Pruning Based Interestingness of Mined Classification Patterns

Ahmed Al-Hegami
Department of Computer Science, University of Sana'a, Yemen

Abstract: *Classification is an important problem in data mining. Decision tree induction is one of the most common techniques that are applied to solve the classification problem. Many decision tree induction algorithms have been proposed based on different attribute selection and pruning strategies. Although the patterns induced by decision trees are easy to interpret and comprehend compare to the patterns induced by other classification algorithms, the constructed decision trees may contain hundreds or thousand of nodes which are difficult to comprehend and interpret by the user who examines the patterns. For this reasons, the question of an appropriate constructing and providing a good pruning criteria have long been a topic of considerable debate. The main objective of such criteria is to create a tree such that the classification accuracy, when used on unseen data, is maximized and the tree size is minimized. Usually, most of decision tree algorithms perform splitting criteria to construct a tree first, then, prune the tree to find an accurate, simple, and comprehensible tree. Even after pruning, the decision tree constructed may be extremely huge and may reflect patterns, which are not interesting from the user point of view. In many scenarios, users are only interested in obtaining patterns that are interesting; thus, users may require obtaining a simple, and interpretable, but only approximate decision tree much better than an accurate tree that involves a lot of details. In this paper, we proposed a pruning approach that captures the user subjectivity to discoverer interesting patterns. The approach computes the subjective interestingness and uses it as a pruning criterion to prune away uninteresting patterns. The proposed framework helps in reducing the size of the induced model and maintaining the model. One of the features of the proposed approach is to capture the user background knowledge, which is monotonically augmented. The experimental results are quite promising.*

Keywords: *Knowledge discovery in databases, data mining, decision tree, domain knowledge, interestingness, novelty measure.*

Received December 9, 2007; accepted March 30, 2008

1. Introduction

Classification is an important data mining task that analyzes a given training set and develops a model for each class according to the features present in the data. The generated model is used to classify unseen data tuples. There are many approaches to develop the classification model including decision trees, neural networks, nearest neighbour methods and rough set-based methods [4, 7].

Decision tree induction methods are the most widely used to construct classification model [4, 7, 12, 15, 21]. These methods partition the data recursively until all tuples in every partition have the same class value. The resultant model is a tree that is used for predicting the class label.

Many algorithms for inducing decision trees have been proposed in the literature (e.g., C4.5 [17], CART [2], SPRINT [19]), based on different attribute selection and pruning strategies. Most of these algorithms operate in a construction phase followed by pruning phase that make an impact on the time and efficiency of the algorithms. However, algorithms such as PUBLIC [4], BOAT [6] address such issues by constructing and pruning the tree in one stage. This process is established by pushing constraints such as accuracy and size into the decision tree in order to prune the tree dynamically and hence result in reduction the size and improvement the performance of decision tree.

Although these approaches require the user to provide constraints, the user background knowledge is not implicitly/explicitly stated. This lack of incorporating user Domain Knowledge (DK) and Previously Discovered Knowledge (PDK) into the tree induction process results in a decision tree which may be optimal in size and accuracy but may generate branches that are similar to the earlier discovered tree and hence does not reflect the user interest.

Commonly used techniques to discover interesting patterns in most KDD endeavours are partially effective unless combined with subjective measures of interestingness. Subjective measures quantify interestingness based on the user understandability of the domain [1, 8, 9]. Capturing the user subjectivity in dynamic environment requires a great deal of knowledge about databases, the application domain and the user's interests at a particular time [12, 13, 20]. Therefore, it is difficult for the user to analyze the discovered patterns and to identify those patterns that are interesting from his/her point of view.

In this paper, we propose a pruning approach that uses interestingness measures to reduce the volumes of discovered patterns. The subjective interestingness evaluates the patterns on the basis of novelty, actionability and unexpectedness measures. The quantification of subjective interestingness is based on the analysis and computation of the deviation of recently discovered rules with respect to known knowledge, i.e., DK and previously PDK, and the importance that the user gives to different types of deviations. The computed degree of interestingness is then compared with the user given threshold to report interesting rules to the user.

The proposed pruning strategy is applied to the tree that is already pruned by traditional pruning criteria, to reflect the user subjectivity and prune away the branches that do not meet the interestingness criterion and subsequently extract interesting patterns. The approach makes sure that the overall accuracy of the discovered model is not compromised.

2. Related Work and Motivation

One of very important issues that attracted the researchers in the area of machine learning is that of simplifying the decision tree. The simplest decision tree with the highest accuracy is the optimal goal to achieve. The need for simplicity and comprehensibility has introduced many algorithms that deal with various methods of pruning decision trees [5, 10, 15, 16, 17, 18]. The general aspect of these approaches is to obtain a smaller decision tree without incurring a high classification error as a consequence of obtaining such a smaller tree.

Although getting an optimal-sized, an accurate decision tree is desirable, some researchers have argued that increasing the size of the tree will result in monotonically decreasing the accuracy while on the other hand a small decrease in accuracy will result in a dramatic decreasing of the size of the tree [5]. An example of such argument is the constructing of a decision tree for the legality of a white-to-move position in chess. It is demonstrated that, while a decision tree with 11 leaves is 100% accurate, a sub tree with only 5 leaves is 99.57% accurate. Thus, more than 50% of the size of accurate tree leads to less than 0.5% reduction of the accuracy [5, 18].

There are many approaches that push the accuracy and size constraints into the decision tree in order to prune the tree dynamically and hence result in reduction the size and improvement the performance of decision tree [5, 6, 18]. In [6], an algorithm called BOAT is proposed. It is a scalable algorithm with the ability to update the tree incrementally over time. The approaches presented in [5, 18] push the accuracy and size constraints into the decision tree in order to prune the tree dynamically. They proposed a classifier called PUBLIC that integrates building and pruning in one stage. In PUBLIC, a node in not further expanded in the construction stage of decision tree if it is determined that it is certain to be pruned in the subsequent pruning stage.

Although these approaches reduce the size of the tree and improve the accuracy of the classifiers, the PDK and the user DK are not implicitly/explicitly stated. This lack of incorporating Known Knowledge (KK) into the decision tree pruning results in a decision tree which may be optimal in size and accuracy but does not reflect the user interest and hence providing the user with knowledge that may not be interesting. In order to obtain interesting patterns, the user has to be involved by providing the system with his/her feeling and/or general impressions about the domain. These feelings can be pushed into the mining algorithm in the form of constraints to discover the interesting patterns. This motivated us to incorporate our interestingness criterion into the pruning stage of decision tree to form a constraint that guarantees small, accurate and interesting decision tree. The advantage of integration interestingness filter into the pruning stage of decision tree is that the size of the tree built can be radically reduced. In addition, the discovered knowledge will reflect the user's requirement by allowing him/her to specify the type of knowledge required. Certainly, it is interested to the analyst to apply the interestingness measure to the pruning stage in order to compare the results at different periods of time and finally the analyst can make a decision to select the appropriate set.

3. Interestingness Measures of Discovered Patterns

Data mining research has shown that we can measure a rule's interestingness using both objective and subjective measures [7, 11, 12, 13]. Objective measures involve analyzing the rule's structure, predictive performance, and statistical significance. Such measures include accuracy, support and confidence. However, objective measures are insufficient for determining a discovered rule's interestingness. Subjective measures are needed [7, 11, 12, 13]. Subjective interestingness, the topic of this article, has three main measures:

- Unexpectedness: rules are interesting if they are unknown to the user or contradict the user's existing knowledge (or expectations).
- Actionability: rules are interesting if users can do something with them to their advantage.
- Novelty: rules are novel if they add knowledge to the user prior knowledge.

Although novelty, actionability and unexpectedness of the discovered knowledge are the basis of the subjective measures, their theoretical treatment still remains a challenging task. Actionability is the key concept in most applications. Actionable rules let users do their jobs better by taking some specific actions in response to the discovered knowledge.

Actionability, however, is an elusive concept because it is not feasible to know the space of all rules and the actions to be attached to them. Actionability is therefore is implicitly captured by novelty and unexpectedness.

In this work we introduce a pruning strategy that is based on the computation of a comprehensive interestingness measure. The interestingness measure quantifies the unexpectedness and novelty by involving the user background knowledge and the previously discovered knowledge. The approach computes the interestingness on the basis of computation of deviation of discovered rules with respect to the domain knowledge and previously discovered rules. Subsequently the user determines a certain threshold value to report interesting rules. The architecture of the proposed approach is shown in Figure 1.

At time t_i, database D_i is pre-processed and subjected to the mining algorithm, resulting into the construction of decision tree K_i. The proposed filter processes K_i, in light of knowledge that the user has already acquired because of his experience and expertise in the domain and the previously discovered knowledge, together termed as KK stored as a rule base. We assume that all rules in KK (PDK+DK) have the same knowledge representation.

In order to evaluate the interestingness of a branch (rule), we use syntactic matching technique to compute the deviation at conjunct level. When the deviation of conjuncts of a rule has been detected, it can be generalized to rule level. The following sections describe the approach for computing these two types of deviation.

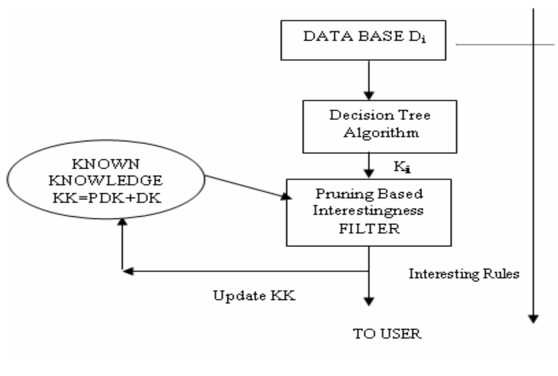

Figure 1. Interestingness as a pruning filter for decision tree algorithm.

3.1. Deviation at Conjunct Level

Degree of deviation of a conjunct c_i with respect to another conjuncts c_j is determined based on the result of comparison of the attributes, operators, and attribute values of the two conjuncts.

Definition 1: let c_1 and c_2 be two conjuncts (A_1 O_1 V_1) and (A_2 O_2 V_2) respectively. The deviation of c_1 with respect to c_2 is defined as follows [11]:

$$\Delta(c_1, c_2) = \begin{cases} 0, & \text{if } A_1 = A_2, O_1 = O_2 \text{ and } V_1 = V_2 \text{ | Indentical conjuncts.} \\ \gamma_1, & \text{if } A_1 = A_2, O_1 = O_2 \text{ and } V_1 \neq V_2 \text{ | Test value has changed.} \\ \gamma_2, & \text{if } A_1 = A_2, O_1 \neq O_2 \text{ and } V_1 = V_2 \text{ | Test condition has changed.} \\ \gamma_3, & \text{if } A_1 = A_2, O_1 \neq O_2 \text{ and } V_1 \neq V_2 \text{ | Condition and value have changed} \\ \gamma_4, & \text{otherwise. | Not compatible} \end{cases} \quad (1)$$

where $\gamma_1, \gamma_2, \gamma_3,$ and γ_4 are user specified numeric quantities, such that $0 \leq \gamma_1 \leq \gamma_2 \leq \gamma_3 \leq \gamma_4 \leq 1$. The possibilities of deviation at conjunct level as mentioned in definition 1, are exhaustive. The degree of deviations vary from 0, which indicates no deviation between the two conjuncts to maximum (γ_4), which indicates incompatible conjuncts. The intermediate degrees of deviation ($\gamma_1, \gamma_2, \gamma_3$) indicate varying extents of deviations between two conjuncts. Note that user subjectivity is captured by seeking parameters ($\gamma_1, \gamma_2, \gamma_3, \gamma_4$) from the user as per the importance that the user gives to different types of deviations.

3.2. Conjunct Set Deviation

Since both antecedents and consequents are considered to be sets of conjuncts, it is necessary to define the deviation Ψ between two conjunct sets. The deviation $\Psi(S_1,S_2)$ between two conjuncts sets is obtained by computing the deviation between the member conjuncts and subsequently combining the deviation between the member conjuncts. The following definition is the basis for computation of deviation at conjunct set level.

Definition 2: let S_1 and S_2 be two sets of conjuncts. We define a matching function $\Psi(S_1, S_2)$ as follows:

$$\Psi(S_1, S_2) = \begin{cases} 0, & \text{iff } |S_1| = |S_2|, \forall c_i \in S_1, \exists c_j \in S_2 \\ & \quad \text{such that } \Delta(c_i, c_j) = 0 \text{ | Indentical sets.} \\ 1, & \forall c_i \in S_1, \neg\exists c_j \in S_2 \text{ such that } \Delta(c_i, c_j) = 1 \text{ | Totally different.} \\ \beta, & \text{otherwise | Intermediate} \end{cases} \quad (2)$$

where

$$\beta = \frac{1}{|S_1|} \sum_{c_i \in S_1} \min_{c_j \in S_2} \Delta(c_i, c_j)$$

As per definition 2, $\Psi(S_1, S_2) = 0$ indicates that S_1 and S_2 are identical, $\Psi(S_1,S_2) = 1$ indicates the extreme deviation and the computed value of β quantifies an intermediate degree of deviation. The value of β is computed as a linear combination of the minimum deviation of each conjunct of S_1 with respect to S_2 divided by the number of conjuncts of S_1.

4. Computation of Interestingness Measure

Since both antecedents and consequents of rules are considered to be sets of conjuncts, it is necessary to compute the deviation between conjunct sets. In order to compute the interestingness of a rule, the deviation between conjunct sets is computed by combining the deviation at conjunct level. Subsequently, the interestingness of a rule can be computed by combining the deviations between the antecedents and consequents. Interestingness of a rule R_1 with respect to another rule R_2 is calculated as follows:

Definition 3: let R_1: $Å_1 \rightarrow C_1$ and R_2: $Å_2 \rightarrow C_2$ be two rules and $S_1 \in Å_1$ and $S_2 \in Å_2$ be two sets of conjuncts ($A_1O_1V_1$ and $A_2O_2V_2$ respectively). The Interestingness measure of R_1 with respect to R_2 denoted by $\Omega\,(R_1,R_2)$ is computed as follows:

$$\Omega(R_1,R_2)=\begin{cases}\Psi(S_1,S_2) & \text{such that } \Psi(S_1,S_2)>\Phi \text{ and } \Psi(C_1,C_2)=0.\\ 1 & \text{iff } \Psi(C_1,C_2)=1.\\ 0 & \text{othetwise}\end{cases} \qquad (3)$$

A rule R_1 is considered to be interesting with respect to R_2 if either the deviation of both antecedents are greater than the user threshold value (Φ) or the rules are not compatible. A rule is not compatible if the deviation of the consequents is 1.

5. Integration of Interestingness Criterion into Pruning Stage

In the previous section, we explained our interestingness criterion. More detailed descriptions of this framework can be found in [9]. This framework can be adapted to solve the pruning problem in that, the node of a tree along with a branch associate with it, can form a conjunct and a rule can be formed by traversing a path form the root to a certain class value. In order to prune a decision tree based on interestingness filter, the interestingness has to be quantified. To achieve this goal, we have to measure the deviation at each branch of the tree. More specifically, let T denote the accurate decision tree constructed using a decision tree algorithm. Our aim is to reduce the size of T tree as well as pruning T in such away that only interesting nodes are maintained. This is can be achieved by removing the branches which have interestingness degree bellow a given threshold value. Once the branches are removed, the node under consideration becomes a leaf and holds the most frequent class of the tuples used to create the branch.

Suppose that, at time t_0, the user may have an intuitive feeling that a particular attribute or a combination of attributes may lead to a particular class value. This domain knowledge can be provided by the user in the form of $C_1 \cap C_2 \cap \ldots \cap C_k \rightarrow K_i$, where C_j is a conjunct of the form *A Op V*, where A is an attribute, *Dom(A)* is the domain of *A*, and $V \in Dom(A)$, $op \in \{=, \prec, \succ, \geq, \leq\}$ and K_i is a class value. There may be domains in which the user does not have any knowledge about domain especially at time t_0, in this case, the DK may not be required at this time, however, the user's knowledge may be increased at time t_i, $i \neq 0$.

Our proposed interestingness based pruning method removes sub trees of fully-grown tree by quantifying the deviation degree of each path in the tree with respect to DK and PDK. The grown tree is traversed in a breadth first manner until a leaf is reached. Then, the path from the root to that leaf is considered for evaluation. This path is compared using a matching technique against DK and PDK. If this path found to be interesting (the deviation degree is higher than a user given threshold value), the leaf node is marked "interesting", otherwise, prune this node away if its father has no more other child and replace its father node with the most frequent class in the dataset used to create this branch. If the father of this node has other branches, simply prune this leaf away. This process traverse each level of the tree looking for the leaves to be pruned based on interestedness taking into consideration the nodes which are already marked "interesting", until no more nodes in the tree with deviation degree higher than a user given threshold. Once a pruned tree is found to be interesting at time t_0, any future dataset at time t_i can build a decision tree and prune it in the same way, however, the matching technique considers the decision tree built at time t_0 as PDK.

6. Example

For better understanding to our proposed integration of interestingness filter into pruning stage of decision tree, let us consider the tree in Figure 2 that is constructed using a traditional decision tree algorithm.

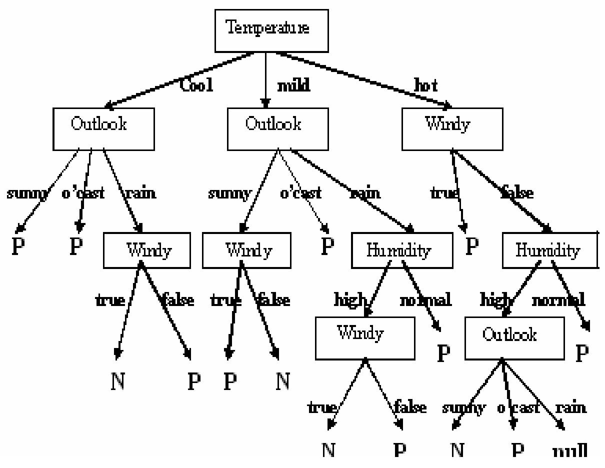

Figure 2. A complicated decision tree.

Suppose that, the user specified the following domain knowledge as DK:

$$\begin{array}{c}\text{Temperature=hot} \cap \text{windy=true} \rightarrow \text{P}\\ \text{Outlook=sunny} \cap \text{windy=false} \rightarrow \text{N}\end{array} \qquad (4)$$

The interestingness Integrated pruning method uses the known knowledge (DK/PDK) as constraints to filter the decision tree in order to maintain up to-date tree. It works by traversing the tree level by level until a leaf node is encountered. Given the tree in Figure 2, the approach starts by examining the nodes of 1st level which are outlook, outlook, and windy. Since there is no leaf node in this level, further level is examined. In the 2nd level there are four leaves which have the corresponding conjuncts, outlook=sunny→*P*, outlook =o'cast→*P*, outlook =o'cast→*P*, and windy =true→*P*. These conjuncts along with their paths from root are compared against DK and PDK (the nodes marked "interesting" with respect to the threshold value, using our matching techniques to compute the deviation degree of this path from the root to that node. In our example, the 1st path to be examined is temperature=cool ^ outlook=sunny → *P*. This path is compared against DK which contains two rules. The deviation degree of the path with respect to the 1st rule in DK is found to be (0.1+1)/2=0.55 and with respect to 2^{nd} rule in DK is found to be 1 because the class values are different. In this case, the minimum deviation is selected to be the deviation degree for this path. Suppose that the user specify 50% as a threshold value, then this node is marked with "interesting".

Similarly the path that involves temperature = cool ^ outlook =o'cast → *P* is examined next and the computed deviation with respect to DK as well as to the node marked "interesting": temperature=cool^ outlook =sunny → *P* is found to be 0.55 and 0.05 respectively. The minimum value is selected to be the deviation degree for this path, which is lower than threshold value and thus it is pruned away from the tree. This process continues until all paths in the tree lead to interesting nodes.

7. Implementation and Experimentation

The proposed approach is implemented and tested using our experimental datasets and real life datasets. The system is built using c programming language. A C4.5 algorithm is modified to incorporate our pruning based interestingness measure. Since, there are no other approaches available, which handle the pruning based interestingness; we could not perform any comparison against our approach. We will explain our framework through the following experiments:

7.1. Experiment 1

The first experiment was run on the German credit database. This dataset created by Professor Hans Hofmann, University of Hamburg, appears on the UCI ML. data repository at [3]. This dataset contains 20 attributes (7 numerical, 13 categorical) and 1000 instances, recording situations with a binary class. The two types of classes determine either a customer is good or bad depending on other attribute values. Based on our framework we obtain a set of rules to represent a user's domain knowledge. With the same confidence and support of first experiment we run CBA against 333, 333 and 334 instances representing instances arrive at T_1, T_2, and T_3 respectively. In addition when our approach is applied to the tree constructed, different categories of knowledge are generated as shown in Table 1.

Table 1. The discovered rules at time T_1, T_2, and T_3.

Time	No. of tuples	No. of discovered rules	Interesting rules	Confirming rules
T_1	333	117	68	49
T_2	333	118	40	78
T_3	334	133	48	85
Total	1000	368	116	210

Consider the following set of rules discovered by C4.5 as follows:

- R_1: Installments = none ^ Personal_status = single_male ^ Status = no_account →good.
- R_2: Telephone = yes ^ Purpose = radio_tv ^ Status = no_account →good.
- R_3: Personal_status = single_male^ Savings_account = less100DM^ Credit_history = all_paid_duly →bad.
- R_4: Debtors = none^ Credit >= 9340.5 → bad.
- R_5: Employment = seven_years ^ Status = no_account →good.
- R_6: Housing = own^ Installments = none^ Purpose = new_car^ Duration >= 14.5^ Status = less_200DM →bad.
- R_7: Installments = none ^ Employment = over_seven ^ Status = no_account →good.
- R_8: Job = skilled^ Installments = bank^ Debtors = none^ Savings_account = less100DM^ Status = 0DM →bad.
- R_9: Telephone = no ^ Credit < 9340.5 ^ Purpose = used_car →good.
- R_{10}: Job = skilled ^ Property = car ^ Purpose = new_car ^ Status = 0DM →bad.

The user specified Domain Knowledge (DK) as follows:

- Job = skilled ^ Debtors = none ^ Duration < 14.5 ^ Status = no_account→good.
- Telephone = no^ Credit >= 9340.5 →bad.
- Savings_account = over1000DM^ Credit < 9340.5 →good.
- Housing = own^ Purpose = education^ Duration >= 14.5 →bad.

- Foreign = yes^ Debtors = none^ Purpose = new_car →bad.

If the user provides the interestingness threshold = 0.5, different types of rules are generated when the proposed approach is applied. Table 2 shows interesting/conforming rules generated at time T_1 along with the degree of deviation in both DK and PDK.

Table 2. The categorization of rules discovered at time T_1.

Rules	**Deviation w.r.t DK**	**Deviation w.r.t PDK**	**Interestingne ss degree**	**Categorization of knowledge**
R_1	0.5	1	0.5	Interesting
R_2	0.5	1	0.5	Interesting
R_3	1	1	1	Interesting
R_4	0.33	1	0.333	Conforming
R_5	0.275	1	0.275	Conforming
R_6	0.700	1	0.700	Interesting
R_7	0.500	1	0.500	Interesting
R_8	1	0.6	0.64	Interesting
R_9	1	1	1	Interesting
R_{10}	1	0.4	0.4	Conforming

Similarly, the same process is performed at time T_2 and T_3 taking into account the interesting rules discovered at time T_1.

7.2. Experiment 2

The second experiment was run on the census-income database that appears at [12]. The dataset contains 6 continuous, 8 nominal attributes 48842 instances, mix of continuous and discrete (train=32561, test=16281), and two classes determine either a person's income is <= 50K or >50K. We run C4.5 against 8 partitions of the dataset representing instances arrive at time T_1, T_2, T_3, T_4, T_5, T_6, T_7 and T_8 respectively. We assume that the user does not have any background knowledge about domain and hence the discovered knowledge is compared against PDK only. Also assume that a user specifies the following values for γ_1, γ_2, and γ_3 as follows: $\gamma_1=0.1$, $\gamma_2=0.5$, $\gamma_3=0.9$.

Table 3 shows the size of the tree induced using C4.5 and its size when using our approach at different point in time.

7.3. Experiment 3

The objective of the third experiment is to compare the size and overall accuracy of the classifier generated by our approach with C5.0 and CBA. The experiment was performed using four datasets. We partitioned them into two increments D_1 and D_2 assumed to have arrived at times t_1 and t_2 respectively.

The decision tree of Figure 3 is generated with C5.0 using Tic-Tac-Toe dataset results into 19 rules. Then, we run our approach against D_2 taking into account PDK that is discovered by *C5*.0 using D_1. The dataset D_1 was run against *C5*.0 for two reasons. Firstly, to show that our approach can take the output of any classifier and consider it as PDK. Secondly, our approach discovers many rules at time t_1 because the PDK is null. The PDK rule base used by our approach is shown in Table 4 and the decision tree constructed by our approach, which generated 14 rules, is shown in Figure 4.

Table 3. The size of the tree before and after pruning at time T_1 to T_8.

Time	**Number of tuples**	**Before pruning**	**After pruning**	**Conforming rules**
T_1	4000	345	42	303
T_2	4000	279	16	263
T_3	4000	318	15	303
T_4	4000	387	37	350
T_5	4000	327	25	302
T_6	4000	348	23	325
T_7	4000	298	19	279
T_8	4561	338	49	319
Total	32561	2640	196	2444

Table 4. The discovered knowledge at time t_1 (PDK) using C5.0 (D_1).

R_1	t7 = b ^ t8 = b → class = positive
R_2	t6 = b ^t7 = x → class = positive
R_3	t7 = b ^ t8 = o → class = positive
R_4	t5 = x → class = positive
R_5	t4 = x ^ t7 = x → class = positive
R_6	t5 = b ^ t9 = o → class = negative
R_7	t5 = o → class = negative

Furthermore, the same experiment is performed on different datasets. Table 5 shows the size and accuracy of C5.0, CBA and our approach on different datasets. Note that the threshold values Φ maintained in the Table 5 is the threshold values that are used in our approach to generate the best classifier in terms of accuracy and size.

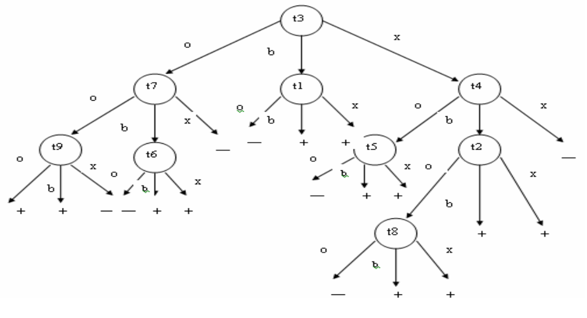

Figure 3. Decision tree constructed using C5.0 (D_1+D_2).

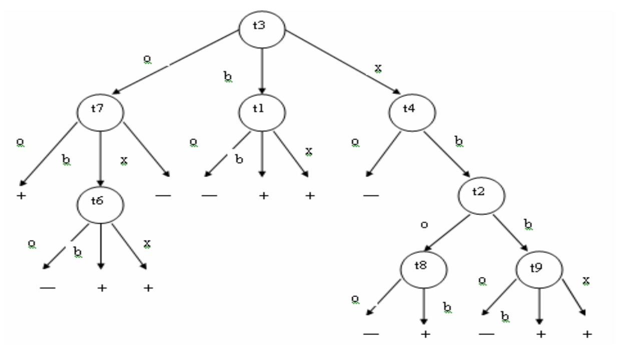

Figure 4. Decision tree constructed using our approach (D_2).

Table 5. The size and accuracy of C5.0/CBA (D_1+D_2) and our approach (D_2) with different datasets.

Dataset	Our Approach				C5.0		CBA	
	PDK (D_1)	Size (D_2)	Accuracy %	Φ	Size	Accuracy %	Size	Accuracy %
Zoo	6	5	98	0.8	8	99	9	100
Tic-Tac-Toe	7	14	84	0.8	19	83	28	100
Monk	4	4	100	0.8	6	93.5	26	98.46
Vote	3	3	96	0.8	2	96.7	38	99.97

8. Conclusion

In this paper, we propose a pruning approach that is based on self-upgrading interestingness filter used in the pruning stage of decision tree. The filter is based on quantification of subjective interestingness of the Currently Discovered Rules (CDR) with respect to DK, and PDK. We capture the user subjectivity by asking for thresholds to determine the degree of interestingness and the importance that the user is given for each deviation. The approach is implemented and evaluated using public datasets and has shown encouraging results.

References

[1] Bhatnagar V., Al-Hegami S., and Kumar N., "A Hybrid Approach for Quantification of Novelty in Rule Discovery," *in Proceedings of International Conference on Artificial Learning and Data Mining (ALDM'05)*, Turkey, pp. 39-42, 2005.

[2] Breiman J., Friedman H., Olshen A., and Stone J., *Classifications and Regression Trees*, New York, Chapman and Hall, 1984.

[3] Data Mining, http:// www.comp.nus.edu.sg /~dm2/index.html, 1998.

[4] Duda O., Hart E., and Stork G., *Pattern Classification*, John Wiley & Sons, 2002.

[5] Garofalakis M., Hyun D., Rastogi R., and Shim K., "Efficient Algorithms for Constructing Decision Trees with Constraints," *Bell Laboratories Technical Memorandum*, 2000.

[6] Ghrke J., Ganti V., Ramakrishnan R., and Loh Y., "BOAT Optimistic Decision Tree Construction," *in Proceedings of the ACM SIGMOD International Conference on Management of Data*, USA, pp. 13-23, 1999.

[7] Han J. and Kamber M., *Data Mining: Concepts and Techniques*, Harcourt India Private Limited, 2001.

[8] Hegami S., Bhatnagar V., and Kumar N., "Novelty as a Measure of Interestingness in Knowledge Discovery," *in International Journal of Information Technology*, vol. 2, no. 1, pp. 37-49, 2005.

[9] Hegami S., Bhatnagar V., and Kumar N., "Novelty Framework for Knowledge Discovery in Databases," *in Proceedings of 6thInternational Conference on Data War1ehousing and Knowledge Discovery (DaWaK 2004)*, Spain, pp. 48-57, 2004.

[10] Karimi K. and Hamilton J., "Temporal Rules and Temporal Decision Trees: A C4.5 Approach," *Technical Report CS-2001-02*, Canada, 2001.

[11] Liu B. and Hsu W., "Post Analysis of Learned Rules," *in Proceedings of the 13th National Conference on AI (AAAI'96)*, pp. 1194-2001, 1996.

[12] Liu B., Hsu W., and Chen S., "Using General Impressions to Analyze Discovered Classification Rules," *in Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD 97)*, USA, pp. 73-83, 1997.

[13] Liu B., Hsu W., Mun F., and Lee Y., "Finding Interesting Patterns Using User Expectations," *Technical Report TRA7/96*, 1996.

[14] Mingers J., "An Empirical Comparison of Pruning Methods for Decision Tree Induction," *Computer Journal of Machine Learning*, vol. 4, no. 2, pp. 213-22, 1987.

[15] Quinlan J., *Induction of Decision Trees*, Academic Publishers, 1986.

[16] Quinlan R., "Simplifying of Decision Trees," *International Journal of Machine Learning Studies*, vol. 27, no. 3, pp. 389-401, 1987.

[17] Quinlan R., *C4.5: Programs for Machine Learning*, San Mateo, 1993.

[18] Rastogi R. and Shim K., "PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning," *in Proceedings of the 24th*

International Conference on Very Large Data Bases (VLDB), New York, pp. 24-27, 1998.

[19] Shafer J., Aggrawal R., and Mehta M., "SPRINT: A Scalable Parallel Classifier for Data Mining," *in Proceedings of 22nd VLDB Conference*, San Antonio, pp. 962-969, 1996.

[20] UCI KDD Archive, http:// kdd.ics.uci.edu/, 2005.

[21] Utgoff E., "Incremental Induction of Decision Tress," *Computer Journal of Machine Learning*, vol. 4, no. 2, pp. 161-186, 1989.

Ahmed Al-Hegami received his BSc degree in computer science from King Abdul Aziz University, Saudi Arabia, and his Master and PhD in computer application from Jawaharlal Nehru University, New Delhi, India.