


# A Novel Incremental Framework for Building Classifier Using Constraint Class Association Rules

B. Subbulakshmi, Dept. of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, India

C. Deisy, Dept. of Information Technology, Thiagarajar College of Engineering, Madurai, India

S. Parthasarathy, Dept. of Applied Mathematics & Computational Science, Thiagarajar College of Engineering, Madurai, India

 <https://orcid.org/0000-0001-7439-6878>

## ABSTRACT

Associative classification (AC) performs much better than other traditional classifiers. It generates a huge number of class association rules (CARs). Since users are interested in the subset of rules, constraints are introduced in the generation of CARs. Real-world databases are record-based in which data is continuously added which demands incremental mining. Hence, constraint class association rules (CCAR) is mined from incremental data. To limit the number of rules and to remove the duplicate rules, redundant rule pruning and duplicate rule pruning techniques are applied. To improve the accuracy of the classifier, the rule selection using principality metric has been applied and the classifier is constructed with rules possessing high principality. Then, classifier is evaluated using single rule and multiple rule prediction methods and the accuracy of the proposed classifier are measured. Experimental results show that the accuracy of the proposed classifier is relatively higher when compared to other algorithms.

## KEYWORDS

Associative Classification, Class Association Rules, Constraints, Prediction, Pruning

## 1. INTRODUCTION

### 1.1 Associative Classification

Associative Classification (AC) (Vishwakarma et al. 2013; Phan-Luong, 2013) is an integration of association and classification methods aims to build a classifier describing the classes of the input training data set. Class Association Rules (CARs) (Mabu et al. 2011) are basically used to build a classification model for which prediction defines the relationship between the itemsets and the class

DOI: 10.4018/IJIRR.316125

\*Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

labels. It is mainly used in the medical dataset. Based on the generated rules, it is possible to identify population at high risk for a particular disease.

Constructing an associative classifier involves two steps. Given the training dataset, class association rules are generated first. Then, a small subset of high- quality rules are obtained using pruning methods and an accurate classifier or model is built for the training data. However, in large or correlated data sets, rule mining may yield a huge number of classification rules. Hence, pruning techniques, in particular, support- based pruning, are exploited to reduce the complexity of the extraction task.

## 1.2 Motivation

A class association rule is a special case of association rule where the rule consequent contains only a class label (Nguyen et al., 2016). The basic objective of any user in mining class association rules (CARs) is to determine a complete set of CARs in such a way that it satisfies user-specified minimum support and minimum confidence thresholds. In literature there exists several methods and techniques have been proposed to manage this issue in CARs. In fact, the CAR mining has been widely used in various practical domains, namely tourism management ((Rong et al., 2012; Leung et al., 2013), social media security (Zhang et al., 2007), healthcare (Nahar et al., 2013), and education sectors (Luna et al., 2015). However, it is also observed by previous researchers that the results obtained as a result of CARs mining has redundancy.

Many real time applications are dynamic in nature such that the data is incrementally added to the database. The major problem with incremental mining is updating knowledge which was mined from the original database. Methods for mining frequent itemsets from incremental datasets have been proposed by (Cheung et al. 1996; Hong et al. 2001; Le et al. 2011, 2012 ; Vo et al. 2014). These methods reduce the execution time and memory usage compared to those obtained from rescanning the original dataset. Thus the techniques of incremental frequent itemset mining can be integrated into constraint CAR Mining. Hence, the incremental method generates all CARs satisfying the given constraint whenever the original dataset is added with new records. However, there exists the possibility of generation of duplicate and redundant CCARs. It can be addressed by applying post processing techniques such as pruning and selection of rules.

In the post-processing step, the Constraint Class Association Rules (CCARs) can be pruned, ranked and then quality rules are selected in order to build a compact and high quality classifier. The classifier is evaluated by measuring the number of correct classifications made for a given set of test instances. Finally, the prediction of test instances is made by selecting the single rule or multiple rules satisfying the antecedent of test instances.

## 1.3 Constraint Class Association Rule (CCAR) Mining

Class Association Rules (CARs) (Mabu et al. 2011) are basically used to build a classification model for prediction. These rules define the association between the itemsets and the class labels and it is useful in various application domains. However, the users are interested in only the subset of CARs, for instance, those rules that contain at least one itemset from a user-defined set of itemsets. Giving constraints on itemsets reduces the number of generated CARs and decrease the search space. Thereby, it improves the performance of the mining process. Moreover, constrained CARs help to discover interesting or useful rules specific to the end user. For example, while analyzing the patient data to identify those people who are at the high risk of HIV infection, analyst may focus on the rules that include information such as age, marital status, and the gender information in the rule antecedents. Similarly, biologists focus on rules involving new drugs to understand the effectiveness of new treatment strategies for cancer treatment applications.

Constraint-based mining (Nguyen & Vo 2014; Nguyen et al. 2015; 2016) provides

- Flexibility to user to express their interestingness using constraints

- Reduce search space by exploring only the constrained patterns rather than looking for all possible patterns. It also improves effectiveness and efficiency of mining methods.

The constraints can be considered in the form of Boolean expressions over the presence of itemsets in the antecedents of classification rules. For example, to find the most popular shoes in winter, the constraint can be set to season as winter. The general approach for constraint based mining tries to extract all frequent patterns and then validate those patterns according to the constraints. Otherwise, the constraints are integrated while looking for frequent patterns. This will generate only the constrained frequent patterns.

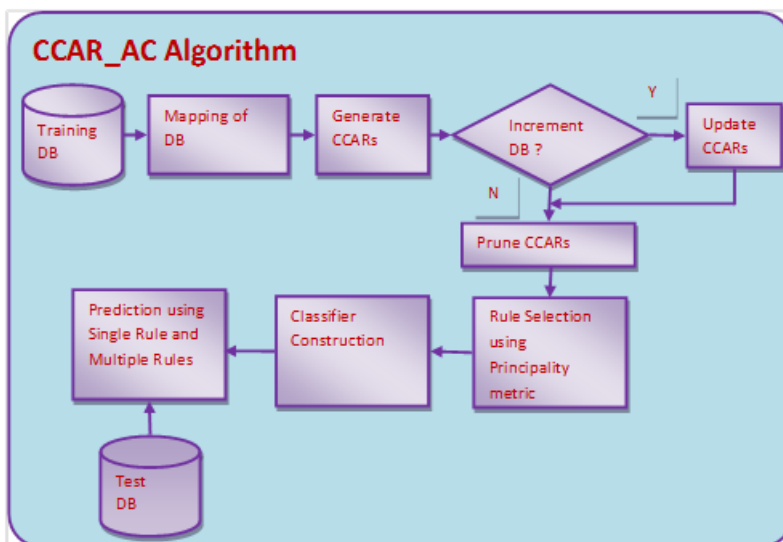
Hence, in this work, Constraint Class Association Rule based Associative Classifier (CCAR\_AC) algorithm is proposed and the framework of the proposed CCAR\_AC algorithm is given in Figure 1. The objective of this CCAR\_AC algorithm is to mine the Constraint Class Association Rules incrementally and apply the rule pruning and selection methods to build a compact and accurate associative classifier. For prediction, single and multiple rule prediction methods are used to evaluate the accuracy of the proposed classifier.

The algorithm generates CCARs incrementally whenever the original database is added with new records using the proposed Incremental CCAR (ICCAR) algorithm. For pruning the Constraint Class Association Rules (CCARs), redundant rule pruning is applied along with the removal of duplicate nodes. Then, the rule ranking is done using the support, confidence, length of the rule. The principality (Chen et al. 2014) of the ranked rules is then measured and the rules are sorted based on the principality. Classifier is constructed using these sorted rules and test instances are classified using single rule prediction and multiple rule prediction methods.

## 2. RELATED RESEARCH

Constraint based class association rule mining algorithms were developed to restrict the number of rules according to the user specific constraints. There are two approaches for the generation of CARs with constraints: Pre-processing and Post processing approaches. In Post processing method (Yafi

Figure 1.  
Framework of the Proposed CCAR\_AC Algorithm



et al. 2012), the CAR mining methods suggested in Nguyen et al. (2013) and Vo & Le (2008) can be used to generate all CARs from the given input and then filtering of CARs is done against the constraints. The issue in this method is the generation of large number of CARs and lots of CARs has to be tested in post processing step. CAR-Miner-Post (Nguyen & Vo 2014) is an example for Post Processing approach. In pre processing method, input dataset is filtered for so that it contains only constraint itemsets. The benefits of this method are size of the input dataset is greatly reduced and an improvement of mining time is obtained. However, this method has to generate all CARs which are complex one and degrade the performance of mining. To resolve the issues in Pre and Post processing methods, a method was proposed in Nguyen et al. (2015) in which constraints are integrated into the rule generation phase.

These algorithms work with static datasets. However, real world databases are dynamic i.e set of new transactions are added to the original input. When the class association rules are derived from the input data, some redundant rules (i.e rules that can be inferred from others) are generated. These rules could be eliminated because they may occupy high storage and leads to poor classification accuracy. Selecting high quality rules is a challenge while constructing the classifier.

The accuracy of the associative classification depends on finding the relevant and appropriate rule for predicting the value of the unknown test instance. CBA algorithm proposed in Ma & Liu (1998) uses single rule prediction method. In this method, rule with higher precedence and matches the attributes of test instance is selected for predicting the class label of the test instance. CMAR algorithm proposed in Li et al. (2001) uses multiple rules that matches the given test instance for prediction instead of a single rule. These rules are grouped according to the class value and the strength of each group is measured based on the correlation levels of the elements in that group.

Some of the limitations observed in the existing approaches are:

- In case of incremental data, the Constraint Class Association Rules (CCAR) Algorithm (Nguyen et al. 2015; 2016) has to be invoked for the whole dataset i.e original records and the additional records in order to update the rules.
- Existing algorithm generates huge number of CCARs along with duplicate nodes (Nguyen et al. 2015;2016).
- Selecting the interesting and appropriate rules for constructing the classifier is a challenge.

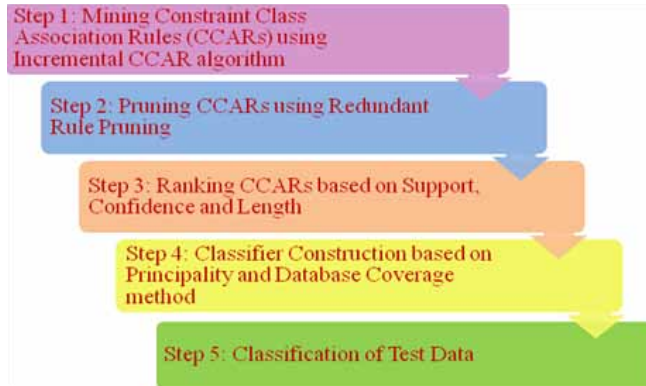
These limitations influence the need of proposed CCAR\_AC algorithm. The proposed algorithm integrates the constraints during rule generation and the generated CCARs are updated whenever the database is added with new records by controlling the number of re-scans over the database by adapting the safety threshold measure proposed in Hong et al. (2001). Then, rule pruning and selection methods are applied to restrict the number of rules and to build a compact classifier. For rules selection, principality metric proposed in Chen et al. (2014) is applied to select the high quality rules. Finally, the built classifier is evaluated using single and multiple rule prediction methods.

### 3. PROPOSED APPROACH

The proposed Constraint Class Association Rule based Associative Classifier (CCAR\_AC) algorithm consists of five modules as shown in Figure 2. Each module of the CCAR\_AC algorithm is described below.

The detailed description about the phases of the proposed Constraint Class Association Rule based Associative Classifier (CCAR\_AC) algorithm is given below.

Figure 2.  
Steps in Proposed CCAR\_AC Algorithm



### 3.1 Mining CCARs using Incremental CCAR Algorithm

Let  $D$  be a dataset consists of 'n' attributes  $\{A_1, A_2, \dots, A_n\}$  and  $|D|$  records or objects in which each record has an object identifier called OID. Let  $C = \{C_1, C_2, \dots, C_k\}$  be a list of class labels. The distinct value of an attribute  $A_i$  and class  $C$  are denoted by  $a_{im}$  and  $c_j$ , respectively. For instance, an item is considered to be attribute- value pair denoted by  $\langle A_i, a_{im} \rangle$ . For example,  $\langle A, a1 \rangle$  and  $\langle C, c3 \rangle$  are items and an itemset is described as collection of items given by  $\langle A, a1 \rangle, \langle B, b2 \rangle, \langle B, b3 \rangle, \langle C, c2 \rangle$ .

A Constraint\_Itemset is a user-defined set of itemsets in which each itemset is said to be a constrained itemset. For example,  $\text{Constraint\_Itemsets} = \{ \langle A, a1 \rangle, \langle B, b1 \rangle, \langle C, c1 \rangle \}$ .

A Class Association Rule  $R$  is of the form  $\text{itemset} @ C_j$  where  $C_j \in C$  is a class label. A rule  $R$  satisfies Constraint\_Itemsets, if its antecedent (itemset) contains atleast one itemset in Constraint\_Itemsets.

An incremental algorithm called ICCAR proposed in Subbulakshmi & Monisha (2016) is used for generating CCARs from incremental data. The proposed algorithm makes use of a tree structure called Incremental Constraint Class Rule-tree (ICCR-tree) (Subbulakshmi & Monisha 2016) for incremental data.

#### 3.1.1 ICCR-tree structure

The ICCAR algorithm proposes the ICCR-tree structure for the input data. In the ICCR-tree, every node consists of the following fields in addition to the itemsets:

- i.  $\text{Obidset}_1, \text{Obidset}_2, \dots, \text{Obidset}_k$ : It is the set of object identifiers of the itemsets of each class. Here,  $k$  is the number of classes in the dataset.
- ii. Position: Stores the position of the class with the maximum cardinality of  $\text{Obidset}_i$ .
- iii. Total: stores the sum of cardinality of all  $\text{Obidset}_i$  where  $i$  varies from 1 to  $k$
- iv. Tsupport: It stores the temporary support value of itemsets for updating the ICCR-tree (in case of incremental data).

The itemset is converted to the form  $\text{att} \times \text{values}$ , where

Att: a list of attribute.

Values: a list of values, each of which contains one attribute in Att.

Consider the sample dataset given in Table 1 with three attributes and six objects.

For example, itemset  $X = \langle A, A2 \rangle, \langle B, B1 \rangle$  is denoted as  $X = 3 \times A2B1$ . A bit representation is used for itemset attributes. Attributes AB can be represented by 11 in bit representation and the value

Table 1.  
 Sample dataset

Object Identifier	Attribute A	Attribute B	Attribute C	Class
1	A2	B1	C1	Yes
2	A2	B1	C1	Yes
3	A2	B1	C1	No
4	A1	B2	C2	Yes
5	A1	B2	C2	No
6	A1	B1	C2	Yes

of these attributes is 3. The itemset  $X = \{A, A2, B, B1\}$  is contained in objects 1, 2 and 3, of which, two objects belong to class Yes. Therefore, the node that contains itemset X has the form  $3 \times A2B1(12,3)$  in which  $Obidset_1 = \{1,2\}$  (i.e. objects 1 and 2 contain both itemset X and class Yes),  $Obidset_2 = 3$  (i.e. object 3 contain both itemset X and class No),  $pos = 1$  (denotes class with highest count) and total = 3.

Rule R, which is generated from a node in the ICCR-tree, has the form itemset @  $c_{pos}$  with  $Supp(R) = |Obidset_{pos}|$  and  $Conf(R) = (|Obidset_{pos}|)/(total)$ .

For example, node  $3 \times A2B1(12,3)$  generates rule  $R: \{A, A2, B, B1\} @ Yes$  (i.e. if  $A = A2$  and  $B = B1$ , then Class = Yes) with  $Supp(R) = 2$  and  $Conf(R) = 2/3$ .

The first level node in the ICCR-tree is a tuple containing the following:

<attr-values, ( $Obidset_1, \dots, Obidset_k$ ), position, total>

The first level of ICCR-tree stores constrained itemset and frequent 1-itemsets. The constrained itemset is joined with other frequent 1-itemsets to generate constrained frequent k-itemsets for  $k \geq 2$ . At successive levels, the tree contains only those itemsets that satisfy the constraints. The successive levels store the difference in Obidsets, when the constrained nodes are added to the tree. Finally, the tree is mined to generate all Constraint Class Association Rules (CCARs) incrementally. While the nodes of the tree are joined together, nodes with the same attributes and nodes with parent-child relationship need not be combined. Each node contains attribute values, differences of object identifiers, support count of each class, the position of the class and temporary support to handle incremental data. This ICCR-tree stores only those itemsets that satisfy the constraints. Hence, it reduces the memory storage and improves the mining time.

When new transactions are added to the original database, a measure called safety threshold ' $f$ ' given in Hong et al. (2001) is used to maintain and update the Constraint Class Association Rules. This measure controls the number of re-scans made over the original database in order to update the previously mined class rules.

The safety threshold ' $f$ ' is computed using the formula given in equation (1):

$$f = \frac{(S_U - S_L) \times |D|}{1 - S_U} \quad (1)$$

where,  $|D|$  denotes the number of records in the original database and  $S_U$  and  $S_L$  are the upper and lower support thresholds respectively.

If the number of added transactions exceeds the safety threshold value, then ICCR-tree is to be reconstructed for both original and inserted database. Otherwise, the ICCR-tree is updated (i.e. temporary support, support values and object identifiers are updated). If the updated support value of the node satisfies the minimum support, then that node is marked for future reference. Finally, the ICCR-tree can be mined to extract all possible CCARs.

The proposed ICCAR algorithm (Subbulakshmi & Monisha 2016) has the following advantages:

- Itemset constraints reduce the search space by extracting only the constrained patterns and minimize the number of rules.
- The use of safety threshold measure controls the number of re-scans over the original database.
- Since the nodes of ICCR-tree store the difference of object identifiers, memory usage of each node is reduced.

### 3.1.2 Incremental CCAR (ICCAR) algorithm

**Input:** Original Dataset  $D$ ,  $minSup$ ,  $minConf$ ,  $Constraint\_Itemsets$

**Output:** CCARs, ICCR-tree

**Procedure Initialization (Dataset  $D$ ,  $minSup$ ,  $Constraint\_Itemsets$ )**

1. Find the constraint nodes using itemset constraints and  $minSup$
2. Scan the dataset to find frequent 1-itemsets using  $minSup$  and store them as frequent nodes
3. Initialize the level- 1 tree ( $L$ ) with constraint nodes and frequent nodes
4. *Invoke ICCAR ( $L$ ,  $minSup$ ,  $minConf$ )*

**Procedure ICCAR (Level  $L$ ,  $minSup$ ,  $minConf$ )**

- 1.
- 2.
- 3.
- 4.
5.  $CCARs = null;$
6. For all  $I_i \in L.Children \ \& \ I_i \in \text{constraint node}$  do
7. **GENERATE-RULE ( $I_i$ ,  $minConf$ )**
8.  $T_i = null;$
9. For all  $I_j \in L.Children$  with  $j > i$  do
10.  $O.attribute = I_i.attribute \ | \ I_j.attribute;$
11.  $O.values = I_i.values \ \cup \ I_j.values;$
12.  $O.obidset = I_i.obidset \ \setminus \ I_j.obidset;$
13. set  $O.Tsupport = 0;$
14. Update  $O.obidset$ ,  $O.position$ ,  $O.total$
15. if new node  $O$  satisfies  $minSup$  then
16.  $T_i = T_i \ \cup \ O;$
17. *Invoke Recursive\_ICCAR ( $T_i$ ,  $minSup$ ,  $minConf$ )*

**Procedure Recursive\_ICCAR (Level  $L$ ,  $minSup$ ,  $minConf$ )**

18. For all  $I_i \in L.Children$  do
19. **GENERATE-RULE ( $I_i$ ,  $minConf$ )**
20.  $T_i = null;$
21. For all  $I_j \in L.Children$  with  $j > i$  do
22.  $O.attribute = I_i.attribute \ | \ I_j.attribute;$
23.  $O.values = I_i.values \ \cup \ I_j.values;$
24. Set  $O.Tsupport = 0;$

```

25. O.obidset = Ii.obidset \ Ij.obidset;
26. Update O.obidset, O.position, O.total
27. If new node O satisfies minSup then
28. Ti = Ti ∪ O;
29. Invoke Recursive_ICCAR (Ti, minSup, minConf)
Procedure GENERATE-RULE (l, minConf)
30. conf = |l.Obidsetl.position| / l.total;
31. if conf ≥ minConf then
32. CCARs = CCARs ∪ {l.itemset → Cposition }
    
```

### 3.1.3 Incremental method for mining CCARs

For incremental mining, the ICCAR algorithm is modified with the following changes:

The ICCAR algorithm has to use two support thresholds ( $S_L$  and  $S_U$ ): For rule generation, upper support threshold ( $S_U$ ) is used and for updating the support of itemsets, lower support threshold ( $S_L$ ) is used.

- The GENERATE-RULE procedure is called with three parameters: l, Upper Support ( $S_U$ ) and minConf. To generate rules, this procedure checks whether the support and confidence of node satisfy  $S_U$  (with respect to original database) and minConf.
- For building the ICCR-tree, the procedures INITIALIZATION, ICCAR and Recursive\_ICCAR use the  $S_L$  value (with respect to original database)

### 3.1.4 Algorithm for updating the ICCR tree for incremental datasets

**Input:** ICCR-tree with L as root node, Original Database D, Incremental Database D', upper support threshold  $S_U$ , lower support threshold  $S_L$  and minConf

**Output:** CCARs satisfying the  $S_U$  and minConf from D+D'

**Procedure Inc\_CCAR ()**

1. Compute safety threshold for the original database using equation (1)
2. If  $f == 0$  or  $f < |D'|$  then
3. Invoke ICCAR algorithm to mine CCARs in D+D' using  $S_L$
4. Compute safety threshold  $f = \frac{(S_U - S_L) \times |D| + |D'|}{1 - S_U}$
5. Else
6. Clear the Obidset information in all the nodes in the ICCR-tree
7. Update the node information in level 1 (L1) of the ICCR-tree
8. Mark those nodes whose information has been changed and support satisfies the minSup ( $S_L$ )
9. **Invoke UpdateICCRTree (L1)**
10. **Invoke GENERATE-UPDATED-RULES** to generate CCARs from the updated tree
11.  $f = f - |D'|$
12.  $D = D + D'$

**Procedure UpdateICCRTree (Level L)**

13. For all  $I_i \in L.Children$  do
14. If  $I_i$  is marked then
15. For all  $I_j \in L.Children$  with  $j > i$  do



16. If  $I_j$  is marked then
17. Let node O be the direct child of  $I_i$  and  $I_j$
18. if O exists then
19. if O.itemset.count = 1 then
20. O.obidset =  $I_i.obidset \setminus I_j.obidset$ ;
21. else
22. O.obidset =  $I_j.obidset \setminus I_i.obidset$ ;
23. O.Tsupport =  $I_i.Tsupport - O.obidset.count$ ;
24. if O.Tsupport > 0 then
25. O.support = O.support + O.Tsupport;
26. if O.support  $\geq S_L \times (|D| + |D'|)$  then
27. mark the node O
28. **Invoke UpdateICCRTree( $I_i$ )**

**Procedure GENERATE-UPDATED-RULES( $L, S_v, minConf$ )**

29. For all  $I_i \in L.Children$  do
30. Compute the support of  $I_i$
31. If support ( $I_i$ )  $\geq S_v \times |D|$  then
31. Compute the confidence for  $I_i$
32. If confidence of  $I_i$  satisfies minConf then
33. CCARs = CCARs  $\cup \{ I_i.itemset \rightarrow class \}$
34. **Invoke GENERATE-UPDATED-RULES ( $I_i, S_v, minConf$ )**

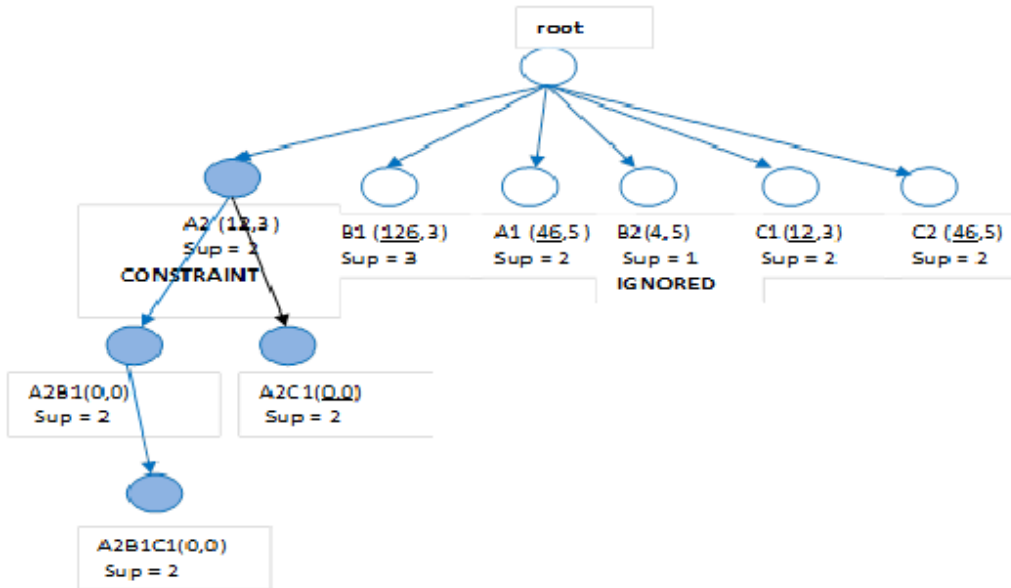
Procedure Initialization is invoked to find the constrained nodes using itemset constraints and frequent nodes containing frequent 1-itemsets by satisfying the minimum support. The constraint nodes are inserted at the first level of the tree along with the frequent 1-itemsets. Then, the constrained nodes are combined with all frequent nodes to generate successive levels of the ICCR –tree.

The procedure ICCAR takes each node and joins it with all other nodes by checking the condition that the attributes of these nodes are not same to generate a child node O. For this child node, it then computes its attribute-values, difference of object identifiers, position and total values. It also checks whether the support of new child node O satisfies minSup. Then, the new child node O is added to  $T_i$ . Then, the Recursive\_ICCAR procedure is called with this new set of child nodes ( $T_i$ ). The procedure GENERATE-RULE is used to generate a rule from the nodes of ICCR-tree by checking the rule's confidence with the minConf value.

Figure 3 shows the generated ICCR-tree for the sample dataset given in Table 1 with minsup = 2. The ICCR-tree is mined for rules that satisfy the minimum confidence. In this example, attribute A=A2 is set as a constraint and it is inserted into the ICCR-tree as first level nodes. The dataset is scanned and frequent 1-itemsets are also inserted in the first level of the tree. Since attribute B2 does not satisfy the minimum support, it is ignored. The constraint node (A= A2) is combined with other nodes containing frequent 1-itemsets to generate only constrained nodes in the successive levels.

For incremental mining, initially the CCARs are mined from the original database and the safety threshold value is calculated based on the number of records in the original database. If this value is equal to zero or less than the number of new transactions in the increment database, then the procedure ICCAR is called using  $S_L$  for generating rules for both original and increment databases and the safety threshold is updated. Otherwise, the incremental CCAR algorithm updates the frequent and infrequent nodes in the ICCR-tree by calling the updateICCARTree procedure and generates updated rules by calling GENERATE-UPDATED-RULES procedure. The safety threshold value is updated as  $f = f - |D'|$ . Finally, the database is updated so that, it contains both records of original and inserted records.

Figure 3.  
 ICCR-tree for the sample dataset in Table 1



### 3.2 Pruning CCARs using Redundant Rule Pruning

The generated Constraint Class Association Rules (CCARs) are pruned using a redundant rule pruning technique. The concept of redundant rules is defined in Vo & Le (2008) is given as:

Given a rule  $r_i$  in the set of rules mined from a dataset  $D$ . The rule  $r_i$  is called a redundant rule, if there is another rule  $r_j$  in the set of CARs such that  $r_j$  is a sub-rule of  $r_i$ , and  $r_j$  has higher precedence over  $r_i$ . Based on this definition, a method as defined in Nguyen et al. (2012) is used to remove the redundant rules i.e. if a rule  $r$  has a confidence of 100%, then all the other rules that are generated later than  $r$  and having  $r$  as a sub-rule, are redundant. Hence, the rule pruning removes the redundant rules from the set of rules mined.

### 3.3 Ranking CCARs based on Support, Confidence, Length of the Rule

The rules are sorted based on rule ranking procedure. The rule ranking procedure given in Chen et al. (2014) considers a rule  $R_i$  over  $R_j$ , if the confidence of  $R_i$  is greater than that of confidence of  $R_j$ . When the confidence is equal, it considers the rule with higher support. If the support is also same, then the length of the rule is considered. The longer rules are preferred over shorter rules because, the longer rule includes more constraints. Thus, the rules with high significance and necessity are retained to build the classifier.

### 3.4 Classifier Construction based on Principality

Confidence based rule generation methods generate all rules satisfying minconf as the candidate rules. To classify a new test instance, the candidate rule with the highest confidence is chosen. But, confidence alone cannot be used as the best criterion to choose a rule in classification. Hence, as in Chen et al. (2014), the completeness of the rule is given by: A rule of the form  $R: X@C$ , the completeness of  $R$  is the proportion of objects in class  $C$  that are correctly predicted by the rule  $R$ :  $Comp(R) = |X_C|/|C|$ .

Completeness is the proportion of instances that are predicted by the rule, while confidence is the fraction of correct predictions made by the rule. Confidence measures the predictive accuracy of a rule, whereas completeness measures the coverage or applicability of a rule to a specific class label.

Considering both the accuracy and coverage of a rule for a given class, a Principality metric for each CAR of the form  $R: X \rightarrow C$  is given as:

$$\text{Principality} = \text{Confidence} \times \lambda - (1 - \lambda) \times \text{Completeness} \quad (2)$$

i.e.  $P(R,C) = \lambda \times \text{conf}(R) + (1 - \lambda) \times \text{comp}(R) = \frac{\lambda \times |X_C|}{|X|} + (1 - \lambda) \times \frac{|X_C|}{|C|}$  where,  $|C|$  is the number of objects belonging to class  $C$  in the training data set,  $|X|$  is the frequency count of the condition of  $R$ , and  $|X_C|$  is the co-occurrence count of rule  $R$ 's condition  $X$  and consequent  $C$ .  $0 \leq \lambda \leq 1.0$  is a weight specified according to the user preference.

The principality metric provides richer information than the confidence metric. This is especially useful, when multiple rules with close degrees of confidence satisfy a test instance. The metric evaluates the accuracy and coverage of rule and it has a higher influence on the classifier performance.

Hence, the principality measure is calculated for each rule. For an associative classifier with multiple class association rules, the confidence and completeness of each rule contribute towards determining the performance of the classifier. However, the performance of the classifier does not solely depend on these metrics, and also on how the rules are chosen to construct the associative classifier. Thus, the database coverage technique is applied to construct the classifier using principality measure.

Therefore, a classifier is constructed by finding the rule with higher principality and then, it is inserted into the classifier until all the database objects are covered by at least one rule. After that, all the training objects covered by the rule will be removed from the training set and the remaining candidate rule's principality is recalculated.

### 3.5 Classification of Test Data

This phase assigns a class label to a new test instance using the trained associative classifier.

Let  $T_t$  be the test dataset. For each object  $t_t \in T_t$  in the test dataset, find all rules that match the given object. If there exists only one rule and that fully matches  $t_t$ 's antecedent, then the class label in the rule consequent is assigned to  $t_t$ ; otherwise, select the rule with the highest precedence to classify  $t_t$ . If there is no rule matching  $t_t$ , then the default class, i.e. the dominant class in training dataset is assigned to it. Two types of predictions are applied to classify test instances.

#### 3.5.1 Single Rule prediction

In single rule prediction, the first rule with higher principality, that matches the test instance, is taken into consideration. The rule that matches first has higher principality than the other rules. Hence, the test instance is assigned with the class label of the matched rule.

#### 3.5.2 Multiple Rule Prediction

All the rules covering the test instance are selected and principality is used to weigh the predictions made by each of these rules. The majority of this weighted prediction is selected as the prediction for the test instance. Rules matching the test instance are selected and principality is used as the weight to select the appropriate class. The weight of each prediction is calculated by summing up the principality values for different classifications from the rules that cover the test instance.

Thus, in multiple rule prediction, the sum of the principality is measured for each class and the test instance is assigned with the class having higher principality value. For instance, if three rules match the test instance, then the rules are grouped based on their class and the weight of each group is calculated i.e. for every group, the sum of principality values of all rules is computed. Finally, the

test instance is assigned with the class having a higher weight. The accuracy measures the ability of the classifier to correctly classify the unseen data objects and it is defined as:

$$Accuracy = \frac{\text{Number of Instances correctly classified}}{\text{Total number of Instances}} \quad (3)$$

### 3.6 Constraint Class Association Rule Mining with Post-processing

The Incremental CCAR algorithm updates the mined CCARs, whenever the original database is added with new records. And then, the Incremental CCAR algorithm has been extended with the additional procedures to implement rule pruning, selection, and classifier construction phases.

**Input:** Root node L of the updated-ICCR tree after the increment database, minConf

**Output:** Set of rules in the classifier

**Invoke procedure PRUNE-RULE (L, R)**

**Invoke procedure CONSTRUCT-CLASSIFIER (R, S, T)**

**Procedure PRUNE-RULE (Level L, RuleSet R)**

1. For all  $I_i \in L.children$  do
2. For all  $I_j \in L.children$  with  $j > i$  do
3. Node O be the direct child of  $I_i$  and  $I_j$
4. Compute the confidence of node O
5. If confidence (O) == 100% then
6. Generate rule from O and add it to RuleSet R
7. Delete all the child nodes of O
8. Else if confidence (O) < minConfidence
9. No rule is generated
10. Else if confidence (O) < minConfidence( $I_i$ )
11. No rule is generated
12. **Invoke PRUNE-RULE( $I_i$ , R)**

**Procedure CONSTRUCT-CLASSIFIER(RuleSet R, New-RuleSet S, Training Data T)**

13. For all  $R_i \in R$  do
14. Rank rules in R based on support, confidence and the length of the  
     rule
15. Calculate the principality for each rule in R using equation 5.2
16. While R is not empty or T is not empty do
17. Select the rule  $r^*$  with the highest principality
18. If the rule  $r^*$  classifies at least one instance correctly
19. Add the rule to new RuleSet S
20. Delete  $r^*$  from R
21. Delete all instances covered by  $r^*$  from T

The PRUNE-RULE procedure is used to eliminate all redundant rules. Every node is traversed by breadth-first traversal and the node satisfying minimum support and confidence is checked for 100% confidence. If so, the child nodes are deleted from the tree. The remaining nodes are mined for further processing and a subset is fed to the CONSTRUCT-CLASSIFIER procedure to build an associative classifier using rule ranking and selection methods. The selection method computes principality metric for each rule and sorts them based on their highest value of the principality. Only those rules with the highest principality value are then used to build a classifier.

## 4. RESULTS AND DISCUSSION

The proposed associative classifier CCAR\_AC has been tested using five datasets namely, Hungarian Heart Disease, German Credit, Breast Cancer, Mushroom and Diabetes dataset. These datasets are downloaded from UCI machine learning repository (<http://mllearn.ics.uci.edu>), and are pre-processed according to the implementation of the proposed algorithm. For the continuous-valued attributes, the data are discretized into bins and the discretized values are used. Pre-processing and discretization are done using WEKA tool, an open source software package for machine learning. The description of each dataset is given in Table 2.

### 4.1 Results for Incremental Mining of CCAR

The proposed approach checks whether the original database is added with new records or not. If the original database is not added with new records, then the constraint class association rules are mined from the original database only. Otherwise, the mined rules are updated, whenever the database is added with new records using the Incremental CCAR algorithm.

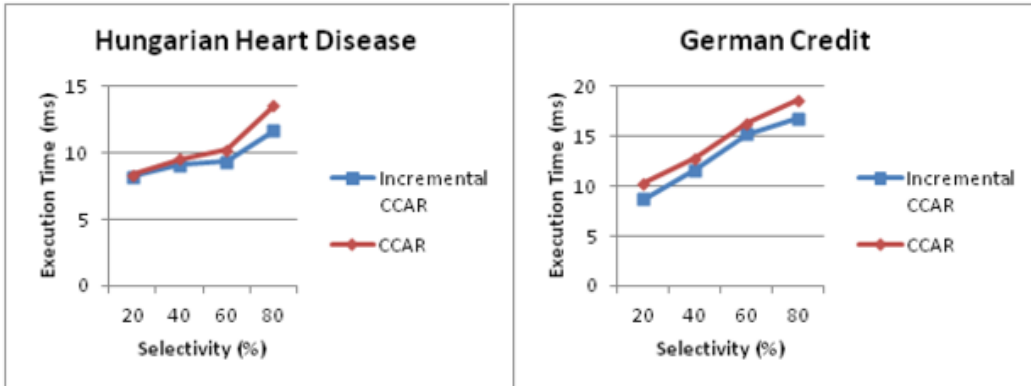
In this experiment, comparison of execution time is made between the CCAR and Incremental CCAR algorithms with minimum support of 5% and minimum confidence of 25%. The algorithms are executed with different selectivity values ranging from 20 to 80. Selectivity is defined as the ratio of number of constraint itemsets selected to the total number of itemsets. If there are 20 items, the selectivity of 10% implies that 2 items are selected as constraints. The total run time of the incremental algorithm for five runs is measured with  $S_U = 5\%$  and  $S_L = 3\%$  by considering two records for each increment and it is compared with a non-incremental algorithm with  $S_U = 5\%$ . For example, in the heart disease dataset, the original database size is varied from 262 records to 270 records. The CCAR algorithm (non incremental) is executed five times by considering a different sized original databases with 262, 264, 266, 268 and 270 records for each execution and the run time of each execution is added. Similarly, the Incremental CCAR algorithm is executed five times. For the first execution, the algorithm considers that the original database contains 260 records and two records are added as an increment. Then, the runtime of incremental CCAR algorithm is noted. For the second execution, the original database contains 262 records and another two records are added. The run time of incremental algorithm for each execution is summed up to compare with the batch - processing method (non-incremental algorithm). Experiments are done to show the execution time of Incremental CCAR and CCAR algorithms for different selectivity values. The increment size varies for different datasets. Depending upon the number of records inserted, the proposed Incremental CCAR algorithm either updates the information in the ICCR tree to generate the updated CCARs or rebuilds the tree for both original and inserted database by rescanning the original database.

For example, consider the heart disease dataset. As shown in Figure 4, the execution time of Incremental CCAR is 11.65ms whereas, for CCAR, it is 13.54ms for a selectivity value of 80%.

Table 2.  
 Description of datasets

Dataset	Instances	Classes	Class Distribution	Attributes	Items
Hungarian Heart Disease	270	2	(150,120)	13	52
German Credit	1000	2	(700,300)	7	30
Breast Cancer	699	2	(458,241)	9	18
Diabetes	768	2	(500,268)	8	35
Mushroom	8124	2	(4208,3916)	21	115

Figure 4.  
 Execution time of CCAR, incremental CCAR in the Heart and German dataset with the constraint selectivity



Therefore, the speedup achieved is 1.16ms. However, for a selectivity of 20%, the CCAR and Incremental CCAR algorithm give only minimal variation in execution time.

From the Figures 4 – 6, it is depicted that the incremental CCAR algorithm gives minimal execution time compared to a CCAR algorithm for Heart disease, German credit and Mushroom datasets. However, for Diabetes and Mushroom datasets, CCAR algorithm (batch-processing method) gives minimal execution time compared to an incremental CCAR algorithm for some selectivity values.

On an average, for a heart disease dataset, the execution time of CCAR is 10.4ms and for Incremental CCAR, it is 9.6ms. Therefore, the speedup achieved is 1.09ms. Similarly, the speed-ups achieved for German, Breast, Diabetes and Mushroom datasets are 1.11, 0.98, 1.03 and 1.1ms.

#### 4.2 Performance Analysis of proposed CCAR\_AC algorithm

During the mining or updating of rules, the proposed approach applies the rule pruning method to remove the redundant rules. Then, for constructing the classifier, the pruned rules are ranked and then evaluated using principality metric. The rules with higher principality values are used to build a classifier. Hence, the proposed classifier makes use of only a subset of rules for classification, due to the application of rule selection and pruning techniques. The proposed approach eliminates the

Figure 5.  
 Execution time of CCAR, Incremental CCAR in the Breast and Diabetes dataset with the constraint selectivity

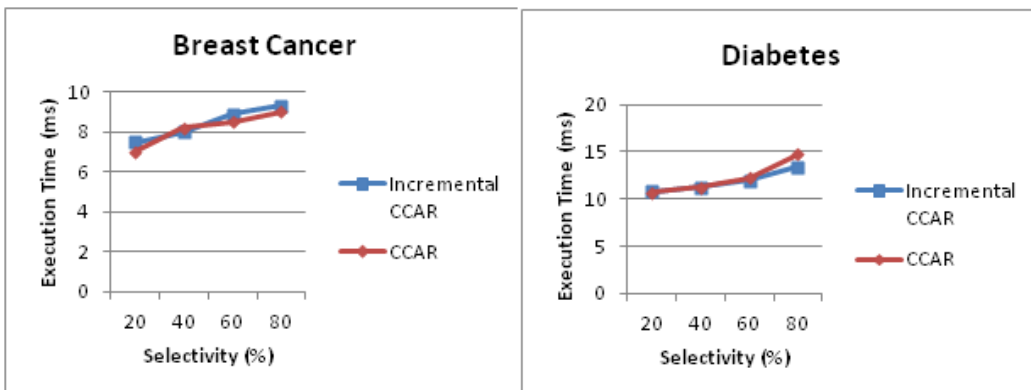
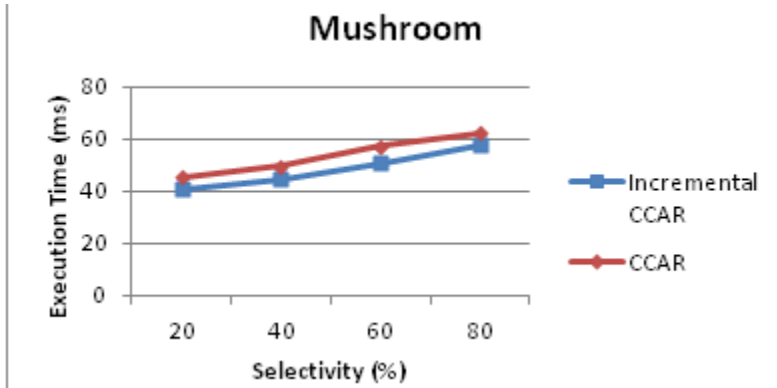


Figure 6.  
 Execution time of CCAR, incremental CCAR in the Mushroom dataset with the constraint selectivity



redundant rules from further processing. The results shown in Figures 7 – 9 indicate that the number of rules is significantly reduced by applying the pruning method.

Then, the rule selection technique using principality measure is applied, and it computes the confidence and completeness of the rules.  $\lambda$  is a key parameter for the principality metric, and it determines the weight of confidence and completeness. A value of  $\lambda = 0.5$  is used while performing rule selection. The total number of rules in classifier for various datasets is summarized in Table 5.3 and the results are depicted in Figures 7 – 9. It is observed that choosing rules of high principality leads to a more compact classifier. Finally, the classifier is constructed using rules of high principality values and by applying the database coverage method.

To evaluate the performance of the proposed Constraint Class Association Rule based Associative Classifier (CCAR\_AC), accuracy is considered as the performance metric and the percentage split method (60% of data for training and 40% for testing) is used as a model evaluation technique. Accuracy denotes the ability of the classifier to correctly classify the unseen data. It is defined as the ratio of the number of correctly classified instances over the total number of instances. The accuracy of the proposed classifier is compared with CBA and a ZeroR algorithm by incorporating the constraints. The execution of CBA and ZeroR algorithms is done using Weka, an open source software package for machine learning. The accuracy of the proposed classifier is relatively higher compared to other algorithms for all the datasets as shown in Table 3 and the corresponding results are depicted in Figure 10.

Figure 7.  
 Number of rules before, after pruning and in classifier for Heart and German datasets

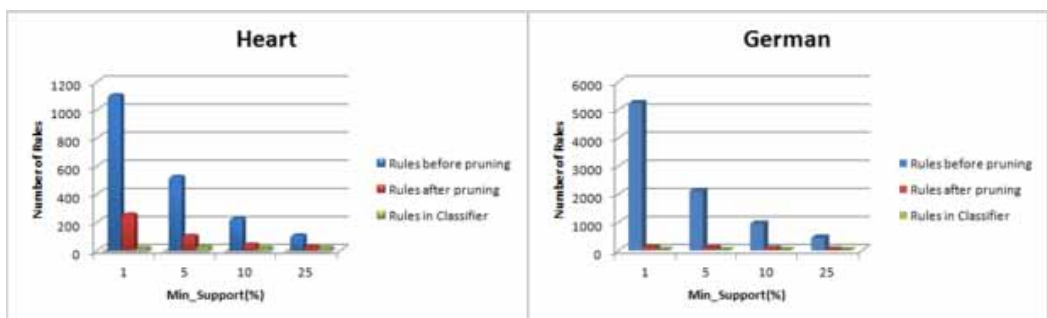


Figure 8.  
 Number of rules before, after pruning and in classifier for Breast and Diabetes datasets

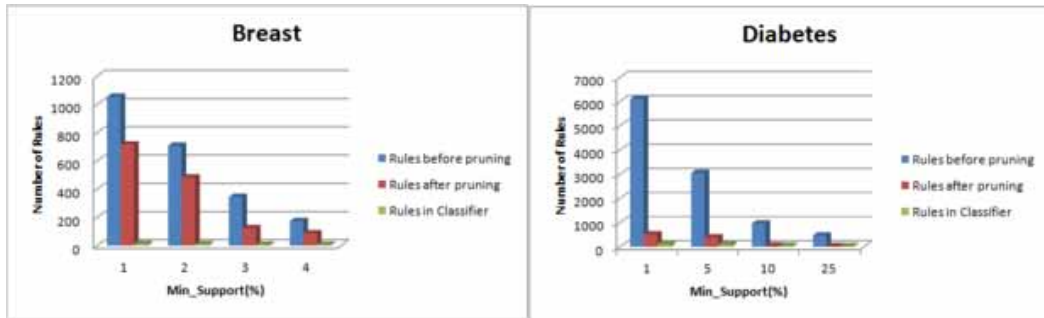


Figure 9.  
 Number of rules before, after pruning and in classifier for Mushroom dataset

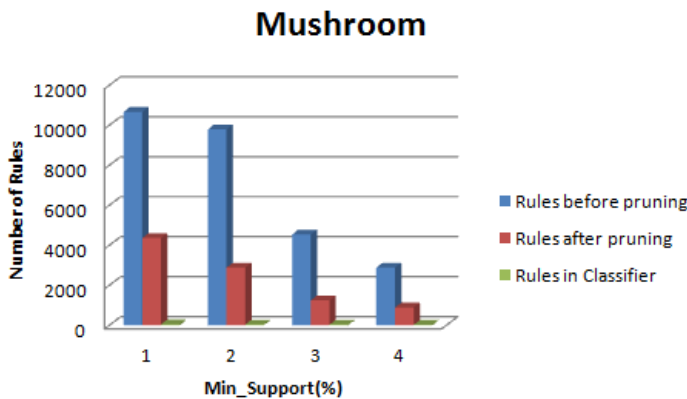


Table 3.  
 Comparison of testing accuracy (%)

Datasets	CBA	ZeroR	CCAR_AC
Heart	77	79	81
German	81	69	90
Breast Cancer	92	91	97
Diabetes	74	70	76
Mushroom	78	75	80
<b>Average</b>	<b>80.4</b>	<b>76.8</b>	<b>84.8</b>

Further, single rule and multiple rule prediction methods are used to assess the performance of the proposed classifier. These prediction methods are used to identify the class label of the test instances. Experiments are done to study the effect of the parameter  $\lambda$ , i.e. the weight of confidence. Table 4 shows the results of accuracy for single rule prediction, as  $\lambda$  varies from 0.5 to 1.0. Note that bold figures of each row in Table 4 indicate the best accuracy ratio with various  $\lambda$  values for the same data set. The corresponding results are shown in Figure 11. The classification accuracy tends to be balanced on most datasets, when  $\lambda > 0.75$  and the maximal accuracy is reached, when  $\lambda \in [0.75, 1.0]$ .



Figure 10.  
 Comparison of classification accuracy of CBA, ZeroR and CCAR\_AC algorithms

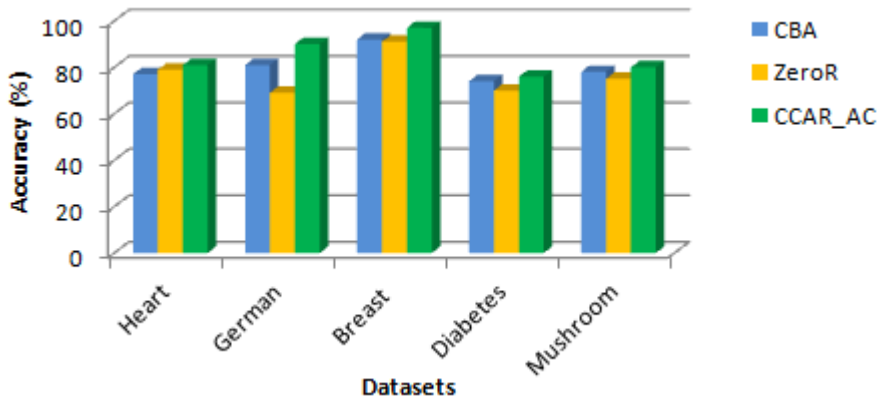


Table 4.  
 Comparison of testing accuracy (%) with different  $\lambda$  for single rule prediction

Datasets	Testing accuracy (%) with different $\lambda$				
	0.5	0.65	0.75	0.9	1.0
Heart	78	75	82	82	82
German	76	78	80	80	80
Breast	88	90	90	90	91
Diabetes	62	61	65	70	70
Mushroom	74	72	76	80	81
Average	75.6	75.2	78.6	80.4	82

Figure 11.  
 Testing accuracy with different  $\lambda$  values using single rule prediction

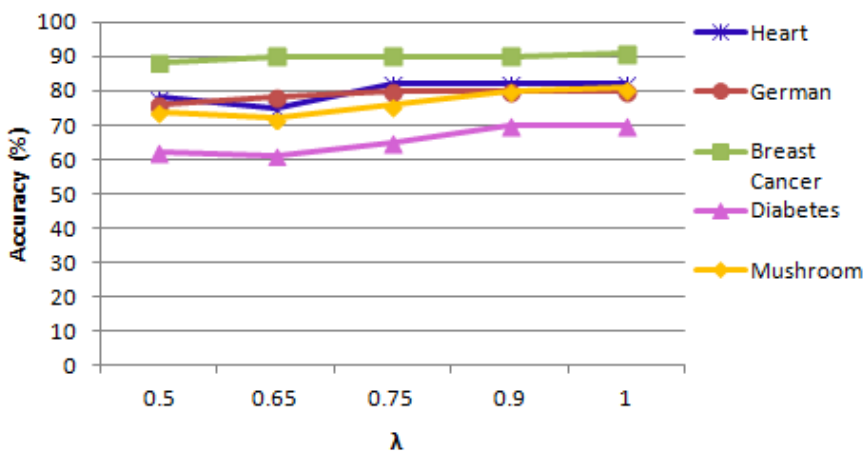


Table 5 shows the accuracy results of multiple rule prediction as  $\lambda$  varies from 0.5 to 1.0. Note that bold figures of each row in Table 5 indicate the best accuracy ratio with various  $\lambda$  values for the same data set. The corresponding results are shown in Figure 12. The classification accuracy tends to be high for breast and diabetes datasets, when  $\lambda = 0.5$ . This shows that the incorporation of completeness during rule selection yields better accuracy for these datasets. For the heart and mushroom datasets, the higher weight of confidence gives higher accuracy i.e. when  $\lambda = 0.9$  and 1.0. The larger value of  $\lambda$  reduces the effects of completeness during rule selection. For the German dataset, maximal accuracy is reached, when  $\lambda = 0.75$ .

It is seen from the results that the accuracy using multiple rule prediction is relatively higher compared to accuracy using single rule prediction for all the datasets.

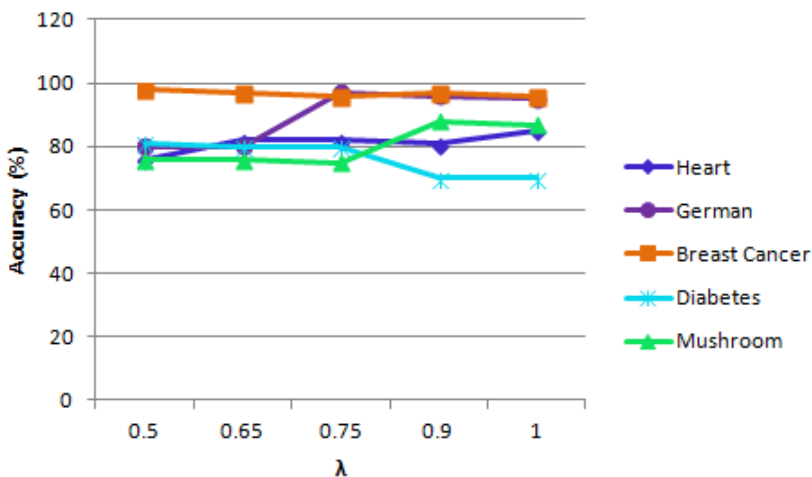
## 5. CONCLUSION

In this work, an efficient method for mining of class association rules using itemset constraints for the incremental data is proposed. Initially, CCAR algorithm mines only those rules satisfying the

Table 5.  
 Comparison of testing accuracy (%) with different  $\lambda$  for multiple rule prediction

Datasets	Testing accuracy (%) with different $\lambda$				
	0.5	0.65	0.75	0.9	1.0
Heart	76	82	82	81	<b>85</b>
German	80	80	<b>97</b>	96	95
Breast	<b>98</b>	97	96	97	96
Diabetes	<b>81</b>	80	80	70	70
Mushroom	76	76	75	<b>88</b>	87
Average	<b>82.2</b>	<b>83</b>	<b>86</b>	<b>86.4</b>	<b>86.6</b>

Figure 12.  
 Testing accuracy with different  $\lambda$  values using multiple rule prediction



itemset constraints and an incremental algorithm is used whenever the original database is added with new records. The Incremental CCAR algorithm computes the safety threshold value which is used to control the number of re-scans over the database. If the number of inserted records is less than the safety threshold, then the ICCR - tree is updated and the updated rules are generated. Otherwise, the ICCR - tree has to be rebuilt for the original and inserted database in order to generate the rules.

The proposed incremental algorithm is tested to compare the execution time of the incremental and non-incremental CCAR algorithm with different selectivity values. The execution time of the proposed Incremental CCAR algorithm gives a speedup of 0.93ms to 1.11ms compared to CCAR algorithm.

Since the number of rules produced by the incremental algorithm is large, the rule pruning and selection methods are used to extract high-quality rules. The rule pruning method gives a significant reduction in the number of rules and for rule selection, principality metric is used and it computes both the completeness and confidence of the rules. The use of completeness results in the reduction of rules. As the weight of the completeness decreases, more rules are generated i.e. higher weight is given to confidence. Results of rule selection highlight that the number of rules in the classifier is minimal, since the selection methods use only those rules with high principality values.

And then, the performance of the proposed Constraint Class Association Rule based Classifier (CCAR\_AC) is evaluated with test data. The accuracy of the proposed classifier is compared with the existing associative classifiers such as CBA and ZeroR by incorporating constraints. Results demonstrate that the average accuracy of the proposed classifier (CCAR\_AC) across the different datasets is higher (85%) than the other two classifiers (80% and 77%). For prediction, the proposed algorithm uses single rule and multiple rule prediction methods with different  $\lambda$  values. It is observed from the results that multiple rule prediction gives higher accuracy values compared to the single rule prediction method for all datasets.

Hence, this work has proposed a method for extracting the CCARs incrementally, when the database is added with new records and the rule pruning and the selection methods are applied to construct the classifier. The classifier's performance is evaluated using single and multiple rule prediction methods.

The proposed work can be extended by applying constraints not only to the rule antecedent but also to the rule consequent. Also, some other rule pruning and rule selection techniques can be used to further improve the accuracy of the classifier.

## REFERENCES

- Chen, F., Wang, Y., Li, M., Wu, H., & Tian, J. (2014). Principal Association Mining: An efficient classification approach. *Knowledge-Based Systems*, 67, 16–25. doi:10.1016/j.knsys.2014.06.013
- Cheung, D. W., Han, J., Ng, V. T., & Wong, C. Y. (1996). Maintenance of discovered association rules in large databases: An incremental updating technique. *Proceedings of the Twelfth International Conference on Data Engineering*, 106-114. doi:10.1109/ICDE.1996.492094
- Hong, T. P., Wang, C. Y., & Tao, Y. H. (2001). A new incremental data mining algorithm using pre-large itemsets. *Intelligent Data Analysis*, 5(2), 111–129. doi:10.3233/IDA-2001-5203
- Le, T. P., Hong, T. P., Vo, B., & Le, B. (2012). An efficient incremental mining approach based on IT-tree. *IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, 1-5. doi:10.1109/rivf.2012.6169826
- Le, T. P., Vo, B., Hong, T. P., & Le, B. (2011). Incremental mining frequent itemsets based on the trie structure and the pre-large itemsets. *IEEE International Conference on Granular Computing (GrC 2011)*, 369-373. doi:10.1109/GRC.2011.6122624
- Leung, R., Rong, J., Li, G., & Law, R. (2013). Personality differences and hotel web design study using targeted positive and negative association rule mining. *Journal of Hospitality Marketing & Management*, 22(7), 701–727. doi:10.1080/19368623.2013.723995
- Li, W., Han, J., & Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. *Proceedings of IEEE International Conference on Data Mining (ICDM 2001)*, 369-376.
- Luna, J., Romero, C., Romero, J., & Ventura, S. (2015). An evolutionary algorithm for the discovery of rare class association rules in learning management systems. *Applied Intelligence Journal*, 42(3), 501–513. doi:10.1007/s10489-014-0603-4
- Ma, B. L. W. H. Y., & Liu, B. (1998). Integrating classification and association rule mining. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 80-86.
- Mabu, S., Chen, C., Lu, N., Shimada, K., & Hirasawa, K. (2011). An Intrusion-Detection Model Based on Fuzzy Class-Association-Rule Mining Using Genetic Network Programming. *IEEE Transactions on Systems, Man, and Cybernetics*, 41(1), 130–139. doi:10.1109/TSMCC.2010.2050685
- Nahar, J., Imam, T., Tickle, K., & Chen, Y. P. (2013). Association rule mining to detect factors which contribute to heart disease in males and females. *Expert Systems with Applications*, 40(4), 1086–1093. doi:10.1016/j.eswa.2012.08.028
- Nguyen, D., Nguyen, L. T. T., Vo, B., & Hong, T. P. (2015). A novel method for constrained class association rule mining. *Information Sciences*, 320, 107–125. doi:10.1016/j.ins.2015.05.006
- Nguyen, D., Nguyen, L. T. T., Vo, B., & Pedrycz, W. (2016). Efficient mining of class association rules with the itemset constraint. *Knowledge-Based Systems*, 103, 73–88. doi:10.1016/j.knsys.2016.03.025
- Nguyen, D., & Vo, B. (2014). Mining class-association rules with constraints. *Proceedings of the Fifth International Conference on Knowledge and Systems Engineering (KSE2013)*, 2, 307-318.
- Nguyen, D., Vo, B., & Le, B. (2015). CCAR: An efficient method for mining class association rules with itemset constraints. *Engineering Applications of Artificial Intelligence*, 37, 115–124. doi:10.1016/j.engappai.2014.08.013
- Nguyen, L. T., Vo, B., Hong, T. P., & Thanh, H. C. (2012). Classification based on association rules: A lattice-based approach. *Expert Systems with Applications*, 39(13), 11357–11366. doi:10.1016/j.eswa.2012.03.036
- Nguyen, L. T., Vo, B., Hong, T. P., & Thanh, H. C. (2013). CAR-Miner: An efficient algorithm for mining class-association rules. *Expert Systems with Applications*, 40(6), 2305–2311. doi:10.1016/j.eswa.2012.10.035
- Phan-Luong, V. (2013). Mining normal and abnormal class-association rules. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)* (pp. 968-975). IEEE.
- Rong, J., Vu, H., Law, R., & Li, G. (2012). A behavioral analysis of web sharers and browsers in Hong Kong using targeted association rule mining. *Tourism Management*, 33(4), 731–740. doi:10.1016/j.tourman.2011.08.006

- Subbulakshmi, B., & Monisha, M. (2016). Incremental Constraint Class Association Rule Mining of Student Performance Dataset. In *5th International Conference on Recent Trends in Information Technology (ICRTIT)*. Anna University.
- Vishwakarma, N. K., Agarwal, J., Agarwal, S., & Sharma, S. (2013). Comparative analysis of different techniques in classification based on association rules. In *Computational Intelligence and Computing Research (ICCRIC), 2013 IEEE International Conference on* (pp. 1-7). IEEE.
- Vo, B., & Le, B. (2008). A novel classification algorithm based on association rule mining. *Proceedings of Pacific RIM Knowledge Acquisition Workshop*, 61-75.
- Vo, B., Le, T., Hong, T.-P., & Le, H. B. (2014). An effective approach for maintenance of pre-large-based frequent-itemset lattice in incremental mining. *Applied Intelligence*, 41(3), 759–775. doi:10.1007/s10489-014-0551-z
- Yafi, E., Al-Hegami, A. S., Alam, M. A., & Biswas, R. (2012). YAMI. *Incremental Mining of Interesting Association Patterns. Int. Arab J. Inf. Technol.*, 9(6), 504–510.
- Zhang, H., Zhao, Y., Cao, L., & Zhang, C. (2007). Class association rule mining with multiple imbalanced attributes. In *AI 2007: Advances in Artificial Intelligence* (pp. 827–831). Springer. doi:10.1007/978-3-540-76928-6\_100

*B. Subbulakshmi is currently working as Assistant professor in the department of Computer Science and Engineering at Thiagarajar College of Engineering, Madurai, Tamil Nadu, India. She has 20 years of teaching and research experience. She has completed her Ph.D from Anna University, Chennai in the year 2019. She is a life member in Computer Society of India (CSI). She has published research papers in various reputed journals and conferences. She is guiding under graduate and post graduate students in the area of sentiment analysis, data analytics and machine learning. Her research interest includes Databases, Data mining, Machine Learning and Data Analytics. She has authored several chapters in the refereed edited books of springer and acting as Reviewer in International Journals from IEEE, Springer.*

*C. Deisy is currently working as Professor & Head of Information Technology department at Thiagarajar College of Engineering, Madurai Tamil Nadu, India. She received her PhD in the year 2010 from Anna University, Chennai. She is a life member of Indian Society for Technical Education (ISTE) and Computer Society of India (CSI). She has published more than 60 research papers in various reputed journals and conferences. She has completed AICTE sponsored project on Mobile Based Cardio Vascular Disease Diagnosis System for Rural India and had worked on collaborative project with multimedia university, Malaysia. She has produced more than 10 research scholars working in the area of data mining, analytics and text mining. Her research interest includes Data Mining, Text and Web Mining. She has authored several chapters in the refereed edited books of springer, IGI and acting as Reviewer in International Journals from IEEE, Springer and Elsevier.*

*S. Parthasarathy is currently working a Professor of Data Science & HOD of Applied Mathematics and Computational Science at Thiagarajar College of Engineering, Madurai, Tamil Nadu, India. Previously he was an Associate Professor of Computer Applications in the same institution. He has more than two decades of teaching and research experience. He completed his Ph.D., Degree in Computer Applications from Anna University, Chennai in the year 2009. He has published research papers in peer reviewed international journals from Elsevier, Springer and Emerald such as Computers in Industry, Software Quality Journal, Computer Standards and Interfaces, Journal of Systems and Software, Business Process Management Journal, International Journal of Project Management and national and international conferences organized by IEEE. He has authored several chapters in the refereed edited books of IGI, USA. Reviewer in International Journals: IEEE Transactions on Engineering Management, Enterprise Information Systems – An International Journal, Taylor & Francis, International Journals from Elsevier, Springer and Emerald.*