is no data mining method that inherently extracts specific means of actions, while considering domain or problem knowledge. In this work we introduce a novel proactive data-mining approach. We show that considering domain knowledge, which is exogenous to the training data, is mandatory in proactive data mining. We describe an implementation of our approach, using decision trees, and propose a novel splitting rule that is driven by the proactive agenda.

II. Proactive Data Mining

Let $A = \{A_1, A_2,\ldots,A_k\}$ be a set of explaining attributes that were drawn from some unknown probability distribution p_0, and $D(A_i)$ be the domain of attribute A_i. We denote by $D = D(A_1)\times D(A_2)\times\ldots\ \times D(A_k)$ the Cartesian product of $D(A_1)$, $D(A_2),\ldots, D(A_k)$ and refer to it as the input domain of the task. Similarly, let T be the target attribute, and $D(T) = \{c_1,c_2,\ldots c_{|D(T)|}\}$ the discrete domain of T. We refer to the values in $D(T)$ as the possible classes. It is assumed that T depends on D, usually with an addition of some random noise.

Classification algorithms receive training data, as input. Let $<X;Y> = < x_{1,n}, x_{2,n},\ldots,x_{k,n} ; y_n >$, for $n = 1,2,\ldots,N$ be a training set of N classified records, where $x_{i,n}\in D(A_i)$ is the value of the i-th explaining attribute in the n-th record, and $y_n\in D(T)$ is the class relation of that record. In an ordinary classification task, we search for a function $f : D \to D(T)$, so that given $x\in D$, a random realization of the explaining attributes, and $y\in D(T)$, the corresponding class relation, the probability of correct classification, $\Pr[f(x) = y]$, is maximized.

In ordinary classification tasks, the underlying assumption is that the target class, for a given record, cannot be changed. This assumption is often incorrect. For example, consider the business case of customer retention, which often triggers the extraction of churn-prediction models [9]. Churn-prediction models explain churning patterns. Clearly, by taking some means of action (for example, offering the customer a more attractive price plan), the company can affect the explaining attributes and in turn change the actual churning probability. Namely, in some scenarios, rather than asking "what will be the target result in that case?" the business user is actually interested in knowing "what should I do, in order to affect the value of the target result according to the company's interests?" [1], [8], [4]. In this work, we allow the values of the explaining attributes to be proactively changed by the user. Changing the values of the explaining attributes can subsequently affect the value of the target attribute in the direction of desired values.

The shift from classification to optimization requires us to consider additional knowledge about the business domain, which is exogenous to the actual training records. The additional knowledge is intended to cover various aspects of the business case, and the potential of changing the input values, such as: what is the objective function that needs to be optimized? What changes in the explaining attributes can and cannot be achieved? At what costs? etc. The exact form of that knowledge may differ from one task to another. In this work we consider a certain form of additional knowledge, which consists of an attributes-change cost and a benefit functions.

The attributes-change cost function C: $D \times D \to R$ assigns a real-valued cost for each possible change in the values of the explaining attributes. If a particular change is not allowed, the associated cost is infinite. The benefit function $B : D \times D(T) \to R$ assigns a real-valued benefit for each possible observation (potentially customer). This benefit depends both on the explaining attributes and the class relation.

The objective of the task that is implied is finding the optimal change (move) in the values of the explaining attributes: $O : D \to D$, which maximizes the expected value of some utility function. The utility function, we consider in this work, consists of the contribution to the benefit due to the move, minus the associated attribute change cost. The cost is provided directly from C, but the contribution to the benefit requires us to know the effect of the change on the target attribute. For this reason, we need to follow a two-phase procedure. In the first phase, we train some classifier, and in the second phase we utilize that classifier to find the optimal change.

The main limitation of the ordinary classification task, which we tackle, is the unfixed nature of the explaining attributes. Nonetheless, the explaining attributes are not typical decision-variables, which can be changed and set as desired. When one changes the probability distribution of the input variables, can she or he be sure that the classification model still correctly describes the functional dependency between the (changed) explaining attributes and the target? The answer to this question is not answered in this work. Instead, we refer to the optimal change O as a recommendation on seemingly attractive means of actions, and not as an automatic mechanism that set values to decision variables. Another question that rises from the above task statement is whether the ability to change the value of the explaining attributes is deterministic or not. The empirical probability for succeeding in the change can be added as part of the exogenous additional knowledge, but we do not consider it in this research.

III. A Proactive Approach Using Decision Trees

Let $DT = (V,E)$ be a decision tree with the set of vertices $V=\{v_0,v_1,\ldots,v_{|V|}\}$, were $|V|$ is the finite cardinality of V, and the set of edges (arcs) E, where each $e \in E$ is an ordered pair of vertices: $e=<v_i,v_j>$ indicating that v_j is a direct son of v_i. We denote the decision-tree's root by v_0. Let us consider decision trees that were trained based on the training set: $<X;Y>$. We define $|v_i(<X;Y>)|$ as the size of the vertex v_i, that is, the number of records in $<X;Y>$ that reach the vertex v_i, when sorted by DT in a top-down manner. Let us further define $p_0(c_j,v_i)$ as the estimated proportion of cases in v_i that belong to class c_j. We calculate $p_0(c_j,v_i)$ according to Laplace's law of succession:

$$p_0(c_j,v_i) = [m(c_j,v_i)+1]/[v_i(<X;Y>)+2], \quad (1)$$

where $m(c_j,v_i)$ is the number of records in $<X;Y>$ that reach the vertex v_i and relate to class c_j. We refer to nodes with no direct sons as leaves (or terminals), denote the set of leaf-nodes by L and define a branch of the tree (denoted by β) as a sequence of nodes $v(0)$, $v(1)$, $v(2)$, ..., $v(|\beta|)$, where $|\beta|$ is the length (number of nodes) of the branch, so that:

(i) $v(0) = v_0$ (i.e., $v(0)$ is the decision-tree's root),
(ii) for all $i = 0,1,\ldots,|\beta|$-1, $v(i+1)$ is a direct son of $v(i)$, and
(iii) $v(|\beta|) \in L$.

We define the total benefit of a branch, as the sum of benefits of all the observations that if sorted down the tree, reach the branch's terminal, and denote the total benefit of the branch β by $TB(\beta)$. We denote the total benefit of the entire tree DT:

$$TB(DT) = \sum_{\beta \in DT} TB(\beta). \quad (2)$$

We can use any given decision tree to examine the expected consequences of proactively act on the records of a certain branch, in order to change the values of their explaining attributes. Such a change actually "moves" records from one branch to another. Let β_1 and β_2 be the source and destination branches, respectively. The estimated merit of moving from β_1 to β_2, may be defined as the difference in the total benefits of β_2 to β_1, minus the cost that outcomes from the change in values that is required in order to move from β_1 to β_2:

$$\text{merit}(\beta_1,\beta_2) = TB(\beta_2) \cdot \frac{|v(|\beta_1|)(\langle X;Y\rangle)|}{|v(|\beta_2|)(\langle X;Y\rangle)|} - TB(\beta_1) - C(D_1,D_2) \text{ , } (3)$$

where D_1 and D_2 are the sub-domains of D that correspond to the terminals of β_1 and β_2, respectively. The term $|v(|\beta_1|)(<X;Y>)|/|v(|\beta_2|)(<X;Y>)|$ normalizes the total benefit of β_2 to the number of records that are currently in β_1. We refer to moves from one branch to another as *single-branch moves*.

When assessing the attractiveness of a branch, we must consider the number of records in it. A branch that has a small number of records is inherently less certain than a branch with large number of records. More specifically, even if moving from β_1 to β_2 is associated with positive merit, knowing that there are only few records in β_2 we might want to avoid the move, since we are unconfident regarding that merit. We take the number of records in a branch into consideration by adding a weight to each possible single-branch move. We denote the weight associated to the move from β_1 to β_2 as $w(\beta_1,\beta_2)$, define the utility of a single-branch move, as follows:

$$\text{utility}(\beta_1,\beta_2) = \left[TB(\beta_2)\cdot\frac{|v(|\beta_1|)(\langle X;Y\rangle)|}{|v(|\beta_2|)(\langle X;Y\rangle)|} - TB(\beta_1)\right]\cdot w(\beta_1,\beta_2) - C(D_1,D_2) \quad , \qquad (4)$$

and consider the move as advantageous, if utility(β_1,β_2) exceeds some pre-defined threshold.

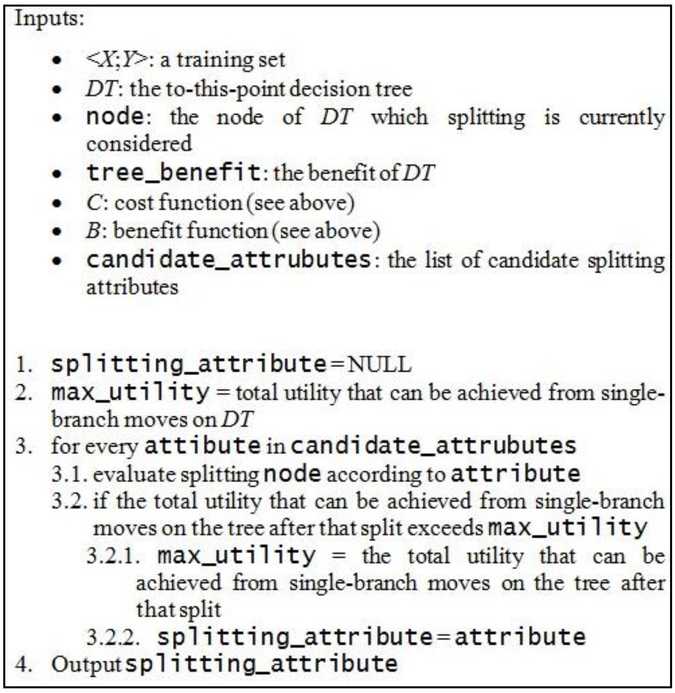
In this work, we have used the lower bound of a 1-α=95% confidence interval, on the probability of the majority class as the weight. That is, denoting the majority class by c^*, we use the following weight:

$$w(\beta_1,\beta_2) = p_0\left(c^*, v(|\beta_2|)\right) - Z_{1-\frac{\alpha}{2}}\cdot\sqrt{\frac{p_0(c^*,v(|\beta_2|))\left(1-p_0(c^*,v(|\beta_2|))\right)}{|v(|\beta_2|)(\langle X;Y\rangle)|}} \quad , \qquad (5)$$

We use these weights as a heuristic, noticing that fewer are the observations in β_2, the smaller is the lower bound of the confidence interval, and the more significant is the suppression of the difference in the total benefits. We do not allow negative weights, and in practice use the maximum between $w(\beta_1,\beta_2)$ and zero as the weight.

Having the utility function being defined for single-branch moves, we propose a simple algorithm for systematically scanning the branches of any given decision tree, and extracting a list of all the advantageous single-branch moves (i.e., all the single-branch moves, which provide a utility that exceeds a pre-defined threshold). It can be seen that as long as the attribute-change cost and benefit functions maintain the Triangular Equality, the selected single-branch moves can be used in any order (and still result in the same total gained utility). From our experience with real-life examples, it is often the case that the Triangular Equality is indeed maintained.

Although a decision tree is the main required input to the scanning algorithm, using a decision-tree that was trained while pursuing classification accuracy might miss the maximal-utility objective. For example, certain attributes might highly contribute to the classification accuracy, but be impossible for change, whereas different attributes might be slightly inferior in terms of accuracy, but be changed easily, with a significant potential utility. In order to tackle the motivation for maximal-utility, we define a novel splitting criterion for decision-tree algorithms. The novel criterion is termed *maximal-utility criterion*, and means: "split according to the values of the explaining attribute that maximizes the potential total utility that can be gained from the tree". This splitting criterion is described in Fig. 1.



```
Inputs:
  • <X;Y>: a training set
  • DT: the to-this-point decision tree
  • node: the node of DT which splitting is currently considered
  • tree_benefit: the benefit of DT
  • C: cost function (see above)
  • B: benefit function (see above)
  • candidate_attrubutes: the list of candidate splitting attributes

1. splitting_attribute=NULL
2. max_utility = total utility that can be achieved from single-branch moves on DT
3. for every attibute in candidate_attrubutes
   3.1. evaluate splitting node according to attribute
   3.2. if the total utility that can be achieved from single-branch moves on the tree after that split exceeds max_utility
        3.2.1. max_utility = the total utility that can be achieved from single-branch moves on the tree after that split
        3.2.2. splitting_attribute=attribute
4. Outputsplitting_attribute
```

Fig. 1. The maximal-utility splitting criterion: split a node according to the values of the explaining attribute, which maximizes the potential for benefit enhancement.

In the following section we provide an illustrative example and compare the utility enhancements from two different decision trees: a traditional, passive, decision tree and a proactive decision tree which is built using the maximal-utility criterion.

IV. Illustrative Example

In this section we demonstrate the properties and potential usage of the proposed approach and its implementation with two different decision trees: passive and proactive. We use a toy dataset of 160 historical observations: 68 churners and 92 non-churning customers, of a wireless operator. This toy example tries to be like a real case study.

The customers are described by three explaining attributes: A_1 – describing the customer's package, which can take the values: 'Data', 'Voice' and 'Data&Voice'; A_2 – describing the customer's sex, which can be either 'Female' or 'Male' and A_3 – describing the customer's monthly rate in US dollars, with the following possible values: 75, 80, 85, 90 and 95. Table 1 describes the empirical joint distribution of the explaining and the Target attributes.

We begin our illustration by generating a decision-tree for predicting the target attribute. We used the well-known J48 implementation of Weka [11]. The output decision tree is described in Fig. 2. J48 builds the tree without considering the benefit of having a customer staying or leaving, and the costs of potential changes in the values of the explaining attributes. Notice that the most explaining attribute in this tree is the

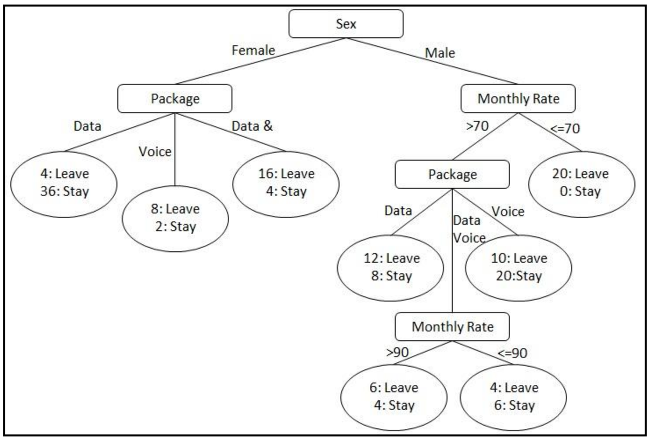


Fig. 2. The J48 decision tree that was produced to predict the churning probability in our illustrative example.

customer's sex, which clearly cannot be changed by the company.

The tree in Fig. 2 was built while completely ignoring the objective of maximum utility. In order to create a meaningful utility, we specify the following costs for changes in the values of the explaining attributes. First, we have specified infinite costs for changes in the customer's . Moreover, we have assumed that due to regulations, the customer's monthly rate can only be reduced (having the magnitude of the reduction stand for the cost of the reduction) and not increased. Finally, the costs we've assumed for changes in the Package attribute are described by the following matrix.

	Data	Voice	Data & Voice
Data	0	5	10
Voice	0	0	5
Data & Voice	0	0	0

The benefit function we have considered assigns the value of a Monthly-Rate for a staying customer and minus that value for a leaving customer. We have then produced a second decision tree, which uses the maximal-utility splitting criterion along with the cost and benefit functions, described above. The output decision tree is described in Fig. 3. Notice that since the customer's sex cannot be changed by the company, the customer's sex attribute was not selected at the root of the tree.

TABLE 1. THE WIRELESS OPERATOR'S TOY DATASET

		Month.	Did	# of
Package	Sex	Rate	Churn?	Observ.
Data	Female	70	Stay	18
Data	Female	70	Leave	2
Data	Male	70	Leave	20
Data	Female	75	Stay	18
Data	Female	75	Leave	2
Data	Male	75	Stay	8
Data	Male	75	Leave	12
Voice	Female	80	Leave	4
Voice	Male	80	Leave	6
Voice	Male	80	Stay	14
Voice	Female	85	Stay	2
Voice	Female	85	Leave	4
Voice	Male	85	Stay	6
Voice	Male	85	Leave	4
Data & Voice	Female	90	Stay	10
Data & Voice	Female	95	Stay	6
Data & Voice	Female	95	Leave	4
Data & Voice	Male	90	Stay	6
Data & Voice	Male	90	Leave	4
Data & Voice	Male	95	Stay	4
Data & Voice	Male	95	Leave	6

In a post processing procedure we searched for the single-branch moves that contribute a positive utility gain. We first scanned the tree in Fig. 2 (which was built by J48). Notice that although the cost and benefit matrices were not considered during the J48 induction of the decision tree (in Fig. 3), during the post processing procedure these cost and benefit were considered. The respective beneficial single-branch moves are described in Table 1. Notice for instance that the most valuable move strive to shift customers from a node with churn rate of 80% of a node with churn rate of merely 10%. The initial overall benefit of the tree in Fig. 2 is: 2020. After implementing all the valuable moves of Table 1, we result with a tree with overall pessimistic benefit of: 3000. We then scanned the maximal-utility generated tree in Fig. 3. The respective beneficial single-branch moves are described in Table 2. Although the tree in

TABLE 2. POTENTIAL ACTION WITH RESPECT TO THE TREE IN FIG. 2.

From Branch(β_1)	To Branch (β_2)	Utility
Sex = 'Female' and Package = 'Voice'	Sex = 'Female' and Package = Data	835.23
Sex = 'Male' and Monthly-Rate > 90 and Package = 'Data & Voice'	Sex = 'Male' and Monthly-Rate > 70 and Package = 'Voice'	90.35
Sex = 'Female' and Package = 'Data & Voice'	Sex = 'Female' and Package = 'Data'	54.64

Fig. 3 has an initial overall benefit of 2020 as well, after implementing all the beneficial single-branch moves, the overall pessimistic benefit raises to 5435.

V. CONCLUSION

This paper proposed a novel, proactive approach to data-mining. This approach involves intervention in the distribution of the input data, with the aim of maximizing an economic utility measure. This intervention requires the consideration of domain-knowledge, which is exogenous to the typical classification task. The paper is focused on decision trees, and based on the idea of moving observations from one branch of the tree to another. We propose a novel splitting criterion for decision trees, termed maximal-utility, which maximizes the potential for profitability enhancement in the output tree.

Our work demonstrates that by taking the proactive approach, it becomes possible to solve business problems that cannot be approach through traditional, passive data-mining methods. We also show that our proposed splitting criterion may outperform passive splitting rules in terms of the potential for utility enhancement.

TABLE 3. POTENTIAL ACTION WITH RESPECT TO THE TREE IN FIG. 3.

From Branch (β_1)	To Branch (β_2)	Utility
Package = 'Data' and Sex = 'Male'	Package = 'Data & Voice' and 'Monthly-Rate <= 90	1873.70
Package = 'Voice' and Sex = 'Female' and Monthly-Rate > 70	Package = 'Data' and Sex = 'Female'	835.23
Package = 'Data & Voice' and 'Monthly-Rate > 90 and Sex = 'Male'	Package = 'Data & Voice' and 'Monthly-Rate <= 90	380.01
Package = 'Data & Voice' and 'Monthly-Rate > 90 and Sex = 'Female'	Package = 'Data' and Sex = 'Female'	304.43
Package = 'Voice' and Sex = 'Male' and Monthly-Rate > 80	Package = 'Voice' and Sex = 'Male' and Monthly-Rate <= 80	21.65

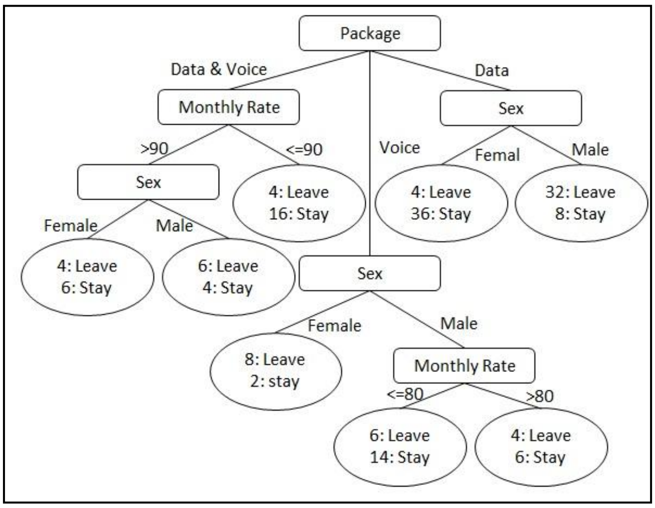


Fig. 3. Proactive classification (with reasonable benefit and cost matrices) generated decision tree over dataset presented in Table 1.

REFERENCES

[1] J. Boulicaut and B. Jeudy, "Constraint-based data mining", The Data Mining and Knowledge Discovery Handbook, pp. 399-416 Springer, 2005.

[2] L. Cao, "Actionable knowledge discovery and delivery", WIREs: Data Mining and Knowledge Discovery, vol 2, pp. 149-163, 2012.

[3] L. Cao, "Domain driven data mining, challenges and prospects", IEEE Trans on Knowledge and Data Engineering, 1100(17), pp. 3067-30105, 2010.

[4] L. Cao, P.S. Yu, C. Zhang and Y. Zhao, "Domain driven data mining", Springer, 2010.

[5] L. Cao and C. Zhang, "Evolution of KDD: towards domain-driven data mining", International Journal of Pattern Recognition and Artificial Intelligence, 21(4), pp. 677-692, 2007.

[6] L. Cao, "Domain driven actionable knowledge discovery in real world", PAKDD2006, pp. 1021-1030, 2006.

[7] H. Dahan and O. Maimon, "Active control for data mining models", 2003.

[8] R. Hema and N. Malik, "Data mining and business intelligence", Proceedings of the 4th National Conference; INDIACom-2010.

[9] S. Y. Hung, D.C. Yen and H.Y. Wang, "Applying data mining to telecom churn management", Expert Systems with Applications 31, pp. 515-524, 2006.

[10] L. Rokach, L. Nahamani, and A. Shmilovici, "Pessimistic cost-sensitive active learning of decision trees for profit maximizing targeting campaigns", Data Mining Knowledge Discovery, 17, pp. 283-316, 2008.

[11] Waikato Environment for Knowledge Analysis, The University of Waikato, Hamilton, New Zealand, 2011.

[12] Q. Yang, Jie Yin, C. X. Ling and T. Chen, "Postprocessing of decision to extract actionable knowledge", In Proc. Of ICDM'03, 2003.

[13] Z. He, X. Xu and S. Deng, "Data mining for actionable knowledge: A Survey", 2005.