INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD • INDIA

Research and Publications

Classification Using Association Rules

Rajanish Dass

W.P. No. 2008-01-05
January 2008

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD-380 015
INDIA

Classification Using Association Rules

Rajanish Dass
Computer and Information Systems Group
Indian Institute of Management Ahmedabad

Abstract

Association rule mining is a well-known technique in data mining. Classification using association rules combines association rule mining and classification, and is therefore concerned with finding rules that accurately predict a single target (class) variable. The key strength of association rule mining is that all interesting rules are found. The number of associations present in even moderate sized databases can be, however, very large – usually too large to be applied directly for classification purposes.

This project compares and combines different approaches for classification using association rules. This research area is called classification using association rules. An important aspect of classification using association rules is that it can provide quality measures for the output of the underlying mining process. The properties of the resulting classifier can be the base for comparisons between different association rule mining algorithms. A certain mining algorithm is preferable when the mined rule set forms a more accurate, compact and stable classifier in an efficient way. First, in this project we are interested in the comparison of the quality of different mining algorithms. Therefore, we use classification using association rules. Secondly, classification using association rules can be improved itself by using a mining algorithm that prefers highly accurate rules. The author of the report is indebted to several students and research assistants who showed interest and got involved in the work.

Classification Using Association Rules

1. Introduction

Information gathering is ubiquitous. For example, at supermarket checkouts information about customer purchases are recorded. When payback or discount cards are used information about customer purchasing behavior and personal details can be linked. Evaluation of this information can help retailers devise more efficient and personalized marketing strategies. The amount of information stored in modern databases makes manual analysis intractable. Data mining provides tools to reveal previously unknown information in large databases. A well-known data mining technique is association rule mining. It is employed to uncover all interesting relationships (called associations) in a database. Nowadays the discriminative power of these descriptive patterns the association rules is used to build accurate classifiers. In this paper we will look at different classification techniques that use association rule mining.

1.1 Data Mining

Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summaries the data in novel ways that are both understandable and useful to the data holder. The Knowledge Discovery in Databases (KDD) process consists of several stages as follows: selecting the target data, preprocessing the data, transforming them if necessary, performing data mining to extract patterns and relationships, and interpreting and assessing the discovered structures.

Association rule mining is a data mining application to extract patterns. When we use classification using association rules we go one step further and start interpreting and analyzing the patterns into predefined classes by techniques of supervised learning. Data Mining is categorized into different tasks depending on what the data mining algorithm under consideration is used for. The data management strategy is important for the efficient access of large databases. It defines the way the data is stored, sorted, indexed and retrieved.

1.2 Association Rule Mining

Association rule mining is a well-known technique in data mining. It is able to reveal all Interesting relationships, called associations, in a potentially large database. However, how interesting a rule is depends on the problem a user wants to solve. Existing approaches employ different parameters to guide the search for interesting rules. Classification using association rules combines association rule mining and classification, and is therefore concerned with finding rules that accurately predict a single target (class) variable.

The key strength of association rule mining is that all interesting rules are found. The number of associations present in even moderate sized databases can be, however, very large usually too large to be applied directly for classification purposes.

Association Rule are the out put of the process of finding Associations or correlations among an item set. The rule form is LHS ->RHS [support, Confidence]
Example:-
Buying beer and chips -> [Ketchup 5%, 60%]
Therefore, any classification learner using association rules has to perform three major steps: Mining a set of potentially accurate rules, evaluating and pruning rules, and classifying future instances using the found rule set.

1.3 Classification

The goal of classification in data mining is to build a model from a given set of training data records in which the classes of the objects are known and uses the model to assign classes to new records. For example, a risk classification model can be built from a dataset of previous credit card customers and applied to classify the risk levels of new customers. Classification is a frequently encountered data mining task in enterprise applications. Several data mining techniques are available for building classification models.

1.3.1 Classification using association rules

Classification using association rules can be divided into three fundamental parts:

1. Association rule mining,
2. Pruning and
3. Classification.

The mining of association rules is a typical data mining task that works in an unsupervised manner. A major advantage of association rules is that they are theoretically capable of revealing all interesting relationships in a database. But for practical applications the number of mined rules is usually too large to be exploited entirely. This is why the pruning phase is stringent in order to build accurate and compact classifiers. The smaller the number of rules a classifier needs to approximate the target concept satisfactorily, the more human-interpretable is the result.

2. Association Rule Mining

2.1 Genrating frequent item sets

Finding association rules can be seen as a simple search problem. But an exhaustive search is intractable because the possible number of association rules is exponential with respect to the number of attributes. For n binary attributes there are O (n^{2n-1}) rules. It is even worse for discrete valued attributes assuming there are n attributes and each can take m values, there are O (m^n) possible rules.

An item set X of length k is frequent if and only if all subsets of X with length k-1 are frequent. This property allows the search space to be pruned substantially. Frequent item sets consists of all individual items that have a support above a user defined minimum support. This can be done with one pass over the data in linear time. To get the frequent item sets of size 2 there are two steps: First the frequent item sets of size 1 are combined in every possible way to build candidate item sets of size 2, then, in another pass over the data, the candidate item sets are checked to make sure that they are really frequent. All infrequent ones are deleted.

Termination is obvious either the set of frequent item sets is empty for a distinct k, or k equals the number of attributes. At most n linear passes over the data are required if n is the number of attributes. From the frequent item sets both algorithms generate association rules in different ways using a different measure for interestingness.

2.2 Apriori Algorithm

The Apriori algorithm is one of the earliest algorithms for mining association rules and has become the standard approach in this area. The search for association rules is guided by two parameters: support and confidence.

The Apriori Mining process is composed of two major steps. The first one generating frequent item sets .This step can be seen as support based pruning, because only item sets with at least minimum support were considered. The second step is the generation of rules out of the frequent item sets. In this step confidence based pruning is applied. Rule discovery is straightforward. For every frequent item set f and every non-empty subset s of f, apriori outputs a rule of the form s => (f - s) if and only if the confidence of that rule is above the user specified threshold.

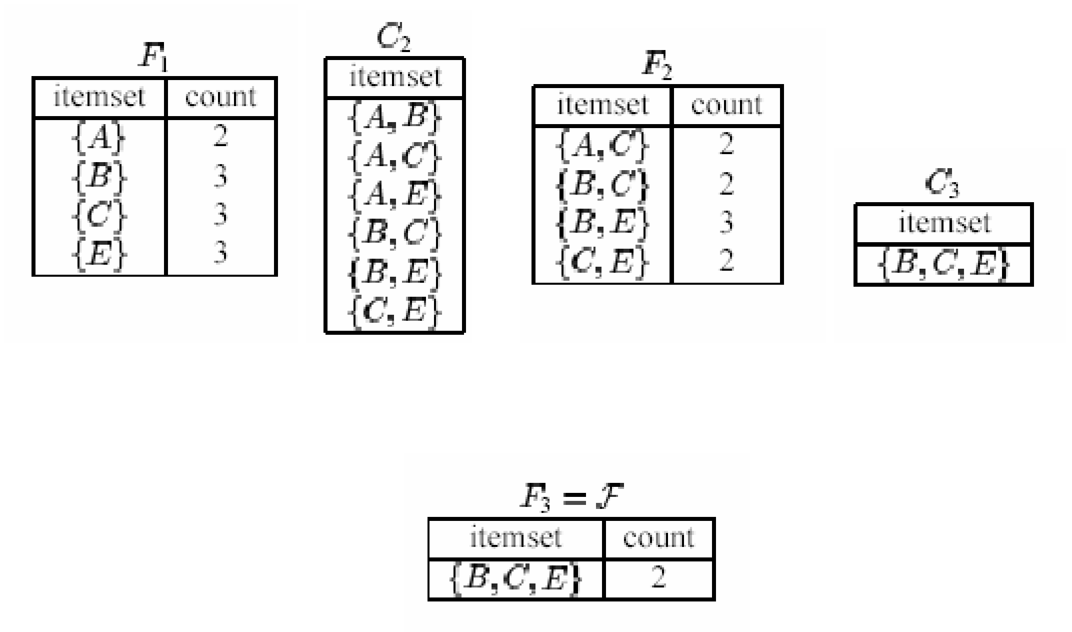
In the first pass of the algorithm, we count for each item $I \in T$ the number of transactions it contains. If the result exceeds |D| Smin then the item becomes a frequent 1-itemset. All subsequent passes consist of two phases. Let the current pass be Ck of Candidate K item sets from the set Fk-1 of frequent (k-1) items in the previous pass. The result set Ck is a super set of frequent k –item sets.

In second phase it scans Database D and Check for each transaction t which of the candidates is contained in t. If candidate is contained in t its counter is increased by one The function Apriorigen consist of two phases .It takes as argument the set of frequent (k-1)-itemset Fk-1.The First phase Called Join step, it Join Fk-1 with itself .It assumes that the items of an item set are stored in their lexicographic order. If the first k-2 items of an itemset ti match those of itemset i2 ≠ i1 , it construct a new candidate k itemset by concatenating this common prefix with the (K-1) th item of i1 and i2 observing the item sort order.

The algorithm concludes Apriori that k item set I only becomes a frequent itemset if there is at least one frequent (k-1) itemset that consist of first k-1 itemset of i. This Join step generates more itemset then necessary. Some of them are eliminated in the prune step. Not only do the first k -1 items of a Candidate have to be contained in an itemset of Fk-1, but all (k-1)-subsets have to be frequent

Example

Consider items set I={A,B,C,D,E} Database {{A,C,D},{B,C,E},{A,B,C, E},{B,E}}.the minimum Support is 0.5 Which corresponds to two transactions



F_1

itemset	count
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{E\}$	3

C_2

itemset
$\{A,B\}$
$\{A,C\}$
$\{A,E\}$
$\{B,C\}$
$\{B,E\}$
$\{C,E\}$

F_2

itemset	count
$\{A,C\}$	2
$\{B,C\}$	2
$\{B,E\}$	3
$\{C,E\}$	2

C_3

itemset
$\{B,C,E\}$

$F_3 = \mathcal{F}$

itemset	count
$\{B,C,E\}$	2

Fig. 1 Execution of Apriori Algorithm.

3. Classification

3.1 Problem Definition

Let database D is a set of instances where each instance is represented by < a_1, a_2...........a_m , c>, where a_1, a_2...........a_m are non class attributes and c is a class attribute C. A common rule is defied as

$$x \rightarrow c ,$$

Where x is a set of non class attributes and c is class label.

A rule classifies an instance if the instance contains all the attribute values in X and the class label c.

The quality measurement factor of a rule is support and confidence where support is (num(x, y) / |D|), |D| denotes the total number of instances in database and confidence is (num(x, y) / num(x)). Rule items that satisfy min_sup are called frequent rule items, while the rest are called infrequent rule items.

The task is to generate the CARs that satisfies both min_sup and min_conf constraints. These CARs can be used to build a classifier which would be able to classify new instance accurately.

The input to build a classifier is a pre processed set of Class Association Rules.

3.2 Schemes for classification with rule sets

To build a Classifier there are three ways to use Classification rules .The basic decision involves whether to consider every rule that covers an instance to certain extent or to consider only a single rule; the simplest way to consider each rule equally.

An unseen instance is classified using just the class label that is favored by the majority of rules in the set that are covering this instance. This kind of classification Algorithm is called *majority vote*.

One another scheme is called *weighting scheme*. The use of weighted scheme is easily accommodated, because association rule mining produce a sorted set of rules according to their interestingness measure the majority vote scheme does not take any information out of the sort order. Hence in order to reveal the differences between associations rule mining algorithms weighted scheme is preferred.

The other scheme is to look only at single rule therefore the sorted set of class association rule is used as a sorted list and the first rule that covers the instance to be classified is used for prediction .This type of approach is called *decision list* algorithm.

These three schemes are the basic schemes used in classification based on association rule set.

3.2.1 Classifiers using a weighted approach

As explained, the simplest voting scheme is a majority vote where each rule r is equally weighted with weight w(r) = 1. In addition we explore two different weighting schemes.

First we use a simple linear weighting function by assigning the weight w = 1 to the highest ranked rule in the sort order based on the interestingness measure. The weight decreases linearly according to this ordering. Therefore the weighting function w for a rule r with rank(r) $\in \{1 \ldots\ldots\ldots\ldots\ldots\ldots \text{rankmax}\}$ where rankmax is the maximal rank of a rule (the maximal rank equals the number of rules) is:

$$w(r) = -1/\text{rankmax} + 1(\text{rank}(r) + 1)$$

The denominator has (rankmax + 1) instead of rankmax because the weight w of the rule with the maximum rank should be non-zero. A practical problem in a comparative study with a linear vote occurs when the number of classification rules in the rule sets differs to a great extent. In this case the weighting functions are too different to be compared directly. To avoid this problem the maximum rank rankmax should be the same. This means that the number of classification rules should be the same. We use another weighting scheme that does not suffer from this problem to such an extent. Instead of applying a linear weighting function, we use the inverse function

$$f(x) = 1/x$$

for weighting. For each rule r its weight w (r) is calculated using:

$$w(r) = 1/rank(r)$$

This weighting scheme emphasizes the difference between the top ranked rules, because rules at the end of the sorted list only get a little weight and so only have small influence.

3.2.2Decision list Based classifiers

The standard classifier for classification using association rules the CBA algorithm is a decision list classifier. Classification is very simple in CBA[11]. It just searches in the pruned and ordered list for the first rule that covers the instance to be classified. The prediction is the class label of that classification rule. If no rule covers the instance, CBA uses the default class calculated during pruning .If the decision list is empty, the majority class of the training instance will be assigned to each test instance as default.

The advantage of a decision list classifier in a comparative study for rule mining algorithms is that the ranking induced by the rule mining algorithm is very important. There is no weighting scheme which could compensate a non-optimal sort order, because the first rule that covers a test instance is used for prediction.

4. Related Work

The authors of Apriori have adapted the popular stepwise association rule mining algorithm for extracting class association rules that represent characteristics of the data classes in two applications, i.e. telecommunication and medical diagnoses. Their aim was to discover a set of overlapping rules that are individually accurate and not prediction of future class labels. Thus, the presented method cannot be considered as a complete classification method since the ultimate aim for classification in data mining is prediction. The results of the two case studies indicated that association rule can be used for partial classification in which many useful rules for general practitioners have been derived for the medical diagnoses study. Finally, the authors speculated whether association rule mining can be used for complete classification.

One of the first algorithms to use association rule approach for classification was CBA [11]. CBA implements the famous Apriori algorithm [13] that requires multiple passes over the training data set in order to discover frequent items. Once the discovery of frequent items finished, CBA proceeds by converting any frequent item that passes the minimum user confidence into a rule. In doing that, only one subset of the generated rules is considered in the final classifier. Evaluating all the generated rules against the training data set does the selection of the subset. The frequent items discovery and rules generation are implemented in two separate phases in CBA.

An associative classification algorithm that selects and analyses the correlation between high confidence rules, instead of relying on a single rule, has been developed. It uses a set of related rules to make a prediction by evaluating the correlation among them. The correlation measures how effective are the rules based on their support values and class distributions. In addition, a new prefix tree data structure named CR-tree to handle the set of rules generated and to speed up the retrieval process of a rule has been introduced. The CR-tree has proven to be effective in saving storage since many conditions of the rules are shared in the tree.

A new approach for building classification systems based on both positive and negative rules has been introduced [2]. The interestingness of the rules for the proposed algorithm is based on the correlation coefficient that measures the strength of the linear relationship between a pair of variables. Besides confidence and support thresholds, correlation coefficient has been used for pruning the final classifier, giving a much reduced rules set if compared with support and confidence pruning methods. The algorithm generates the rules similar to Apriori approach and ranks the rules similar to CBA rules ranking method. A greedy associative classification algorithm called CPAR was proposed [20]. CPAR adopts FOIL [4] strategy in generating the rules from data sets. It seeks for the best rule condition that brings the most gain among the available ones in the data set. Once the condition is identified, the weights of the positive examples associated with it will be deteriorated by a multiplying factor, and the process will be repeated until all positive examples in the training data set are covered.

The searching process for the best rule condition is time consuming process for CPAR since the gain for every possible item needs to be calculated in order to determine the best item gain. Thus, CPAR uses an efficient data structure, i.e. PN Array, to store all the necessary information for calculation of the items gain. In the rules generation process, CPAR derives not only the best condition but all close similar ones since there are often more than one attribute items with similar gain. It has been claimed that CPAR improves the efficiency of the rule generation process if compared with popular associative classification methods like CBA and CMAR[9].

CAAR was proposed [19] The existing association-based classification algorithms suffer from two major shortcomings: (1) they generate classifiers containing a lot of rules; (2) they consume a large amount of system resources. To overcome these problems, this Algorithm presents a novel algorithm, namely classification based on atomic association. Atomic rule mining generates the smallest and the simplest rule set for classification. The strong atomic rules with the highest and near-highest confidences *can* realize partial classification accurately. Multiple passes of partial classification generates the concise and accurate classifier.

The new approach is compared with decision tree induction and the existing associative classification. The results show that the proposed algorithm not only achieves the highest classification accuracy but also generates the smallest classification rule *set.*

Harmony was proposed [18]. It directly mines the final set of classification rules. HARMONY uses an instance-centric rule-generation approach and it can assure for each training instance, one of the highest-confidence rules covering this instance is included in the final rule set, which helps in improving the overall accuracy of the classifier. HARMONY also has high efficiency and good scalability.

5. Various Algorithms for Classification Using Association Rule Mining

5.1 CBA: Integrating Classification and Association Rule Mining

Classification rule mining and association rule mining are two important data mining techniques. Classification rule mining aims to discover a small set of rules in the database to form an accurate classifier. Association rule mining finds all rules in the database that satisfies some minimum support and minimum confidence constraints. For association rule mining, the target of mining is not predetermined, while for classification rule mining there is one and only one pre-determined target, i.e., the class.

Both classification rule mining and association rule mining are indispensable to practical applications. Thus, great savings and conveniences to the user could result if the two mining techniques can somehow be integrated. The integration is done by focusing on a special subset of association rules whose right-hand-side is restricted to the classification class attribute.

Data mining in the proposed associative classification framework thus consists of three steps:

1. Discrediting continuous attributes, if any
2. Generating all the class association rules (CARs), and
3. Building a classifier based on the generated CARs.

CBA makes the following contributions:

1. It proposes a new way to build accurate classifiers.
2. It makes association rule mining techniques applicable to classification tasks.
3. It helps to solve a number of important problems with the existing classification systems.

Generating the Complete Set of CARs The proposed algorithm is called algorithm CBA (Classification Based on Associations). It consists of two parts, a *rule generator* (called CBA-RG), and a *classifier builder* (called CBA-CB).

The CBA-RG algorithm

The CBA-RG algorithm exploits the concept of Apriori algorithm for generating the frequent rule items by making multiple passes over the data. In the first pass, it counts the support of individual rule item and determines whether it is frequent. In each subsequent pass, it starts with the seed set of rule items found to be frequent in the previous pass. It uses this seed set to generate new possibly frequent rule items, called candidate rule items. The actual supports for these candidate rule items are calculated during the pass over the data. At the end of the pass, it determines which of the candidate rule items are actually frequent. From this set of frequent rule items, it produces the rules (CARs).

For each subsequent pass (pass k) the algorithm performs 4 major operations.

The frequent rule items F_{k-1} found in the (k-1)th pass are used to generate the candidate rule items C_k using the condidateGen function.

It then scans the database and updates various support counts of the candidates in C_k.

After those new frequent rule items have been identified to form F_k, the algorithm then produces the rule CAR_k using the genRules function.

Finally, rule pruning is performed on these rules using the pruneRules function.

The condidateGen function is quite similar to the function Apriori-gen in algorithm Apriori. The genRules function is used for generating the CARs and the pruneRules function uses the pessimistic error rate based pruning method in C4.5 (Quinlan 1992) for pruning the rules.

Building a Classifier

This section presents the CBA-CB algorithm for building a Classifier using CARs. Before performing pruning a global order is imposed on the set of mined rules. Let R be set of CARs and their are two rules, r_i and r_j, $r_i \succ r_j$ (also called r_i precedes r_j or r_i has a higher precedence than r_j) if the confidence of r_i is greater than that of r_j, or Their confidences are the same, but the support of r_i is greater than that of r_j, or Both the confidences and supports of r_i and r_j are the same, but r_i is generated earlier than r_j.

Naïve Approach M1 Classifier

As we are looking for only highest precedence rules for our classifier, so we sort the R (CARs) according to the precedence rules.

For each rule r in R we go through D to find those cases covered by r (they satisfy the conditions of r). r is marked if it correctly classifies a case d. did is the unique identification number of d. If r can correctly classify at least one case (i.e., if r is marked), it will be a potential rule in the classifier. Those cases it covers are then removed from D. A default class is also selected (the majority class in the remaining data), which means that if we stop selecting more rules for the classifier C this class will be the default class of C. We then compute and record the total number of errors that are made by the current C and the default class. This is the sum of the number of errors that have been made by all the selected rules in C and the number of errors to be made by the default class in the training data. When there is no rule or no training case left, the rule selection process is completed.

The first rule at which there is the least number of errors recorded on *D* is the cutoff rule. All the rules after this rule can be discarded because they only produce more errors. The rules that are not discarded and the default class of the last rule in *C* form the classifier.

M2 CLASSIFIER

For each instance in D we find two kind of highest precedence rules, first that correctly classifies d called cRule and the next is that wrongly classifies d called wRule. In this stage we create a matrices which contains <**CovRule, cRule, wRule, U, Q, A**> where **CovRule** means all rules that match with this particular instance, *U* is the set of all *cRule*s, and *Q is* the set of *cRule*s that have a higher precedence than their corresponding *wRule*s. ***A*** denotes the set of <*dID*, *y*, *cRule*, *wRule*> where *dID* is the unique identification number of the case *d*, *y* is the class of *d* when $wRule \succ cRule$.

It goes through D again to find all rules that classify it wrongly and have a higher precedence than the corresponding *cRule* of *d* because in stage 1 we could not decide which rule should cover a particular instance. So in this stage wRules may replace *cRules* to cover the case because they have higher precedence.

5.2 CMAR: Accurate and Efficient Classification Based on Multiple Class Association Rules

Building accurate and efficient classifiers for large databases is one of the essential tasks of data mining and machine learning research. Given a set of cases with class labels as a *training set*, *classification* is to build a model (called *classifier*) to predict future data objects for which the class label is unknown. To achieve high accuracy, a classifier may have to handle a large set of rules, including storing those generated by association mining methods, retrieving the related rules, and pruning and sorting a large number of rules.

CMAR has been used for accurate and efficient classification and has made the following contributions.

First, *instead of relying on a single rule for classification,* CMAR *determines the class label by a set of rules.* Given a new case for prediction, *CMAR* selects a small set of high confidence, highly related rules and analyzes the correlation among those rules.

Second, to improve both accuracy and efficiency, CMAR employs a novel data structure, CR-tree, to compactly store and efficiently retrieve a large number of rules for classification. CR-tree is a prefix tree structure to explore the sharing among rules, which achieves substantial compactness. CR-tree itself is also an index structure for rules and serves rule retrieval efficiently.

Third, *to speed up the mining of complete set of rules,* CMAR adopts a variant of FP-growth method. *FP-growth* is much faster than *Apriori*-like methods used in previous association-based classification, especially when there exist a huge number of rules, large training data sets, and long pattern rules.

In general, given a training data set, the task of **classification** is to build a classifier from the training data set such that it can be used to predict class labels of unknown objects with high accuracy. A**ssociative classification** method finds the complete set of class association rules (CAR) passing the thresholds. When a new (unknown) object comes, the classifier selects the rule which matches the data object and has the highest confidence and uses it to predict the class label of the new object. Recent studies show that associative classification is intuitive and effective and has good classification accuracy in many cases.

CMAR, which performs **C**lassification based on Multiple **A**ssociation **R**ules. *CMAR* consists of two phases: *rule generation* and *classification*.

In the first phase, *rule generation*, *CMAR* computes the *complete* set of rules .*CMAR* prunes some rule and only selects a subset of high quality rules for classification.

In the second phase, *classification*, for a given data object, *CMAR* extracts a subset of rules matching the object and predicts the class label of the object by analyzing this subset of rules.

To find rules for classification, *CMAR* first mines the training data set to find the complete set of rules passing certain support and confidence thresholds. This is a typical frequent pattern or association rule mining task .To make mining highly scalable and efficient, *CMAR* adopts a variant of *FP-growth* method.

FP-growth is a frequent pattern mining algorithm which is faster than conventional *Apriori* like methods, especially in the situations where large data sets, low support threshold, and long patterns exist.

There are two major differences in the rule mining in *CMAR* and the standard *FP-growth* algorithm. On one hand, CMAR *finds frequent pattern and generates rules in one step.* Conventionally, association rules must be mined in two steps. This is also the case for traditional associative classification methods. First, all the frequent patterns (i.e., patterns passing support threshold) are found. Then, all the association rules satisfying the confidence threshold are generated based on the mined frequent patterns.

The difference of *CMAR* from other associative classification methods is that for every pattern, *CMAR* maintains the distribution of various class labels among data objects matching the pattern. This is done without any overhead in the procedure of counting (conditional) databases. On the other hand, CMAR *uses class label distribution to prune.* Once a rule is generated, it is stored in a *CR-tree*, which is a prefix tree structure.
The number of rules generated by class-association rule mining can be huge. To make the classification effective and also efficient, we need to prune rules to delete redundant and

noisy information. According to the facility of rules on classification, a global order of rules is composed.

CMAR employs the following methods for rule pruning.

1. First, using general and high-confidence rule to prune more specific and lower confidence ones.
2. Second, selecting only positively correlated rules.
3. Third, pruning rules based on database cover

5.3 CPAR: Classification based on Predictive Association Rules

Associative classification, approach also suffers from two major deficiencies:

1. It generates a very large number of association rules, which leads to high processing overhead;
2. Its confidence based rule evaluation measure may lead to over fitting.

In comparison with associative classification, traditional rule based classifiers, such as C4.5, FOIL and RIPPER, are substantially faster but their accuracy in most cases, may not be as high.So, we propose a new classification approach, CPAR (Classification based on Predictive Association Rules), which combines the advantages of both associative classification and traditional rule based classification. Instead of generating a large number of candidate rules as in associative classification, CPAR adopts a greedy algorithm to generate rules directly from training data. Moreover, CPAR generates and tests more rules than traditional rule based classifiers to avoid missing important rules.
CPAR inherits the basic idea of FOIL in rule generation and integrates the features of associative classification in predictive rule analysis. In comparison with associative classification, CPAR has the following advantages:

1. CPAR generates a much smaller set of high quality predictive rules directly from the dataset;

2. To avoid generating redundant rules, CPAR generates each rule by considering the set of already generated rules;
3. When predicting the class label of an example, CPAR uses the best k rules.
4. CPAR employs the following features to further improve its accuracy and efficiency:

(a) CPAR uses dynamic programming to avoid repeated calculation in rule generation; and
(b) When it generates rules, instead of selecting only the best literal, the entire close to the best literals are selected so that important rules will not be missed.

CPAR generates a smaller set of rules, with higher quality and lower redundancy in comparison with associative classification. As a result, CPAR is much more time efficient in both rule generation and prediction but achieves as high accuracy as associative classification.

Two important association rule based classifiers are CBA and CMAR. CBA first generates all the association rules with certain support and confidence thresholds as candidate rules. It then selects a small set of rules from them to form a classifier. When predicting the class label for an example, the best rule whose body is satisfied by the example is chosen for prediction.

CMAR generates and evaluates rules in a similar way as CBA (but uses a more efficient FPtree structure). When datasets contain a large number of rows and columns, both rule generation and rule selection in CBA and CMAR are time consuming.

5.4 MSCBA: Improving an Association Rule Based Classifier

Existing classification algorithms in machine learning mainly use heuristic search to find a subset of regularities in data for classification. In the past few years, extensive research was done in the database community on learning rules using exhaustive search under the name of association rule mining. Although the whole set of rules may not be used directly for accurate classification, effective classifiers have been built using the rules. . The main strength of this system is that it is able to use the most accurate rules for classification.

Building effective classification systems is one of the central tasks of data mining. Past research has produced many techniques and systems. The existing techniques are, however, largely based on heuristic/greedy search. They aim to find only a *subset* of the regularities that exist in data to form a classifier. In the past few years, the database community studied the problem of rule mining extensively under the name of association rule mining. The study there is focused on using exhaustive search to find all rules in data that satisfy the user-specified minimum support (minsup) and minimum confidence (minconf) constraints.

Although the complete set of rules may not be directly used for accurate classification, effective and efficient classifiers have been built using the rules. The major strength of such systems is that they are able to use the most accurate rules for classification. However, they also have some weaknesses, inherited from association rule mining.

1. Traditional association rule mining uses only a single minsup in rule generation, which is inadequate for unbalanced class distribution.
2. Classification data often contains a huge number of rules, which may cause combinatorial explosion. For many datasets, the rule generator is unable to generate rules with many conditions, while such rules may be important for classification.

It tackles the first problem by using *multiple class minsups* in rule generation (i.e., each class is assigned a different minsup), rather than using only a single minsup as in CBA. This results in a new system called msCBA [12] .

The second problem is more difficult to deal with directly as it is caused by exponential growth of the number of rules. We deal with it indirectly. We try to find another classification technique that is able to help when some rules from msCBA are not accurate. The decision tree method is a clear choice because decision trees often go very deep, i.e., using many conditions. We then propose a technique to combine msCBA with the decision tree method as in C4.5.

The basic idea is to use the rules of msCBA to segment the training data and then select the classifier that has the lowest error rate on each segment to classify the future cases falling into the segment. This composite method results in remarkably accurate classifiers.

After all rules (CARs) are found, a classifier is built using the rules. In CBA, a set of high confidence rules is selected from CARs to form a classifier (this method is also used in msCBA). The selection of rules is based on a total order defined on the rules.

A CBA classifier is of the form:
<*r*1, *r*2, …, *rn*, *default_class*>. (**1**)
Where *ri* □ *R*, *ra* □ *rb* if *b* > *a*. In classifying an unseen case, the first rule that satisfies the case classifies it. If no rule applies, the default class is used.

The most important parameter in association rule mining is the minsup. It controls how many rules and what kinds of rules are generated. The CBA system follows the classic association rule model and uses a single minsup in its rule generation. We argue that this is inadequate for mining of CARs because many practical classification datasets have uneven class frequency distributions. If we set the minsup value too high, we may not find sufficient rules of infrequent classes. If we set the minsup value too low, we will find many useless and over-fitting rules for frequent classes.

To solve the problems, msCBA adopts the following (*multiple minimum class supports*):

For many datasets, the rule generator is unable to generate rules with many conditions (i.e. long rules) due to combinatorial explosion. When such long rules are important for classification, our classifiers suffer. Here, we propose a combination technique. The aim is to combine msCBA with a method that is able to find long rules. Clearly, the decision tree method is a natural choice because decision trees often go very deep, i.e. using many conditions.

The proposed combination method is based on the competition of different classifiers on different segments of the training data. The key idea is to use one classifier to segment the training data, and then choose the best classifier to classify each segment.

5.5 CAAR: Construct Concise and Accurate Classifier by Atomic Association Rules

Atomic rule mining generates the smallest and simplest rule set for classification. The strong atomic rules with the highest and near-highest confidences *can* realize partial classification accurately. Multiple passes of partial classifications generate the concise and accurate classifier.
CAAR[19] also uses two important strategies:

1. Mining the atomic rules to generate the smallest classification rule set.
2. Using strong atomic rules with highest and near highest confidence for accurate prediction.

Class Association Rule

A ***class association rule*** is the rule in which categorical variable is included in the rule's consequent.

Atomic Class Association Rule

An ***atomic class association rule*** is the class association rule in which both antecedent and consequent have only one item. For short, the atomic class association rules are called ***atomic rules.*** An atomic rule can be represented in the form A_x=a $\Rightarrow$ C=c, given **as** AR (A_x,a,c) where $A_x \in$ A, a $\in V[A]$,c $\in V[C]$.

Strong Atomic Rule

A ***strong atomic*** rule is the atomic rule satisfying the minimum support (called min_sup) and the minimum confidence (called min_conf) thresholds. In CAAR, min_conf is set to 0.98 x min_conf where min_conf is the maximum confidence. The rational behind this technique is to extract the strong atomic rules with the highest and near-highest confidences to construct a concise and accurate classification model.

Selectivity

To evaluate whether a dataset is suitable for CAAR an efficient technique is proposed. The mean confidence and support *of* top-10 strong atomic rules are used to measure the selectivity of CAAR. When there are less than **10** strong atomic rules, all of them *are* used for measurement. If they are less then min_sup then that dataset belongs to type –N (Not suitable), otherwise it belongs to type –P(Suitable).

Generation of strong atomic rules

The data structure used for atomic rule generation is a class Counter which uses a 3 dimension array to store the counts of 2 itemsets relevant to the atomic rules. This class is defined in Java language is as follows.

```
public class Counter {

public int [][][] count = new int [ml][nVl][nC];
public void counting (Dataset d)
{
... ………
}
public void counting (Instance inst)
{
• • …..
}
public int getcount(int X, int Vx, int c)
{
...
 }
public int getCount(int X, int Vx)
{
.............
 }
...
}
```

The getCount method of counter can readily get the count of both an item and atomic rule.

The steps to generate strong atomic rules are given **as** follows.

a. Scan the dataset and count the atomic rule-related 2-itemsets occurring in the instances.
b. Extract candidate atomic-rules from the counter.
c. Compute the support and confidence of the candidate atomic-rules and find the maximum confidence max_conf of all candidate atomic-rules.
d. Extract all strong atomic rules. The threshold min_sup is set to 1% and min_conf is set to confCoef * max_conf. The default value of confCoef is 0.98 which was determined by experiments.

Building a CAAR classifier

This algorithm is to generate CAAR classifier as given below. The counting method of the counter scans the dataset-D and counts the all atomic rules related 2-itemset.

NumOfClass function gets the number of classes included in the valid instances in the dataset. AtomicRule-Gen function generates the strong atomic rules using the approach mentioned previously. The Prune rules function prunes the redundant rules and counts the 2-itemset's occurrence in the valid (non-deleted) instances for the next partial classifications.

When the class labels of remainder instances in the dataset are the same, the algorithm generates a special rule which indicates the default class. Finally, combine all obtained classification rules in sequence to form a classifier.

The ***PruneRules*** function has three steps:

a. Sort the strong atomic rules according to the strength order of classifying capacity.

b. Test the strong atomic rules on the dataset. If an instance contains the rule's antecedent item, the rule's test-count increases by 1, and this instance is logically deleted from the dataset D otherwise, count the 2-itemsets in this instance for the next partial classification.

c. Prune the redundant rules.

5.6 MCAR: Post-Bagging and Conviction

Post-bagging algorithm[1] consists of in resampling parts of a classification model rather then the data. With a particular kind of model: large sets of classification association rules and in combination with ordinary best rule and weighted voting approaches. We empirically evaluate the effects of the technique in terms of classification accuracy.

Introduction

One can use an association rule discovery strategy to obtain a large set of rules from a given dataset, and subsequently combine a subset of the rules to obtain a classification model. This two-step training process is typically heavier than building directly a model, such as a decision tree. The motivation for going the long way lies on the possibility for delaying heuristic decisions in model building, while maintaining the scalability of the process. On the other hand, association rules can be seen as Bayesian statements about the data, and can be combined using Bayesian principles in a justified way.

For each sample, a classification model is learnt, and new cases are classified by combining the decisions of the resulting models for the new case. Bagging is therefore an ensemble method that requires a single training data set and a single model generator algorithm. We propose and empirically evaluate a *post-bagging* method. From the training data, we obtain one set of association rules, and from that single set of rules we build a number of (partial) classification models using a bootstrap sampling approach on the set of rules.

Classifications from Association

An association rule discovery algorithm such as APRIORI, takes a set of transactions $D = \{T \mid T \text{ is a set of items } i\}$, a minimal support threshold σ and a minimal confidence threshold φ, and outputs all the rules of the form $A \rightarrow B$, where A and B are sets of items in D and $sup(A \quad B) \geq \sigma$ and $sup(A \quad B)/sup(A) \geq \varphi$. $sup(X)$ is the support or the relative frequency of an item set X observed in D.

Association rule discovery can be directly applied to tabular datasets, such as the typical UCI dataset, with one column for each attribute by regarding each example as a set of items of the form < *attribute* = *value* >. Likewise, continuous attributes can be dealt with if discretized in advance. Despite the fact that an association rule algorithm finds ALL rules that satisfy σ and φ, the discovery process can be relatively fast and discovery time grows linearly with the number of examples.

The discovery of association rules can then be seen as a step preceding model building, or a computationally feasible way of having a quasi-complete search on the space of rules. A classification rule model built from such an unrestrained set of rules can potentially be more accurate than another using a greedy search approach Which is the best way of obtaining a classification model from a set of association Rules.

Obtaining Classifiers from Association Rules

We can regard classification from association rules as a particular case of the general problem of model combination, either because we see each rule as a separate model or because we consider subsets of the rules for combination. We first build a set of rules R. Then we select a subset M of rules that will be used in classification, and finally we choose a prediction strategy π that obtains a decision for a given unknown case x. To optimize predictive performance we can fine tune one or more of these three steps.

1. **Strategy for the Generation of Rules**
2. **Choice of the Rule**
3. **Strategy for Prediction**

Rule Generations

Typically, the generation of association rules is done after the identification of frequent item sets. For efficiency purposes, it is desirable to push the rules generation task into the frequent pattern mining phase. Frequent item set identification is typically done as follows: first, all frequent items are identified, and then candidate item sets are generated following an imposed order.

Rule Selections

Rule selection, or pruning, can be done right after rule generation. However, most of the rule selection techniques can be used earlier when the rules are being generated. Pruning techniques rely on the elimination of rules that do not improve more general versions.

The rule selection method *RC* builds a decision list by traversing the generalization lattice of the rules and by looking at the training error of the rules. It starts with the most general rules, which will be at the bottom of the decision list. After that, it moves to the next level of the generalization lattice and chooses the rules that better handle the exceptions of the more general rules, while discarding the other rules at the same generalization level. This is done iteratively until the bottom of the lattice is reached.

Combining the Decisions of Rules

We can use some prediction strategy to combine the rules in *R*.

1. **Best Rule**
2. **Voting**
3. **Weighted Voting**

Bagging Association Rules for Classification

Bagging is the generation of several models from bootstrap samples of the same original dataset. The prediction given by the set of resulting models for one example *e* is done by averaging the predictions of the different models. Bagging has the effect of improving the results of an unstable classifier by reducing its variance. In the case of decision trees, bagging works because it increases the probability of choosing more complex models.

In the case of classification from association, we obtain a large set of rules *R* that contain many alternative possible models. This technique called *post-bagging* consists of sampling repeatedly the set of rules a posteriori to obtain an ensemble of models similar to bagging. The models in a particular ensemble will be similar, but their differences will tend to reflect the variability of rule sets obtained from the same source of data. New cases are classified by obtaining the prediction of each of the models in the ensemble and using simple voting to combine those predictions.

5.7 HARMONY: Efficiently Mining the Best Rules for Classification

Some of the previous algorithms are suffering from large number of closed itemset; expensive itemset generation and rule generation, this algorithm directly mines the final set of classification rules. HARMONY[18] directly mines for each training instance one of the highest confidence frequent classification rules that it supports, and builds the classification model from the union of these rules over the entire set of instances. It employs an instance-centric rule generation framework and is guaranteed to find and include the best possible rule for each tuple in database.

Rule Enumeration

It uses 'projection-based' item set enumeration for classification rule mining. For a given training database TrDB and minimum support min_sup, it first compute the frequent item of length one and sort them lexicographically and then it applies the divide–and-conquer method plus the depth-first-strategy.

In mining the rules, first item in F1 list is treated as the current prefix P and its conditional database (i.e. rules containing P) $TrDB|_P$ is built and the divide-and-conquer method is applied recursively with the depth-first search strategy.

Ordering of local items

Although there is a lexicographic is a default mechanism for ordering the items yet HARMONY propose 3 new schemes

Maximum confidence descending order

For mining the highest confidence rule, it sorts the local frequent items in frequent items in their maximum confidence descending order.

Entropy ascending order

If the entropy of the set of instances containing P $\cup$ {x} ($1 \leq j \leq m$) is small, it is highly possible to generate some high confidence rules with body P $\cup$ {x}. Thus another good ordering heuristic is to rank the set of local frequent items in their entropy ascending order, and the entropy with respect to item x_j is defined as follows:

$$-\frac{1}{\log k}\sum_{i=1}^{k}\left(\frac{\sup_p^{c1}\cup\left(x_j\right)}{\sup\ _p\cup\left(x_{j1}\right)}\right)\log\left(\frac{\sup_p^{c1}\cup\left(x_j\right)}{\sup\ _p\cup\left(x_j\right)}\right)$$

Correlation coefficient ascending order

More similar the class distribution between conditional databases $TrDB|_P$ and $TrDB|_{P\cup\{x_j\}}$($1 \leq j \leq m$), the lower is the possibility to generate higher confidence rules from $TrDB|_{P\cup\{x_j\}}$. Because the correlation coefficient is a good metric in measuring the similarity between two vectors (the larger the coefficient, the more similar the two vectors), it can be used to rank the local items. In HARMONY, the correlation coefficient ascending order by default adopted to sort the local items.

Pruning

HARMONY proposed some effective pruning method to improve the rule mining process.

Support equivalence item elimination

Given the current prefix P, among its set of local frequent items {x1,x2,..x_m}, some may have the same support as P. We call them support equivalence items and can be safely pruned if

$$\sup_{p\cup(x_j)} = \sup_p$$

Unpromising item elimination

Any rule mined by growing prefix P will have a confidence that is no greater than the current highest confidence rules (with the same rule head) of any conditional instance in TrDB$|_P \cup \{x_j\}$ thus item X_j can be safely pruned.

Unpromising conditional database pruning

For any conditional instance $<t_l, X_l, c_i> \in$ TrDB$|_P$, if the following always holds, the conditional database TrDB$|_P$ can be safely pruned.

$$HCCR_{t1}^{conf} \geq \min\left(1, \frac{\sup_p^{ci}}{\min\_\sup}\right)$$

The Integrated Rule Mining Algorithm

The algorithm firstly initializes the highest confidence classification rules with respect to each training instance to empty, then enumerates the classification rules by calling subroutine ruleminer (TrDB). Subroutine ruleminer takes as input a prefix itemset pi and its corresponding conditional database cdb. For each conditional instance, it checks if a classification rule with higher confidence can be computed from the current prefix pi, if so, it replaces the corresponding instance's current highest confidence rule with the new rule. It then finds the frequent local items by scanning cdb, prunes invalid items based on the support equivalence item pruning method and the unpromising item pruning method. If the set of valid local items is empty or the whole conditional database cdb can be pruned based on the unpromising conditional database pruning method, it returns directly. Otherwise, it sorts the left frequent local items according to the correlation coefficient ascending order, and grows the current prefix, builds the conditional database for the new prefix, and recursively calls itself to mine the highest confidence rules from the new prefix.

Building the Classification Model

For building the Classification Model HARMONY first groups the set of the highest confidence covering rules into k groups according to their rule heads (i.e., class labels), where k is the total number of distinct class labels in the training database. Within the same group of rules, HARMONY sorts the rules in their confidence descending order, and for the rules with the same confidence, sorts them in support descending order. In this way, HARMONY prefers the rules with higher confidence and the rules with higher support if the confidence is the same.

5.8 CorClass: Correlated Association Rule Mining for Classification

CorClass[21] directly finds the best correlated associations rules for classification by employing a branch-and-bound algorithm. It follows the strategy in which calculating the upper bounds on the values attainable by specializations of the rule currently considered. The upper bound finally allows dynamic rising of the pruning threshold, differing from the fixed minimal support used in existing techniques. This will result in earlier termination of the mining process. Since the quality criterion for rules is used directly for pruning, no post-processing of the discovered rule set is necessary.

The CorClass Algorithm

The upper bound allows for two types of pruning w.r.t. the actual mining process. First, the user can specify a significance threshold for χ^2 test. Second, the goal can be to mine for a user specified maximum number k of rules in which case the threshold is raised dynamically. We will only describe the k-best algorithm here since deriving the threshold-based algorithm should be straightforward.

The algorithm starts from the most general rule body (denoted by T). We use an optimal refinement operator (denoted by ρ in the listing). This refinement operator is defined as follows

Optimal Refinement Operator: Let L be a set of literals, $\prec$ a total order on L, $\tau \in$ R.

$$(r) = \{r \wedge l_i \mid l_i \in L, ub_\sigma(l_i) \geq \tau , \forall l \in r : l \prec l_i \}$$

is called an optimal refinement operator. The operator guarantees that all rules can be derived from T in exactly one possible way. So, no rule is generated more than once. Since only literals are added whose upper bound exceeds the threshold, the value of the refinement has a chance of exceeding or matching the current threshold, if $ub_\sigma(l_i) \geq \tau$. In each iteration of the algorithm, the rule body with the highest upper bound is selected for refinement.

Now 2 points have to be discussed

1. Raising the threshold τ dynamically and including the current rule in the temporary solution set S.
2. Including the current rule in the set of promising rules P i.e. the set of candidates for future refinement.

The decision on whether to include the current rule r in the temporary solution set is based on three criteria. First, significance value of current rule has to greater than or equal to the significance threshold. Second, if there already is a rule with the same significance-value than check that, whether it is a generalization of r. So, r is only included if it has a different support, since otherwise it includes at least one literal giving no additional information. If r is included, r_k is removed and the threshold rose to $\sigma(r_{k-1})$. After processing all refinements the set of promising rules is pruned by removing all rules with $ub_\sigma(r_i) < \sigma(r_k)$. Also, all refinements whose upper bound exceeds the threshold are included in the promising set. The algorithm terminates once there is no candidate remaining for refinement. During mining, the rules are already ranked by (1) score, (2) generality, and (3) support. The majority class of all instances satisfying a rule is set as the head. Should not all training instances be covered, the majority class of the remaining ones is set as the default class.

Classification

This algorithm uses two strategies for classifying a new object

1. Decision List

Rank all the rules (rules are ranked by quality according to some criterion) and use the first rule satisfied by an example for classification.

2. Weighted Combination

The general way to do this is to collect all such rules, assign each one a specific weight and for each c_i predicted by at least one rule sum up the weights of corresponding rules. The class value having the highest value is returned.

6. Previous Comparison Studies

Many comparisons have been intra subject comparisons, e.g. within symbolic learning [5] within Statistics and within neural networks. There have been fewer but still many, inter-subject studies involving algorithms form two or more of these fields ,e.g., [5,7,15,17] reported that ID3 was preferable on an engineering control problem to two neural network algorithms. [3] reported back propagation did better than CART. [16] showed that back-propagation outperformed nearest neighbor for classifying.

The comparisons by [7] found that ID3 performed better than discriminant analysis for classifying the gait cycle if artificial limps. [17] reported that ID3 was preferable on an engineering control problem to two neural network algorithms. Many comparisons reported that various neural networks performed similarly to, or slightly better than symbolic and statically algorithms [10] .

7. Experimental Evaluation

***Characteristics of datasets*:** Table includes datasets which are used in experiment to compare these algorithms in terms of Accuracy and generation time. Data sets are taken from the UCI Machine learning Repository , which have been discretised/ normalized.

Datasets	**No. of Classes**	**No. of records**	**Missing values**	**No. of Col.**	**file size in bytes**
adult.	2	48842	6465	15	2034781
anneal	6	898	22175	39	36705
auto	7	205	59	26	17150
breast	2	699	16	11	18126
cylBands	2	540	999	40	56675
glass	7	124	0	11	5946
heart	5	303	6	14	11943
hepatitus	2	155	167	20	8335
horseColic	2	368	1927	28	18303
ionosphere	2	351	0	35	40300
iris	3	150	0	5	2017
led7	10	3200	0	24	61254
mushroom	2	8124	2480	23	539424
pimaIndians	2	768	0	9	18462
pageBlocks	5	5473	0	11	169663
pima	2	768	0	12	18642
ticTacToe	2	958	0	10	25866
waveform	3	5000	0	22	324562
wine	3	178	0	14	7146
zoo	7	101	0	18	4670

Table 1: Characteristics of datasets.

All the experiments are performed on a Pentium(R) 4 CPU 2.40 GHz, 248 MB of RAM. Comparison is based on Accuracy and generation time. Analysis is shown below.

Datasets	CBA		CMAR		CPAR		CARM		MCAR	
	Accuracy	Gen time	Accuracy	Gen. time	Accuracy	Gen time	Accuracy	Gen time	Accuracy	Gen time
adult.	74.5	23.01	72.5	78.1	76.2	80.9	74.2	2.899	74	0.2
anneal	84.2	5.8	82.4	2.3	90.2	1.8	80.1	0.512	82.5	0.45
auto	45	53.3	78	70.9	48	1.2	70.6	3.33	70.2	10.2
breast	93.95	0.89	89.843	0.515	94	0.7	89.98	0.813	90.25	0.3
cylBands	66.85	0.69	62.08	1.208	69.33	0.89	57.77	16.094	68.34	0.42
glass	73.9	0.8	70.1	0.8	74.4	0.57	72.2	0.435	77.7	0.32
heart	81.9	1.42	82.2	0.9	82.96	1	80.4	0.612	80.67	0.24
hepatitus	42.5	1.39	40.06	0.4	74.34	0.31	79.33	7.438	67.78	0.83
horseColic	78.89	0.77	63.73	0.36	81.57	0.57	66.64	0.845	75.68	0.33
ionosphere	90	23.56	91.5	3.989	92.9	1.1	91	2.3	91.78	10.35
iris	94	0.325	93.3	0.359	95.7	0.23	94.8	0.322	95.49	0.13
led7	71.4	0.72	72	0.678	72.24	5.7	67	0.435	71.96	0.23
mushroom	99.9	12.86	99.9	12.86	98.52	10.45	98.65	14.43	93.65	5.12
pimaIndians	69.76	0.8	54.39	0.343	76.24	0.25	65.1	0.844	77.8	0.42
pageBlocks	90.9	2.2	90.1	0.856	76.2	15.5	90	2.234	76.12	1.35
pima	73.62	0.77	56.096	0.203	74.82	0.36	65.1	0.84	73	0.35
ticTacToe	92.6	0.7	92.3	0.312	91.5	0.3	90.14	0.828	92.56	0.42
waveform	72.4	93.4	83.2	16.73	80.9	38.1	67.7	93.4	79.89	32.2
wine	90	11.723	90.5	7.32	92.5	0.32	90.4	11.7	90.5	6.21
zoo	89.9	3.25	96	3.112	95.8	0.2	93.4	3.2	90.89	1.2

Table2: Comparison of Classification Algorithms

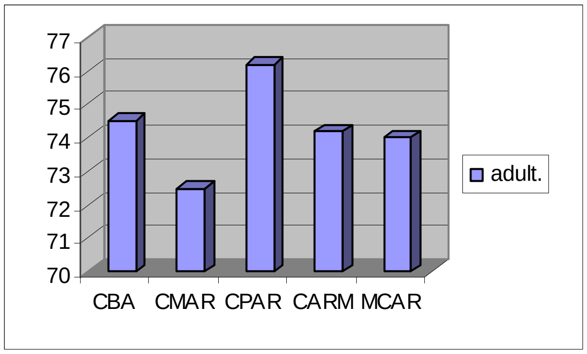


Fig.1 Accuracy comparison of different algorithms for adult dataset.

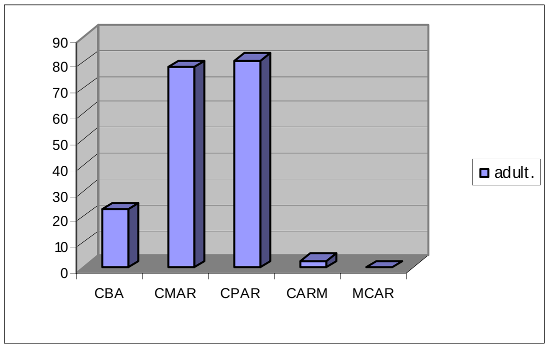


Fig2.Rule Generation time comparison of different algorithms for adult data set

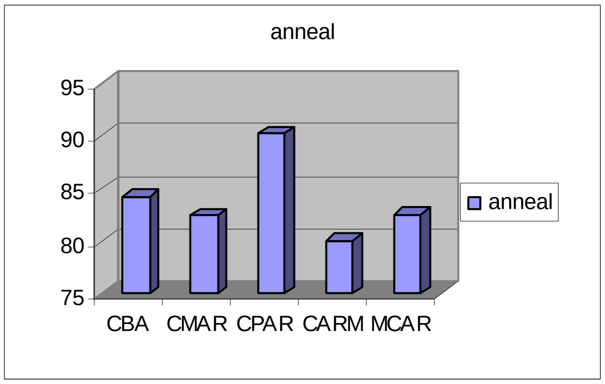


Fig.3 Accuracy comparison of different algorithms for anneal data set

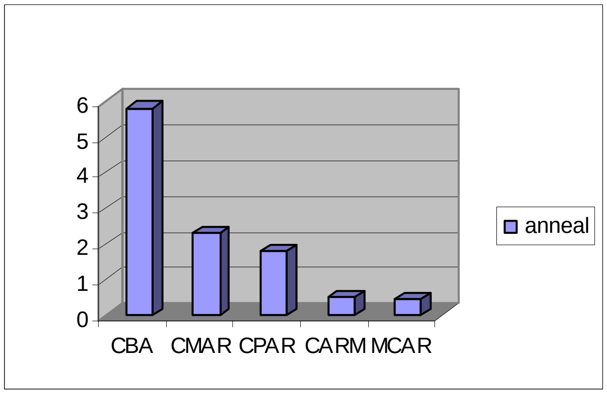


Fig4. Rule Generation time comparison of different algorithms for anneal dataset

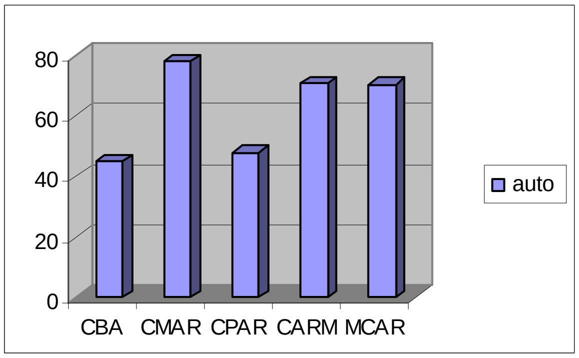


Fig.5 Accuracy comparison of different algorithms for auto data set.

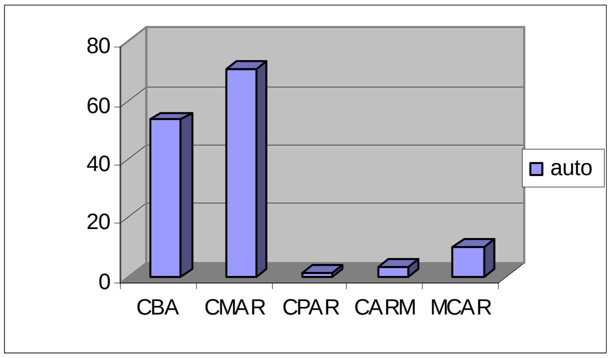


Fig 6. Rule Generation time comparison of different algorithms for auto dataset

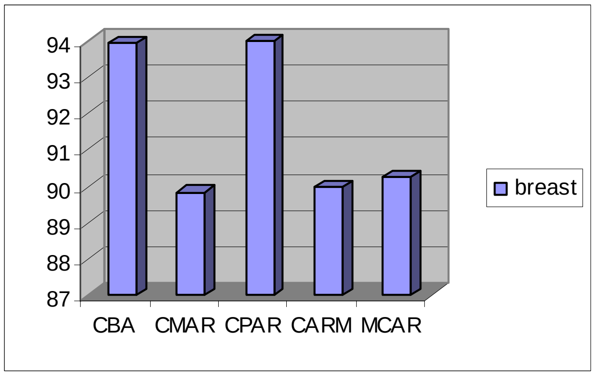


Fig.7 Accuracy comparison of different algorithms for breast data set.

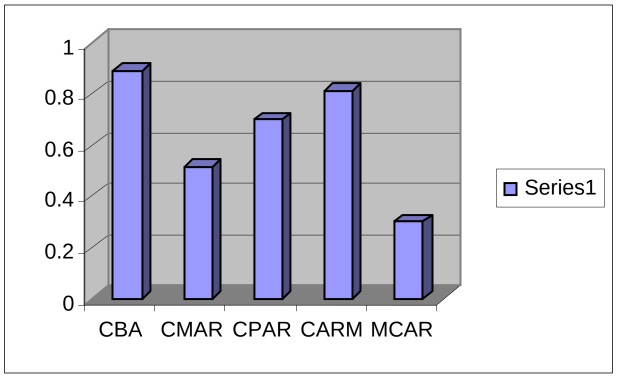
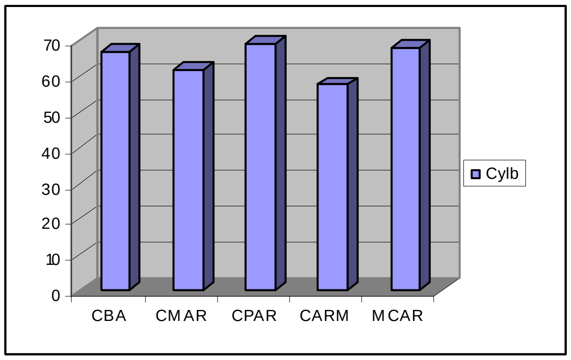


Fig.8 Rule Generation time comparison of different algorithms for breast dataset



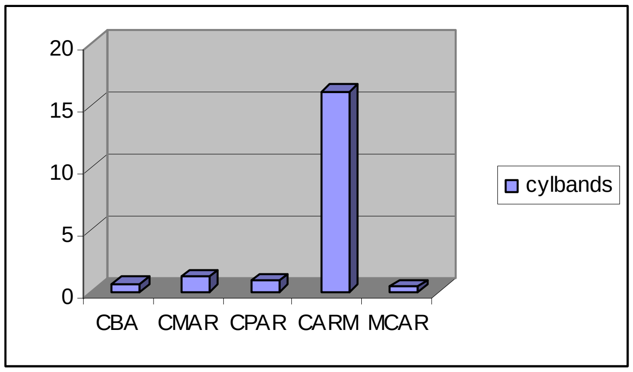
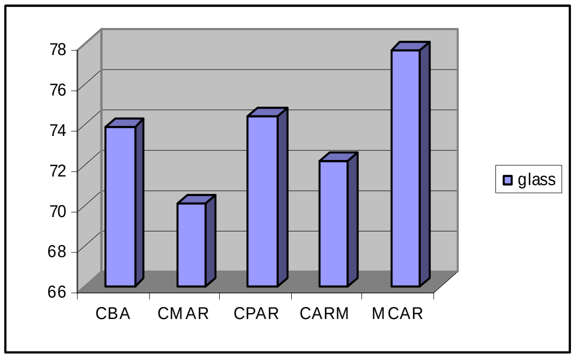


Fig.9 Accuracy comparison of different algorithms for cylBand dataset.

Fig.10. Rule Generation time comparison of different algorithms for cylBands dataset



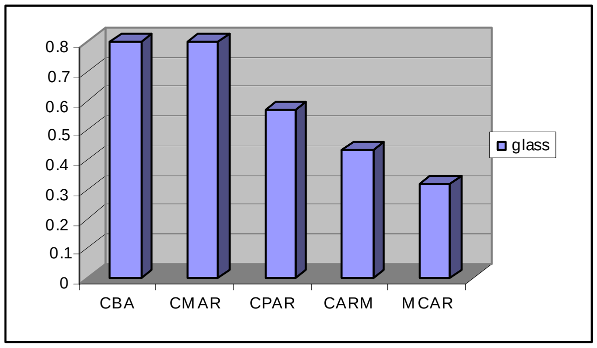
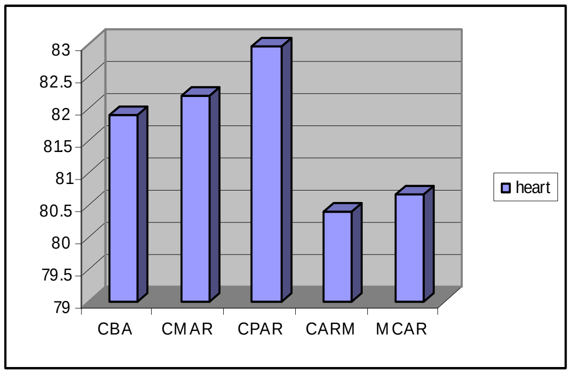


Fig.11. Accuracy comparison of different algorithms different for glass dataset.

Fig.12. Rule Generation time comparison of algorithms for glass dataset



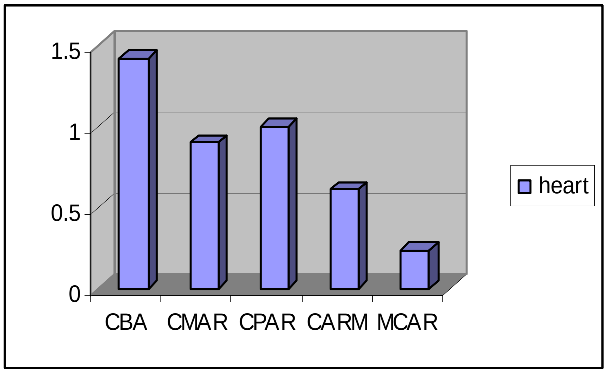


Fig13 Accuracy comparison of different algorithms for heart dataset.

Fig.14. Rule Generation time comparison of different algorithms for heart dataset

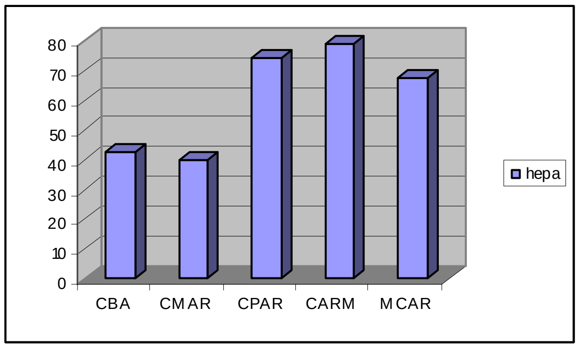


Fig15 Accuracy comparison of different algorithms for hepatitis dataset.

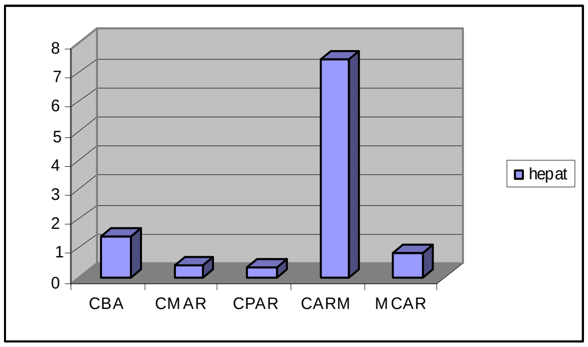


Fig 16. Rule Generation time comparison of different Algorithms for hepatitis dataset

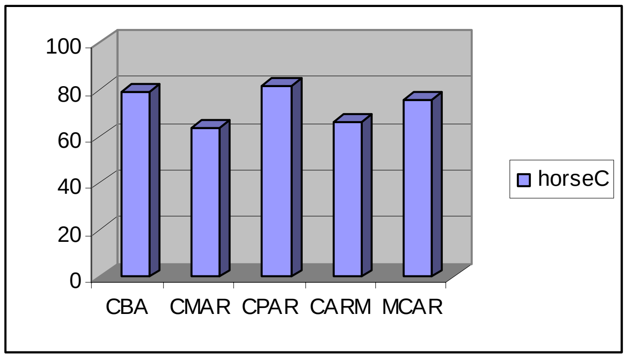


Fig17 Accuracy comparison of different algorithms for horsecolic dataset.

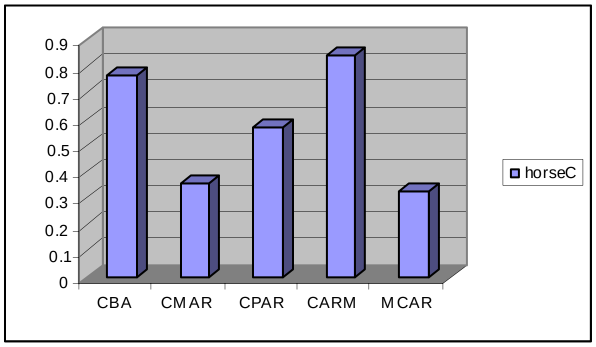


Fig 18. Rule Generation time comparison of different algorithm for horsecolic dataset

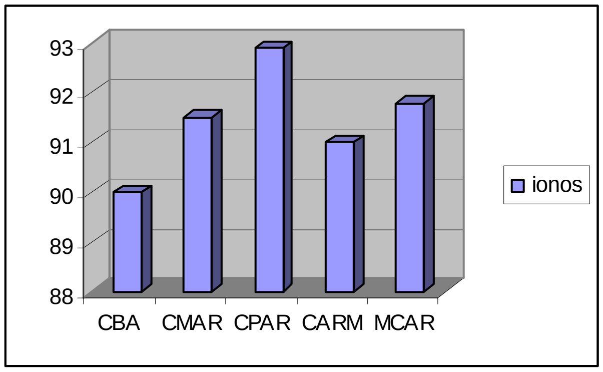


Fig19Accuracy comparison of different algorithms for ionosphere dataset.

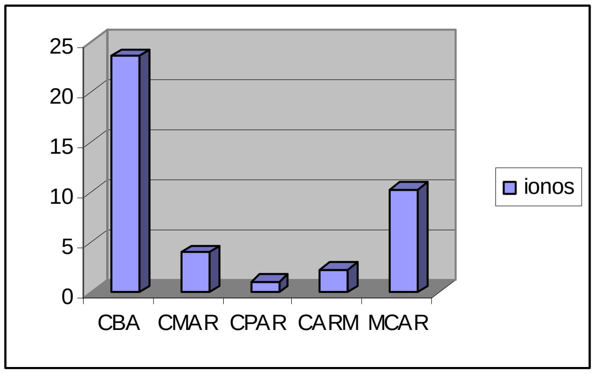


Fig 20 Rule Generation time comparison of different algorithms for ionosphere dataset.

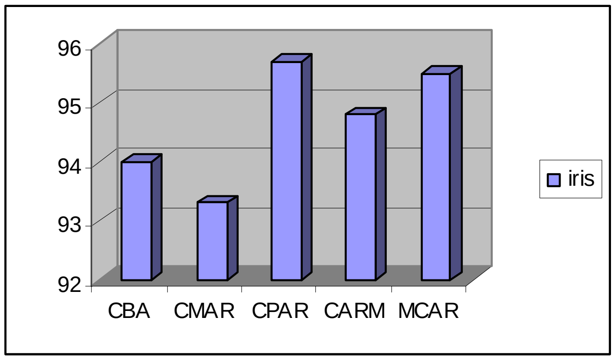


Fig21Accuracy comparison of different algorithms for iris dataset.

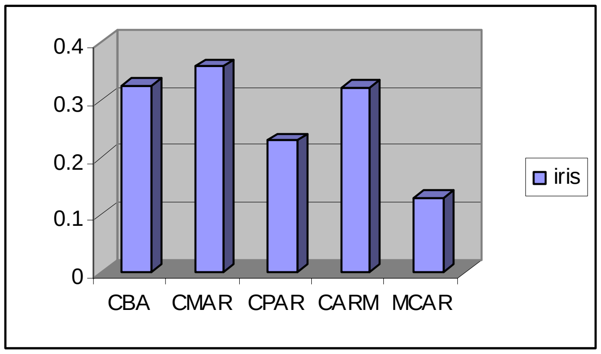


Fig 22 Rule Generation time comparison of different algorithms for iris dataset.

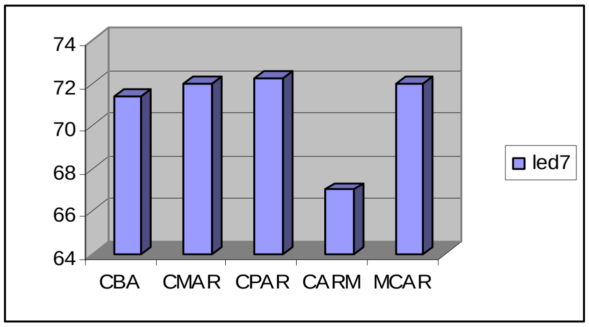


Fig23Accuracy comparison of different algorithms for led7 dataset.

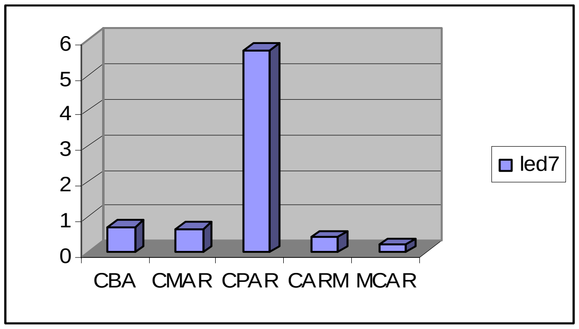


Fig 24 Rule Generation time comparison of different algorithms for led7 dataset.

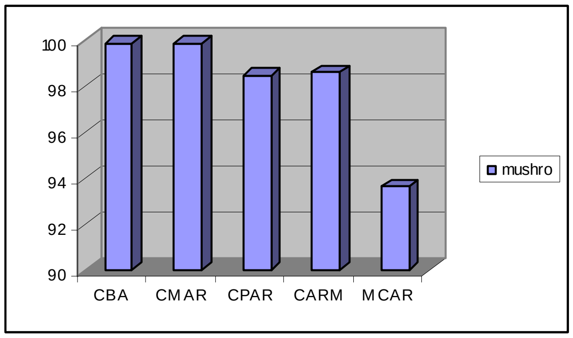


Fig25Accuracy comparison of different algorithms for mushroom dataset.

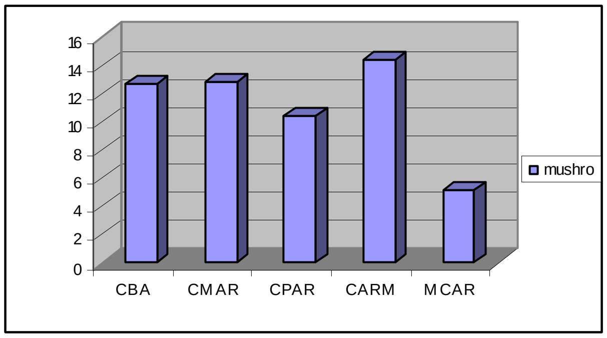
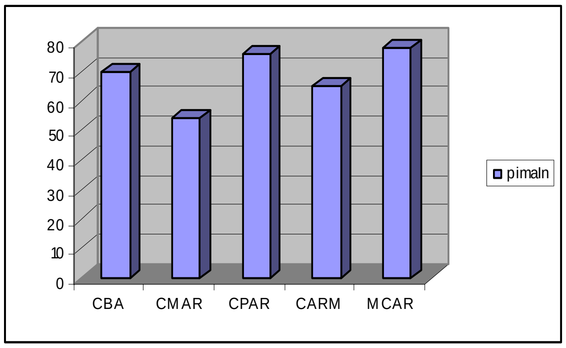


Fig 26 Rule Generation time comparison of differ algorithms for mushroom dataset.



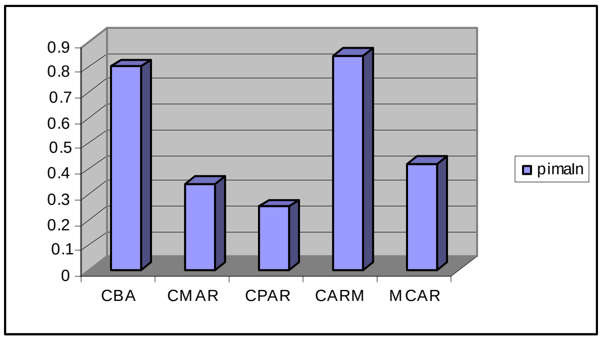
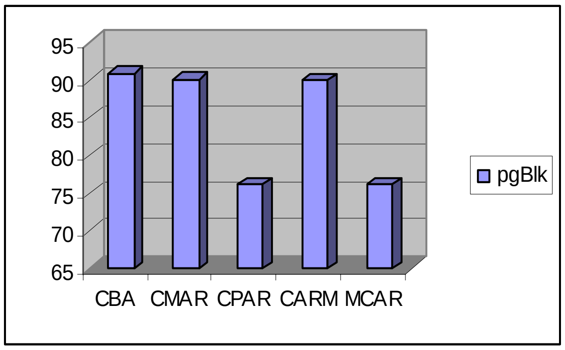


Fig27Accuracy comparison of different algorithms for pimaIndians dataset.

Fig 28 Rule Generation time comparison of differ Algorithm for PimaIndians dataset.



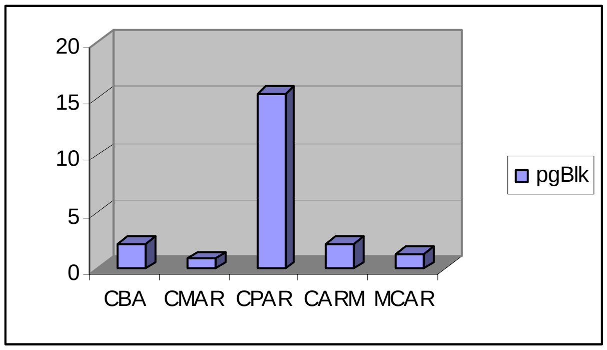
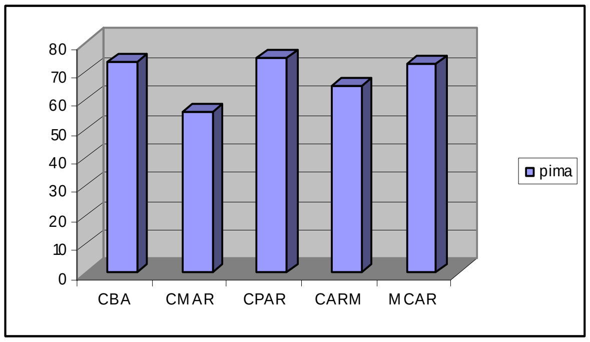


Fig29Accuracy comparison of different algorithms for page block dataset.

Fig 30 Rule Generation time comparison of differ algorithms for Page block dataset.



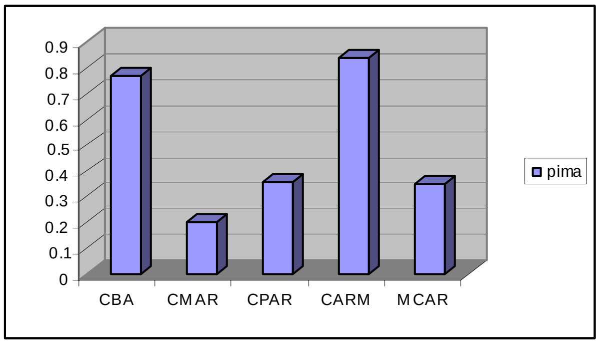


Fig31.Accuracy comparison of different algorithms for pima dataset.

Fig32 Rule Generation time comparison of differ algorithms for pima dataset.

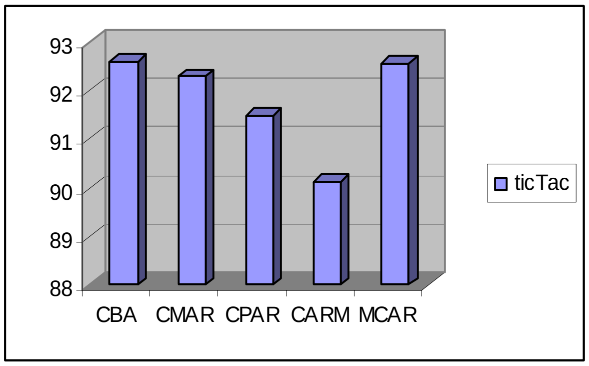


Fig33.Accuracy comparison of different algorithms for ticTacToe data set.

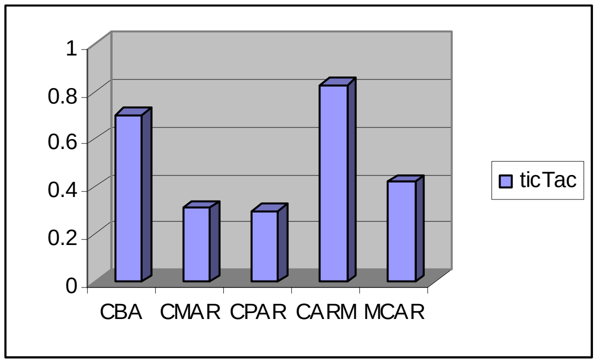


Fig34 Rule Generation time comparison of differ algorithm for ticTacToe dataset.

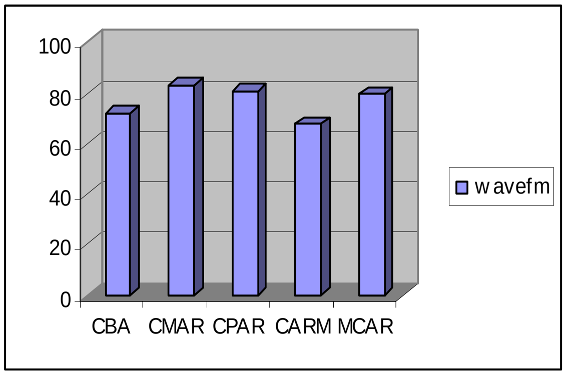


Fig35.Accuracy comparison of different algorithms for waveform dataset.

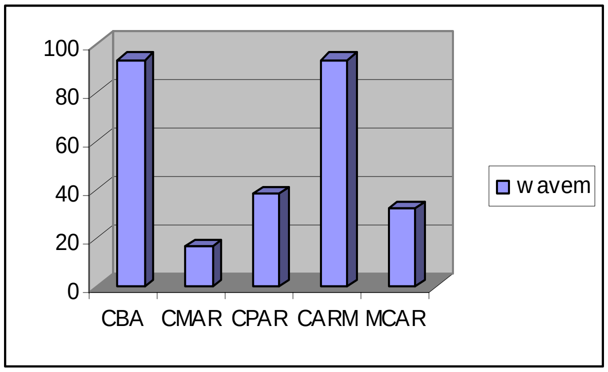


Fig 36 Rule Generation time comparison of differ algorithm for waveform dataset.

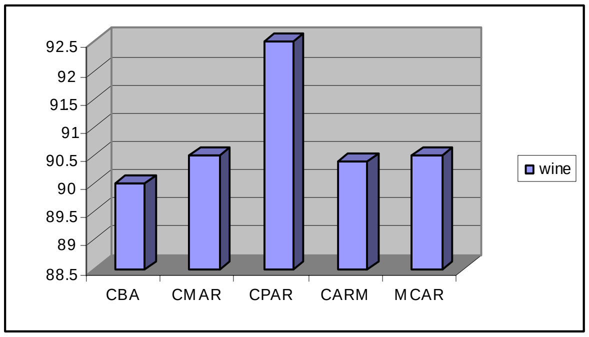


Fig37.Accuracy comparison of different algorithms for wine dataset.

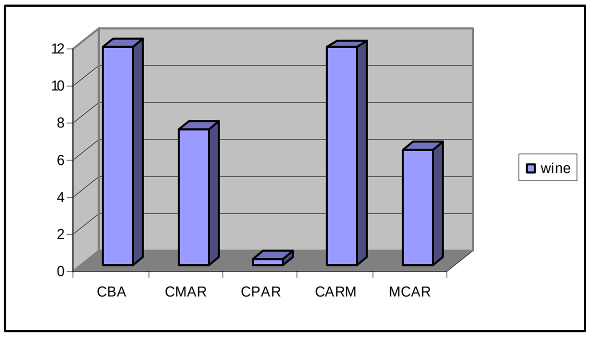


Fig 38 Rule Generation time comparison of differ algorithms for wine dataset.

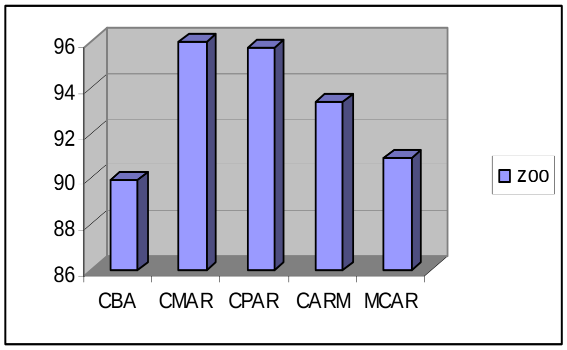


Fig39.Accuracy comparison of different algorithms for Zoo dataset.

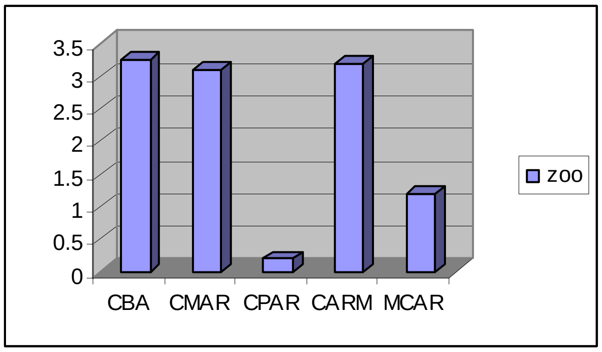


Fig 40 Rule Generation time comparison of differ algorithm for zoo dataset.

This table includes the comparison of different Classification algorithms in terms of number of generated rules.

Datasets	**CBA**	**CMAR**	**CPAR**	**CARM**	**MCAR**
	No. of Rules	No. of Rules	No. of Rules	No. of Rules	No. of Rules
adult.	228	236	158	288	332
anneal	34	38	30	42	58
auto	54	65	48	64	98
breast	49	48	42	52	86
cylBands	78	79	68	73	102
glass	27	32	23	38	79
heart	52	55	52	58	98
hepatitus	23	32	22	34	76
horseColic	97	121	78	89	115
ionosphere	45	54	43	47	139
iris	32	33	31	32	79
led7	71	76	60	71	119
mushroom	89	93	70	91	105
pimaIndians	45	48	34	55	88
pageBlocks	84	85	75	89	98
pima	79	82	58	78	154
ticTacToe	68	72	45	76	123
waveform	386	389	298	378	452
wine	45	43	37	65	112
zoo	97	93	67	98	198

Table 3. Comparison of Classification Algorithms Based on number of rules

In this analysis Classification Algorithm is shown which made a significant contribution in the classification rule mining .Some popular Associative Classification techniques that are CBA, CMAR, CPAR, CARM and MCAR has been compared in terms of Classification Accuracy and Generation time and number of rules generated in Classification . Experiments were conducted using java 1.4 version.

The support threshes hold play a major role in over all classification accuracy, in this experiment it is noticed that the support rate between 1 to 5 usually achieve the best balance between accuracy rates and the size resulted of classifiers therefore the minimum support was set to 1 and confidence threshold set to 50%.

Classifying new instance CBA simply select a rule with highest confidence and support only which is not right always but CMAR and CPAR are using multiple rules in classifying new object.

This table represents the Accuracy of classification systems CPAR have slightly better accuracy then CBA and CMAR CARM and MCAR is more faster then CBA ,CMAR, CARM and CPAR. Here the goal is to determine the one that generates accurate classifiers. CPAR represents a new approach towards efficient and high quality classification System. MCAR algorithm also derived slightly more accurate classification system then CBA and CPAR.

In terms of number of rules MCAR generates much more rules then other Algorithms but CPAR generates slightly less number of rules then CBA and CMAR .The increase in accuracy suggests that due to small increase in the number of generated rules by MCAR, which not simply over fit.

Impact of Larger Dataset

In this section Impact of larger datasets on different algorithm is analyzed. To analyze efficiency and scalability of algorithms on much larger dataset, datasets are increased by itself by copying the data in one time or two times and so on. Here we increased our data set up to 50 times. Performance of algorithms (CBA, CMAR, CPAR, CARM, MCAR) with blown up of waveform dataset is analyzed here. Waveform data set is selected here because it is a large dataset with 5000 records. Data set size is increased up to fifty times and accuracy performance of algorithms is analyzed each time by executing algorithms.

Data size increased(waveform)	CBA	CMAR	CPAR	CARM	MCAR
1	68.82	71.44	70.66	74.9	74.12
2	69.39	71.32	72.36	75.7	74.35
3	69.56	71.6	73.55	76.1	73.89
5	69.53	71.88	71.88	77.9	73.14
10	69.58	71.52	74.52	78.7	73.45
15	69.53	71.4133	74.08	78.5	73.25
20	69.66	71.52	74.6	78.7	73.12
40	69.7	71.52	74.64	78.6	73.45
50	69.72	71.52	76.4	78.7	74.59

Table 4. Accuracy of different algorithms with blown up dataset

Data size increased(waveform)	CBA	CMAR	CPAR	CARM	MCAR
1	2.77	55.234	6.64	83.41	1.22
2	5.02	58.125	17.33	92.562	1.55
3	7.28	64.984	29.66	119.688	1.71
5	12.35	72.375	58.23	133.672	2.31
10	22.92	93.391	80.3	122.079	5.63
15	33.94	105.078	132.99	137.157	11.12
20	47.63	128.625	394.52	131.14	20.14
40	93.52	187.156	3396.2	160.23	39.49
50	115.09	230.09	5630.1	174.766	85.31

Table.4 Rule generation time of different algorithms with blown up dataset

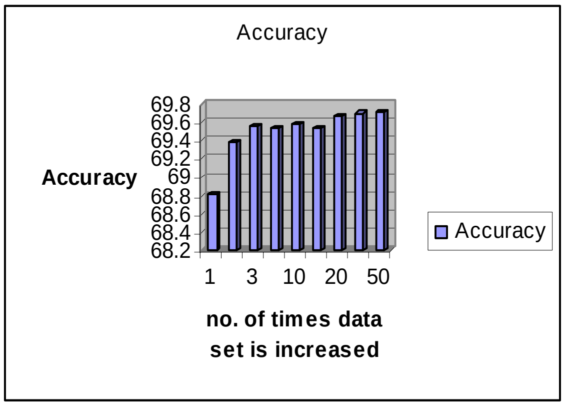


Fig.41 Impact of blown up wave form dataset on accuracy of CBA algorithm.

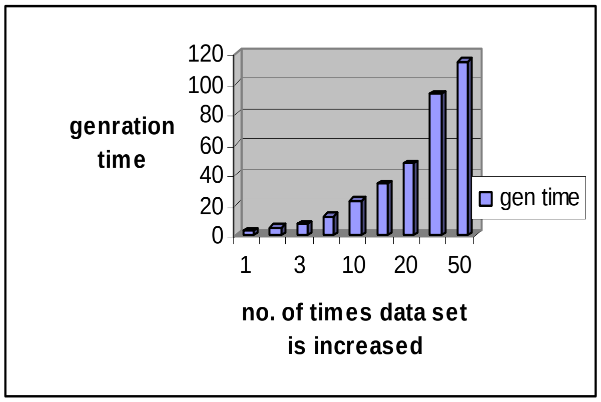


Fig 42 Impact of blown up waveform Dataset on generation time of CBA algorithm.

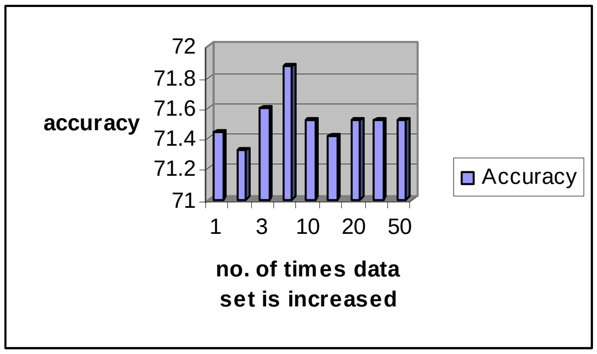


Fig 43 Impact of blown up wave form dataset on accuracy of CMAR algorithm.

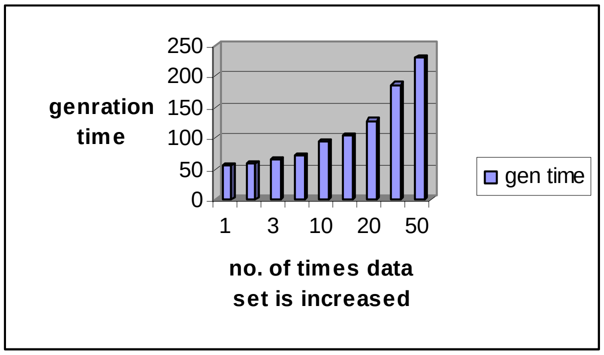


Fig.44 Impact of blown up waveform dataset on generation time of CMAR algorithm.

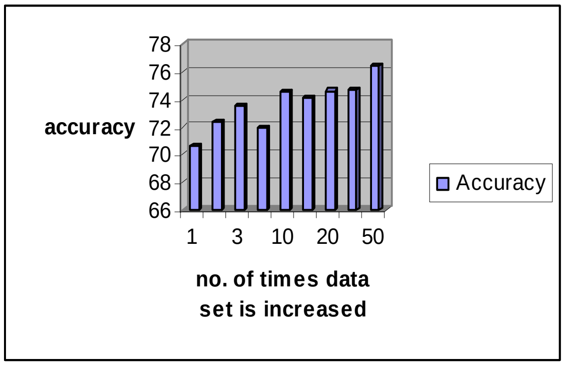


Fig 45 Impact of blown up wave form dataset on accuracy of CPAR algorithm

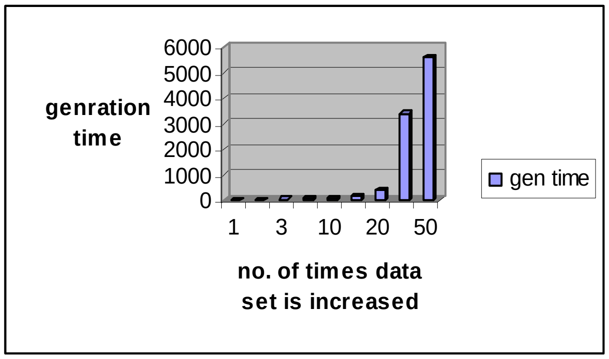
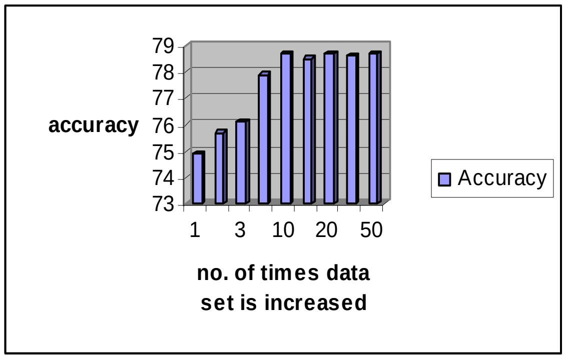


Fig.46 Impact of blown up waveform data set on generation time of CPAR algorithm



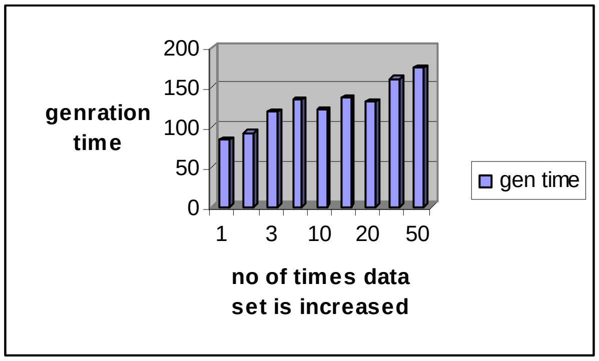
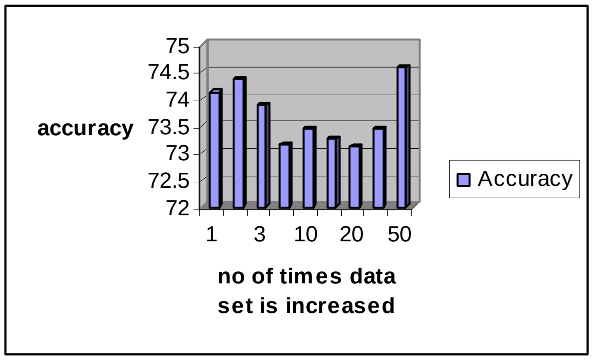


Fig.47 Impact of blown up wave form dataset on accuracy of CARM algorithm

Fig.48 Impact of blown up waveform dataset on generation time of CARM algorithm



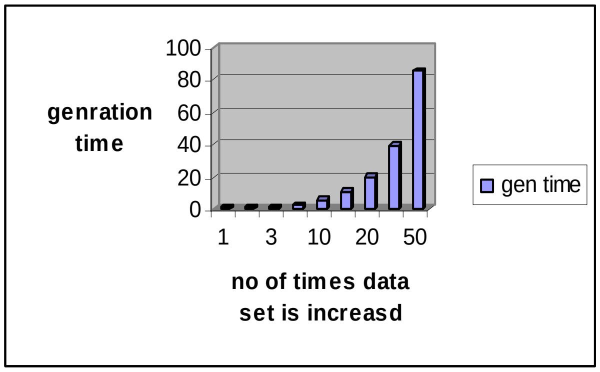


Fig.49 Impact of blown up wave form dataset on accuracy of MCAR algorithm

Fig.50 Impact of blown up waveform data set on generation time of MCAR algorithm

In this section the impact of blown up data set is analyzed .In accuracy comparison with blown up data set CPAR generates more accurate results then CBA and CMAR algorithms .we can say that CPAR can used for large dataset also but it takes much time to generate rules. CMAR also takes more time then CBA but the accuracy percentage of CBA is quite low with large data sets and CARM is also not much bad with accuracy but generation time of MCAR is quite low then CARM, MCAR takes less time to generate rules rather then other algorithms but the number of rules generated by MCAR is more then CPAR. In terms of Accuracy and less number of rules CPAR is preferred to use with large data sets.

8. Conclusion and Further Work

Previous Studies have rarely considered more than handful of different algorithms and these are often restricted to one or two classes of algorithms. Ripley examined ten varied algorithms and their variations. [6] examined nine different algorithms (linear discrenation, quadratic discrimination, nearest neighbor , naïve Bayes, Bayes 2nd order , back propagation, PVM, Assostant and CART), but did not include any statistical methods. [6] looked at four datasets (including two artificial datasets).

Just as there are complications caused by different variations of algorithms, there are complications caused by variations of datasets for example, there are several “thyroid” data sets. At the 1989 International Workshop on Machine Learning [14], there were several papers giving the accuracy for the ID3 on the thyroid data: each was different.

Data sets are also commonly artificially created and investigated the performance of six statistical discriminates applied to data with a mixture of binary nominal ordinal and continuous attributes. [15] studied the MONK’s problem involving simulated robots classified in to classes using six attributes. Using artificial data has the advantage that conditions can be altered at will, but the disadvantage is that in defining the class and type of noise one almost defines the best algorithms for finding the class. To judge on the empiric ability of algorithms on large real world problems, it is better to use large real world data.

There have also been various different measures of success and testing methodologies. The most common measure of success has been prediction accuracy or error rate. Prediction accuracy is normally measured on a separate test set. Traditionally this is 30% of the total data-although there appears to be no strong reasons for using this split. Cross-validation should be used, or if the dataset is small one output should be used. If the data sets are very small the bootstrap method can be used. [8] Describes these various techniques.

Learning speed is also studied. All previous study found that back-propagation was slower than decision tree algorithms If the speed of classification is important; a nearest neighbor would be slow. One other important measure of success of an algorithm is the understandability of the classification function/rule.

The trouble in testing this is that the result of symbolic learning are more human friendly than the black box result of statically and neural network algorithms. The main focus of the experiment is on the rule mining algorithms to analyze a much accurate and efficient algorithm which takes less time to generate rules and generates less number of rules using association rule mining approach. Apart from the mining algorithms we have also explored which classification schemes are preferable in classification using association rules. This evaluation from a classification perspective includes a comparison with standard machine learning techniques.

10. References

[1] P.J.A. Al´ýpio M. Jorge, An experiment with association rules and classification: post-bagging and conviction, In Proceedings of the 8th International Conference on Discovery Science, Singapore on Discovery Science (2005).

[2] M.-L. Antonie, O.R. Z, An Associative Classifier based on Positive and Negative Rules, in: Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (Paris, France, 2004).

[3] Atlas, Performance comparisons between backpropagation Network Classification Trees On Real Time World Application, In proceeding of IEEE, 78(10):1614-1619 (1990D).

[4] Q.a. Cameron-Jones, FOIL: First Order Inductive Learner, In Proceedings of the 6th European Conference on Machine Learning (1993).

[5] Clark and Boswell Rule Induction with CN2: Some Recent Improvements, In international Work Shop on Machine Learning (1991).

[6] J.H.G.P.L.D. Fisher, Model of incremental concept formation, In journal of Artificial Intellegence (1989).

[7] Kirkwood, Decision Analysis Application in the Operation Research, In international Work Shop on Machine Learning (1998).

[8] W.a. Kulikowski, Data Mining with Decision Trees and Decision Rules, (1991).

[9] W. Li, J. Han, J. Pei, CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules, Proceedings of the 2001 IEEE International Conference on Data Mining (2001) 369-376.

[10] H.a. Lippmann, Neural Net and Traditional Classifiers, In international Work Shop on Machine Learning (19987).

[11] B. Liu, W. Hsu, Y. Ma, Integrating Classification and Association Rule Mining, in: A. Press. (Ed.), In Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98). (Menlo Park, CA, 80–86, 1998) 443 - 447.

[12] B. Liu, Y. Ma, C.K. Wong, Improving an Association Rule Based Classifier, in: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (Lyon ,France, 2000) 504-509.

[13] A.R.a.S. R, Fast Algorithms for Mining Association Rules, (September 1994).

[14] Serge, The accuracy for the ID3 on thyroid data, In international Work Shop on Machine Learning (1989).

[15] Thurn, The Monk's problem :A Performance Comparison of Different Learning Algorithms, In international Work Shop on Machine Learning (1991).

[16] J.T.A.S. Tomita, implementation of Backpropogation Nueral Netwoks on Large Parallel Computers, In international Confrence on Machine Learning (1988).

[17] Tsaptsinos, analysis for classifying the gait cycle if artificial limps, In international Work Shop on Machine Learning (1990).

[18] J. Wang, G. Karypis, Harmony:Efficiently Mining The Best Rules For Classification, in: Appears In Siam 2005 Data Mining Conference (2005).

[19] X.-Y. XU, G.-Q. HAN, H.-Q. MIN, CONSTRUCT CONCISE AND ACCURATE CLASSIFIER BY ATOMIC ASSOCIATION RULES, in: Proceedings of the Third International Conference on Machine Learning and Cybernetics, (IEEE, Shanghai, 2004).

[20] X. Yin, J. Han, CPAR: Classification based on Predictive Association Rules, in: Proceedings of the Third SIAM International Conference on Data Mining, , . SIAM 2003, (San Francisco, CA, USA, 2003).

[21] A. Zimmermann, L.D. Raedt, CorClass: Correlated Association Rule Mining for Classification, in: Proceedings of 7th International Conference, Discovery Science 2004 (Padova ,Italy, 2004).