Inteligencia Artificial 22(64), (2019), 63-84
doi: 10.4114/intartif.vol22iss64pp63-84

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

An Efficient Probability Estimation Decision Tree Postprocessing Method for Mining Optimal Profitable Knowledge for Enterprises with Multi-Class Customers

Janapati Naga Muneiah[1,2,A] and Ch D V Subba Rao[3,B]

[1] Research Scholar, Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Kakinada, Andhra Pradesh, India.
[2] Associate Professor, Department of Computer Science and Engineering, Chadalawada Ramanamma Engineering College, Tirupati, Andhra Pradesh, India
[3] Professor, Department of Computer Science and Engineering, Sri Venkateswara University College of Engineering, Tirupati, Andhra Pradesh, India.
[A] nagamuni513@gmail.com, [B]chdvsrao@svuniversity.ac.in

Abstract Enterprises often classify their customers based on the degree of profitability in decreasing order like C_1, C_2, ..., C_n. Generally, customers representing class C_n are zero profitable since they migrate to the competitor. They are called as attritors (or churners) and are the prime reason for the huge losses of the enterprises. Nevertheless, customers of other intermediary classes are reluctant and offer an insignificant amount of profits in different degrees and lead to uncertainty. Various data mining models like decision trees, etc., which are built using the customers' profiles, are limited to classifying the customers as attritors or non-attritors only and not providing profitable actionable knowledge. In this paper, we present an efficient algorithm for the automatic extraction of profit-maximizing knowledge for business applications with multi-class customers by postprocessing the probability estimation decision tree (PET). When the PET predicts a customer as belonging to any of the lesser profitable classes, then, our algorithm suggests the cost-sensitive actions to change her/him to a maximum possible higher profitable status. In the proposed novel approach, the PET is represented in the compressed form as a Bit patterns matrix and the postprocessing task is performed on the bit patterns by applying the bitwise AND operations. The computational performance of the proposed method is strong due to the employment of effective data structures. Substantial experiments conducted on UCI datasets, real Mobile phone service data and other benchmark datasets demonstrate that the proposed method remarkably outperforms the state-of-the-art methods.

Keywords Data mining, Knowledge Engineering and Applications, Machine Learning: Methods and Applications, actionable knowledge discovery, profit maximization.

1 Introduction

Most of the service providing sectors such as Mobile Phone service, Internet, Banking, Insurance, IT Services, Retail, etc. are encountering the crisis due to certain classes of their customers. The prime reason for massive losses taking place in these sectors is due to the attritors, a class of customers who close their account and shift to the competitor over a period of time [1] and this phenomenon is called attrition or churning. Apart from attritors and non-attritors still, there are other classes of customers who stay with the service provider but inactive in using the services and lead to very insignificant profits and uncertainty. Most often, enterprises classify their customers based on the amount of profit gained from them over a certain period of time and perceive them in a certain order of priority based on the degree of profitability. Among these multiple classes of customers in the decreasing order

ISSN: 1137-3601 (print), 1988-3064 (on-line)

of profitability those who stand at the first tier yields highest profit and the last tier yields very less or zero profit. Most of the times last tier customers are attritors. There are many reasons for attrition or lesser profitability nature of the customers and some of them are huge industry deregulations, low service levels, high tariffs, not updating with the technology, etc. If necessary actions are taken, then, a customer who is predicted to be an attritor or of any less profitable class can be converted as a non-attritor and high profitable.

It is a known fact that retaining an existing customer is cheaper than finding a new customer [2]. However, enterprises consistently try hard to retain their customers by organizing customer retention campaigns and provide suitable offers to the likely attritors. In some businesses, they also take actions such as changing the current plan, reducing interest rates on loans etc. to convert other inactive classes of customers as active and more profitable. It is a difficult task to detect probable attritors, and less profitable classes of customers among a large number of customers through campaigns. Hence, a machine learning model for attrition or less profitability prediction is required.

Some of the researchers of the machine learning community have focused on the attrition problem and they treated the attrition prediction as a classification problem and limited only on constructing a classification model using the customers' base. Thereafter, they mainly concentrated on improving the technical interestingness measures like accuracy and AUC, etc. of the constructed model [3-8]. When an existing customer's record is given as input to the constructed model; it only classifies the customer as a probable attritor or non-attritor. This knowledge is not useful to the enterprise since it does not suggest any actions to change the customer from a less profitable class to a higher profitable one and no direct benefit is obtained. Hence, some manual work by the business expert has to be performed on the model to find the actions to change the customer from a less profitable class to a higher profitable one.

A customer's sample can be re-classified by changing the values of the required attributes. Here the idea of *changing a customer's class* means, changing her/him from a lesser or zero profitability category to a higher profitability category. Till now, very less study has been done on extracting profit maximizing knowledge automatically from the machine learning models. Though some of the past research has addressed this problem, in that they treated the problem as a 2-class problem and presented the methods to convert the customer from class C_2 (attritor) to class C_1 (non-attritor) only [9-13]. Moreover, there is no specific focus on computational performance while achieving the objective of profit maximization.

When the customers are of more than two classes then, enterprises perceive them in the decreasing order of profitability as C_1, C_2, C_3,..., C_n classes. Class C_1 customers are highest profitable and in most of the applications, class C_n customers are attritors. In the service oriented business sectors, in one set-up, customers are classified as platinum class (C_1), gold class (C_2), iron class (C_3), and lead class (C_4), where n=4, in the decreasing order of profitability [14]. In one context of the Retail industry, customers are categorized as true friends (high loyal and highest profitable), butterflies (low loyal and high profitable), barnacles (high loyal and low profitable), and strangers (low loyal and lowest profitable) [15]. In another circumstance, they are classified as stay customers, discount customers, impulsive customers, need based customers, and wandering customers or churn customers (zero profitable) [16]. In such kind of applications [17], if a customer is predicted as belonging to class C_m ($m \leq n$) then, it is necessary and beneficiary to the enterprise to re-classify her/him to another class C_k ($k<m$ and $k \geq 1$) by applying the required actions. An action is changing an attribute's value of a customer. For instance, in the wireless mobile phone sector, changing the *data plan* of a customer from one category to another is an action.

To address these limitations and challenges, we propose an efficient algorithm namely EDest_Leaf_Finder (EDLF). Our research considers the attrition avoidance and profit maximization problem as a multi-class classification problem and employs a decision tree specifically probability estimation decision tree (PET) which is built using the customers' profiles. With the help of the PET if a customer is predicted to be as an attritor or a low profitable class customer, then our method tries to convert her/him as non-attritor and higher profitable class customer by changing the values of the required attributes of the customer. However, efforts are made only to change the values of flexible attributes, whose values are possible to change (eg. Service Level, data plan, etc. in Telecom sector, discount rate, interest rate, etc. in other sectors), and not the values of the non-flexible attribute's whose values cannot be changed (eg. Gender, age, and Income of the customer, etc.).

In the process of achieving a profit maximizing solution, the constructed PET is represented in the compressed form as a Bit patterns matrix (BPM). Thereafter, PET is used only to find the class label and class probability estimation of a customer's sample. All the remaining postprocessing work is performed on the BPM only rather than on the actual form of the PET. EDLF follows a novel approach and employs bitwise AND operations on the elements of the 2-D array (BPM) for attaining the objective. Due to the effective organization of the bit patterns and efficient usage of the array data structures and bitwise AND operations, the computational achievement of the proposed method is strong. The proposed method is applicable to the business domains/applications with *n* number of classes of customers, where $n \geq 2$, and it is designed such that it does not leave any option and provides an optimal solution with a maximum possible net profit for a customer who is predicted as attritor/less profitable.

Even when it is not possible to re-classify the customer from class C_m to C_k ($k<m$), the proposed method tries to find a profit enhancing solution.

In summary, this paper focuses on providing a profit maximization solution for the service providing business sectors which classify their customers into multi-classes. When a customer is predicted as attritor/less profitable, the proposed method suggests customized actions to reclassify her/him such that the net profit obtained is the maximum. The solution is provided while achieving the remarkable computational performance by employing efficient data structures. We have illustrated the working of the proposed method using a synthetic dataset from the Banking sector. By conducting experiments on synthetic data, real-world data belonging to Mobile phone service, UCI and other benchmark datasets, the efficiency of the proposed method has been compared with a single tree based [9] and also ensemble tree based state-of-the-art methods [12,13]. These experiments demonstrate that our method achieves remarkable computational performance and outperforms the state-of-the-art methods with respect to runtimes and profits.

The rest of the paper is organized as follows: In section 2 we review the literature and discuss the related work. In section 3, we provide the preliminaries and discuss mining profitable knowledge from PET using our algorithm EDLF for 2-class, 3-class problems and finally, a mathematical model has been formulated to compute the net profit for multi-class applications. Performance evaluation of EDLF is presented in section 4 by comparing its computational times and profits with state-of-the-art methods. In Section 5, we have given the conclusions and discussed the possibilities for future work.

2 Literature Survey and Related work

Earlier, many researchers of machine learning and data mining have studied the attrition problem and handled it as a classification problem. They have built various data mining models for attrition prediction. However, till today there is more focus on building churn prediction models only rather than automatic extraction of the profitable actionable knowledge from the model. And most of this attrition prediction research has only targeted on improving the performance metrics like accuracy, AUC, and robustness, etc. of the constructed model [3-8]. Recently, Sivasankar and Vijaya have presented [18] a hybrid method for building a churn prediction model which is based on the combination of classification and clustering. They claimed that the predictive accuracy of their method is high. Bahnsen et al. [19] presented a new measure based on monetary constraints for evaluating the effectiveness of the attrition methodology. Their method makes use of several fixed offers which is also dependant on product based cost and likeliness of offer acceptance by the customer. As different classes of customers have a varied monetary effect, they projected within the customers that the charge assessed and specified by the new methodology will be different.

The method proposed by Eugen Stripling et al. [20] has taken the profit into account and hence included the profitability concept in the churn prediction model. They have introduced a genetic algorithm based classifier for churn prediction which also incorporated the notion of profit maximization. The study of Pednault et al. [21] has observed the sequential nature of some of the CRM problems and addressed the issue of sequential decision making using the Markov decision process. Yang et al. [9] presented a single decision tree based cost-sensitive profit-maximizing method viz. Leaf_Node_Search that suggests an optimal destination leaf for an instance which is predicted as belonging to an un-loyal class(attrition). Their algorithm follows a conventional tree traversal approach for searching the solution that leads to more computational time when the size of the prediction model is huge.

Instead of class probabilities, Nasrin Kalanat et al. introduced [10, 11] fuzzy based methods which makes use of fuzzy membership to mine the profitable actions from the data. Their methods build the fuzzy decision tree and postprocess the tree to mine the actions to improvise the fuzzy profit by taking fuzzy costs into account. Sebastiaan Höppner, et al. introduced a new classification approach [22] for mining the profitable knowledge for 2-class applications of attrition prevention. Their approach integrates the profit calculation measure within the decision tree construction phase itself.

To change one input sample from an undesired class to a desired one, Zhicheng Cui et al. proposed a cost-sensitive method [12] to extract actionable knowledge from additive tree models(ATM) such as Random forest. They have formulated the problem as an Integer Linear Programming (ILP) problem. Qiang LU et al. [13] also adopted ATM classifiers to address the problem of mining the actionable knowledge for achieving a maximum net profit from each individual. They have formulated the problem of extracting optimal actions from the ATM as a state space graph search problem and solved it by the state space search algorithm. Further, to reduce search time and achieve the runtime performance, they presented a sub-optimal search algorithm with an acceptable and rational heuristic function. However, ATM’s are complex and lacks the interpretability. When dataset size is huge, the model built using algorithms like Random forest can be enormously huge and deep. Consequently, search time for finding an optimal solution highly increases.

Gao and Yao [23] presented a method which follows a three phase model. Trisecting is the first phase where a universe of samples is partitioned into three disjoint sections. The second phase is the acting step where distinct action plans for the three sections are discussed. In the third phase, they discussed the method of changing the samples from undesired to the desired zone. To extract actionable knowledge, Cao et al. proposed four types of generic frameworks [24, 25] which incorporate domain knowledge to some extent. They studied on providing solutions for the applications of various domains using a framework which not only considers technical interestingness but also domain-specific expectations.

Though research on actionable knowledge discovery from data mining models is limited, some researchers have even shown their interest in surveying the existing methods and reviewed the issues with the present research [26]. Zhang et al. studied the rate of prediction, prediction capability and withholding capacity [27] and proposed a method for obtaining profit. Nasrin Kalanat and Eynollah Khanjari proposed a new cost-sensitive method [28] for mining actionable knowledge from graph data belonging to social networks where there can be relationships between the objects. To the best of our knowledge, all the existing research has treated the problem as a 2-class problem only. However, some of the studies [12, 13] followed a tricky method while dealing with multi-class datasets eventually treated as a binary classification problem.

3 Automatic extraction of profit maximizing knowledge from PETs

3.1 Modelling using a PET

Automatic extraction of actionable knowledge for profit maximization in the context of business problems is considered as a classification problem. PET has been used for modelling in our research. The motivation for using decision trees is, they are simple and easy to comprehend. They also work well for high dimensional data and also accuracy is in general high. The research in this paper can also be explained at ease with the help of the decision tree. In the business sectors, customers' profiles are described by a large number of different types of attributes like income, age, gender, education and nationality, service usage time, service plan, service level, rate of interest on loans, etc. To build the decision tree using the customers' profiles, C4.5 technique [29] has been used since it is one of the best off-the-shelf classifiers. C4.5 has become highly popular after ranked as #1 algorithm in the field of data mining [30]. Normally, CRM datasets tend to be very large. When the model is built on such datasets using the other effective decision tree induction methods like Random forest, the size of the model is high and moreover, those ensemble methods produce multiple trees. Consequently, the CPU time for achieving our objective significantly increases and also the process to attain the solution turn out to be more complex.

The proposed algorithm is designed to work on a single tree. Because of all these reasons, C4.5, a benchmark algorithm has been used in our research for model construction. C4.5 uses gain ratio [29] as the splitting criterion for attribute selection at a node during the tree construction. The ratio of Information gain and Split information with respect to an attribute *A* results in gain ratio as shown in Eq. (1). Information gain, as given in Eq. (2), is the amount of information that can be gained if attribute *A* is chosen as a splitting attribute at a node. If the entropy concerning an attribute is 0 then the Information gain can be maximum. Entropy is the expected information required to classify a sample in the dataset *D* as given in Eq. (3). Information gain method favours the attribute with maximum outcomes and selects it as a splitting attribute. Gain ratio measure overcomes this drawback by applying a type of normalization to Information gain using split information. The amount of Split information with respect to an attribute *A* denotes the potential information generated by splitting the training dataset *D* into *o* number of partitions, belonging to the *o* number of outcomes of a test on attribute *A*. Computation of Split Information with respect to an attribute *A* is given in Eq. (4).

$$\text{Gain Ratio (A)} = \frac{\text{Info Gain(A)}}{\text{Split Information(A)}} \quad \text{where} \tag{1}$$

$$\text{Information Gain (A)} = \text{Entropy(D)} - \sum_{j=1}^{o}\left(\frac{|D_j|}{|D|} \times \text{Entropy}(D_j)\right), \tag{2}$$

$$\text{Entropy(D)} = \left(-\sum_{i=1}^{|C|}(P_i * \log_2 P_i)\right) \text{ and} \tag{3}$$

$$\text{Split Information}_A(D) = -\sum_{j=1}^{o}\left(\frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)\right) \tag{4}$$

During the tree construction, the attribute computed as possessing the highest gain ratio is determined as the splitting attribute at a node of consideration. In the given equations, $|C|$, *o* and |D| are the number of classes, number of outcomes of an attribute *A* and total number of records in the training dataset respectively. Various steps in C4.5 algorithm for tree construction are described in the pseudocode in Algorithm 1.

The proposed algorithm though depends on the predicted class label for providing an optimal solution, it strictly makes use of the probability estimation of each class which can be offered from the decision trees. For the induction of probability estimation decision tree, the extension of C4.5 viz., C4.4 algorithm is adopted. The tree output by C4.4 estimates the probability of belonging to each class for an instance fallen into a particular leaf node. C4.4 is totally based on the frequency and adopts the maximum likelihood estimate for probability estimation [31]. Further, Laplace correction is also applied to overcome extreme probabilities scenarios. The probability that a customer's instance X which has fallen into a leaf node belongs to class C is given in Eq. (5).

$$P(C/X) = \frac{\sum_{j=1}^{|D|} \delta(R_j,\ C) + 1}{|D| + |C|} \tag{5}$$

If the jth customer's record, R_j, belongs to class C, then $\delta(R_j,C)=1$, or else 0. After the PET is constructed, its effectiveness is evaluated using the required metrics like accuracy, Area Under Receiver Operator Characteristic Curve (AUC), etc.

Algorithm 1. C4.5. Construct a decision tree using the instances of data partition of D

```
Input  : An attribute-valued training data partition D associated with class labels
Output : A decision tree

1   create a node N ;
2   if all the instances in D are of the same class, C then return N as a leaf node labeled with the class C;
3   if attribute_list is empty then return N as a leaf node labeled with the majority class in D;
                        // attribute_list - List of available attributes at that node or partition
4   Apply Attribute_selection_method, Gain ratio (D, attribute_list) to find the best Splitting_Attribute
     and label the node N with Splitting_Attribute;
5   if Splitting_Attribute is discrete_valued then
      attribute_list=attribute_list- Splitting_Attribute; // Splitting_Attribute is removed from the attribute_list
6   for each outcome j of Splitting_Attribute  // partition the instances and grow subtrees for each partition
7       let D_j be the set of data tuples in D satisfying outcome j;  // a partition
8       if D_j is empty then
9           attach a leaf labeled with the majority class in D to node N;
10      else attach the node returned by C4.5(D_j, attribute list) to node N;
11   end for
12   return N;
```

3.2 Finding Destination leaf for profit maximization

When a customer's instance is input to the PET it reaches one of the leaf nodes which represent a class label and also the probability of belonging to each class. Based on the class label and the class probability information, the degree of customer's loyalty and the amount of profit that the service provider can make from the customer is predicted. If a customer is predicted as attritor/less profitable, then, it is required to change her/him as a non-attritor and maximum profitable. To achieve such profit maximizing solution, we introduce our algorithm EDLF. When a customer's instance X falls into a leaf node of the PET that represents attrition/less profitability then, our proposed method finds an optimal destination leaf (defined below) with a maximum net profit by applying necessary actions. Providing the optimal solution is the goal. While finding the optimal destination leaf for a customer, EDLF algorithm considers each potential leaf node and then a leaf with a maximum net profit will be chosen.

Definition 1 (Source). The leaf node of the PET in which a customer's sample X falls into based on its attributes' values is defined as a Source(L_S).
Definition 2 (Desirable leaf). Irrespective of the number of classes of customers that the dataset has, only the leaf node possessing highest class C_1 probability among all the leaf nodes is a Desirable leaf.
Definition 3 (Candidate). A leaf node which estimates a profit more than that of the L_S is a Candidate (L_C). Amount of profit obtained from a customer if she/he falls into a leaf is computed with respect to the probabilities of all classes except class C_n represented by that leaf.
Definition 4 (Action). An action Actn is defined as a change in the value of an attribute *A* which is denoted as Actn = {*A*, $p \rightarrow q$}, q=1, 2, ..., *o* where $q \neq p$. *p* and *q* are original and changed values of *A* respectively. *o* is the number of outcomes of attribute *A*. If *A* is continuous valued attribute then *o*=2.

Definition 5 (Action set). If an instance is required to fall from L_S into an L_C of the PET, then a set of actions are required. When the path along root to L_C of the PET is considered, it contains say *len* number of decision nodes (non-class attributes). The attribute-value pairs along the path of L_C are $\{(A_i=q_i)$, i=1,2,...,$len\}$. For the instance X, the corresponding attribute-value pairs are $\{(A_i=p_i)$, i=1,2,...,$len\}$. In the act of shifting the instance from L_S to L_C, a set of actions in the form $Ast=\{(A_i,\ p_i \rightarrow q_i)$, i=1, 2, ..., $len\}$, where $p_i \neq q_i$, are required.
Definition 6 (Net Profit). To make the instance fall into an L_C from the L_S, an action set is required. For each action Actn=$\{A,\ p \rightarrow q\}$ in *Ast*, some cost is incurred. For each attribute a cost matrix is maintained where the entries in them are provided by the domain expert. Total cost for all actions in *Ast*, i.e. Tot_Cost is calculated. Net profit is then computed as the difference in the profit when the instance X is in L_C and in L_S minus Tot_Cost.
Definition 7 (Destination). A leaf node in which a customer's sample has been eventually shifted to, from the L_S with a maximum Net Profit is the Destination (L_D).
Definition 8 (Goal). If the L_S is not a Desirable leaf then, the Goal is finding the action set, *Ast*, to make the instance X fall into Destination L_D and symbolized as (X, $L_S \rightsquigarrow L_D$, Max_NetProfit) where Max_NetProfit is the Net Profit obtained which is eventually a maximum value.

With our proposed method, the tree is represented as a compressed model in the form of a 2-D array. Research framework of our method is presented in Figure 1. The BPM representing PET, input instance and the output given by the PET for the instance are inputs to our proposed algorithm. The dataset is preprocessed before it is used for PET construction. Prior to applying the proposed method on the input instance, the constructed PET is evaluated using 10-fold cross validation. If the input instance does not fall into the Desirable leaf of the PET, then the proposed method determines the Destination for it. This is achieved by applying the bitwise AND operations on the attributes' values of the input instance X and the Candidate which are organised as an array data structure.

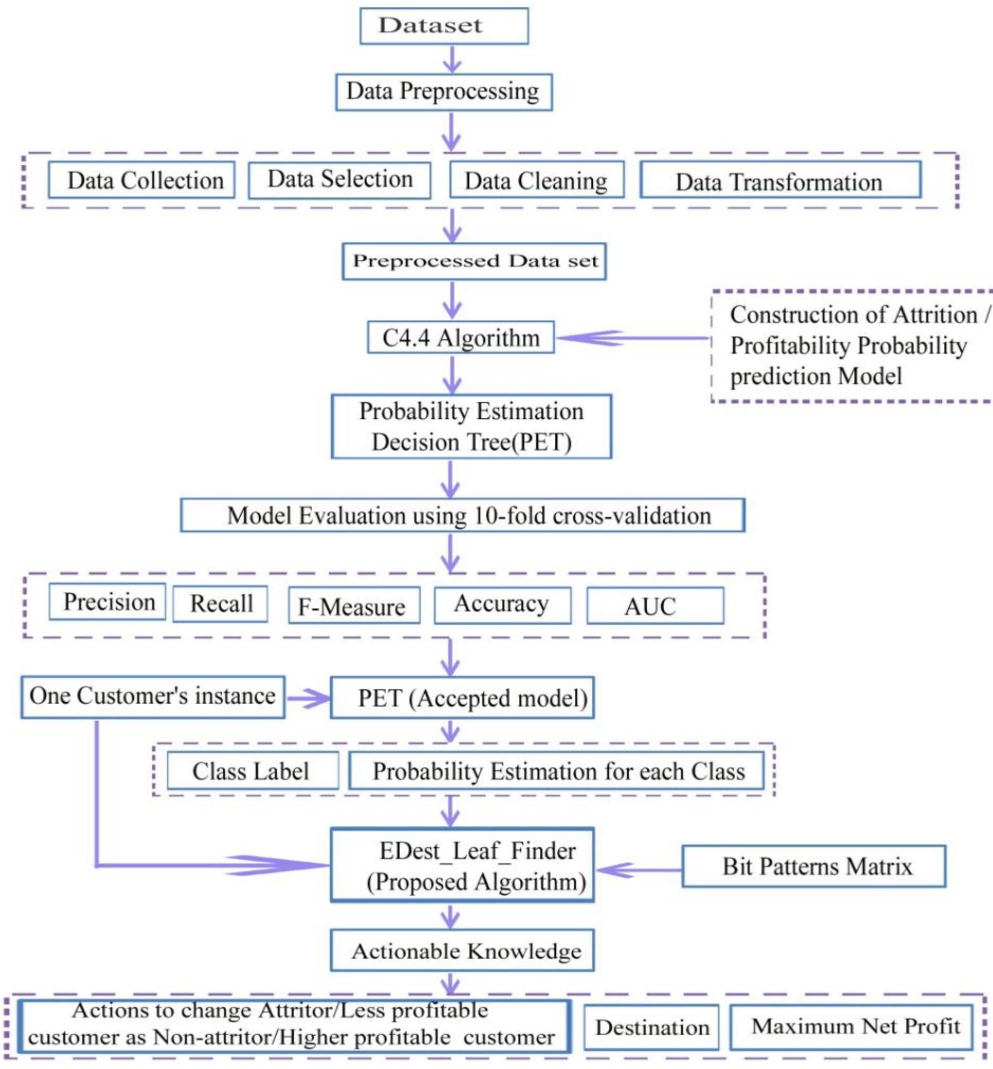
Dataset
Data Preprocessing
Data Collection
Data Selection
Data Cleaning
Data Transformation
Preprocessed Data set
C4.4 Algorithm
Construction of Attrition / Profitability Probability prediction Model
Probability Estimation Decision Tree(PET)
Model Evaluation using 10-fold cross-validation
Precision
Recall
F-Measure
Accuracy
AUC
One Customer's instance
PET (Accepted model)
Class Label
Probability Estimation for each Class
EDest_Leaf_Finder (Proposed Algorithm)
Bit Patterns Matrix
Actionable Knowledge
Actions to change Attritor/Less profitable customer as Non-attritor/Higher profitable customer
Destination
Maximum Net Profit

Figure. 1 Research framework of EDLF

3.2.1 Profit maximization for 2-class problems

There are many business sectors which generally classify their customers as Positive and Negative classes. The Positive class customers are loyal and stay with the enterprise and contribute profits. On the other hand, Negative class customers, who are familiarly called as attritors or churners, leave the enterprise and go for the service of the competitor. Banking sector is highly facing the attrition problem due to a number of options to the customer. Hence, for illustration of our proposed work, a synthetic dataset in Table 1 belonging to the Banking service sector, where each customer belongs to either of the two classes Positive/Negative is considered. In this sector, customer's profiles are included different attributes such as socio demographic (e.g., age, income, gender, job status, education and nationality, etc.), service levels provided to the customer, behavioural information (e.g., duration of service usage, amount of revenue generated, etc), etc. This information is used to predict the customer's nature.

The details of the synthetic dataset are shown in Table 2. This dataset consists of 17 records composed of four significant discrete attributes of the customers and a class label viz., Positive (C_1), Negative (C_2). Among 17 records of the dataset, 64.71% are Positive and 35.29% are Negative. When this data set is given as input to the C4.4 algorithm, the PET shown in Figure 2 is obtained as output. The induced PET depicts what sort of customer's features lead to either the class C_1 or C_2 and also the probability of belonging to each class. Thus, for a customer's instance, the PET can provide a class label and also the probability of belonging to each class. To provide more clarity, each leaf node of the PET in Figure 2 is associated with class label and also the probability of belonging to each class.

Table 1 Synthetic Bank customers dataset

RID	Service Level	Income Of Customer	Gender	Rate_Of_ Interest	Class
1	Low	High	Male	Low	Negative
2	Low	High	Male	High	Negative
3	High	High	Male	Low	Positive
4	Medium	Medium	Male	Low	Positive
5	Medium	Low	Female	Low	Positive
6	Medium	Low	Female	High	Negative
7	High	Low	Female	High	Positive
8	Low	Medium	Female	Low	Negative
9	Low	Low	Female	Low	Positive
10	Medium	High	Female	Low	Positive
11	Low	Medium	Female	High	Positive
12	High	Medium	Male	High	Positive
13	High	High	Female	Low	Positive
14	Medium	Medium	Male	High	Negative
15	Medium	Low	Male	Low	Positive
16	High	High	Male	Low	Positive
17	Low	Medium	Male	High	Negative

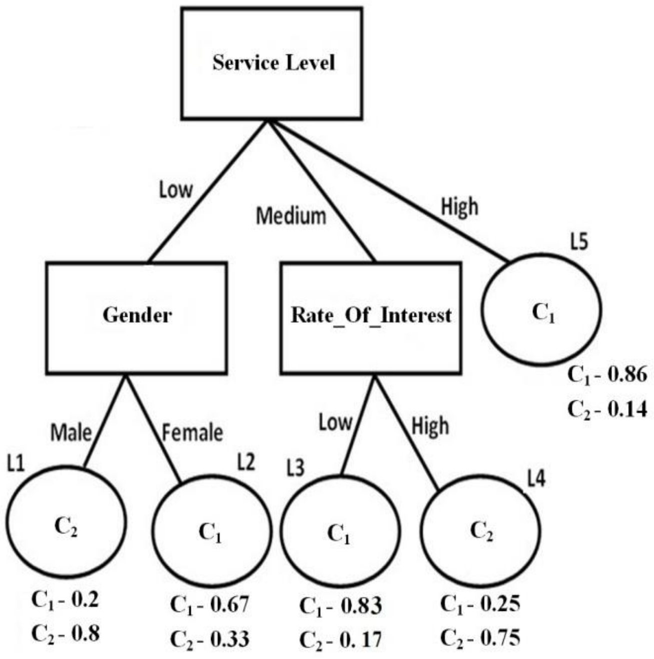

Figure. 2 PET representing the dataset in Table 1

Table 2: Features of synthetic Bank customers dataset in Table 1

Attribute	Description	Flexible/ Non-flexible	Values
Service Level	Level of service provided to the customer	Flexible	Low, Medium, High
Income Of Customer	Customer's annual income	Non-flexible	Low, Medium, High
Gender	Gender of the customer	Non-flexible	Male, Female
Rate_Of_Interest	Interest Rate on the loans taken by a customer	Flexible	Low, High
Class Label	Whether customer left (Negative) or stayed (Positive) with the Service provider	Flexible	Positive, Negative

3.2.2 Bit patterns matrix representation of the PET

EDLF is based on bitwise operations and array data structures. Normally bitwise operations are utilized and incorporated when we are required to encode or decode data in a packed manner and need the computations to be performed quickly. Bitwise AND can be used to execute set intersection and bitwise OR to execute set union. We use the former point which is being employed in our proposed method. To increase efficiency arrays are used in the course of attaining the Goal since; accessing array elements can be done in a faster way.

The BPM, B[ ][ ] is formed by *p* number of *bit patterns vector*s where *p* is the number of leaf nodes representing one of the class labels among C_1, C_2, ..., C_{n-1} and *n* is the number of customers' classes. Hence, BPM for the PET in Figure 2 is formed with the bit patterns vectors of the leaf nodes L2, L3, and L5, since, n=2. Bit patterns of the leaf nodes representing class C_2 are not necessary to furnish in the BPM since; no customer will be changed to such leaf nodes. For each of these *p* leaf nodes of the PET, the path along the root to a leaf is considered and each decision node's (non-class label attribute's) value in that path is represented in terms of the fixed size *bit pattern*. Bit patterns are formed for all the attributes present in that path and also for the attributes not present in the path but in the tree. However, the method considers only the attributes which are present in the PET and not all the attributes in the dataset. If an attribute has *v* outcomes, then that attribute's value is represented with a *v-bit pattern*. For each value of an attribute, among *v* bits of the pattern, only one bit is set to 1 and all the remaining bits are set to 0. The number of *v-bit patterns* in each row of the BPM is *q* where *q* is the number of distinct decision nodes in the PET. For the PET shown in Figure. 2, bit patterns of the attributes are formed as shown below.

Gender (Male-10, Female-01), *Service Level* (low-100, medium-010, high-001), *Rate_Of_Interest* (low-10, high- 01). If an attribute is not found in the path from root to a leaf then, its bit pattern is represented with all 1's for our computational requirements. For the PET in Figure 2, L3 is one of the leaf nodes. Fixed size bit patterns vector for the path from the root to leaf L3 is formed as: Attribute values to reach L3 shall be *Gender*=NULL, *Service Level*=medium, *Call Charges*=low. Thus, *bit patterns vector* of the path of L3 is (11, 010, 10). Similarly, bit pattern vectors of the other potential Candidates L3 and L5 are also formed.

The BPM of the PET in Figure 2, whose order is *p*×*q*, is shown in Figure 3. The bit patterns in each row of BPM are arranged in the order of *Gender*, *Service Level* and *Rate_Of_Interest*. Class label and class probabilities estimations of the rows (leaf nodes) in BPM are maintained as furnished in Table 3, which are needed during Net Profit computation. With this approach, once the PET is represented in the form of BPM, PET is used only to find the class label and class probability estimation of the input instance. The total remaining process is performed on the 2-D array formed by fixed length bit patterns vectors. However, the method of representing the PET as a BPM is the same irrespective of the number of classes of customers. After the PET is represented as a BPM, EDLF starts its post processing. It considers bit pattern vectors of all the Candidates and determines the Destination among them. In the PET in Figure 2 only the leaf node L5 is a Desirable leaf.

$$B = \begin{array}{c} \begin{array}{ccc} \textit{Gender} & \textit{Service Level} & \textit{Rate\_Of\_Interest} \end{array} \\ \left(\begin{array}{ccc} 01 & 100 & 11 \\ 11 & 010 & 10 \\ 11 & 001 & 11 \end{array}\right) \end{array} \begin{array}{l} \\ \leftarrow L2 \\ \leftarrow L3 \\ \leftarrow L5 \end{array}$$

Figure. 3 Bit patterns matrix representation of the Candidates of PET in Figure 1

Table 3: Class label and class probabilities Estimation of the leaf nodes of the PET in Figure 1

Leaf	Class	Class-C_1 probability	Class-C_2 probability
L2	C_1	0.67	0.33
L3	C_1	0.83	0.17
L5	C_1	0.86	0.14

3.2.3 Determining the Candidates and Destination for an instance in 2-class problems

In the case of 2-class problems, say for a customer's instance X Source is L_S, representing class C_k, where k=1(Positive) or k=2 (Negative), which is not a Desired leaf. Then, in these two contexts, the proposed method determines the Destination among all the probable Candidates, L, as:

$$k=1: \ (X,\ L_S \rightsquigarrow \text{Max\_NetProfit}\ ((\forall L)\ ((L \in C_1)\ \wedge\ (Pc_1(L) > Pc_1(L_S)))))$$

$$k=2: \ (X,\ L_S \rightsquigarrow \text{Max\_NetProfit}\ ((\forall L)\ (L \in C_1)))$$

In the above two cases, $Pc_1(L)$, $Pc_1(L_S)$ are class C_1 probabilities of the probable Candidate, L, and Source respectively. In the case when k=1, if L represents class C_1, and its C_1 class probability is more than that of the C_1 probability of L_S, then L is a Candidate. In the other case when k=2, any leaf node L representing class C_1 is a Candidate. In each of these cases, among all the Candidates one which yields maximum Net Profit is chosen as the Destination.

In the case of 2-class problems, when a customer's instance has fallen into a leaf node then the profit obtained from her/him is computed with respect to the class C_1 only. If the probability of belonging to class C_1 is 1.0 (customer is 100% of Positive class) for a leaf node, then enterprise makes a certain amount of profit P_x from the customer who has fallen into this leaf. For example, if an instance X has fallen into the leaf L2 of the PET in Figure 2, then 0.67*1000=$670 is made by the service provider when P_x value is considered as $1000. If X is shifted to a Candidate then, the profit obtained after shifting is calculated in terms of the difference of class C_1 probabilities in Source and Candidate which is the net gain in the class C_1 probability after the shifting. If X is shifted from L1 to L5 then the profit is 860-200=$660. Eventually, the Net Profit, N_P, obtained after transformation of X from L_S to an L_C is formulated in Eq. (6). By subtracting the total cost of actions incurred for this transformation, the EDLF computes the Net Profit.

$$N_P = P_x * (P_{C_1}(L_C) - P_{C_1}(L_S)) - Tot\_Cost \tag{6}$$

For illustration, a customer's sample X is considered, where the attributes' values are *Gender*=Male, *Service Level*=Low, *Rate_Of_Interest*=High. X falls into the leaf node L1 of the PET in Figure 2 which is not a Desirable leaf. Bit patterns vector representation of instance X is (10, 100, 01). Then, the Net Profits, if X is shifted to each of the Candidates, are computed and the one which produces the maximum is the Destination. For the case of X, Candidates are L2, L3, and L5.

For shifting X to a Candidate, a set of actions are required. To confirm whether an action is required or not on an attribute, a bitwise AND operation is performed between the bit patterns of X and the Candidate on the corresponding attribute's values. If the bitwise ANDing results in *False* then, it denotes that an action is required. Consequently, during finding the Destination for a customer's sample X, only the attributes, after bitwise ANDing, which are resulted as *False* are considered for further steps. Cost for this action will be obtained from the cost matrix of the corresponding attribute. *True* cases need not be considered since the result will be *True* in two cases. The first case, if a particular attribute's value is the same for the customer's instance and also the Candidate's path, then there is no need to change the value of that attribute. Second, if an attribute is not present in the path of a Candidate then, the case of changing that attribute's value does not arise. To make this case *True*, that nonpresent attribute's value's bit pattern is represented with all 1's.

In the process of attaining the Goal, bitwise AND operations are performed between X and the Candidates. The results after the bitwise AND operations are shown in Table 4 where the operations resulted in *True* and *False* are denoted as T and F respectively. During the Net Profits computation, every action cost is considered as $100 and P_x value is taken as $1000. Then, L1⤳L2: Not possible since a non-flexible attribute's (Gender) value has to be changed since 10 AND 01=*False*. L1⤳L3: Ast={(*Service Level*, Low→Medium), (*Rate_Of_Interest*, High→Low)}, N_P=1000*(0.83-0.2)-(100+100)=$430. L1⤳L5:Ast={(*Service Level*, Low→High)}, N_P=1000*(0.86-0.2) -100=$560. Goal : (X, L1⤳L5, $560). In this scenario, by shifting the customer X from L1 to L5 a maximum amount of Net Profit is obtained and no other option can improve the Net Profit more than this amount.

As the other case, for the sample Y={*Gender*=Female, *Service Level*=Low, *Rate_Of_Interest*=High}, Goal is (Y, L2⤳L5, $90) with Ast={(*Service Level*, Low→High)}. In this case, for Y, the leaf nodes L3 and L5 are the Candidates. Finally, the sample Y is shifted from a leaf node representing class C_1 to another leaf node representing class C_1 but possessing a higher probability of C_1.

Further, to achieve extra computational performance, attributes order in the bit patterns vector is formed such that in each row of BPM, bit patterns of all the non-flexible attributes are placed first followed by flexible attributes'. In the BPM shown in Figure 3, bit pattern of *Gender* is the first element in every row. This is done irrespective of the order of the attributes along the path from the root to leaves. The reason for this arrangement is for early stopping i.e. along the path of a Candidate, if any non-flexible attribute's value is required to be changed, then that Candidate can be ignored in that stage itself so that unnecessary extra computations are avoided. This situation is observed while performing L2 & X on the attribute *Gender* where we have stopped and not performed bitwise ANDing on the remaining attributes of the L2 as shown in Table 4.

Table 4: Bitwise AND (**&**) operation between instance X and Candidates

L2 & X			**L3 & X**			**L5 & X**		
10	100	01	10	100	01	10	100	01
01	100	11	11	010	10	11	001	11
F			T	F	F	T	F	T

3.3 Profit maximization for 3-class problems

In real time, several business sectors categorize their customers into three classes based on the amount of profit obtained from them within a period of time. In one scenario of retail Banking, customers are categorized as Medium-high profit customers (C_1), Low profit customers (C_2), Unprofitable customers (C_3) [17]. The amount of profit m ade from the class C_1 and class C_2 customers is high and mediocre respectively, while from class C_3 customers profit contribution is zero or negative. In the case of 3-class applications, when an instance X falls into a leaf node, then, that instance most likely contains the qualities of all the three classes but with different ranks.

In this scenario, as both the classes C_1 and C_2 are profitable, profit is calculated with respect to these two classes. For example, L2 in Figure 4 is one of the leaf nodes of the hypothetical PET representing a 3-class dataset that comprises the class probabilities C_1=0.8, C_2=0.1, C_3=0.1. This means for a customer who has fallen into L2 contains the qualities of all the three classes in different degrees. If class C_1 probability is 1.0 then the service provider earns an amount of P_{X_1} and if class C_2 probability is 1.0 then an amount of P_{X_2} is made. Let the assumption is P_{X_1}=\$1000 and P_{X_2}=\$500. An instance X is considered for illustration and assumed as fallen into the leaf node L2. If X remains in L2, then the amount of profit made is 0.8*1000 + 0.1*500 =\$850.

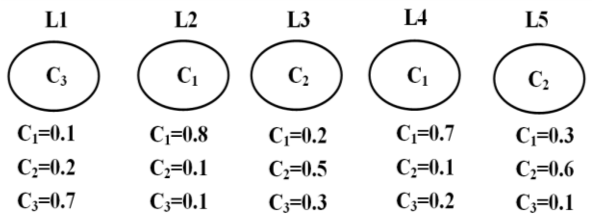

Figure. 4 Leaf nodes of a hypothetical PET

While computing the profit (Pt) resulted after migration of the instance from Source to a Candidate, sum of the differences of class C_1 probabilities in Source and Candidate and C_2 probabilities in Source and Candidate is considered as given in Eq. (7). Profit is calculated with respect to the gain in the C_1 and C_2 class probabilities. The amount of profit (Pt) and Net Profit (N_P) obtained from a customer after changing her/him from L_S to an L_C is computed as given in Eqs. (7) and (8) respectively. In Eq. (7), $Pc_2(L_C)$, $Pc_2(L_S)$ are class C_2 probabilities of the Candidate and Source respectively.

$$Pt = (P_{X_1}*(P_{C_1}(L_C) - P_{C_1}(L_S))) + (P_{X_2}*(P_{C_2}(L_C) - P_{C_2}(L_S))) \qquad (7)$$

$$N_P = Pt - Tot\_Cost \qquad (8)$$

3.3.1 Determining the Candidates and Destination for an instance in 3-class problems

In 3-class problems, for the customer's instance X, say the L_S is representing class C_k (k=1 or 2 or 3) which is not a Desired leaf. For each of these three scenarios, our method determines the Destination among the probable Candidates, L, as:

k=1: (i) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) ((L \in C_1) \wedge (Pc_1(L) > Pc_1(L_S))))).$

k=2: (i) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) (L \in C_1))),$

(ii) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) ((L \in C_2) \wedge (Pt>0)))).$

k=3: (i) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) (L \in C_1))),$

(ii) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) (L \in C_2))).$

In the case of k=1, L_S represents class C_1 and then, another leaf representing C_1 but with a higher probability of C_1 is a Candidate. In the case of k=2, the leaf nodes representing C_1 are the Candidates of a top priority since they can provide more profit than the other options. If it is not possible to shift to any of the Candidates representing class C_1 (due to the actions on non-flexible attributes or due to the high action costs, no Candidate is yielding a positive Net Profit), then, the other option is finding the best one among the Candidates representing class C_2. In this context, the leaf node which can provide a profit (Pt), computed using Eq. (7), more than that of the Source is a Candidate. In the case of k=3, our method first tries to shift the instance X from L_S to a leaf representing class C_1. If this is not possible, then it tries to shift to a leaf representing class C_2. Hence, all the leaf nodes representing either class C_1 or class C_2 are the Candidates. A Candidate which yields maximum Net Profit is the Destination. However, the Net Profit from a Candidate also depends on the cost of actions.

For example, for a customer's instance X, Source is L1 of the hypothetical PET in Figure 4. Then, the Candidates are L2, L4, L3, and L5. P_{X_1} and P_{X_2} are considered as \$1000 and \$500 respectively. For simplicity, it has been assumed that to shift Y to any of the Candidates, it requires two actions and every action requires a cost of \$100. First priority goes to L2 and L4. L2 yields a Net Profit (1000 * (0.8-0.1) + 500 * (0.1-0.2)) – 200=\$450. L4 yields a Net Profit of \$350. Then, the Goal can be achieved by making X fall into the leaf L2, which produces a maximum Net Profit. Goal : (X, L1 $\rightsquigarrow$ L2, \$450). For this sample, since a Candidate representing class C_1 and producing some positive Net Profit is found, Net Profits w.r.t. L3 and L5 are not computed. As another example, let Y is an instance whose Source is L3. Then the Candidates for Y are L2, L4, and L5. If it is not possible to change Y to either L2 or L4 then, L5 can be the Candidate and also Destination. In this case, it is assumed that one action is needed to shift Y from L3 to L5 and hence, stood as Destination leading to the Goal (Y, L3$\rightsquigarrow$L5, \$50).

3.4 Profit maximization for n-class problems

In various business applications, there are datasets composed of multi-classes like four, five and more number of customer classes [14-16]. In CRM, in one scenario customers are classified as high stay (contributes highest level of profitability), latent stay, spurious stay, and low stay. In such applications, when there are *n* number of classes of customers (n≥2) then, they are perceived as arranged in the decreasing order of degree of profitability viz., C_1, C_2, ..., C_n where C_n represents the class of attritors. In the multi-class scenario, a customer who falls into any of the leaf nodes L of the PET is most likely to have the qualities of all the n classes in different degrees. In such a case, except class C_n, all the other classes are profit yielding in different amounts. Hence, the amount of profit earned from a customer if she/he falls into a leaf L is:

$$\sum_{k=1}^{n-1} (P_{X_k} * P_{C_k}(L)) \tag{9}$$

In Eq. (9), P_{X_k} is the amount of profit earned by the enterprise if probability of belonging to class C_k for the customer is 1.0 and P_{C_k} is the probability that the customer belongs to class C_k. Profit (Pt) obtained from a customer X by shifting her/him from a Source L_S to a Candidate L_C is computed w.r.t all the class probabilities except class C_n probability. Pt is generalized as shown in Eq. (10) and the Net Profit (N_P) is shown in Eq. (11).

$$Pt = \left(\sum_{k=1}^{n-1} (P_{X_k} * P_{C_k}(L_C)) - \sum_{k=1}^{n-1} (P_{X_k} * P_{C_k}(L_S)) \right)$$

$$Pt = \sum_{k=1}^{n-1} (P_{X_k} * (P_{C_k}(L_C) - P_{C_k}(L_S))) \tag{10}$$

$$N_P = Pt - Tot\_Cost \tag{11}$$

3.4.1 Determining the Candidates and Destination for an instance in n-class problems

In the case of n-class problems, for a customer X if the L_S is not a Desirable leaf representing class C_k, where k=1, 2, .., n then, for each of these n scenarios, the proposed method searches and finds the Destination among all the probable Candidates, L, as:

k=1 : (i) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) ((L \in C_1) \wedge (Pc_1(L) > Pc_1(L_S)))))$

⋮

k=m : (i) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) (L \in C_1)))$,

(ii) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) (L \in C_2)))$,

⋮

(m) $(X, L_S \rightsquigarrow Max\_NetProfit((\forall L)((L \in C_m) \wedge (Pt>0))))$.

⋮

k=n : (i) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) (L \in C_1)))$,

(ii) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) (L \in C_2)))$,

⋮

(n-1) $(X, L_S \rightsquigarrow Max\_NetProfit ((\forall L) (L \in C_{n-1})))$.

In the case of k=m (m>1 and m<n), L_S represents class C_m and then all the leaf nodes representing the classes C_1 through C_{m-1} are the Candidates. Then, EDLF tries to find a Destination by giving priority to the Candidates in the order C_1, C_2,..., C_{m-1}. If any Candidate of the higher profitable class provides some positive Net Profit then the algorithm stops and does not go for the Candidates of the other lower profitable classes. When even the Candidates of the class C_{m-1} cannot yield a positive Net Profit, then, the leaves representing class C_m which can give the profit more than that of the L_S are the Candidates. In the case of k=n, the leaves representing any of the classes C_1, C_2, ..., C_{n-1} are the Candidates. If none of the Candidates representing class C_i (i=1, 2, ..., n-2) can yield a positive Net Profit, then only the algorithm tries a Candidate representing class C_{i+1}.

To achieve much better computational performance, the bit patterns vectors in the matrix are arranged in the order of class labels of the leaf nodes starting from class C_1 and ending with C_{n-1} so that unwanted Candidates need not be processed. Further, we don't consider arranging all the rows representing one class C_k (k=1, 2, ..., n-1) of B[ ][ ] in the descending order of the values of class C_1 probabilities. The reason is that there is no guarantee that a Candidate possessing higher class C_1 probability can yield maximum Net Profit among all the Candidates of that class. This is because; Net Profit also depends on the number of actions and cost of actions. Hence, all the Candidates representing one class are required to be processed to find the optimal one. For illustration, we have presented Table 5 which represents the organization of Bit patterns vectors of 7 leaf nodes of an imaginary PET in the order C_1, C_2, and C_3 where n=4. Among 10 leaves, 3 are representing class C_4. Hence, the BPM contains 7 rows. PET contains 5 input attributes A1, A2, A3, A4, and A5. Out of them A2 and A4 are non-flexible and remaining are flexible. It can be observed that patterns within one class are not arranged based on class C_1 probabilities. Bit patterns of the non-flexible attributes A2 and A4 occupied the first two positions in each row.

As discussed in the above sections, for finding the Destination and computing the Net Profit from one customer in the multi-class scenario, a method is devised and presented in algorithm 2. We have designed the algorithm such that it handles all the three different cases where the Source represents one among them i.e. 1. $L_S \in C_1$ 2. $L_S \in C_m$, and 3. $L_S \in C_n$. The overall working of the algorithm is briefly described in the four steps below:

1. Initialize Max_NetProfit , c, Dest **(line 1).** c is the class of a row (leaf) of BPM represented for a path from the root to a leaf which obviously starts from C_1. Therefore c=1.
2. Case 1 : **k = 1** i.e. $\mathbf{L_S \rightsquigarrow C_1}$. If the Source ($L_S$) represents class C_1, then, find Destination among the leaf nodes having better C_1 class probability than that of the L_S **(lines 2-15).**
3. Case 2 : **k < n** i.e. $\mathbf{L_S \rightsquigarrow C_m}$. If the L_S represents class C_m where m>1 and m<n, then, find Destination from the Candidates of class C_1, if not possible from the Candidates of class C_2, ..., and if not possible from the Candidates of class C_{m-1} **(lines 17-33)** and if not possible from the Candidates of class C_m **(lines 34-47).**
4. Case 3 : **k = n** i.e. $\mathbf{L_S \rightsquigarrow C_n}$. If the L_S represents class C_n then, find Destination from the Candidates of class C_1, if not possible from the Candidates of class C_2, ..., and if not possible from the Candidates of class C_{n-1} **(lines 49-68).**

The methods Profit( ) and Tot_Cost( ) computes profit when X is shifted from L_S to L_C and total cost incurred for actions respectively. These methods are presented in algorithm 3 and algorithm 4.

Table 5: Organization of BPM of an imaginary PET representing 4-class data

S. No.	Leaf No.	Class	Class C_1 Probability	Class C_2 Probability	Class C_3 Probability	Class C_4 Probability	BPM				
							A2	A4	A1	A3	A5
1	L1	C_1	0.90	0.04	0.03	0.03	**10**	**111**	**100**	**111**	**111**
2	L5	C_1	0.70	0.10	0.10	0.10	**11**	**111**	**010**	**100**	**111**
3	L8	C_1	0.80	0.05	0.10	0.05	**11**	**100**	**001**	**111**	**111**
4	L2	C_2	0.05	0.75	0.05	0.15	**01**	**111**	**100**	**111**	**100**
5	L9	C_2	0.10	0.80	0.05	0.05	**11**	**010**	**001**	**111**	**111**
6	L3	C_3	0.20	0.05	0.70	0.05	**01**	**111**	**100**	**111**	**010**
7	L6	C_3	0.05	0.05	0.85	0.05	**11**	**111**	**010**	**010**	**111**

Algorithm 2: ***EDest_Leaf_Finder (EDLF)***

Inputs : - One customer's instance X
- Output given by PET for input instance X i.e. Source L_S information for X in PET including **class k** and each class probabilities
- Bit patterns vectors Matrix B representing PET
- n (number of customers' classes)
- Cost matrices for all the flexible attributes in the PET
- P_{X_k} for all k values.

Output : - Destination
- Net Profit

```
Max_NetProfit ← 0,   c ← 1, Dest ← $L_S$ ;     // c = 1. Rows in B obviously starts with class $C_1$, Dest - Destination for X.
if  k = 1   then                                // $L_S \in C_1$
    for  i ← 1  to  p  do          // p is number of  rows in B
        if  $L_i \notin C_1$   then    // If the class represented by Leaf $L_i$ (given in B[ ][ ]'s i$^{th}$ row) is not $C_1$
            break;
        else
            if  $Pc_1(L_i) > Pc_1(L_S)$   then
                $N_{Pi}$ ← Profit($L_S$,  $L_i$) - Tot_Cost(X, $L_i$);   //  $N_{Pi}$ - Net profit w.r.t. Candidate-i
                if  $N_{Pi}$ > Max_NetProfit  then
                    Max_NetProfit ← $N_{Pi}$ ;
                    Dest ← $L_i$;
                endif
            endif
        endif
    endfor
else
    if  k < n   then                             // $L_S \in C_m$
        for  i ← 1  to  p  do
            if  c < k   then
                if  $L_i \in C_c$   then          // If the class represented by Leaf $L_i$ is c. c is initially 1 since, classes start from $C_1$.
                    $N_{Pi}$ ←  Profit($L_S$,  $L_i$) - Tot_Cost(X, $L_i$);    //  $N_{Pi}$ - Net profit w.r.t. Candidate-i
                    if  $N_{Pi}$ > Max_NetProfit   then
                        Max_NetProfit ← $N_{Pi}$ ;
                        Dest ← $L_i$ ;
                    endif
                else
                    if  Max_NetProfit > 0   then      // Eventually if some positive profit got with class $C_1$.
                        break;
                    else
                        c ← c + 1;
                        i  ← i - 1;
                    endif
                endif
            else
                if  c = k   then             // If  $L_i \in C_m$ and  $L_S \in C_m$
                    Pt = Profit($L_S$,  $L_i$);
                    if  Pt > 0   then
                        $N_{Pi}$ ←  Pt - Tot_Cost(X, $L_i$);
                        if  $N_{Pi}$ > Max_NetProfit    then
                            Max_NetProfit ← $N_{Pi}$ ;
                            Dest ← $L_i$;
                        endif
                    endif
                else
                    break;
                endif
            endif
        endfor
    else                                 // $L_S \in C_n$
        for  i ←1  to  p  do
            if  c < n   then      //  If the class represented by Leaf $L_i$ is less than n  i.e. $C_n$
                if  $L_i \in C_c$   then
                    $N_{Pi}$ ←  Profit($L_S$,  $L_i$) - Tot_Cost(X, $L_i$);
                    if  $N_{Pi}$ > Max_NetProfit   then
                        Max_NetProfit ← $N_{Pi}$ ;
                        Dest ← $L_i$;
                    endif
                else
                    if  Max_NetProfit > 0   then
                        break;
                    else
                        c ← c+1;
                        i  ← i-1;
                    endif
                endif
            endif
        endfor
    endif
endif
Output (Dest, Max_NetProfit) ;
```

Algorithm 3: Profit calculation for multi-class problems

```
Profit ($L_S$, $L_C$)
  begin
    Pt ← 0;
    for k ←1 to n-1 do
      Pt ← Pt + ( $P_{X_k}$ * ($P_{C_k}(L_C)$ - $P_{C_k}(L_S)$) )
    end for
    return (Pt);
  end
```

Algorithm 4: Calculation of total cost of actions for Source to Candidate transformation

```
Tot_Cost(X, $L_i$ )
  begin
    for j ← 1 to q do    // q is the number of columns in the Matrix B
      T_Cost ← 0;        // T_Cost - Total cost of actions
      if (B(i, j) & $X_j$ ) is False then  // Bitwise AND operation between $i^{th}$ Candidate's $j^{th}$ attribute's values of Bit pattern vector and X
        if attr[j] is a non-flexible attribute then  // attr[j] – $j^{th}$ attribute
          skip this Candidate-i and continue with next one  // X can't be shifted to Candidate-i
        else
          T_Cost ← T_Cost + attr[j]_Cost_matrix(corresponding value) ;
        endif
      endif
    end for
    return(T_Cost);
  end
```

4 Experimental Evaluation

In order to evaluate the performance of the proposed algorithm, extensive experiments are conducted using several multi-class UCI datasets, real Telecom dataset, and other benchmark datasets. We have compared the performance of the proposed method with a single crisp tree based method, and two ensemble tree based methods. The proposed method exhibited a similar performance in all the experiments and outperformed the state-of-the-art methods.

4.1 Performance analysis of EDLF

The strength of EDLF algorithm lies in the effective usage of two vibrant points which are suitable for the research discussed in this paper. First is, it completely counts on bitwise AND operations. In computer programming, a bitwise operation is performed in between bit patterns at the level of their individual bits. For performing comparisons and other calculations efficiently, bitwise operations are far preferable as they straight away executed by the processor. After compilation of any bitwise operation, irrespective of the programming language used for implementation, it is translated as a single assembly language instruction in all most all assembly languages though that assembly language is overly simplified. When that assembly instruction turn comes for execution, it takes only one CPU cycle for evaluation. Thus the bitwise operations are basic and quick operations which can most significantly enhance the computation performance. Second is, performing the bitwise operations on the elements of a 2-dimensional array. Using the array data structure for storing and accessing a group of elements can greatly enhance the computational performance since an array is an efficient data structure for this purpose. In the context of accessing the elements, arrays are simple and efficient data structures than any other data structures like linked lists, etc. Moreover, accessing any single element in an array is done in a faster manner and in the constant time O(1).

To compute the Net Profit obtained from a customer X it is necessary to process the elements of the BPM which contains *p* rows and *q* columns. For complexity analysis, we consider that on average, the number of rows and the number of columns in BPM is *p*. In the case of successful search (Destination found), best case runtimes can be observed if any of the Candidates of class C_1 yields a positive Net Profit for the input instance X. In this case, the remaining Candidates of the classes C_2, C_3, ..., C_{n-1} need not be processed and the algorithm's time complexity is O(1). If L_S for an instance X represents class C_n and if the Destination is found among the

Candidates of class C_{n-1} only and if most of the attributes in each row of BPM are flexible and their values are required to be changed then, the proposed algorithm exhibits worst case runtime i.e. $O(p^2)$. For successful search, average case time complexity is also $O(p^2)$.

In the case of unsuccessful search (Destination not found), if each row's first element represents a non-flexible attribute and if that attribute's value has to be changed, then the algorithm exhibits the best case with runtime $O(p)$. Another best case scenario for unsuccessful search arises when L_S represents class C_1 and if the Destination is not found among the Candidates of class C_1. If L_S represents class C_n and if no Candidates of the classes C_1 through C_{n-1} yields a positive Net Profit, and if majority number of attributes in each row of BPM are flexible and their values are required to be changed then the proposed algorithm exhibits worst case runtime $O(p^2)$. During the unsuccessful search, average case time complexity when L_S represents class C_n is $O(p^2)$ and if L_S represents class C_1 it is $O(1)$. However, in real life, time complexities in the order of quadratic are acceptable and the customer's class changing problem can be solvable in a finite amount of time.

4.2 Experimental setup

To evaluate the efficiency of our algorithm, datasets which are appropriate to the research discussed in this paper are used. To construct a PET with C4.4 algorithm, Weka (version of 3.8) source code in Java, has been used. All numeric attributes in the datasets used in the experiments are discretized (except to the datasets in section 4.4) by applying the supervised filter viz., *Discretize* in Weka which is based on the class information, via MDL method [32]. The constructed PET's performance is evaluated using 10-fold cross-validation and the precision, recall, F-measure, accuracy and also AUC to figure out the probability estimation capability [33] are recorded for each dataset. Then, the proposed method, EDLF, is implemented in Java programming language and the experiments are conducted on a dual core Pentium 4, 2.5 GHz processor with 4GB RAM running on Windows7 Operating System.

We have used the UCI and other standard datasets to demonstrate the efficiency of the proposed method. However, we have incorporated the required features to these datasets such that they fit to demonstrate our research. During the experiments on UCI and other benchmark datasets, when working on 2-class datasets, one appropriate class has been considered as Positive (C_1) and the other one as Negative (C_2). Then, while computing the Net Profit, P_x value is set to \$1000. While applying the proposed method on datasets composed of multi-classes ($n>2$), we assumed and treated the classes aptly in the descending order of profit and computed the Net Profit. If the dataset is composed of n-classes viz., C_1, C_2, ..., C_k, ..., C_n, and if L_S represents class C_k ($k \geq 1$ and $k<n$), then a profit value of \$(1000/k) has been considered. However, for all datasets, if L_S purely represents class C_n, then a profit value of \$0 has been considered during Net Profit calculations. Action costs are taken appropriately for the flexible attributes in the range \$[0-200]. Based on their characteristics, some of the attribute are considered as non-flexible.

4.3 Comparison with single crisp tree based method

We first compare the computational performance of the proposed method with Leaf_Node_Search [9]. To our knowledge, this is the state-of-the-art method based on a single crisp tree and best fits for comparison with our method. Though Leaf_Node_Search algorithm is designed to work for 2-class problems, we have made required modifications to it such that it works for multi-class datasets and conducted experiments on UCI datasets, and a real Telecom dataset.

4.3.1 Experiments on UCI ML Datasets

For comparing runtimes and Net Profits of EDLF with Leaf_Node_Search, experiments are performed on 10 UCI datasets [34] with multi-classes that supports classification task and have sufficient examples. The details of the datasets used in the experiments along with the information of the PET constructed using each dataset and the PETs' technical evaluation measures are furnished in Table 6. The experimental setup which is discussed in section 4.2 is employed. The methods are implemented in the Java programming language.

Table 6: UCI datasets used in the experiments and the evaluation metrics of the PETs

Data set	No. of Instances	No. of Attributes	No. of Classes	Precision	Recall	F-measure	Accuracy (%)	AUC
Anneal	898	39	6	0.924	0.923	0.920	92.31	0.961
Autos	205	26	7	0.846	0.839	0.839	83.90	0.940
Balance Scale	625	4	3	0.658	0.696	0.673	69.60	0.768
Connect-4	67557	42	3	0.791	0.808	0.797	80.84	0.868
German	1000	20	2	0.706	0.721	0.710	72.10	0.696
Glass	214	10	7	0.743	0.738	0.732	73.83	0.852
Heart-c	303	14	5	0.791	0.789	0.787	78.87	0.801
Hypothyroid	3772	30	4	0.994	0.995	0.994	99.46	0.990
Nursery	12960	8	5	0.970	0.971	0.970	97.05	0.995
Solar	1066	12	6	0.723	0.745	0.728	74.48	0.916

The two methods try to find the Destination. Each time a set of instances are taken randomly from one dataset and given as input to the algorithms and the total runtime for finding the Destination for each of those instances, and total Net Profit obtained from all those set of instances are recorded. The graphs presented in Figure 5(a) through 5(j) describe the runtime behavior of the proposed method and Leaf_Node_Search on 10 UCI datasets. In all the graphs, x-axis and y-axis represent the number of instances taken as input from one dataset and the total runtime respectively. All the graphs shown in Figure 5(a) through 5(j) describes the fact that on all datasets, runtime of EDLF is significantly less as compared with Leaf_Node_Search whose runtime is increasing exponentially. Leaf_Node_Search allows a large number of candidate actions to be considered, complicating the computation. The computational complexity of Leaf_Node_Search is high since it is necessary to perform more number of primitive operations(comparisons). If the average number of Candidates, average number of attributes, and average length of path from the root to a Candidate is considered as *p* then the runtime of Leaf_Node_Search is $O(p^3)$. The time complexity of Leaf_Node_Search is cubic whereas the time complexity of the proposed method is polynomial with degree 2.

When the training data size is huge, the constructed PET's size can be large and deep. Then, the average length of each path from the root to a Candidate and the number of Candidates increases. When the tree size increases, then there can be a possibility for the increase in the number of actions. This eventually increases the computational time since the number of bit patterns in each row of the Bit pattern matrix gets increased and more bitwise operations need to be performed for finding the Destination. In the same scenario, Leaf_Node_Search requires higher run times than EDLF since it has to identify the attribute and then its value. However, Net Profit also decreases in this case as the Tot_Cost value increases. Hence, while working on large datasets with too many number of attributes, runtimes of both the methods increase. This fact is observed with the Connect-4 dataset, where the constructed tree contains 15952 nodes. However, this increase is high with Leaf_Node_Search. When more number of non-flexible attributes are encountered along a path, then, there is a possibility for early stopping of the processing which causes a reduction in the runtime of EDLF. The potential use of bitwise operations, best use of the array data structures and placing non-flexible attributes' bit patterns as the starting elements in each row of BPM are substantially helping improve the efficiency of EDLF.

Table 7 presents the Net Profits produced by the Leaf_Node_Search and EDLF on each of the 10 UCI datasets used in the experiments. On all the 10 UCI datasets, our algorithm is generating a higher total Net Profit in comparison with Leaf_Node_Search. Leaf_Node_Search produced a total Net Profit of $10695525 and on the other hand, EDLF produced $12000760 which is 12.2% more than that of the Net Profit produced by the Leaf_Node_Search. Leaf_Node_Search only tries to change an instance from a leaf node with class C_k to C_j ($j<k$) and stops if these transformations are not possible. On the other hand, in the same scenario, the proposed method tries to shift the instance from Source representing class C_k to a Candidate representing class C_k that can yield some positive Net Profit. Though a Candidate contains the higher probability of class C_1, along its path from the root, if it contains more non-flexible attributes, then there is a chance for discarding that Candidate. If more Candidates are of such kind, then, eventually there is a chance for the drop in the Net Profit. In the other case, if more the number of actions required for shifting an instance from L_S to L_C then more the chances for an increase in total cost that leads to reduction in Net Profit.

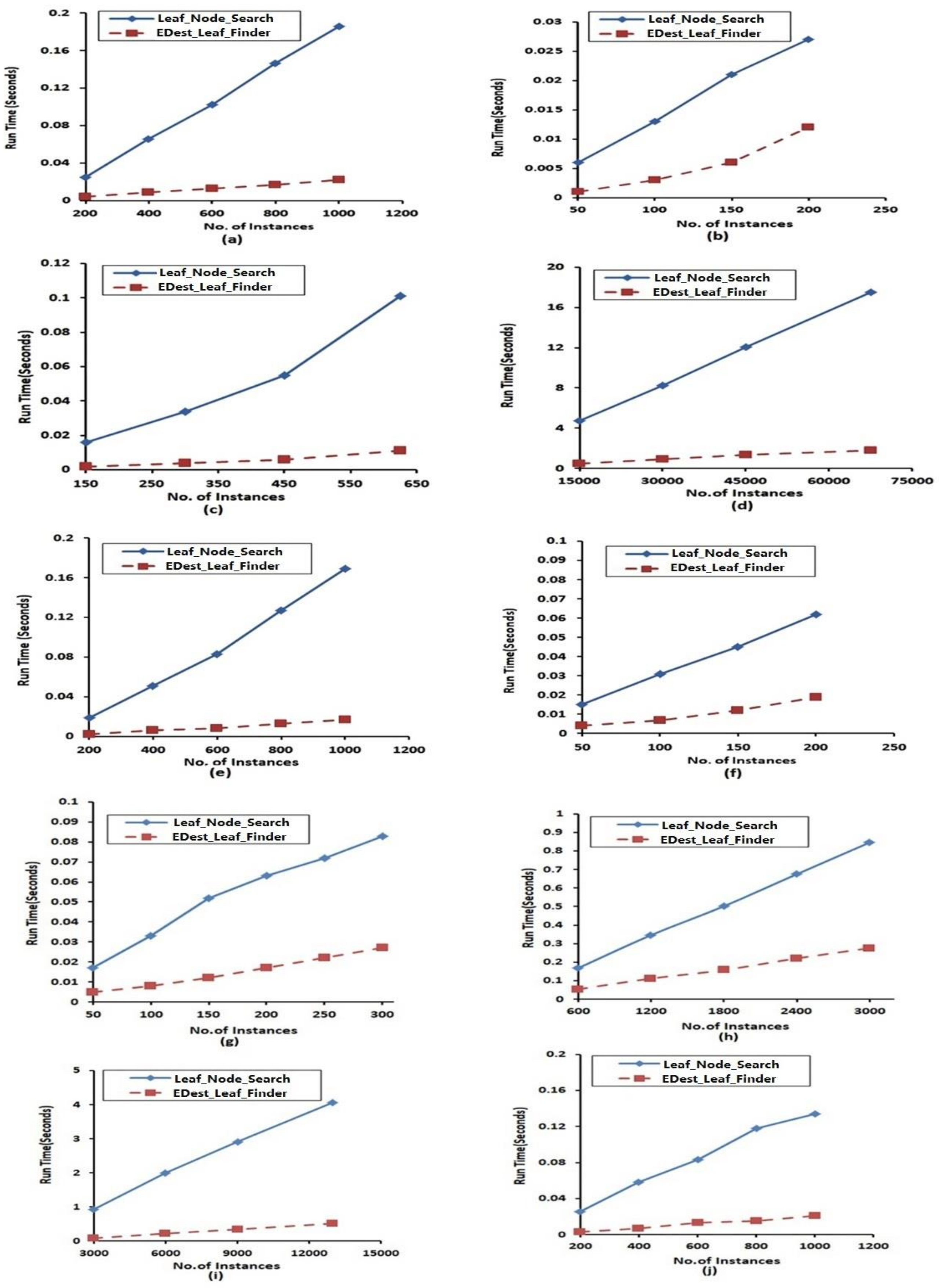

Figure. 5 Runtime comparisons of EDLF and Leaf_Node_Search on UCI datasets.
(a) Anneal (b) Autos (c) Balance Scale (d) Connect-4 (e) German (f) Glass (g) Heart-c (h) Hypothyroid (i) Nursery (j) Solar

Table 7: Net profits comparison of EDLF and Leaf_Node_Search on UCI datasets

Name of the Dataset	Net Profit($)	
	Leaf_Node_ Search	EDLF
Anneal	140610	153070
Autos	53400	67860
Balance Scale	105490	117330
Connect-4	8903200	9981700
German	169025	191080
Glass	57820	67070
Heart-c	49300	56440
Hypothyroid	228650	247620
Nursery	877580	995520
Solar	110450	123070
Total	10695525	12000760

4.3.2 Experiments on real data

Mobile phone service sector is facing more problems due to attrition which is leading to huge losses. Hence, we have considered and performed experiments on the large real time Telecom dataset obtained from an operator in India and demonstrated the performance of EDLF against the Leaf_Node_Search. This dataset has 15000 records of the customers where 2500 customers are classified as Negative and 12500 are as Positive. Each customer's instance is described by 40 input attributes which are divided into four categories viz., Socio-demographic (age, Gender, job status, and education etc.), behavioural (calling behaviour, short message usage, data usage, and frequency of calls made to customer care center, etc.), tariffs levied (Call Charges, Data Charges, etc.), and Service Levels (Customer complaint resolving, internet speed, network coverage, and call drop rate reduction, etc.). Among the 40 attributes, 19 are flexible and the remaining 21 are non-flexible. The decision attribute is a class label that denotes whether the subscriber has retained or left the service provider within a period of 3 months.

To maintain the balance in the distribution of class labels of the subscribers' records in the dataset and to avoid predicting many examples as Positive, we have randomly chosen 2500 Positive samples from the dataset. Totally 5000 records are used in the experiments where the share of the Positive and Negative samples is equal. By giving these 5000 samples as input to C4.4 algorithm, a PET is constructed which contains 734 leaves. Among these leaves, 346 represent the class Positive and the rest of the 388 have Negative class label. However, we have also applied the required data pre-processing methods (eg. data cleaning, data transformation, etc.) before building the tree. 10-fold cross validation is used to evaluate the induced PET where the accuracy and AUC values are observed as 88.04% and 0.894 respectively. During profit calculations, action costs are considered in the range \$[0-200] depending on the attribute. P_x value is considered as \$1000. After this setup, we have applied our EDLF and also Leaf_Node_Search to compare their runtimes and Net Profits.

Table 8: Runtime and Net Profit comparisons of EDLF and Leaf_Node_Search on real Telecom dataset

#Records	Runtime(Seconds)		Net Profit($)	
	Leaf_Node_ Search	EDLF	Leaf_Node_ Search	EDLF
1000	0.2501	0.04659	104380	115090
2000	0.4113	0.08432	195330	218110
3000	0.6397	0.11457	256020	278430
4000	0.8511	0.15039	350790	387020
5000	1.2637	0.21231	432880	466580

Table 8 presents the runtimes and Net Profits comparison of EDLF and Leaf_Node_Search on the real Telecom dataset. Each time, a set of samples are given as input to the algorithms and the total runtime to determine Destination for each of those instances and the total Net Profit obtained from the set of samples are recorded. It has been noted that proposed method is notably outperforming Leaf_Node_Search in all cases. On average, the runtime for finding the optimal solution for one instance with Leaf_Node_Search is 0.0002269 seconds and with EDLF is 0.0000414 seconds which is 5.48 times less than that of the former method. For all cases, the total Net Profit produced by EDLF is 9.39% more than that of generated by the Leaf_Node_Search. In the case when 5000 records are given as input, the total Net Profit generated by EDLF is 7.78% more than that of produced by the Leaf_Node_Search. On average, Net Profit yielded for one instance by Leaf_Node_Search is $92.33 and by the EDLF is $101.40 which is 9.82% more than that of the one produced by Leaf_Node_Search.

When a minimal number of actions are required to be performed and those actions are not required on non-flexible attributes, then, high Net Profit values can be expected. In the cases where there is no possibility to shift instances from a leaf node with class C_k to another one with class C_j (j<k) EDLF generates more Net Profit than that of generated by the Leaf_Node_Search.

4.4 Comparison with ensemble tree based state-of-the-art methods

Computation times of proposed method are compared against two ensemble tree based state-of-the-art methods i.e. Suboptimal search [13] and Integer Linear Programming (ILP) [12] and the results are presented in Table 9. To the best of our knowledge, these two are the leading state-of-the-art methods for optimal action mining.

These state-of-the-art techniques find a set of actions that can convert the input sample from an undesired status to the desired one using additive tree Model (ATM) which are based on an ensemble of trees. With the same parameter setting, experiments are performed on nine datasets from LibSVM[1] website and UCI which are used in Suboptimal search and ILP's original experimental analysis. For runtime comparison, from each dataset 30 samples are randomly picked and given as input and the average computation time to provide the solution for each of these 30 samples is recorded. The main motive of our EDLF is to search and find a Destination among all the Candidates. Hence, no other leaf node produces profit more than that of the one produced as output. Therefore, we focus on comparing the methods with respect to the computation times. These ensemble tree based methods also provide the solution by treating the problem as 2-class. However, while processing the multi-class datasets, these techniques follow a tricky method by using the one-verses-remaining all strategy which eventually perceives the problem as 2-class. They set the desired class as Positive and all the remaining classes as Negative.

Table 9: Runtime comparison of proposed method and two state-of-the art methods on nine benchmark datasets

Dataset	#Instances	# Attributes	# Classes	Accuracy (%)	AUC	Runtime (Seconds)			$\frac{M3}{M2}$ (%)	$\frac{M3}{M1}$ (%)
						ILP (M1)	Suboptimal search (M2)	EDLF (M3)		
A1a	32561	123	2	84.4	0.85	7.54	4.03	7530×10^{-5}	998.67×10^{-5}	1868.48×10^{-5}
Australian	690	14	2	85.6	0.89	108.03	1.87	95×10^{-5}	0.88×10^{-5}	50.80×10^{-5}
Breast cancer	683	10	2	95.0	0.95	31.01	1.46	89×10^{-5}	2.87×10^{-5}	60.96×10^{-5}
DNA	3386	180	3	92.8	0.94	35.39	4.43	6478×10^{-5}	183.05×10^{-5}	1462.3×10^{-5}
Heart	270	13	2	81.8	0.84	5.71	2.33	300×10^{-5}	52.54×10^{-5}	128.7×10^{-5}
Ionosphere	351	34	2	89.2	0.92	48.92	2.91	237×10^{-5}	4.84×10^{-5}	81.44×10^{-5}
Liver disorders	345	6	2	63.2	0.56	31.71	0.17	2.8×10^{-5}	0.088×10^{-5}	16.47×10^{-5}
Mushrooms	8124	22	2	100.0	1.00	3.69	2.71	4835×10^{-5}	1310.3×10^{-5}	1784.1×10^{-5}
Vowel	990	12	11	81.5	0.95	68.15	1.99	185×10^{-5}	2.71×10^{-5}	92.97×10^{-5}
				85.9 (Average)	0.88 (Average)	340.15 (Total)	21.9 (Total)	0.195668 (Total)	284×10^{-5} (Average)	616.24×10^{-5} (Average)

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets// datasets/

On average, the computational time of EDLF is 0.284×10^{-2}% of ILP and 0.61624×10^{-2}% of Suboptimal search. In most of the experiments, EDLF's computational times are drastically less than those of ILP and Suboptimal search. This fact is observed especially on the datasets Liver disorders (0.000088×10^{-2}% and 0.01647×10^{-2}%), Australian (0.00088×10^{-2}% and 0.0508×10^{-2}%), Breast cancer (0.00287×10^{-2}% and 0.06096×10^{-2}%).

When the dataset size and dimensionality increase, the computational time of EDLF has increased to some extent and still very less than those of ILP and Suboptimal search. The runtimes on the datasets A1a (0.99867×10^{-2}% and 1.86848×10^{-2}%), DNA(0.18305×10^{-2}% and 1.4623×10^{-2}%) and Mushrooms(1.3103×10^{-2}% and 1.7841×10^{-2}%) depict this observation. As the Suboptimal search and ILP applies their postprocessing method on an ensemble of trees built using Random forest classifier, they require more computation time to achieve the objective. On the other hand, the proposed method postprocesses and extracts knowledge from a single tree.

From all the experimental results, it can be concluded that the proposed method outperforms two different classes of state-of-the-art methods. However, all the other methods talk about the conversion of an instance from class C_2 to C_1 only and treat as a 2-class problem. Proposed EDLF tries for more optimization for acquiring maximum profits and fits well for multi-class problems.

5 Conclusions and Future Scope

The ultimate goal of the enterprises is to improve their profits. In this regard, they depend on the machine learning models for acquiring automated profit maximizing knowledge. In brief, in this paper, we have strictly focused on two issues. The first one is, providing a profit maximization solution for enterprises with multi-class customers who are facing losses and uncertainty due to attritors and other insignificant profitable classes of customers. Second, providing the solution while achieving remarkable computational efficiency. Our novel algorithm EDLF serves both the purposes. On the other hand, the limited research done in the past has treated this profit maximization problem as a 2-class problem and also there was no focus on computational achievement. We started our work by building a probability estimation decision tree (PET) using customers' profiles. Then, the PET is represented as a compressed form as a BPM on which the entire profit maximization course of action is performed. Customers are supposed to be in the decreasing order of profitability viz., C_1, C_2, ..., C_n. If a customer falls into any leaf nodes of the PET which is not a Desirable leaf then, the proposed method searches and finds another optimal leaf which provides a maximum Net Profit. Our method suggests the cost-sensitive actions, to shift the customer from Source to Destination, such that the solution is most optimum. When it is not possible to migrate a customer from a leaf node representing class C_k to other leaf node representing class C_j (j=1,2,.., k-1), then, the proposed method tries to shift the instance to a leaf node representing class C_k that can provide more profit than that of the Source. Moreover, the proposed method provides tailored profit maximizing actions for each individual customer. Afterwards, we have discussed the solution for a 2-class problem with the help of a synthetic Telecom dataset. Then, for the 3-class scenario, we have discussed computing the Net Profit with an example. Finally, we have generalized and formulated a mathematical model to provide a solution for n-class applications where $n\geq2$.

Due to the merits like the potential use of array data structures, the best use of bitwise operations on the fixed length bit patterns, and the appropriate arrangement of the bit patterns and also the bit pattern vectors of the leaf nodes in a well-organized order, the computational achievement of EDLF is very impressive. The experiments performed on UCI datasets, real Telecom dataset, and other benchmark datasets prove that the proposed EDLF algorithm appreciably outperforms the other classes of state-of-the-art methods with respect to computational time and Net Profits.

As the next step, a study can be performed to improve the proposed algorithm such that the new approach can also find the class label/source leaf node information of the given input sample by using the BPM so that the induced PET can never be used in the postprocessing phase. This work can also be extended by incorporating ontological concepts and can be used for various other domains. We believe the proposed method can evolve as a remarkable method for profitable action extraction for the service providing sectors like Telecom, Banking, Internet, Retail, and online shopping, etc.

Acknowledgments

Sincere thanks are given to the anonymous reviewers for their valuable comments and suggestions.

References

[1] W. Kamakura, C. F. Mela, A. Ansari, A. Bodapati, P. Fader, R. Iyengar, R. Wilcox. Choice models and customer relationship management. Marketing Letters, 2005, 16(3–4): 279–291. http://dx.doi.org/10.1007/s11002-005-5892-2

[2] F. Reichheld. Prescription for cutting costs. Bain & Company, 2014

[3] Long Bing Cao, D. Luo and C. Zhang. Knowledge actionability: Satisfying technical and business interestingness. International Journal of Business Intelligence and Data Mining, 2007, 2(4):496-514

[4] G. Nie, W. Rowe, L. Zhang, Y. Tian and Y. Shi. Credit card churn forecasting by logistic regression and decision tree. Expert Syst Appl., 2011, 38:15273–15285. doi: 10.1016/j.eswa.2011.06.028

[5] Adnan Amin, Sajid Anwar, Awais Adnan, Muhammad Nawaz, Khalid Alawfi, Amir Hussain, Kaizhu Huang. Customer Churn Prediction in Telecommunication Sector using Rough Set Approach. Neurocomputing, 2017, 237:242–254. DOI: http://dx.doi.org/10.1016/j.neucom.2016.12.009

[6] Abbas Keramati, Hajar Ghaneei and Seyed Mohammad Mirmohammadi, Keramati. Developing a prediction model for customer churn from electronic banking services using data mining. Financial Innovation, 2016, 2:10. DOI 10.1186/s40854-016-0029-6

[7] Patrik Berger, Michal Kompan. User Modeling for Churn Prediction in E-Commerce. Intelligent Systems, IEEE, 2019, 34(2):44-52 March-April DOI: 10.1109/MIS.2019.2895788

[8] Thomas Verbraken, Wouter Verbeke and Bart Baesens. Profit optimizing customer churn prediction with Bayesian network classifiers. Intelligent Data Analysis, 2014, 18(1) : 3-24. DOI 10.3233/IDA-130625

[9] Quiang Yang, Jie Yin, Charles Ling and Rong Pan. Extracting Actionable knowledge using decision Trees. IEEE Transactions on Knowledge and Data Engineering, 2007,17(1) : 43-56. doi: 10.1109/TKDE.2007.9

[10] Nasrin Kalanat, Behrouz Minaei-Bidgoli. An optimized fuzzy method for finding action. Journal of Intelligent & Fuzzy Systems, 2016, 30(1):257-265. DOI:10.3233/IFS-151751

[11] Nasrin Kalanat, Shamsinejadbabaki, P. Saraee. A fuzzy method for discovering cost-effective actions from data. Journal of Intelligent & Fuzzy System, 2015, 28: 757–765. DOI:10.3233/IFS-141357.

[12] Zhicheng Cui, Wenlin Chen, Yujie He, Yixin Chen. Optimal Action Extraction for Random forests and Boosted Trees, In:Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015,179-188

[13] L. U. Qiang, Zhicheng Cui, Yixin Chen, Xiaoping Chen. Extracting optimal actionable plans from additive tree models. Frontiers of Computer Science, 1-14, 11(1), 2017, Pages 160-173. https://doi.org/10.1007/s11704-016-5273-4

[14] V. Zeithaml, R. T. Rust, K. N. Lemon. The customer pyramid Creating and serving profitable customers. California Management Review, 2001, 43(4), 118–142

[15] Reinartz, Werner, V. Kumar. The Mismanagement of Customer Loyalty. Harvard Business Review, 2002, 80 (7): 86-94, 125

[16] A. S. Dick, K. Basu. Customer loyalty: Toward an integrated conceptual framework. Journal of the Academy of Marketing Science, 1994, 22(2):99-113. https://doi.org/10.1177/0092070394222001

[17] R. Garland. Non-financial drivers of customer profitability in personal retail banking. Journal of Targeting, Measurement and Analysis for Marketing, 2002, Vol. 10:233, pp. 233-248. https://doi.org/10.1057/palgrave.jt.5740049

[18] E. Sivasankar, J. Vijaya. Hybrid PPFCM-ANN model:an efficient system for customer churn prediction through probabilistic possibilistic fuzzy clustering and artificial neural network. Neural Computing and Applications, May 2018, doi:10.1007/s00521-018-3548-4

[19] Alejandro Correa Bahnsen, Djamila Aouada, Björn Ottersten. A novel cost-sensitive framework for customer churn predictive modelling, Springer, Decision Analytics, 2015, 2(5). DOI 10.1186/s40165-015-0014-6

[20] E. Stripling, S. Vanden Broucke, K. Antonio, B. Baesens, M. Snoeck. Profit maximizing logistic model for customer churn prediction using genetic algorithms. Swarm and Evolutionary Computation, 2018, Vol. 40, Pages: 116-130. doi: 10.1016/j.swevo.2017.10.010

[21] E. Pednault, N. Abe, B. Zadrozny. Sequential Cost-Sensitive Decision Making with Reinforcement Learning. In: Proceedings of Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, 259-268, doi:10.1145/775047.775086

[22] H. Sebastiaan, E. Stripling, B. Baesens, S. Broucke, T. Verdonck. Profit driven decision trees for churn prediction. Computational Intelligence & Information Management, 2018. https://doi.org/10.1016/j.ejor.2018.11.072

[23] C. Gao, Y. Yao. Actionable Strategies in Three-way Decisions. Knowledge-Based Systems, 133(C), 141-155, 2017, doi: 10.1016/j.knosys.2017.07.001

[24] Long Bing Cao, Y. C. Zhao, H. F. Zhang, D. Luo, C. Q. Zhang, E. K. Park. Flexible frameworks for actionable knowledge discovery. IEEE Transactions on Knowledge and Data Engineering, 2010, 22(9): 1299–1312. DOI: 10.1109/TKDE.2009.143

[25] Long Bing Cao. Domain Driven Data Mining, IEEE Transactions on Knowledge and Data Engineering, 2010, 22(6),755-769. DOI: 10.1109/TKDE.2010.32

[26] Long Bing Cao. Actionable knowledge discovery and delivery. WIREs Data Mining and Knowledge Discovery, 2012, 2:149-163. doi:10.1002/widm.1044

[27] Z. Zhaojing, W. Regina, Z. Weihong, L. Shizhan, L. Dong, and J. Hao. Profit Maximization Analysis Based on Data Mining and the Exponential Retention Model Assumption with Respect to Customer Churn Problems. In: IEEE International Conference on Data Mining Workshop (ICDMW), 2015, 1093-1097. DOI: 10.1109/ICDMW.2015.84

[28] Nasrin Kalanat, K. Eynollah. Action extraction from social networks. Journal of Intelligent Information Systems, 2019, https://doi.org/10.1007/s10844-019-00551-2

[29] Tom Mitchell. Machine Learning and Data Mining. Communications ACM, vol. 42, no.11, Nov. 1999, 30-36.

[30] W. Xindong, J. Vipin Kumar, R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, N. Angus, Bing Liu, S. Philip, Y. Z. Zhou, S. Michael, D. J. Hand, D. Steinberg. Top 10 algorithms in data mining. Knowledge and Information Systems, 2008,14: 1–37, DOI 10.1007/s10115-007-0114-2

[31] F. J. Provost, P. Domingos. Tree induction for probability-based ranking. Machine Learning, 2003, 52(3):199–215. https://doi.org/10.1023/A:1024099825458

[32] Fayyad, M. Usama, M. Irani, B. Keki. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, .hdl:2014/35171 Proceedings of the International Joint Conference on Uncertainty in AI, (Q334 .I571 1993),1022-1027

[33] J. Huang, C. X. Ling. Using AUC and Accuracy in Evaluating Learning Algorithms. IEEE Trans on Knowledge and Data Engineering, 2005, 17(3), 299-310. DOI**:** 10.1109/TKDE.2005.50

[34] C. L. Blake, C. J. Merz. UCI Repository of Machine Learning, www.ics.uci.edu/~mlearn/mlrepository.html, 1998

[35] Q. Yang, J. Yin, X. Ling, T. Chen. Postprocessing Decision Trees to Extract Actionable Knowledge. In:Proceedings of the Third IEEE International Conference on Data Mining IEEE, 2003, 685–688. DOI**:** 10.1109/ICDM.2003.1251008