TITLE OF THESIS – LC AN EFFECTIVE CLASSIFICATION BASED ASSOCIATION RULE MINING ALGORITHM

FULL NAME OF AUTHOR – QAZAFI MAHMOOD

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy

The University of Huddersfield

Submission date as May 2014

Copyright statement

Abstract

Classification using association rules is a research field in data mining that primarily uses association rule discovery techniques in classification benchmarks. It has been confirmed by many research studies in the literature that classification using association tends to generate more predictive classification systems than traditional classification data mining techniques like probabilistic, statistical and decision tree. In this thesis, we introduce a novel data mining algorithm based on classification using association called “Looking at the Class” (LC), which can be used in for mining a range of classification data sets. Unlike known algorithms in classification using the association approach such as Classification based on Association rule (CBA) system and Classification based on Predictive Association (CPAR) system, which merge disjoint items in the rule learning step without anticipating the class label similarity, the proposed algorithm merges only items with identical class labels. This saves too many unnecessary items combining during the rule learning step, and consequently results in large saving in computational time and memory.

Furthermore, the LC algorithm uses a novel prediction procedure that employs multiple rules to make the prediction decision instead of a single rule. The proposed algorithm has been evaluated thoroughly on real world security data sets collected using an automated tool developed at Huddersfield University. The security application which we have considered in this thesis is about categorizing websites based on their features to legitimate or fake which is a typical binary classification problem. Also, experimental results on a number of UCI data sets have been conducted and the measures used for evaluation is the classification accuracy, memory usage, and others. The results show that LC algorithm outperformed traditional classification algorithms such as C4.5, PART and Naïve Bayes as well as known classification based association algorithms like CBA with respect to classification accuracy, memory usage, and execution time on most data sets we consider.

Table of Contents

List of Figures

List of Tables

Dedications and Acknowledgements

First of all, I would like to dedicate this research to my ALLAH, whose mercy, blessings and grace has led me to this important moment of my life.

I would like to express my special thanks to my supervisors Dr Fadi Thabtah and Professor Lee McCluskey for their encouragement, patience, support and guidance throughout this research project.

Many many thanks to my father who is always an inspiration for me and mother for her prayers, ongoing support and advice, and my wife, who has been supportive towards me during this period of life.

Special thanks to my friends especially, Munir, Umair, Misbah, Bilal and Ahsan for their support. And many thanks to all my brothers who encouraged me and were supportive during the PhD study.

List of abbreviations

AC	Associative Classification
AUC	Area under the ORC curve
ADT	Association based Decision Tree
AREM	Associative Classification Regression Model based on EM Algorithm
BART	Bayesian Additive Regression Trees
CAEP	Classification by Aggregating Emerging Patterns
CART	Classification and Regression Trees
CAR	Class Association Rule
CBA	Classification based on Association Rule
CMAR	Classification based on Multiple Class-Association Rules
CPAR	Classification based on Predictive Association Rules
DIC	Dynamic Itemset Counting
DHP	Direct Hashing and Pruning
EP	Emerging Patten
ECMC	Evolving Clustering method for Classification
FP	Frequent Pattern
IREP	Incremental Reduced Error Pruning
L^3	Live and Let Live
LB	Large Bayes
LR	Logistic Regression
MCAR	Multi-class Classification based on Association Rule
MCMC	Markov chain Monte Carlo
MMAC	Multi-class, Multi-label Associative Classification
OneR	One Rule

PECM	Phishing Evolving Clustering Method
PST	Prefix Stream Tree
REP	Reduced Error Pruning
RF	Random Forest
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
RMR	Ranked Multi-label Rule
TF/IDF	Frequent inverse document frequency

Chapter 1

Introduction

1.1 Motivation

Storage capabilities have developed tremendously in the past years, hundreds and thousands of files can be saved in a micro memory card. This innovation has tempted and made it feasible for small and large organizations to keep a record of minute details about their customers, like Tesco, ASDA and Morrison's, etc. in retail market businesses and banks and financial institutions. The transactions of the daily sales in a retail store are often called the market basket data (Agrawal, 1993). The retail stores collect and store the daily purchases and information of all customers. Finding the correlations between the items purchased by the customers and the features of the customers in different geographical areas can be very useful in deciding the marketing strategy and launching targeted promotions.

Assume a planning and marketing department in a retail business organization decided to introduce a store card for their customers. Applications are collected and decisions have to be made for each customer, whether to issue a store card and how much credit limit would be recommended for each customer. The decision from the management of the store needs to take into account the shopping frequency of the customer, geographical area where he lives, and the total amount of purchases in a period of time. These associations can be extracted from the data by the application of data mining approaches.

Another example is a medical organization that keeps the record of all the patients with different diseases and symptoms. The information is very beneficial when the associations are extracted between different symptoms. The extracted information is used to narrow down the possibility of a patient's disease by entering all the symptoms of a new patient in an automated system. For instance, the data contain a large number of entries with 'fever' and 'shivering' referring to the possibility of a malarial disease.

An example of associations in a retail supermarket could be assumed as "80% of the customers who have bought a smart phone also have purchased a screen protector", and "60% of the customers who have purchased milk will likely to buy bread as well". For example the

percentage of transactions containing “smart phone and screen protector” is 3%, and the total transactions of “milk with bread” are 10% in the transactional database. The customers who will buy ‘smart phone’ or ‘milk’ are the antecedent of an association rule and ‘screen protector’ or ‘bread’ is the consequent of the rule. The confidence of a rule represents its strength and in the example above “80% “means that if 100 customers have purchased a smart phone, 80 of them have also bought screen protector. The confidence for the other rule is 60%, whereas “3%” and “10%” represent a significant statistical measure known as support of the rules.

In the real world classification data such as medical diagnoses or market basket analysis, the problem is to discover rules from the data set of historical transactions. The rules generated must have significant support and confidence (frequencies of attribute values above user’s thresholds denoted as minimum support and minimum confidence). A subset of the produced rules is selected to build a model that is able to predict the outcome of the new set of attributes in a database or previously unseen data. The approach that uses association rules to build-up classification systems (classifiers) is known as associative classification (AC) (Li et al., 2001), (Hao et al., 2009), (Sangsuriyu et al., 2010), (Lucas et al., 2012), (Jiang and Karyros, 2012) and (Jabbar et al., 2012).

The AC approaches tend to explore all the linkages or the associations between the values of the attributes (items) and their relative classes in the data set and produce classifiers. The classifiers generated by AC algorithms are usually more accurate with respect to classification accuracy than classic rule based classification approaches like decision trees (Quinlan, 1979) and which tend to generate relatively small size classifiers. The AC approaches have made inroads in extracting such knowledge that is missed by traditional classification methods and have also improved the accuracy rate of different applications.

AC is beneficial in many applications and some of them are described in the above sections like decisions to issue store card, medical diagnosis and making a decision to supply utility supply. It also includes the fraud detections in insurance and credit cards for financial institutions, text categorization and especially in detection of phishing in emails. Likewise the decision of the potential disease according to the patient’s symptoms, a decision to assign a journal or a description in text to one or more possible categories can also be done by AC mining. There are large numbers of journals in a digital library and assigning them to their specific category is a time consuming and tedious task when done by humans. But the process of

assigning journals to particular category or categories becomes very efficient and convenient when a classifier is set in the automated system.

The process of producing the complete set of rules needs substantial CPU time as it requires many data set scans and itemsets joining during the training phase (Baralis, et al., 2004; Li et al., 2001). Hence it is very important to use a fast and effective method for rule discovery that can improve the training time and memory usage. The classical AC techniques like CBA (Liu et al., 1998) and CBA (2) (Liu et al., 1999) may not be able to deal with the increase in the database size and the dimensions of the datasets, meaning adding more attribute columns in the data base. These techniques produce many rules and the process of rule generation can be complex and exhaustive, because these consider all the attribute values while joining them. Therefore it is essential to derive the rules that will be used in the prediction process in reasonable time and at the same time using less memory resources, to help decision makers to plan without waiting.

Use of AC approaches or techniques are used successfully in real world applications in the recent years like heart disease prediction (Jabbar et al., 2012), based on the data collected from the patients response for treatment of Hepatitis C, (Enas et al., 2012) has implemented a predictive system by using AC approaches and also successful application of AC in artificial immune systems is done by (Samir, 2012).

Phishing is currently an interesting research topic for the authorities dealing with the money transfer on the web. Researchers are trying to explore the possible nature of the phishing attacks by using different intelligent classification and other traditional techniques that are based on human experience. Now, since classification using association rule is proven to be effective and accurate classification approach in data mining according to several research studies (Tang and Liao, 2007), (Li et al., 2008), (Hao et al., 2009), (Sangsuriyun et al., 2010), (Jabbar et al., 2012), (Samir, 2012) and (Enas et al., 2012). We believe that there is a need to develop and implement an AC technique that will explore the associations in security data based on websites and emails features in order to alarm end-user of possible phishing attacks. The AC algorithms have shown excellent predictive accuracy in detecting phishing in websites as compared with other classification techniques and this fact is also studied by (Aburrous et al., 2010) and (Al Ajlouni et al., 2013). The use of AC to detect phishing in websites is relatively latest research and the main motivation in the thesis to develop an AC algorithm that is fast and effective and can be used in prediction the phishing websites accurately.

In this thesis, an effective and efficient AC algorithm is proposed for extracting useful knowledge missed by classic rules based classification algorithms for the problem of emails and websites categorization. This is since the proposed algorithm considers efficiently all possible correlations between the websites features and the class in the data set. The proposed algorithm improves upon a known AC algorithm called CBA in three main steps to handle appropriately the problem of websites features prediction:

The training step: Unlike CBA algorithm which employs Apriori (Agrawal, 1993) candidate generation function that works by repeatedly merging disjoint candidate itemsets without considering the class attribute to discover the rules. The proposed algorithm significantly improves the training phase of CBA by reducing the number of itemsets and as a consequence training time as well as memory allocation gets minimized. This has been accomplished by looking at class of candidate disjoint itemsets before merging them and using the minority rule.

Building the classifier phase: The proposed algorithm implements a new ranking method based on the confidence, support, and cardinality, different for the method used by CBA of confidence , support and the order of rule generated. The pruning phase of our algorithm uses the same method of database coverage.

Predicting test data: Unlike the majority of current AC algorithms that use one single rule from the classifier to assign class to test cases. The proposed algorithm makes use of group of rules to make the class assignment of test data. This surely improves upon the classification accuracy of the classifiers produced by the proposed algorithm over other AC algorithms since more rules are cooperating in making the prediction.

1.2 Main Aim of the Thesis

The main aim of the thesis is to contruct an effective and efficient novel AC algorithm based on the CBA with specific focus on improving the rule generation, rule ranking and prediction phases and effectively applying this novel AC algorithm on the extracted features based on some set of rules from the large pool of phishing and legitimate websites, so to efficiently detect and predict the outcome of the new website features whether it is a phishing website or legitimate.

Considering all the features in experimental study would be a complex problem, the aim is to select some set of features from the phishing and legitimate websites and extract them by applying some rules. The selected features are then divided into four groups of data sets naming Abnormal Based features, Address Base features, HTML Base features and Domain based features. The classification accuracy of the phishing datasets will be studied and discussed in detail when novel LC algorithm and other AC and classification algorithms are applied.

1.3 Issues Addressed in the Thesis

To achieve the aims of the thesis different issues related to AC mining and the feature selection from the websites are addressed that include the efficiency of the training phase, building the classifier, the prediction phase and the applicability of AC on real world applications such as phishing. These issues will be highlighted here and will be discussed in details in this thesis.

1.3.1 Issue 1: An Associative Classification Algorithm for Emails and Website Prediction

In this thesis we deal with the problem of automatically categorizing websites and emails to their relevant types based on different features collected and assessed by the proposed AC algorithm. Mainly, the proposed algorithm is developed mainly to successfully allocate the appropriate type of the website based on features related to fake and accurate websites in automated manner. We have developed within the LC algorithm three methods described in the following subsections that have enhanced the primarily the efficiency and the accuracy of the classifiers generated by the LC algorithm when contrasted with known AC and traditional classification algorithms. In other words, we showed the applicability of AC classification as an approach on effectively predict the type of emails and websites in two contexts: classification accuracy and efficiency. Finally, we conducted experiments to extract the useful and important features from the world phishing websites data. Then we compared AC methods as well as traditional classification techniques performance on these data sets to reveal the efficiency and accuracy of LC algorithm.

1.3.2 Issue 2: Efficiency of Rule Discovery Phase

The data used in classification problems is large, and therefore the number of candidate *ruleitems* generated also known as potentially frequent *ruleitems* in each iteration is relatively very large. It is computationally expensive to mine this highly correlated data using AC techniques. To generate the frequent *ruleitems* it is really essential to have such an algorithm that executes quickly and able to generate rules with minimum use of memory space. Many of the current AC approaches like CBA have adopted the Apriori (Agrawal and Srikant, 1994) candidate discovery step inherited from association rule mining to extract frequent *ruleitems*. In our thesis "candidate itemset" and "frequent itemset" terms are used when talking about association rule terms like "candidate *ruleitems"* and *"*frequent *ruleitems"* are used when we mention AC. The discovery of frequent itemsets in Apriori from a transactional data set is achieved in steps, where frequent itemsets discovered in the *nth* iteration are used to generate potentially frequent itemsets, also known as candidate itemsets at (*n+1)th* iteration. During the process of each iteration a database scan is necessary to calculate the support and confidence counts of the newly produced candidate itemsets.

Similar to the association rule mining approaches, AC techniques also adopted the concept of Apriori candidate generation function to extract the frequent *ruleitems*. We enhance the process of finding frequent *ruleitems* by only considering frequent *ruleitems* that share common class in the merging process. We also implemented a minority rule method that eliminates all the *ruleitems* that are associated with the minority classes and keeps the *ruleitem* with majority class, if a *ruleitem* is found linked with more than single class. This substantially minimizes the number of frequent ruleitems produced at the end of rule generation phase, reduces the number of rules in the classifier after pruning phase, and also decreases the training time and reduces memory use.

1.3.3 Issue 3: Enhancing Prediction Procedure of Test Data

Predicting test data is considered one of the most vital steps in classification data mining. Most of the current AC algorithms predict test cases using single based rule which is the top ranked rule in the classifier that relate to the test case. However, there could be more than one rules that may related to the test case ignoring an important fact, which make the prediction decision by current AC algorithms doubtful. Our proposed algorithm takes into account all related rules in the classifier to the test case during the prediction phase in which these rules get categorised into

groups based on their class labels. Then, the average confidences for all rules belonging to the same group are computed and the class that belong to the largest average confidence group gets allocated to the test case. This prediction method ensures that all related rules participate in the class assignment decision.

The classifier by evaluating the candidate rules produced at the training phase against the training data set and only keeping rules that have certain training instances coverage. The process of evaluating each candidate rule on the training data set necessitate that the candidate rule body must exactly match at least one of the training instance. This process of removing rules is carried out in pruning phase thus leading to the improvement in prediction accuracy.

1.3.4 Issue 4: Rule Sorting Criteria

The rules ranking have a significant impact on the prediction step where the top ranked rules are used very frequently to classify test objects. The priority of the rule is usually determined according to several factors such as the support, confidence and the length of a rule (cardinality). In AC, normally a very small support is used and since classification data sets are dense, the expected number of rules with identical support, confidence and cardinality is high.

For example, for "balance-scale" database taken from (Weka, 2000) while keeping *minsupp* of 5% and *minconf* of 40% using the CBA algorithm (Liu, et al., 1998), the classifier have produced 15 rules, having same support values and the top 4 having same confidence value, CBA is not able to discriminate between which rules to select. Furthermore, the 15 rules discovered by CBA also have the same length. So it is harder for CBA to rank them.

A more detailed rule ranking technique as compared to CBA's rule ranking approach is introduced in the LC algorithms that by looking into class frequencies in the training data set associated with the tie rules, and selects the rule associated with the largest class frequency. The proposed rule ranking approach provides a more detailed mechanism of favouring one rule on another during constructing the classifier. This ranking method is also a useful tool since it minimises the use of randomisation in rule selection, and therefore it may positively impact the classification accuracy of the resulting rule set.

1.3.5 Experimental Study Comparing Data Mining Approaches on UCI

Many experiments have been conducted while comparing a wide range of classification approaches like decision trees, statistical approaches and AC on different binary and multi-class benchmark problems. We have based our comparisons on accuracy of classification, merging of itemsets during each iteration, execution times and number of rules.

1.4 Outline of Thesis

The thesis comprises of 6 chapters. Chapter 2 describes the data mining with introduction, types of data mining and definitions that will be used in this thesis. Discusses research works performed in association rule and classification techniques in data mining where detailed descriptions of algorithms are given. Chapter 2 also focuses on demonstrating many AC techniques in data mining. It includes the methods used by AC algorithms to produce frequent *ruleitems*, discusses the generation of rules, rule sorting and pruning methods and methods for the classifier building and the prediction processes. Chapter 2 also includes the literature review on phishing and use of AC mining in Phishing. Chapter 3 demonstrates the steps and the process of CBA algorithm in detail. The purpose is to explain for the comparison purposes the modifications in CBA algorithm. Chapter 4 introduces the "Looking at the Class (LC) algorithm" for construction of the frequent *ruleitems* to produce better and efficient classifiers. This chapter presents a fast and effective method to discover frequent *ruleitems* from databases and demonstrates by experiments its effectiveness and efficiency in terms of number of candidate *ruleitems* merging and execution time. It and also studies the impact on the performance in terms of accuracy using a number of well-known data sets from UCI. This chapter also proposes new methods for rule ranking and prediction.

Chapter 5 includes the current emerging real world problem called phishing. The main challenge of extracting the useful features is addressed in this chapter. The chapter also demonstrates in detail the methods and criteria used in the selection of features from the large matrix of phishing data and formulate the rules to classify the data. In this chapter, a detail analysis of experiments conducted on the extracted data using the new AC algorithm and comparison on the performance with well-known classification and AC algorithms are conducted

with respect to several evaluation measures. Critical analysis of the experimental results is discussed in detail in Chapter 6. Chapter 7 summarises the major achievements and findings in the thesis, draws the general conclusions and recommends further research directions.

Chapter 2

Literature Review

2.1 Introduction

This chapter is divided mainly into three parts. The first part of the chapter consists of detailed review of two data mining tasks of classification and association rule mining. The differences between these two approaches are highlighted and the benchmark algorithms are discussed in detail for both classification and association rule mining. In the second part of the chapter, third data mining approach which is the combination of both association and classification rule mining, known as associative classification will be presented and lastly in the third part a very important and critical problem of phishing in websites will be discussed in detail. The first part of the chapter demonstrates the classification problem followed by a focus on the classification approaches and will conclude by comparing the research work done in each technique. The benchmark and well known classification approaches such as decision trees (Quinlan, 1986; Quinlan, 1988; Quinlan, 1993; Blackmore and Bossomaier, 2002), Naïve Bayes (Duda and Hart, 1973) and rule induction (Furnkranz, 1999) will be reviewed. The importance of the above techniques discussion is significant as some of them will be used in the comparative analysis of the experimental results in chapter 4 and 5. In the latter half of this part of the chapter the association rule mining technique will be discussed. The association rule mining problem will be presented with the explanation of famous and pioneer rule discovery approaches such as Apriori (Agrawal and Srikant, 1994). This algorithm will be discussed in details because the proposed work has used the basic concept of Apriori rule generation process. The other AR algorithms like FP-growth (Han, et al., 2000), partitioning (Brin, et al., 1997) and others will also be discussed. The main purpose of the discussion is to understand the tasks in classification and rule generation process in Association rule mining before the third type of data mining technique of Associative classification(AC) is discussed.

The discussion in the second part of the chapter will be organized as, a demonstration of AC general working, the AC problem and detailed review of related work published will be surveyed. The use of AC algorithms in real world applications like heart disease prediction

(Jabber et al., 2012), treatment response data in patients of Hepatitis-C (Enas et al., 2012) and artificial immune system (Samir, 2012) will be discussed. Different pruning methods and the current prediction approaches used in AC will be compared and explained in some detail in this part of the chapter.

In the third part of the chapter a very important information security problem called phishing will be introduced. The selection of this important real world problem is due to a fact that the number of web users are increasing in volumes and so as the phishing attacks. An increase of 59% in phishing attack volumes is reported in 2012 than 2011 and globally the losses due to phishing are estimated at $1.5 billion in 2012 an increase of 22% than 2011(RSA's Report, 2013). The literature is reviewed with emphasis on the use of AC approaches in detecting phishing and different methods been used to extract the features from websites and emails will be discussed in detail.

2.2 Classification Technique in Data Mining

The purpose of classification techniques is to generate rules from a set of training data that contains a set of class labels, and then the model or the set of rules will be used in the prediction of test data while performance metric of accuracy is calculated. Hence the classification is a two stage process as described in Figure 2.1. In the first phase rules are generated by a learning process performed by an algorithm on the training data set. The second phase is the testing of the extracted rules in the first step; these are applied on the test data to predict the outcome of the values in the test data instance.

In classification different approaches for knowledge extraction from a database can be grouped into rule induction separate and conquer (Furnkranz, 1999), divide and conquer (Quinlan, 1987a), statistical approaches (Duda and Hart, 1973; Meretakis and Wüthrich, 1999) and covering (Cendrowska, 1987).

The rule induction approach also known as separate and conquer approach, works by generating rules in a greedy manner, extracting rules one by one. When a rule is found, the process of checking how many instances this rules covers in the training data set is initiated. The instances covered by that rule are then removed from the training data set. The process is continued until the best rule with a large error rate is found. The other approach of divide and

conquer selects a root node at the top level by using information gain (Quinlan, 1979). The root node contains an attribute from the training data set. The training data set is then split into many subsets depending on the possible values of the attribute selected. The process is iterated until all attributes that are linked in one branch represents the same class or the remaining data entry values are not able to add or split any more. Naïve Bayes is a statistical approach in which all the classes' probabilities are calculated in the training database. The probabilities are based on the occurrence of associations between the attribute values and used to classify the test data. The rest of the techniques like covering algorithms pick up all the classes present in the data set one by one. In order to extract highly accurate rules, the approaches tries to find a way of covering the maximum number of training instances of that class.

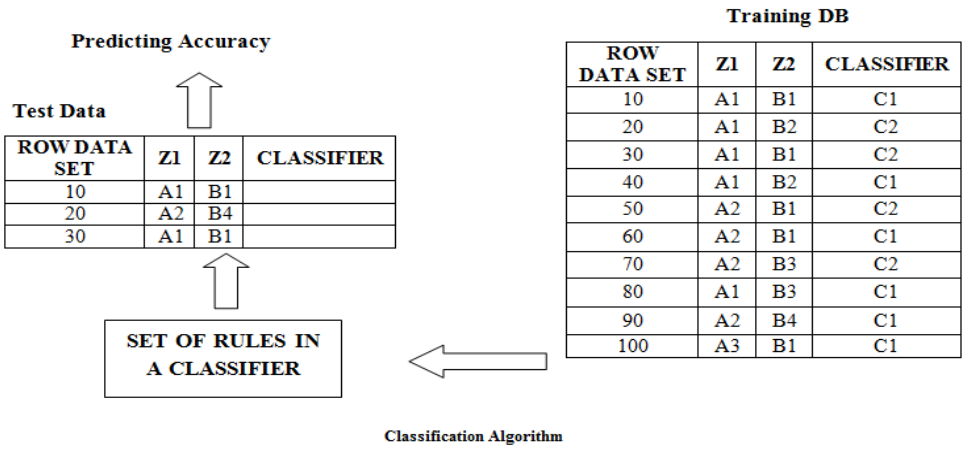

Figure 2.1 Classification as two-step process in data mining

The survey of the classification techniques such as PART (Frank and Witten, 1998), decision trees (Quinlan, 1986; Quinlan, 1993; Quinlan, 1998), Prism (Cendrowska, 1987), RIPPER (Cohen, 1995), and will discuss significance regarding their characteristics and different issues in depth in the next sections.

2.2.1 The Classification Problem

A classification problem is defined as: let '*D*' denote the data table and 'C' as the set of classes present in the table. Each record $d \in D$ has an outcome in the form of a class $c \in C$. The main aim is to find a set of rules known as classifier $z : d \rightarrow c$ that is able to reduce the chances that $z(d) \neq c$ for a new test data instance (d, *c*).

2.2.2 Classification Example

Table 2.1 contains a ‘Lenses’ data set taken from UCI machine learning repository. Depending upon any attribute values of the following attributes: age of patient, spectacle prescription,

Table 2.1: Lenses data set (UCI machine learning repository)

Age of the Patient	Spectacle Prescription	Astigmatic	Tear Production rate	Class of Lens
young	myope	no	reduced	none
young	myope	no	normal	soft
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard

astigmatic and tear production rate, the data set ‘Lenses’ is classified into three classes like soft, none and hard. It would be difficult for an optician or eye specialist to advise the type of lens while looking at the data in the above table or a bigger data set. But when the machine learning approaches are applied on the above data, few simple rules are generated that are easy to understand. These rules generated will help any system or the opticians to make their decision such as to advise a new patient about the type of lens. The test data in the Table 2.2 will be classified with the rules generated by classification approaches.

Table 2.2 : Test data to advice type of lens

Age of the Patient	Spectacle Prescription	Astigmatic	Tear Production rate	Class of Lens
young	myope	yes	normal	?
young	hypermetrope	no	normal	?
pre-presbyopic	myope	yes	reduced	?
presbyopic	myope	yes	normal	?

2.3 Common Classification Techniques

2.3.1 Simple One Rule

One rule (OneR) proposed by(Holte, 1993), is a straightforward and cheapest classification approach in which one-level decision tree is constructed and rules are generated from the training records that are linked with frequent classes. The algorithm checks all the existing attributes in the training database and also loops among all different values of each attribute. It then calculates the occurrence of the attribute value in focus with the class value then extracts the frequent class and marks it as a candidate rule. It will count the error rate in classifying the test instances of each rule and stores the rule that have the lesser error rate. The process of finding the association of class with the attribute values continues until no rules left with a desirable error rate.

The challenges of real-valued attributes and missing values are faced by classification approaches (Quinlan, 1988), (Witten and Frank, 2000). The problem of missing values is solved by OneR, by considering them as any other current attribute value. The challenge of real-valued attributes is solved by changing them into categorical values by the use of discretisation method and lastly for the categorical attributes the algorithm OneR maps all possible value into a positive integer set. The research results in (Holte, 1993) have shown that in many of the classification problems, OneR performs well and finds classifiers with reasonable accuracy.

2.3.2 Decision Trees

A very well-known and mostly discussed technique for classification and then used for prediction is decision trees (Quinlan, 1979, 1986, 1998). This approach can be understood by an intellectual's game of asking some number of questions to guess or identify the place, thing, a personality, or an event from the history or present. Questions are asked in such a fashion that one question is linked and related to the previous question asked. The game starts when one person thinks of any famous thing. A group of intellectual people asks about 15 to 20 questions to reveal the identity of that thing. The series of questions in the above game are represented by decision tree where outcome of one question set up the next question to be asked.

To build a decision tree, from the instance of a data set the attribute is entered in the root node and branches are built for each attribute values that are linked to the root node attribute. This recursive process is continuously applied until all the instances in node have the same class or there is no further room to split the tree (Quinlan, 1979).

When the tree is built, the path from the root node to the end nodes, that is leaf nodes, is termed as a rule. The conditions or antecedent of a rule is the path from the top root node to the leaf node and the consequent of each path is a class attribute value which is assigned by leaf node. The most critical task in this approach is to select a candidate attribute to split or separate the data, because selection affects the class distributions in the branches.

Pruning methods are used to remove the rules that are redundant and unnecessary and as a result of this process, the simpler and easy to understand rules are kept. The tree built in the above process will be pruned either by doing sub-tree replacement or sub-tree rising (Quinlan, 1993) .In the sub-tree replacement substitution is carried out of leaf nodes with sub-trees and in sub-tree rising the nodes that are higher in the tree are replaced. Both these techniques are known as post-pruning techniques (Witten and Frank, 2000). Another pruning method is also very effective; firstly it calculates the error rate of the leaf and internal nodes, and then it compares the error rates of the replaced leaves (Quinlan, 1987b). Decision tree algorithms like ID3 and C4.5 will be discussed in the following section.

2.3.3 ID3 Algorithm

The first decision tree algorithm introduced is ID3 by (Quinlan, 1979). It uses an information gain measure to decide which attribute value should go in the root node. ID3 picks a root node attribute from the available attributes in the training database. As we know that this selection has very crucial impact on the distribution of the classes, so the aim is to select an attribute for the root node with the best information gain value. After the selection is made by the algorithm of the root node, the recursive process of selecting the best attribute in the remaining child nodes continues until the rest of the training data cannot be split further (Utgoff, 1989). A decision tree is constructed after the above process corresponds to an attribute at each node and the arcs contain the attribute's value. Thus the links in the tree starting from the root node to any leaf node refers to rules.

The information gain information that was used to make a decision regarding placing an attribute in the root node, is also helpful in predicting the class that can or should be assigned to the unclassified test instance.

The attribute which contains the maximum information is selected. The information in an attribute is measured by using Entropy. Give a training data objects D of R rules,

$$\text{Entropy (D)} = \sum -P_j \log_2 P_j \qquad (2.1)$$

where P_j is the probability that *D* belongs to class *j*. The information gain of a set of data objects on attribute *X* is defined as

$$\text{Information Gain (D, X)} = \text{Entropy(D)} - \sum ((|D_a| / |D|) * \text{Entropy } (D_a)) \qquad (2.2)$$

Where the sum is over each value *a* of all possible values of attribute *X*, D_a = subset of D for which attribute *X* has value *a*, $|D_a|$ = number of data objects in D_a , $|D|$ = number of data objects in D.

A training database *D* is considered in Table 2.1, which represents 'Lenses' information , with three class labels *none, soft and hard, D* consists of 24 data objects where 15 are associated with class 'none', 4 with class 'hard' and 5 with class 'soft'.

Entropy (*T*) = - (15/24) $\log_2$ (15/24) - (5/24) $\log_2$ (5/24) - (4/24) $\log_2$ (4/24) = 0.940.

To calculate the information gain value of attribute Age of the Patient, which contains three attributes values “young” (8/24), “pre-presbyopic” (8/24) and “presbyopic” (8/24).

Gain (*T*, Age of the patient)=Entropy (*T*) - (8/24)*Entropy(T(young)+(8/24)*Entropy(T(pre-presbyopic))+ (8/24)*Entropy(T(presbyopic))

The missing attributes should be handled in the ID3 by making some modifications to the algorithm. As the decision tree takes into consideration all the training data attributed values it may produce long paths. Hence pruning is performed to generate a small subset of rules. The extension of the ID3 is done by applying different pruning methods like substituting the sub-tree by the leaf node. The substitution is carried out when the expected error rate in the sub tree is more than the leaf node.

2.3.4 C4.5 Algorithm

C4.5 is a decision tree algorithm that was proposed by Quinlan (Quinlan, 1993), that generates rules from data set. C4.5 is a modification of the ID3 algorithm discussed in the section above 2.3.3. The C4.5 algorithm just like ID3 also uses the information gain measure to determine the root attribute. We will explain in detail in the following section, how C4.5 builds a decision tree. Let the training data set shown in Table 2.1, the algorithm will determine the Entropy for all the four attributes, Age of the Patient, Spectacle Prescription, Astigmatic and Tear Production rate is calculated and one is selected as the root. The process of selection continues in the same fashion for the rest of the attributes. C4.5 also handles the missing values by using the probabilities of the different values representing an attribute. The total numbers of missing values of an attribute are calculated and this value is distributed among the different values of an attribute based on their probabilities, in order to keep the uniformity.

As we know that ID3 used sub-tree replacement to perform the pruning. The modification is made in C4.5 in that it uses two pruning methods like pessimistic error rate and sub-tree replacement (Quinlan, 1987b; Breiman, et al., 1984), to simplify the DT. Sub-tree replacement is explained in the above section 2.3.2. Pessimistic error rate pruning was introduced by (Quinlan, 1987b). The rules are read directly from the constructed decision tree. The rule contains node and arc values as the antecedent and the frequent value in leaf node is taken as the consequent. A

new rule can be formed i.e., R' by removing some conditions from R. If the error rate of R' is lower than R on the training data set then R is replaced by R'.

2.3.5 Statistical Approach (Naïve Bayes)

Statistical approach uses all the attributes in the training data set to predict the outcome. This approach works unlike the One Rule Algorithm described in section 2.3.1, which finds a best attribute from the training data set which is then used for prediction. The best known statistical algorithm is Naïve Bayes proposed by (Duda and Hart, 1973), it calculates each class probability for an attribute using the combined probabilities of all attribute values of that data object. The algorithm makes an assumption that the conditional probabilities, of the different attributes with the same class, are independent of each other. Thus same weight is given to the training attributes during the calculation of the probabilities with a class. Naïve Bayes algorithm has proven to work well in several experimental studies (Lewis, 1998b; Meretakis and Wüthrich, B, 1999; Friedman, et al., 1997, Dong, at al., 1999).

A training data set is considered in Table 2.1, which describes the patient prescription of a lens. The Naïve Bayes algorithm works by calculating the probabilistic values. The attribute value frequencies in the training data for the age of the patient attribute, we found 8 occurrences of "young" of which 4 instances are associated with class "none", 2 instances are associated with class "soft" and remaining 2 are with "hard" class from the values in Table 2.1. The probabilistic measure value of each attribute is used to estimate the *likelihood*.

When a new instance has to be classified of Table 2.2, the probabilistic values of each attribute will be used to predict the *likelihood* of a class, to be assigned to the test instance. Considering the first test instance from Table 2.2, which contains the values: young, myope, yes and normal, the *likelihood* of the possible classes is calculated as follow:

Likelihood for class (none) = 4/15 * 7/15 * 8/15 * 3/15 = 0.01327 = 2.42%

Likelihood for class (soft) = 2/5 * 2/5 * 0/5 * 5/5 = 0.16 = 29.18%

Likelihood for class (hard) = 2/4 * 3/4 * 4/4 * 4/4 = 0.375 = 68.40%

From the example above the *likelihood* for class "hard" is larger than that of class "soft", and class "none" and so the test instance shown in Table 2.2 is assigned to class " hard".

There is one problem in Naïve Bayes algorithm; if an attribute value never occurred for any of the classes in the training data set then the final probability value of an attribute with that class will be zero. A method called Laplace estimator method by (Snedecor and Cochran, 1989) is one solution that adds 1 in the numerator and 3 in the denominator. Another solution is applied by a minute addition to the Naïve Bayes method, a small number say '*n'* to the numerator and *n/3* to the denominator (Witten and Frank, 2000).

2.3.6 Rule Induction and Covering Approaches

2.3.6.1 Incremental Reduced Error Pruning (IREP)

Incremental Reduced Error Pruning (IREP) technique was proposed by (Furnkranz and Widmer, 1994). It combines the separate and conquer approach with the Reduced Error pruning (REP). REP is an effective method that prunes and generates a set of rules. The errors are estimated at all nodes of the tree by keeping the chunk of training data as a test data that is independent. The process works by estimating the misclassification rate of the test data at each node and is compared with the error if the concerned node is exchanged with the resultant majority class. While replacing the node if the error is reduced, pruning of the sub tree is carried out. This computational and pruning process is performed repeatedly for each node until there is no reduction in error at each node.

IREP works in greedy manner to generate some rules; two sets of data are constructed by random partitioning the training data. One set is growing which contains 66.6% and the rest of training data is considered as pruning set. IREP starts with an empty rule; a rule condition containing the attribute value is added using a Foil-gain metric (Quinlan and Cameron-Jones, 1993). The algorithm appends those conditions to the current rule continuously that increases the Foil-gain value until that rule is not able to cover any data from the growing set. When the rule is constructed, the algorithm then picks one rule from the generated rules and prunes it backwards. From the selected rule IREP deletes one condition and selects that deletion which showed improvement in the function below:

$$v = (rule, r\_p, r\_m, P, M) \equiv \frac{p(M-m)}{P+M} \tag{2.3}$$

where P, M represents the total number of data instances in the pruning set and r_p, r_m are the total number of data instances that are predicted by the pruned rule. The pruning process will be continued until a deletion in the rule condition does not improve the value of 'v'.

The pruned rules are placed in the classifier and the data instances that are linked with it are deleted from both the sets. An empirical study was performed on some benchmark problems which showed that REP is slower than IREP and both are competitive when compared on the basis of error rate (Furnkranz and Widmer, 1994). When compared with C4.5 algorithm, IREP performed well on 16 data sets and showed less error rate, whereas IREP is out performed by C4.5 on 21 data sets.

2.3.6.2 Repeated Incremental Pruning to Produce Error Reduction

RIPPER is a modified version of the IREP algorithm and was proposed by Cohen (Cohen, 1995). RIPPER builds the set of rules called classifier as follows: firstly it divides the training data set like IREP into two parts, a pruning and a growing set. The process starts by an empty rule set and the algorithm appends heuristically one condition at a time till no error is found on the growing set.

The modification in IREP will be explained in this section. One modification is the introduction of revised stopping condition when the rules are generated. In IREP a stopping condition is used that checks the error rate of a learned rule and stops adding rules, when error rate exceeds 50% on the pruned data. This criterion seems to stop too early, if an application domain contains a large number of low coverage rules. RIPPER uses a minimum description length principle (MDL) to stop adding a rule (Rissanen, 1985). When the rule is added the complete description length of the training and the rule data sets is calculated. If the description length is greater than the shortest description length extracted so far, the algorithm will not add any more rules. This technique of MDL considers the best set of rules that reduces the size of the classifier (set of rules) and also minimizes the quantity of information needed to handle the exceptions relating to these set of rules (Witten and Frank, 2000).

Another modification in IREP algorithm is the procedure that optimizes and reduces the number of rules discovered by pruning the discovered rule set. It means that this process is applied after the first pruning phase, so known as post-pruning. So the classifier produced by

IREP is again processed through an optimization phase, to further simplify the rule set characteristics.

The RIPPER integrates the IREP and optimization procedures. The working is as follows: A pruned rule pr_i is selected from the rule set and two alternative rules such as replacement and revision of pr_i are built. To create the replacement of pr_i, an empty rule pr_i' is constructed and pruned to decrease the error rate of the rule set including pr_i' on the pruning data set. And the revision of pr_i is built in the same fashion but the rule is constructed heuristically and one condition is added at one time to the actual pr_i instead of the empty rule. The rule with the minimum error rate is selected from these three rules when analysed on the pruning data.

A comparative study is conducted on 36 benchmark data sets (Merz and Murphy, 1996) to estimate prediction accuracy of C4.5, IREP and RIPPER algorithms. The results have revealed that C4.5 has shown less error rates on 15 data sets while RIPPER has demonstrated better values than C4.5 on 20 data sets. On the other hand RIPPER has achieved better results when compared with IREP on 28 data sets.

2.3.7 Prism

Prism is categorized as a covering algorithm to generate classification rules and was proposed by (Cendrowska, 1987). This approach considers one class from the training data and finds a method to cover all the records to that class and the data instances which do not belong to that class are excluded. To achieve the maximum accuracy of the created rules, this approach appends one attribute value to the current rule. At each step, the algorithm selects the condition that increases the probability of the desired classification. The process of rule construction continues until the stopping condition is met. Once a rule is constructed, the algorithm stops building the rules for the current class when all the data associated with the current selected class is covered. After this process another class is selected and so on.

The rules generated by prism are nearly perfect (0% error rate). The main advantage of Prism is that it can add a rule in the generated rule set with no impact on the current rules, when compared with decision tree. On the contrary in a decision tree when a rule or a path to the tree

has to be added, it requires changing of the complete tree structure (Witten and Frank, 2000). When an instance has to be classified, decision tree uses rules from the tree directly unlike in Prism due to independence of rules, problems may occur when an instance is linked with more than one rule having different classes.

A study was conducted to investigate the e-learning systems while using classification algorithms to produce set of predicting rules for courseware authors in (Romero, et al., 2005). The findings from 50 log files for students have shown that Prism has successfully extracted more accurate rules when compared with ID3.

2.3.8 Hybrid Approach (PART)

PART is a hybrid approach introduced by (Frank and Witten, 1998). Unlike RIPPER and C4.5 approaches that work in two phases, PART produces rules all in one go while avoiding extensive pruning. PART joins both the approaches of divide-and-conquer as in C4.5 and separate-and-conquer as in RIPPER to explore and generate the rules. It constructs a partial decision tree by using divide-and-conquer approach, from the rules generated by separate-and-conquer approach. The PART algorithm avoids building a full decision tree but instead it constructs and prunes a partial decision tree like C4.5. While handling of missing values and pruning approaches remain same as C4.5. PART is different from RIPPER in the pattern of generate rules; each rule in PART refers to the leaf that has the largest coverage in the partial decision tree. On the contrary RIPPER constructs the rule in a greedy fashion, beginning from an empty rule and appends conditions until the rule does not show any error.

The results are gathered from the experimental studied conducted on PART, RIPPER and C4.5 using data sets from (Merz and Murphy, 1996) and have been written in (Frank and Witten, 1998). The results have shown that simplicity of PART has not affected its performance. The set of rules generated are more accurate than RIPPER though they are lengthy, and are also more accurate than C4.5.

When the comparative study was conducted between PART and C5 (Quinlan, 1998) using missing persons profiling data (Henderson, et al., 2000). The experimental analyses have indicated the resemblance in features and accuracy of the rules generated by PART and C5 using Weka software system (Weka, 2000).

2.4 Issues in Classification

2.4.1 Over fitting

There are many issues in classification and over fitting is one of them; as the training segment goes on too lengthy with the aim of reducing the error rate to zero, then the general performance of the outcome classifier on test data objects may depreciate. That is called general explanation of the over fitting problem, which can happen due to many reasons like restricted number of training data objects or noise among the training objects (Jensen and Cohen, 2000; Freitas, 2000). In decision tree algorithms for example, it is always possible to create a highly precise decision tree for the training data, but, during the construction of the tree it is generally helpful to stop the building process near the beginning in order to simplify the performance of the result on test data objects. For that reason, pruning approaches like pre-pruning and post-pruning (Breiman, et al., 1984; Quinlan, 1993) have been extensively used during building decision trees to keep away from fitting the training data very well and to give precise performance on test data.

There are many methods exercised to stay away from the problem of overfitting in the classification literature, which includes cross validation (Witten and Frank, 2000) and MDL principle (Rissanen, 1985). Cross validation is a well-recognized evaluation approach in data mining. Cross validation is generally employed when the amount of data for mining is small. In cross validation, the training data set is separated at random into *n* blocks, every block is held out once, and the classifier is trained on the remaining *n*-1 blocks; after that its error rate is calculated on the holdout block. Therefore, the learning procedure is executed *n* times on to some extent different training data sets. When there is enough data, for user then user simply divide the data into two sets, these are training and test. User can discover the classifier from the training set and calculate its quality on the test set.

In the end, the MDL principle has been used in decision trees to stop the progress of the growth of the tree. Overall, over fitting is believed to be one of the reasons why classification task in data mining is so difficult (Jensen and Cohen, 2000).

2.4.2 Inductive Bias

Given a training data set, the classification task may be seen as suggest hypotheses (classification rules) from the set of accessible training data objects. Inductive bias can be described as a set of assumptions that lead to the selection of hypothesises (Liu, et al., 2002). With no inductive biases the classification techniques would not be able to help one rule over other ones. In addition, all rules when taken collectively will predict that all data objects are evenly likely, and subsequently cannot supply a basis for prediction.

Classification algorithms are capable to generalise their performance on test data objects by inductive biases because they have contained assumptions of favouring one rule over another. For example, a decision tree algorithms like ID3 (Quinlan, 1979) and C5 (Quinlan 1998) have an obvious bias in their searching for the best attribute decision node, which is, the attribute selection technique based on information gain. Besides, these algorithms help smaller valuable sub-trees over difficult ones by using backward pruning. Probabilistic classification algorithms like Naïve Bayes calculate the probability for every one class in the training data set utilizing joint probabilities of attribute values for a data object. While inductive bias in Naïve Bayes algorithm stands for the assumption that the provisional probability of a data object given a class is independent of the probabilities of other data objects given the same class (Liu, et al., 2002).

At last, because classification algorithms have a bias, the outcomes precision depends a lot on the training data quality. (Freitas, 2000) Just to point out if someone is saying that algorithm X is more accurate than algorithm Y, it is only because of the application domain utilized for the experiment.

2.5 Association Rule Mining

Since 1993 association rule mining introduced by (Agrawal, et al., 1993) is a significant research domain in the machine learning and data mining community. In data mining Association rule mining (ARM) is a very important task which tends to find correlation among the attribute values in a data set. A classic example for association rule mining is a super market basket data, the business community needs to know the shopping trends of the customers (Agrawal and Srikant, 1994). In association rule mining approach, one tries to explore the association between

the shopping basket items of all the customers. For example, in supermarket transactions, if a customer buys a mobile then what is the probability that he will buy a mobile cover as well? Or if a customer buy a 'crisps' what are the chances that he will buy a 'dip sauce'? And if a customer buys 'bread' what is the probability that he will buy a 'jam' or 'butter'? When the business decision makers have the knowledge of association between the frequent sold items, the answer of all the above questions would be much easier. In short the life becomes easy for experts to make strategic decision of buying products, shelving, planning, to launch promotions and advertising. Besides the supermarket example Association rule mining is also helpful in exploring the information in telesales and marketing (Mackinnon and Glick, 1999) , mail ordering (Agrawal and Srikant, 1994) and also in the commerce industry (Pramudiono, et al., 2002).

In association rule mining the task of finding the rules can be described (Agrawal, 1993) in the following points:

Let a data set '*D*' contain the transactions of sales in a super market, and *ITEMS = {item*$_1$*, item*$_2$*, item*$_3$*, item*$_4$*, item*$_5$*……,item*$_n$*}* is the complete set of items present in the data set '*D*'. Any transaction '*T*' in '*D*' that have any number of non-empty items called an itemset, in such a way that $T \subseteq ITEMS$.

An item support is calculated as the total number of occurrence of that item in 'D' to the total number of transactions in the data set '*D*.

An itemset containing two or more items, support is calculated as the total number of occurrence of that itemset in 'D' to the total number of transactions in the data set '*D*.

An Association rule can be described as $item_i \rightarrow item_{i+1}$, where $item_i, item_{i+1} \subseteq$ ITEMS and $item_i \cap item_{i+1} = \theta$.

The association rule confidence can be defined as or calculated as the proportion of the transactions that contains $item_{i+1}$ also have $item_i$, so to find confidence $(item_i \cup item_{i+1}) / \text{support}(item_i)$.

In the association rule approach, inputs of support and confidence values known as minimum support "*minsupp*" and minimum confidence "*minconf*" are defined by the users and the problem is to extract or find all the rules (combination of different number of items) that have the higher values than the thresholds of *minsupp* and *minconf*. As we know that ARM is a unsupervised

learning and only itemsets are used to compute the confidence and support values, in contrary to the AC algorithms which considers the class labels with the itemsets to calculate those values in question.

2.5.1 Problem Solving Strategy

The generation of association rules from market basket transactional data set can be divided into two stages as in (Agrawal and Srikant, 1994).
Stage 1: The itemsets that are produced with higher support values than minsupp are known as frequent itemsets and the rest of the itemsets will be termed as not frequent or infrequent.

Stage 2: All the frequent itemsets from above stage 1 will be passed through a check of *minconf* value. Those frequent itemsets that have higher value than *minconf* are kept as rules. rules.

The first stage of exploring the frequent itemsets, is comparatively more core and tougher problem than the second stage, and requires exhaustive computation and memory (storage) (Cheung, et al., 1997; Lin and Dunham, 1998; Zaki, et al., 1997; Lim, et al., 2000). We will analyse an example from the same sales records data from a supermarket, suppose that data contains 1050 distinct items; we can image a combination explosion of 2^{1050} possible associations of distinct itemsets, many of them might not be frequent in the next step. We will find only a little subset of frequent itemsets from this huge set of candidate itemsets. This is an extensive research area and many scientists have keenly investigated this problem of exploring frequent itemsets in the past many years. The main aim of the researchers is to improve the efficiency (Bayardo and Agrawal, 1999; Liu, et al., 1999; Li, et al., 1999; Zaki, 2000; Baralis, et al., 2004) in the process of generating frequent itemsets.

In the second stage, the rule generation from the set of explored frequent itemsets is relatively easy process and quiet straight forward, given their support values are already known (Han, et al., 2000).

The first association rule mining algorithm is proposed by (Agrawal and Srikant, 1994) is Apriori. The rule discovery process in Apriori is achieved in levels; in the first level the database is scanned and Apriori calculates each item support count and checks whether the support count is greater than the minsupp value. All the items that pass the minsupp value are called frequent 1-items. In the second level the associations or combination of frequent 1-items found in the 1st

level are carried out with the disjoint frequent 1-items. Apriori works by decreasing the number of candidate items or itemsets at each subsequent level by using a property called as "downward closure". It is defined as if an itemset has support value greater than the *minsupp* threshold, then all its subsets have support value greater than *minsupp* threshold. So it infers that the subset of the frequent itemset will also be frequent and on the contrary the superset of an infrequent itemset is also infrequent. By the introduction of this "downward-closure" property, the numbers of potential itemsets are reduced significantly at each level and thus enhancing the efficiency of the frequent itemset generation process. Many of the algorithms in association rule mining proposed after Apriori uses this "downward-closure" approach.

Approaches that use Apriori have performed efficiently and accurately where the size of the data set is small or the candidate itemsets are less (Liu, et al., 1999; Park et al., 1995). However where the support value is kept low, the number of possible itemsets generated may be huge. Multiple scans over the data set is needed to calculate the itemsets support values at each level of the process, and that leads to a significant computational overhead in respect of execution time and the use of memory. As we all know that the data is being collected on daily basis, all the organizations aim to improve their services to get good profits. As the data is growing, to extract and evaluate this data is also becoming computationally expensive. So the main focal point is to minimize the complexity and our research is aiming for that bit to improve the efficiency in terms of run time without affecting the performance when the data sets are large.

To address the shortcomings in Apriori, different ARM techniques have been introduced like Dynamic Itemset Counting (Brin, et al., 1997), multiple-minimum-support (Liu, et al., 1999), Frequent Pattern Growth (Li, et al., 2000) etc. Some of the approaches will be reviewed and discussed that have achieved some significant improvements in the frequent itemset generation step.

2.5.2 Association Rules Mining Approaches

2.5.2.1 Apriori

Apriori is proposed by (Agrawal and Srikant, 1994), and as was discussed earlier that the algorithm works in steps and in each iteration it generates frequent itemsets taking into account

the prior knowledge of the frequent itemsets generated in the previous step. In the discovery process of frequent itemsets "downward-closure" property is used by Apriori. In each iteration, a complete pass over the training data set is done to extract new candidate itemsets from the frequent itemsets found in the previous step.

Apriori starts by taking an input of training data set and explores all candidate items in the data set and those who pass the *minsupp* threshold are stored as frequent 1-items. The algorithm generates the candidate C_n itemsets from merging the frequent 1-items (F_{n-1}) with frequent 1-items (F_{n-1}). Aripori calculates the support counts of all the itemsets found in the previous step. Then it checks whether the support counts of the candidate itemsets generated are greater than the *minsupp* value. The itemsets that passes this filter will be called frequent 2-itemsets. The algorithm continues to generate candidate and frequent itemsets, and terminates when no more frequent itemsets are found. It will combine all the frequent itemsets found in all the iterations.

2.5.2.2 Dynamic Itemset Counting

Researches always desire for the speedy generation of frequent itemsets, so to increase this process a new ARM algorithm is proposed known as Dynamic Itemset Counting (DIC) by (Brin, et al., 1997). The database is split into many partitions by DIC and each partition is marked with a start point. The support counts of all the itemsets generated so far are determined and addition of new candidate itemsets is done dynamically as soon as their subsets become frequent. The major difference between Apriori and DIC is that DIC begins to generate rest of the candidate items based on the candidate item that have just reached the *minsupp* value and not waiting for the scan to complete, unlike Apriori. DIC constructs a prefix-tree to generate the dynamic candidate itemets. DIC is sensitive to homogenous data and if the data set is highly correlated.

The experimental findings have shown that DIC is 30% faster than Apriori using synthetic data while keeping *minsupp* to 0.5% and synthetic data. And on the census data DIC performed with excellence than Apriori by 36%. The itemsets generated in census data sets are very large, and needs long training time when the support threshold is lowered for both DIC and Apriori.

2.5.2.3 Frequent Pattern Growth

It is an Apriori like technique and needs a high execution time and space in main memory to discover all the frequent itemsets. A new ARM technique known as FP-growth (Han, et al., 2000) is introduced that constructs a very condense tree for transactional database known as frequent pattern tree (FP-tree). One path in the tree corresponds to one transaction in the database and the size of the path is the total number of frequent items in the record (transaction).The FP-tree can be seen as a useful structure as there is a lot of sharing between the frequent itemsets and the FP-tree includes all the frequent itemsets present in each transaction of database, thus the FP-tree is smaller in volume than the main database. The construction of FP-tree is done in the second scan after the frequent itemsets are produced with their support values in the first scan.

Unlike Apriori there is no candidate rule generated in FP-growth algorithm during the mining process. The major drawback in FP-method is seen when the FP-tree constructed is dimensionally large and not able to fit in the main memory.

The comparisons between Apriori and FP-growth regarding performance on two data sets having 10000 records have shown that FP-growth is faster than Apriori as the candidate sets that Apriori must maintain becomes very huge. The searching process to update candidate support counts is also very costly for Apriori at all levels.

2.5.2.4 Partitioning

This ARM approach reduces the I/O time by decreasing the frequency of database scan to two is been introduced by (Savasere, et al., 1995). The algorithm splits the database into small portions such that each portion can be accommodated in main memory so that the frequent item generation will be local. It uses Apriori during the first scan and constructs a tid-list for all itemsets in the current portion. The algorithm conducts a union process in the second pass on frequent itemsets discovered locally in each portion to generate frequent itemsets for all data set.
The main disadvantage of this approach is that it divides the data uniformly into equal partitions, if the occurrence of an itemset is evenly distributed in each partition then this approach will produce majority of frequent itemets in the second scan, but in case when the distribution of itemsets are not evenly distributed in all the portions or partitions then in the second pass

majority of the itemsets produced are infrequent and thus causing a significant overhead for I/O (Lin and Dunham, 2000). For the large data sets when the number of partitions is more in number, the size of local frequent itemsets also increases leading to redundant generation and more time needed for the computation, especially when frequent items are present in more than one partitions (Zaki, et al., 1997).

The performance analysis between partitioning algorithm and Apriori has shown that processing time for both increases when the support count value is decreased for 6 market basket data sets (Agrawal and Srikant, 1994). When different sizes of partitions are used for 6 benchmark data sets the decrease in execution time is seen for low number of partitions.

2.5.2.5 Direct Hashing and Pruning

Generally, the execution time of the ARM is mainly dependent on the speed of producing the frequent itemsets in the early stages of rule discovery phase, i.e., in first and second iterations.frequent-1 and frequent-2 itemsets. The findings of (Agrawal and Srikant, 1994) have revealed that the time taken to generate candidate itemsets in the initial stages has the major chunk towards computational cost. It is obvious that if the frequent itemsets are large in 1^{st} iteration, the number of candidate itemsets in 2^{nd} iteration will also be large. So the main focus is to decrease the number of the candidate itemsets at initial scans that may lead to substantial reduction of execution time and use of memory. Direct Hashing and Pruning (DHP), was introduced by (Park, et al., 1995) that is based on hash-based technique to decrease the number of candidate itemsets at the intial stages.

The working of DHP is described as follow: It builds a hash tree, H_1 to discover frequent 1-itemsets during database scan. If an item is present in the hash table and found during the scan then its count is incremented else the item is assigned count 1 in the hash table. After all the 1-itemsets are calculated for the database, a new hash table H_2 is constructed that contains the all 2-itemsets. We can get candidate 2-itemsets from hash table H_2 after the database scan.

The process of pruning involves the removal and reduction of some of the transactions from the data base. In DHP support is set to prune the itemsets, e.g., in H_1 candiate-1 items, t = ACDEF with occurances count of 3, 2,1,4,3. If the support is set to '3', the frequent 1-itemset will only contain the three items in $t = \{A, E, F\}$, having values greater than minsupp are kept while {C, D} items will be removed from *t.*

Experiment results have shown that DHP minimizes CPU time in the 2nd iteration, but it also has shown reduction in execution time in the later cycles (Park, et al., 1995). The processing time of DHP in the second iteration to discover candidate 2-itemsets is less than of Apriori. But the CPU time is higher than Apriori in the first iteration as DHP needs time to build the hash table.

2.5.2.6 Multiple Support Apriori

The number of rules produced is controlled by a constraint of support (Zaki, 2000; Bayardo and Agrawal, 1999; Agrawal, et al., 1993). Most of the ARM algorithms consider single support value. Setting the support value to a low value generates large amount of useless rules (Liu, et al., 1999, Li, et al., 1999) and changing the support value to high leads to non-selection of some important rules.

To tackle the problems as mentioned above, an Apriori like algorithm is build known as MSapriori, which takes into account different support values for every item in the data set. This has given users the flexibility to express themselves and has given option to select support according to different nature of data. The support value of a specific rule in MSapriori is the lowest *minsupp* value between the items in that rule. The generate function of MSapriori is same as Apriori's.

A study against the real data from (Agrawal and Srikant, 1994) has shown that MSapriori produced less number of candidate itemsets as compared to Apriori, however the processing time for both algorithms are the almost the same.

2.5.2.7 Confidence-Based Approach

Another approach proposed by (Li et al., 1999) that does make use of the support threshold and only mines the rules with high confidence. For a dataset, the user sets an itemset target, and this shows the consequent of the desired outcome. All association rules are found keeping in view of the target that is consequent. The process of exploring high confidence rules is divided into two parts. In step 1, database is split into two halves, first half contains all the records with the target itemset T_1, and the other half contains the remaining transactions, T_2. Then the technique

removes all the transactions in T_1 and T_2 with the target item. The remaining set of items left in the original database D, are reduced to $D' = D$ – target (item).

The discovery of all the itemsets, *X*, is carried out in step 2 of those which are present in T_1 but are not present in T_2, and rules like $X \rightarrow tg$ are generated. These kinds of itemsets that have support value of zero in T_2 but contains some support in T_1 are called Jumping Emerging Patterns (JEP). For the discovery of JEP like items authors of (Li, et al., 1999) inherited two border methods from (Dong, 1999). The method of first border explores the data set and discovers itemsets with some support value, called *horizontal* borders. The second border method will take two *horizontal* borders as an input and produce all itemsets with support zero and non-zero for one and the other.

The studies have shown that the high confidence rules extracted by this approach cannot be found by traditional approaches of association rules. But the itemsets generated are much higher than the actual data set.

2.5.2.8 Tid-list Intersection

One of the efforts made to reduce the execution time is by (Zaki, et al., 1997), an Eclat algorithm is build that uses a vertical layout to keep data scans to one. The question is addressed whether it is possible to generate all the frequent itemsets in a single pass over the data; Eclat used a tid-lists intersection approach instead of a complex data structure.

An extension in Eclat algorithm is proposed by (Zaki and Gouda, 2003) known as dEclat. It introduced a new approach of diffset that reduced the memory needed to store the transactions identifiers (tids). A vertical layout is used to keep the differnces in tids of a candidate itemset from its generating frequent itemsets. The new diffset approach does not keep the complete tids of each itemset instead it keeps the differences among class and its member itemsets.

Results are generated on synthetic and real world data (Zaki and Gouda, 2003) and have shown that vertical approaches like dEclat and Eclat performed better than horizontal approached like Apriori and FP-growth in reference to memory use and execution time. On dense data, Eclat is outperformed by dEclat, and on sparse databases the volume of the data stored in dEclat grows at a faster rate as compared to Eclat. The conclusion of the author is to begin with tidlist and then

change to diffset for sparse databases and to start with diffset in case of dense data sets for better results.

2.6 Associative Classification Mining

As learned from the review in the first part of the this chapter that classification and Association rule mining are playing a significant role in problem solving and decision making in the real world. The difference between these two tasks classification and ARM is that former tends to predict class labels in a database and later finds the correlation between the attributes of the database. In 90's (Ali et al., 1997) and (Liu et al., 1998) combined these two tasks to build classifiers with very encouraging results. This new phenomenon is known as Associative classification (AC). The findings of the several researches AREM (Jiang and Karyris, 2012), Fuzzy AC (Lucas et al, 2012), (Niu et al., 2009; Li et al., 2008; Thabtah, et al., 2005; Yin and Han, 2003; Liu et al., 1998, 2001) have concluded that AC approaches build more accurate classifiers than the previously known classification methods as rule induction (Cohen, 1995, Quinlan and Cameron-Jones, 1993;), decision trees (Quinlan, 1998; Quinlan, 1993), and probabilistic (Duda and Hart, 1973) approaches.

Rule induction approaches like ++IREP (Oliver et al., 2003), RIPPER (Cohen, 1995) and IREP (Furnkranz and Widmer, 1994) produce local and human readable set of frequent rules in a greedy way. In the discovery process of the locally derived rule, all the instances related to the rule in the training data are eliminated and this process continues until the error rate is unacceptable. In the rule induction techniques rules are derived from the chunks or partitions of the training database instead of considering full training dataset. A statistical FOIL gain metric measure is used in the above rule induction algorithms, the rule having highest foil-gain metric measure is selected and rules with the highest value are placed at the top of the classifier. A good example is the IREP rule induction algorithm that selects the rule on the basis of Foil-gain value (Quinlan and Cameron-Jones, 1993) and then the rule is inserted at the top of the classifier. In comparison to rule induction approaches, associative classification aims to extract a more global classifier while using full set of training data. Generally, when a classifier is built using AC, a set of frequent ruleitems called Class Association Rules (CARs) are discovered from a training database and based on some methods, a small subset is selected from these large number of CARs to produce a classifier. The production of the classifier is accomplished by different

approaches in number of ways, such as in CAAR (Xu, et al., 2004), L^3 (Baralis and Torino, 2000) and CBA (Liu, et al., 1998) algorithms, selection of these set of rules is carried out by examining and evaluation the full set of CARs on the training database and only those rules are considered that cover at least one training database object. On the contrary, CPAR algorithm chooses a classifier in a greedy manner. The predictive accuracy of the classifier produced is evaluated on test data object to predict the class label.

Several AC algorithms are been introduced in the past decade, such as ARM (Jiang and Karyris, 2012), Fuzzy AC (Lucas et al., 2012), PST (Lakshmi, 2012) and use of AC techniques in applications like heart disease prediction (Jabbar et al., 2012), hepatitis-C treatment response (Enas et al, 2012), in artificial immune system (Samir , 2012), CBAR (Niu et al., 2009), ACCF (Li et al., 2008), BCAR (Yoon and Lee, 2008), ACN (Kundu et al., 2008), MCAR (Thabtah et al., 2005), CACA (Tang and Liao, 2007), 2-PS (Qian et al.,2005), MMAC (Thabtah et al., 2004), CAAR (Xu, et al., 2004), L^3 (Baralis et al., 2004), Negative-Rules (Antonie and Zaïane, 2004), CPAR (Yin and Han, 2003), ARC-AC (Antonie and Zaïane, 2002), CMAR (Li et al., 2001) and CBA (Liu et al., 1998), and The above mentioned algorithms use different techniques to find, rank, store, prune and predict rules and predict the outcome of a new test case.

The discussion in the chapter will be organized as, a demonstration of AC general working in Section 2.6.1. The AC problem and detailed review of related work published will be surveyed in Section 2.6.2. Different pruning methods will be explored in Section 2.6.3, the current prediction approaches used in AC will be compared in Section 2.6.4. And finally, the chapter summary is given in Section 2.6.6.

2.6.1 Working of Associative Classification

AC is a supervised learning method where the outcome or the prediction of a set of attribute values in a dataset is ‘classes’. AC is the combination of two main data mining tasks; association and classification. First step in AC rule mining is the extraction of set of rules by using any of association rule mining algorithm. These set of rules do satisfy user defined input thresholds for minimum support and confidence (*minconf*). The rules generated are very large in quantity and many of them are either redundant or noisy. In the second phase of the AC mining task, pruning methods are applied to reduce the amount of rules by eliminating the noisy and redundant rules

from a set of frequent ruleitems. The remaining sets of rules are called classifier and are used in predicting the outcome of the new data.

2.6.1.1 Main Steps in Associative Classification

The procedure of constructing a classifier involves many steps in AC as demonstrated in Figure 2.1 and can be summarized as follows:

i) The Candidate ruleitems discovery process.

ii) Extraction of all Frequent ruleitems from the Candidate ruleitems that passes the minimum support threshold value.

iii) The rules with values higher than minimum confidence threshold value are selected.

iv) Ranking and pruning methods are applied on all generated frequent ruleitems in order to finalize a set of rules called a classifier.

v) The final step in AC is to evaluate the goodness (accuracy) of the classifier, by applying prediction approaches on the test data.

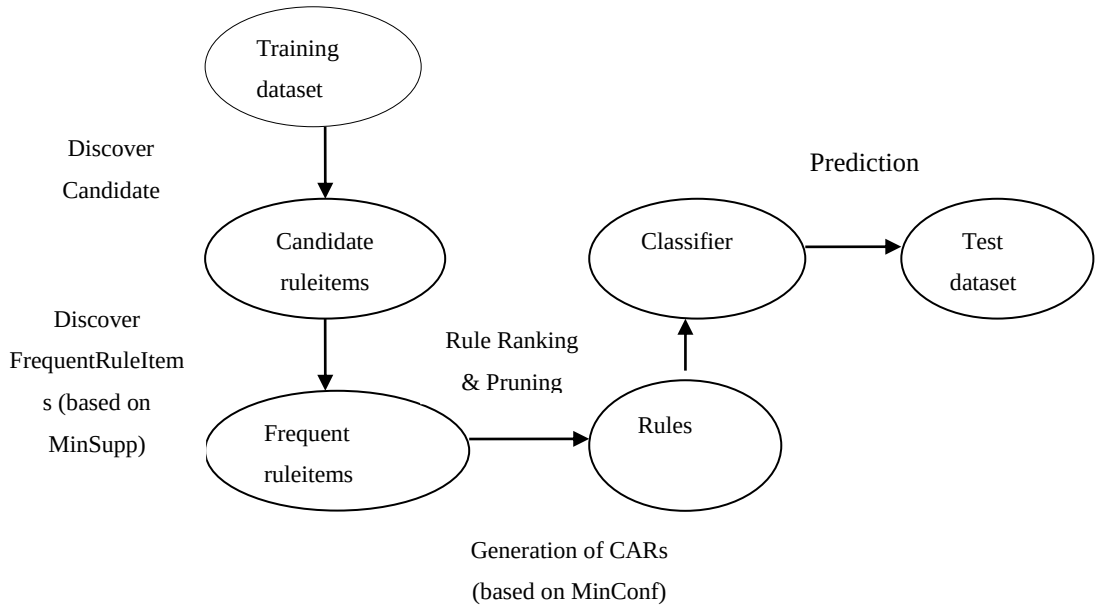

Figure 2.2 Main Steps of an associative classification algorithm

2.6.2 Associative Classification Problem

AC is a special case of association rule mining in which only the class attribute is considered in the rule's consequent (Liu et al., 1998), for example in a rule such as $X \rightarrow Y$, Y must be a class attribute. The AC problem is summarized from (Thabtah, et al., 2004) as follows: A training dataset T has m distinct attributes A_1, A_2, … , A_m and C is a list of class labels. The number of rows in T is denoted $|T|$. Attributes could be categorical (meaning they take a value from a finite set of possible values) or continuous (where they are real or integer). In the case of categorical attributes, all possible values are mapped to a set of positive integers. For continuous attributes, a discretization method is first used to transform these attributes into categorical ones.

Definition 1: A row or a training object in T can be described as a combination of attribute names Ai and values aij, plus a class denoted by cj.

Definition 2: An item can be described as an attribute name Ai and a value ai, denoted < (Ai, ai)>.

Definition 3: An itemset can be described as a set of disjoint attribute values contained in a training object, denoted < ($Ai1$, $ai1$)… (Aik, aik)>.

Definition 4: A *ruleitem r* is of the form <*itemset*, *c*>, where *c*ε*C* is the class.

Definition 5: The actual occurrence (*actoccr*) of a *ruleitem r* in *T* is the number of rows in *T* that match the itemset of *r*.

Definition 6: The support count (*suppcount*) of *ruleitem r* is the number of rows in *T* that match *r's* itemsets, and belong to the class *c* of *r*.

Definition 7: The occurrence of an itemset *i* (*occitm*) in *T* is the number of rows in *T* that match *i*.

Definition 8: An itemset *i* passes the *minsupp* threshold if (*occitm*(*i*)/|*T*|) ≥ *minsupp*.

Definition 9: A *ruleitem r* passes the *minsupp* threshold if (*suppcount*(*r*)/|*T*|) ≥ *minsupp*.

Definition 10: A *ruleitem r* passes the *minconf* threshold if (*suppcount*(*r*)/*actoccr*(*r*)) ≥ *minconf*.

Definition 11: Any *itemset i* that passes the *minsupp* threshold is said to be a *frequent itemset*.

Definition 12: Any *ruleitem r* that passes the *minsupp* threshold is said to be a *frequent ruleitem*.

Definition 13: A class association rule (CAR) is represented in the for (A_{il},a_{il}) ^.....^ (A_{il},a_{il}) → c , where the left-hand-side (antecedent) of the rule is an itemset and the consequent is a class.

2.6.3 AC and ARM Main Differences

The best survey found regarding the difference in AC and ARM is well presented in (Thabtah, 2006), and before going further in the discussion of AC it is necessary to point out the differences in both the approaches and we will summarize them as follows:

- ARM is termed as unsupervised learning as no class outcome is involved, whereas AC is a supervised learning approach and the outcome is always some value from the set of class labels.
- In ARM over fitting is not an issue but in AC approaches, when they aim to decrease the error rate to the lowest while exploring the training data set, the training phase might take too long. This will deteriorate the efficiency of the classifier. (Witten and Frank, 2005) and hence results in overfitting.
- In the consequent of ARM there may be different and more than 1 attribute, but in AC the consequent is only class attribute value.
- The main purpose of AC is prediction of a class label while exploring the correlation among, whereas, ARM tends to discover the associations between the itemsets contained in the transactional data.

Basically, in association rule mining, the algorithm goes through a number of iterations until no more frequent items are left in the dataset. In the first iteration the single item or attribute is said to be frequent in the dataset that satisfies a user defined minimum support value, are extracted. The frequent items are called frequent 1-itemset. In Table 2.3, a training data contains three attribute values (attribute1, attribute2 and attribute3) and two classes (class1, class2). Minimum support value of 20% is used to extract the frequent attributes in the training dataset.

For example, keeping the *minsupp* to 20%, the frequent 1-itemset in training Table 2.3 are <(Attribute1, X1)>, <(Attribute1, X2)>, <(Attribute1, X3)>, <(Attribute2, Y1)>, <(Attribute2, Y2)>, <(Attribute2, Y3)>, <(Attribute3, Z1)>, <(Attribute3, Z2)>, <(Attribute3, Z3)> and <(Attribute3, Z4)>. A combination of itemsets and a class is called ruleitem in the form att(*1)*,

att(*2*),…..att(*m*) →*C* where att(*i*) is the set of itemsets and '*C'* as class. If a ruleitem satisfies a user defined *minsupp* they are called frequent ruleitems.

Table 2.3: Training data

	Attribute 1	Attribute 2	Attribute 3	Class
1	X1	Y2	Z1	class1
2	X1	Y3	Z2	class2
3	X1	Y2	Z1	class2
4	X1	Y1	Z2	class1
5	X2	Y1	Z1	class2
6	X2	Y3	Z3	class1
7	X2	Y1	Z3	class2
8	X1	Y1	Z3	class1
9	X3	Y2	Z4	class1
10	X3	Y3	Z1	class1

Associative Classification algorithms scan the training database or datasets more than once to produce frequent ruleitems. Firstly they produce frequent 1 ruleitems and in the first scan, they find the support of 1- ruleitems, and then in each subsequent scan, they start with ruleitems found to be frequent in the previous scan in order to produce new possible frequent ruleitems involving more attribute values. In other words, frequent 1- ruleitems is used for the discovery of frequent 2- ruleitems, and frequent 2- ruleitems is the input for the discovery of frequent 3- ruleitems and so forth. When frequent ruleitems have been discovered, classification based on association rules algorithms extract a complete set of class-association-rules (CAR) from those frequent ruleitems that pass the *minconf* threshold.

One of the first algorithms to merge classification with association rules was proposed in (Liu et al., 1998). It consists of two main phases: Phase one implements the Apriori algorithm (Agrawal and Srikant, 1994) in order to discover frequent ruleitems, and phase two involves building the classifier. Experimental results indicate that the approach developed in (Liu et al., 1998) produced rules which are competitive to popular learning methods like decision trees (Quinlan, 1988). In the next section, data layouts used in AC will be reviewed.

2.6.4 Association Classification Data Layouts

The term data layout means the manner in which data is kept or represented during the execution of the algorithm. Selection of appropriate data layout is very useful for reducing execution times. The reason for data layout discussion in this section is to demonstrate the advantages and effectiveness in the associative classification algorithms. Generally, two forms of representations will be discussed below; they are the vertical layouts (Shenoy et al., 2000; Zaki et al., 1997; Holsheimer et al., 1995) and horizontal layout (Agrawal and Srikant, 1994). In a horizontal layout, the data table comprises of hundreds of thousands of transactions with each

Table 2.5: Horizontal data layout

Transaction #	Purchased Items			
1	bread	milk	juice	
2	bread	cola	milk	
3	milk	eggs	bread	juice
4	bread	basket	milk	
5	cola	juice	bread	milk

Table 2.4: Vertical data layout tid-list

basket	cola	bread	eggs	juice	milk
4	3	1	3	1	1
	5	2		2	2
		3		5	3
		4			5
		5			

transaction having a unique identification and contains different set of items.

Table 2.5 presents the horizontal layout for a data base that contains the records of daily transactions. In the vertical layout as shown in Table 2.4 constructed in such a way that it contains all the items in a transactions data base followed by an array of their tid-list (transaction id list) (Savasere et al., 1995). The array of tid-list, stores the location of all row numbers of each item that are present in the transactional data set. Unlike the horizontal layout that has high computational costs for support counting; in vertical layout the support counts for frequent items are calculated by simply intersecting the tids. For example, we can calculate the support counts of candidate itemsets of size *k* easily, only by finding the intersections between tid-lists of *(k-1)* subset. The tid-lists are easy to handle and their data structure is relatively simple, and can store all the needed information of all item relating to the database. The execution time to get the support count of any new candidate itemset is relatively less as in the tid-list approach, the database is not scanned again and again. (Zaki and Gouda 2003; Zaki et al., 1997).

2.7 Associative Classification Algorithms

By combining the classification and association rule mining accurate classifiers with efficiency are produced (Liu et al., 1998). Studies followed by (Baralis et al. 2008; Vyas et al., 2008; Li et al., 2008; Kundu et al., 2008; Thabtah, et al., 2004; Thabtah et al., 2005; Liu et al., 1998) revealed that AC techniques construct mostly accurate systems that can predict classes better than the traditional approaches. AC has the capability to produce rules that are much easier to understand or interpret and modified by the end-user as comparison to the models produced by probabilistic approaches and neural networks, which generate models that are difficult to manipulate and interpret. In the following section different AC algorithms will be explained in detail.

2.7.1 Classification Based on Associations (CBA 2)

Some of the classification data sets show uneven distribution of the of class labels. This may cause the production of fewer rules for minority class and more number of rules for the majority class. CBA (2) is introduced to solve the problem and uses multiple support values (Liu et al., 1999). The major enhancements in CBA (2) when compared with CBA are:

- The minimum support is assigned to each class depending on its frequency distribution in the data table.

Minimum Support = total (minimum support) x Class (frequency Distribution)

- For those data sets which are highly correlated, it would be computationally very expensive and sometimes very difficult for original CBA, due to the possibility of combinational explosion to produce long rules containing many conditions. These long rules may be significant in classification hence classifiers may suffer. Here the CBA (2) is merged with decision tree method to extract lengthy rules (Quinlan, 1992). The procedure is to first segment the data by one classifier and the best classifier is chosen for the classification of each segment. Experiment findings show that CBA and C4.5 are outperformed by CBA (2) in regards to accuracy (Liu, et al. 1999).

2.7.2 Classification Based on Multiple Association Rules (CMAR)

This algorithm is proposed by (Han et al., 2000) and adopts an ARM algorithm of FP-growth, to discover and construct an FP-tree and mines the large training datasets with efficiency (Li et al., 2001). It comprises of rule generation and classification phases.

i) Rule Generation:

- FP-growth algorithm is adopted, to scan the training dataset and explore the full rule set that passes a certain confidence and support thresholds.
- F-list is a sorted list in descending order which comprises of frequent items explored in the first scan. The training data is scanned again and FP-tree is constructed. The attribute values of each tuple that appears in the F-list are produced and then sorted as per F-list.
- The complexity of exploring specific frequent item in the complete training data set is minimized to find frequent items in an item-projected dataset.
- The generated rules are sorted in a prefix tree structure known as CR-tree.
- Rules are selected from large amount of generated rules by using pruning techniques like database coverage. This in turn helps to make classification more effective and efficient.

ii) Classification

- To classify a new instance, CMAR starts by matching the pruned rules with any new instance. If the same class label is found, it is assigned by CMAR to the test instance.
- CMAR checks the consistency in the class labels, if the answer is no then the algorithm divides the rules into different class label groups. Each group has a unique label and the rules contained in the group are linked to the same label. The strength of the group is measured by the combined effect of the rules. The group is said to be strong if it contains rules in the group that have high positive correlation and shows high support value. The CMAR uses a weighted Chi-square to measure the strength of each group in order to decide which group class to assign to the test instance.

Experimental results on 26 datasets in UCI database repository have demonstrated that the algorithm CMAR performed efficiently and more accurately as compared to CBA, and is more scalable and efficient than C4.5 and CBA. One drawback documented in CMAR is that it utilizes a large amount of memory resources in the training phase.

2.7.3 Classification Based on Predictive Association Rules (CPAR)

The algorithm of CPAR (Yin and Han, 2003) works in a greedy manner and adopts the main idea of FOIL (Cohen, 1995) in rule production and integrates rule generation step with AC features. CPAR main steps can be summarised as follows:

- Uses FOIL greedy strategy for generating the best rules for each class that cover the training data instances. It deletes all the instances that are positive and are matched by the best rule until all the positive instances in the database are covered.
- It avoids generating redundant rules and therefore, it produces a smaller set of predictive rules.
- It uses a group of rules for the prediction of test data instances class labels.
- It selects the best rule for every class and then the average of expected accuracy is compared for all individual classes and the class with a high expected accuracy is chosen.
- Repeated calculations are avoided by using dynamic programming
- Close-to-the-best literals are selected to avoid rule missing

When compared with C4.5, Ripper, CBA and CMAR in terms of accuracy CPAR achieves high accuracy and efficiency when experiments were performed on datasets from UCI machine learning repository.

2.7.4 Gain Based Association Rule Mining (GARC)

GARC is an approach in AC that uses an extended association rule mining method to construct a classifier. The key idea in GARC lies in the domain of ARM but it is different from the benchmark technique of CBA which apply the Apriori candidate generation mining methods. The key features of the technique are explained in three steps. First to discover candidate itemsets it applies an information gain measure and includes and generates the itemsets that have best-split attribute value. This step also helps in the reduction of candidate itemsets. Secondly combining the rule generation and frequent item generation processes and uses the information kept for rule itemsets and for the itemsets that are excluded. Thirdly to produce a resultant compact set in the process of mining that is more condense and easily understandable. The GARC algorithm gets rid of many candidate itemsets in the training phase and the resultant

numbers of rules are much less than CBA.The GARC gain based association rule classification algorithm works as follows:

Any one transaction in a training data set contains ‘n’ number of items and generates 2^{nd} candidate itemsets. GARC uses an information gain measure to decrease the count of candidate itemsets. In the first scan of the data set, all 1-itemsets that contains the best split-value are generated and stored in a variable.

The performance of GARC when compared with other algorithms, on 30 data sets from UCI including continuous and discrete, has revealed interesting results. GARC has not shown any significant difference in accuracy when compared with two extensions of C4.5, one is C4.5 Tree and other is C4.5 pruning tree. But GARC produced a classifier with explicit rules when compared with NN and SVM. . The pruning strategies applied in GARC has demonstrated that GARC when used with the pruning strategies produced better results in terms of efficiency, accuracy and understandability. The effects of changing the *minconf* value on the results are, as the *minconf* value increases, there is seen an increase in accuracy first and then a decrease. It happens because when the *minconf* is low many useless rules are generated and when its high some significant rules are not discovered leading to a decrease in accuracy. It is concluded that the best accuracy is achieved where a=0.01 and b=0.7. When information gain measures are incorporated in GARC the results regarding the number of rule generated and execution time are extremely good. The average number of rules with gain is 39% of GARC with no information gain measure. The run time with gain is 3.2% with no information gain. But the accuracy measures are fairly similar when results are formulated for both with and without information gain measures in place. The experimental results have shown when GARC is compared with CBA, that GARC produced very few rules by a ratio of 4.3% of that of CBA.

2.7.5 Multi-Class Classification Based on Association rule (MCAR)

MCAR employs an effective technique to produce frequent ruleitems and uses a method to rank rules that keeps the rules with high confidence values for prediction. MCAR works in two steps: generation of rules and classifier building. The first stage starts by searching the training data and extract frequent-1 rule itemsets, and these ruleitems are combined together to generate candidate-2 ruleitems using other attributes. The ruleitems that passes a certain set values of *minimum support* and *minimum confidence* is treated as a frequent rule. In the second stage, effectiveness

of the rules on the training datasets is measured to build a classifier. The rules that cover or can classify a certain number of training data instances are placed in the classifier. The MCAR algorithm is presented in Figure 2.4 Figure 2.3 shows the rule discovery procedure used by MCAR, In Figure 2.4, the Multi-interval discretisation technique of (Fayyad and Irani, 1993) is applied in MCAR for real and integer type of data. MCAR scan the training dataset calculates the frequencies of the itemsets. The itemsets along with their classes, which have support count more than the *minsupp* value, are stored in vertical format in an array. Rest of the itemsets are discarded. Produce function as described in the Figure 2.3 is used to discover ruleitems of size *k* by combining the different column itemsets of size *k-1* and then their rowIds are intersected. The outcome of intersection between two itemsets rowIds is a set that holds the rowIds of both the itemsets occurrence in the training data. The above set and the array containing the frequencies of the class labels were generated in the first scan. The values are used to formulate the

Input:, *minconf* ,*minsupp* thresholds and Training data (*D*),
Output: A Rule Set

Preprocessing phase
If *D* having integer/real attributes
  Discretise continuous columns using a Multi-interval discretisation method.
Shuffle the training objects locations randomly

The Algorithm
Scan *D* for the set S_1 of frequent one-ruleitems
$R \leftarrow S_1$
$i \leftarrow 1$
while $(S_i \neq 0)$
{
  $S_{i+1} \leftarrow produce(S_i)$
  $R \leftarrow R \cup S_{i+1}$
  $i \leftarrow i+1$
}

Rank R according to the method shown in fig 3.7.
Evaluate R on D
Remove all rules $I' \rightarrow c'$ from *R* where there is some rule $I \rightarrow c$ of a higher rank and $I \subseteq I'$.

Figure 2.3 MCAR algorithm

confidence and support of new combinations of ruleitems.

.

The function in Figure 2.5 is called in each iteration of the algorithm and produces a frequent

Function produce
Input: A set of *ruleitems S*
Output: set of S' produced ruleitems

$S' \leftarrow 0$
Do
For each pair of disjoint items I_1, I_2 in *S* Do
If ($<I_1 \cup I_2>$, *c*) passes the *minsupp* threshold
if ($<I_1 \cup I_2>$, *c*) passes the *minconf* threshold
$S' \leftarrow S' \cup (< I_1 \cup I_2 >, c)$
end if
end if
end
end
Return S'

Figure 2.5 Rule discovery algorithm of MCAR

Input: set of created rules (*R*),a array (*Tr*), class array *C*
A rule *r* in *R* has the following properties: Items, class, rowIds(tid-list)
The class array, *C*, contains the occurrences of class labels in the training data
Output: classifier (*Cl*)
R' = sort(*R*);
insert r_1.*rowIds* into *Tr*
for each rule $r \in R'$ in sequence do
if *r* classifies at least a single case
begin
insert *r* at the end of *Cl*;
insert *r.rowIds* into *Tr*
end if
end
If *Tr*.size > 0 then
select the majority class as a default class from (*C-Tr*)
else
select the majority class as a default class from the current *Cl* and add it to *Cl*
end if
for each rule ($r_i : I \rightarrow p$) $\in$ *Cl* in sequence do
if there is a lower ranked rule $r_j : I' \rightarrow p$ where $I \subseteq I'$
prune r_j
end if
end

Figure 2.4 MCAR classifier builder algorithm

ruleitems at iteration *K* in order to discover frequent itemsets at *K+1* iteration. As documented that the number of rules generated by the AC algorithms are large (Baralis et al., 2004; Li et al.,

2001). MCAR ranks and then prunes the redundant rules to form a set of classifier that have less number of rules, which are eventually easy to understand and handle. Rule ordering procedure of MCAR is shown in Figure: 2.5.

MCAR uses a different approach in ranking the rules. Instead of using the confidence, support and cardinality measures, it ranks by taking into consideration the class frequencies distribution in the main data and rules are prioritized which are linked with classes that are dominant. If two rules have the same confidence, support value and length of item, MCAR selects a rule that is associated with the dominant class. Rules with the same class frequencies are selected randomly.

Figure 2.5 shows the build classifier algorithm. After the classifier is built it is used to classify the test case data. MCAR uses a method, which implies that in the ranked rules, the first rule that matches the portion of the test instance classifies it. The default class is assigned to the test instance where no rule match is found for the test instance condition.

MCAR is found very competitive in terms of predictive accuracy when analysed with the traditional approaches like C4.5 and RIPPER, on 20 data sets from collected from UCI data repository. MCAR has shown good scalability when comparison is drawn between well-known AC technique CBA (Liu et al., 1998) with regards to prediction capacity, efficiency and rule features. MCAR has shown 2-5% higher accuracy than CBA and C4.5.

2.7.6 Multi-class, Multi-label Associative Classification (MMAC)

The first multi-label algorithm documented in AC is MMAC algorithm (Thabtah et al., 2004). All the steps in finding the classifier are the same as the algorithms described in the above sections. But the unique feature of MMAC is the capacity to produce rules with multiple classes. The idea behind the research was to keep the rules that are discarded by the existing AC approaches. For example an item '*x'* with class '*c1'* has a support count of 40% and same item '*x'* with class '*c2'* has support count of 20%. In other approaches of AC rule itemset *(x, c1)* is selected over *(x,c2) is* discarded but in MMAC both are kept. The information kept is very useful for the decision makers. In the first step MMAC generates rules; it scans all the training data to form all the possible set of CARs. The training cases associated with the discovered CARs are removed from the data. In the second stage MMAC finds more rules depending on some criterion set by user of *minsupp* and *minconf* from the left unclassified data, until no more rules

are left to be found. Finally, a global classifier is built by merging all the frequent rule itemsets generated in each iteration.

Experiments conducted on 28 unique data sets (Merz and Murphy, 1996) have shown that the MMAC is an effective and accurate classification approach. It has shown high competitiveness and found scalable when compared with traditional and other AC approaches like RIPPER, PART and CBA.

2.7.7 Class Based Associative Classification

In (Tang and Liao, 2007), a new AC technique based on class is proposed called CACA. The algorithm aims at reducing the search space in the generation of rules by introducing some innovations: 1, combine the rule generation and classifier building stages; 2, using the class based criterion to reduce the number of frequent patterns generated; 3, implement an Ordered Rule Tree (OR-Tree) structure to store and then in the iteration update the rules and the information; 4, compact set is redefined to make it unique and non-sensitive to reduction in rules. It scans the training data and keeps the items in a vertical format. Then it calculates the frequency of all the attributes and sorts them in descending order. The rules that pass a user defined criterion of *minsupp* are kept while remaining rules are deleted. For the rest of the attribute values in the first step, intersection between the attributes is carried out based on the class strategic to reduce the searching space to produce frequent pattern. The attributes values in a class group that pass the *minconf* threshold, are added in Ordered Rule Tree (OR-Tree) starting from the root node and the last node in the every path of the rule contains the information of the support and confidence. Like CBA, CACA also has the capacity to classify unseen data. Experiments performed on 15 UCI datasets have shown that CACA and MCAR have better accuracy than CBA. When compared with MCAR, CACA has shown slightly better accuracy while generating less number of candidate rules. The execution time of CACA is also better due to the reduced search space to generate candidate ruleitems than MCAR.

2.7.8 Associative Classification Based on Closed Frequent Itemsets (ACCF)

ACCF is another AC algorithm that was proposed in (Li et al ,. 2008) and consists of two phases, rule generation and Classifier construction. ACCF algorithm employs an extended version of

CHARM (Zaki et al., 1999) in mining the frequent itemsets that would be used later in extracting the set of classification rules (CARs). After passing over the training dataset, ACCF discovered the set of Closed Frequent Items (CFIs) along with their. *tidsets* as well as the class labels i.e. the rows numbers where each item occurred though CHARM approach (Zaki and Hsiao, 1999). By getting *tidsets* for both the items and their class labels, the support and confidence for each rule is then computed. Only those rule that survive the predefined thresholds *Minsupp* and *Minconf* are proceeded to the evaluation step where a rule evaluation procedure is invoked to evaluate the set of CARs produced in the previous step.

The set of CARs are ranked according to their confidence, support, rule length and first rule generated with no redundancy, Then they are evaluated using Database coverage pruning procedure adopted from CBA (Liu et al., 1998) where all rule that can't cover at least one training case will be discarded. In classifying test case *ts*, ACCF starts with the first ranked rule, the first rule applicable to *ts* classify it. In case where no rule is applicable to ts, the default class is assigned.

Experiential results against one AC algorithm states that the proposed algorithm has out-performed CBA with respect to the efficiency and effectiveness, however not enough details are demonstrated. For instance, details such as the time taken for learning and/or classification were absent.

2.7.9 Boosting Association Rules (BCAR)

In the boosting association rule (BCAR) approach proposed by (Yoon and Lee, 2008), very large number of rules are produced in the first step. Then the derived rules are filtered by using a technique that is equivalent to the deterministic boosting algorithm (Freund and Schapire, 1997). BCAR can be used in huge scale classification such as TC data. Experiments performed on text collections showed that the algorithm has achieved prediction better than SVM (Vapnik, 1995) and Harmony (Li et al., 2007).

2.7.10 Association Classification based on Compactness of Rules (ACCR)

ACCR was proposed by (Niu et al., 2009), which also is an extension of the Apriori algorithm. It generates rules and builds a classifier. If the value of support is set very low the rule generated

are large in number and are redundant and time to generate these rules is also high. If the support value is set higher then there is always a danger of ignoring good quality rules. In ACCR a new approach is adopted to keep the high quality rules by developing a metric measure called "compactness". It stores ruleitems with low support but high value of confidence. The compactness is calculated as:

$$Compactnes(I) = \sum_{i=1}^{m} \frac{Lift(R_i)}{m} \quad (2.2)$$

$$R_i = (I - \{I_i\}) \rightarrow I_i \quad (2.3)$$

$$Lift(A \rightarrow B) = \frac{Conf(A \rightarrow B)}{Sup(B)} = \frac{Sup(A \cup B)}{Sup(A)Sup(B)} \quad (2.4)$$

where the "lift" is the degree of independence between two items (A) and (B) of the measured rule A→B. If the value of the lift is near 1, the relationship between two items (A) and (B) is small. ACCR builds similar classifier as CBA. Experimental findings against the UCI datasets have shown that ACCR algorithm performs better when compared with CBA and CMAR.

2.7.11 Improved Classification Based on Predictive Association Rules

Classification based on predictive association rules was used first in CPAR (Yin X. and Han J. 2003) to generate CARs and build classification model, another algorithm adopt predictive rule . Traditional AC algorithms are spending more time in generating the set of rule and learning the classifiers since they have repeated calculations. Alternatively, Classification based on PRM takes less time due to the fact that it discovers the frequent items and generates rules simultaneously.

A new AC algorithm that uses PR in building the classifier is that to generate the set of CARs is proposed in (Hao et al., 2009) which is taking the advantages from AC and traditional association rules mining. The proposed algorithm has been developed in an attempt to fix some falls in its predecessor CPAR which lies in rule evaluation and the classification phases. Hereunder these falls are listed along with the suggested solution by the proposed algorithms:

(1) Class imbalancing since each class has imbalanced number of rules in it because CPAR consider each rule's weight only but not the weight for each class. This may lead to favoring the class with more rules to class the test case than the one with fewer rules. To overcome this fall, the proposed algorithm developed a new "Class Weighting Adjustment" which will help in balancing the power of classification rules by adjusting the weight criterion at each iteration.

(2) In CPAR, particularly in the classification phase, every class is treated uniformly for cases. This will increase the wrong classification rate. To overcome this issue, similarity calculations between the case and the "center vector" of each class is proposed. The algorithm only fetches the rules of a class whose similarity of center vector to the example is above the average similarity. Similarity calculation is done using vector space model (Hao et al., 2009). And (3) CPAR is not applicable for the cases that don't satisfy any rule. ICPAR proposed a Post-processing using SVM (Vapnik et al., 1995) is employed for classification because of the ability of SVM to avoid over-fitting as well as the ability of handling large feature spaces.

Experimental results show that ICPAR has greatly enhanced the efficacy and effectiveness when compared to CPAR.

2.7.12 Hierarchical Multi-Label AC using Negative Rules (HMAC)

A new Hierarchical Multi-Label AC algorithm has been developed in (Sangsuriyun, et al., 2010) that uses negative rules in predicting the class labels for test cases. It should be noted here that "mutli-abel rule" denotes a rule that have more than one class in its consequent such as r:I $\rightarrow$ $c1, c2$, "negative association rule" , given *Pa* and *Pc* where *pa* is the rule antecedent is a combination of three items, X: positive item-set, N: negative item-set and NP: negation of positive item-set and *Pc* is the rule consequent which can be one of three type, C: positive class, NC: negative class or NPC: negation of the positive class means none simultaneous presence of classes; a negative rule can be given as $Pa \rightarrow Pc$.

After generating the complete set of rules (CARs), HMAC invokes the ranking procedure based on a number of parameters, *F-measure*, *Jaccard*, *Support*, *ActOcc* and *rule length*. It should be noted here that HMAC replaced the confidence parameter by *F-measure, Jaccard*. In evaluation the set of rules, the algorithm then fires two pruning procedures (1) Pearson's

correlation coefficient procedure and (2) redundant rule pruning. The classification model is then constructed using only those rules that are not redundant and positively correlated. Lastly, in classifying a test case *t*, HMAC classifier, a test case *ts* is compared with the set of ordered rules, if *ts* matches rule positive class rpc_i without any rules negative class RNC it classifies it, if failed to match any rule then HMAC moves to the next rule set, if still didn't match any rule in all sets then the default class is assigned. Although it has been reported in this article that algorithms using Pearson's test can result in gaining good accuracy results, it is difficult to validate this as inadequate experimental results are available and much of the information relating to their generation is absent.

2.7.13 Probabilistic CBA

Research work on Class imbalancing issue is very rare since very little attention was paid to this issue; SBA (Liu et al., 2003) raised the issue by introducing a scoring mechanism for training cases in order to reveal the likelihood that the case belongs to a rare class. Although SBA employs pessimistic error estimation measure to perform pruning, but still the number of rules is a bit large and results in complication upon scoring step. Yet, a new AC algorithm called PCBA that addresses the class imbalancing issue by proposing a new pruning method has been introduced in (Wen-Chen et al., 2012) called PCBA pruning aiming to improve CBA in accurate prediction of rare data cases. CBA deals only with Class association rules of the form I→ **c**, where I is the set of items and c is the class that can be either positive or negative; these CARs are then ranked according to confidence, support and generated first rule. It should be noted here that CBA ranking procedure is adequate only when classes are evenly/ semi even distributed (Balanced data). On the other hand if the class distribution is not adequate, notably differ due to the fact that positive classes are often much smaller than negative ones when dealing with imbalanced data which may degrade the classification accuracy. Evaluation on six imbalanced dataset (benchmarking and real_life applications) is conducted; results revealed that the proposed algorithm performs better than C5.0 and SBA.

2.7.14 AREM: A Novel Associative Regression Model Based On EM Algorithm

AREM algorithm that uses the regression model is proposed by (Jiang and Karyris, 2012). It algorithm starts with extracting regression rules by applying a pruning technique that is instance based and the aim is to discover the best rules and then a EM algorithm is applied to train the probabilistic model. The AREM model consists of two components. First component is the rule discovery component and regression rules that are frequent are discovered. To find the entire frequent itemsets FP growth algorithm is used and then the instance based technique is applied to prune the frequent set of rules found in the first step. The second component, each regression rule value on the right hand side is updated after learning from a probabilistic model. The EM algorithm is an iterative technique used for learning and optimization.

The AREM performance is measured and evaluated on 10 data sets. The six data sets are downloaded from three websites of CitySearch, BestBuy and Yelp. The other data sets are downloaded from DataExpo09 and CMU StatLib. For the evaluation purposes mean Squared Error(MSE) is used as the performance metrics between predicted and actual variables. The model is trained on the training data sets MSE is calculated for each parameter and average MSE is calculated.

The results have shown that AREM has performed well as compared to all other AR and regressions models such as SVR, CART, Boosted Regression, Cubist when MSE is taken as the performance measure. It is also evaluated that AREM is more suited to sparse and high dimensional data sets.

2.7.15 Prefix Stream Tree (PST) for Associative Classification.

In data stream mining a PST (Prefix Stream Tree) model is used by (K.P. Lakshmi, 2012) in AC. It is not possible to store in the primary and the secondary memory during the data streaming. Extraction of the frequent itemsets from the recent data is a challenge. The efficient updation of the existing frequent itemsets with a addition or deletion of itemsets will increase the performance.Firstly a single full scan of data is carried out to build a compact structure of data.

The compact constructed structure helps in efficient mining in terms of time and memory. For the new data stream the compact tree is reconstructed. A memory efficient data structure called PStree is proposed in this paper.

A prefix schema is used to construct the PSTree. The tree uses the sliding window to capture the data and construct a tree, as the window slides batch by batch the tree is updated. The tree consists of nodes, where first node is known as root node and represented as *"null"*. Other nodes are known as ordinary nodes. The end nodes or last nodes in the tree are called leaf nodes and contains the class label, support and count of batch. A list of all itemsets called I-list is maintained and contains the support counts. Two phases that are used in the construction of PSTree are *Insertion* and *Restructuring*. Both the phases execute in a dynamic fashion and restructuring is carried out after insertion. Reconstruction phase is executed either by Branch Sorting method (Tanbeer et al., 2008) or Path Adjusting method (Koh and Shieh, 2004). The confidence of the extracted rules is calculated by using the FP-Growth mining algorithm (Han et al., 2000) and sorting is carried out in the memory. The frequent itemsets in the classifier are used to predict the class labels of the test dataset.

2.8 Use of AC algorithms in Medical Diagnosis and Recommender System

2.8.1 Use of Associative Classification and Genetic Algorithm in Heart Disease Prediction

AC classifiers are used in the applications where maximum accuracy is desired. The fact is explored by (M. A. Jabbar et al, 2012) by applying AC and genetic algorithm in predicting heart disease. The genetic algorithm is used to explore the prediction rules with high level so the rules discovered will be comprehensible, with high interestingness and high accuracy.

The model of the heart disease prediction system uses the APRIORI algorithm to discover the frequent itemsets that are large in number. To enhance the accuracy of the AC a hypothesis testing Z-statistics and informative attribute entered rule generation are proposed in the study. In the algorithm, use of Gini index is introduced as a filter to minimize the number of

candidate itemsets and the best attribute is selected. The attributes having Gini index value at the lowest are selected for rule discovery.

The evaluations are based on the 2 data sets from UCI repository and 6 data sets from SGI machine learning repository. Tenfold cross validation is used to evaluate the results. The performance of the proposed model is compared with Naïve Bayes, J48, Neural Network and GNP. The accuracy is increased by 1% using GNP than Naive Bayes and 4.6% increase in accuracy in breast cancer data set. An increase of 4% in accuracy is seen in the heart disease data when evaluated with J48 than NN. The proposed approach has shown 7.5% increase as compared to GNP for Pima Indian data. The Proposed algorithm has performed well than the well-known classification algorithms in terms of accuracy.

2.8.2 Artificial Immune System for Associative Classification

The proposed model is introduced by (Samir A. Elsayed, 2012) combines two novel approaches of AC-CS (Associative Classification with Clonal Selection) and ML-DS (Multi-Level Deterministic Sampling). The AC-CS while maintaining the running time enhances the accuracy. The algorithm starts with large sample selected deterministically from the dataset and then it runs in iterations. The data is divided into equal sizes of disjoint groups that are smaller in size. The subgroups with larger distance are discarded and the subgroups with less or minimum distance are kept. This process is iterated until all the rest of the transaction size becomes equal to the threshold values. The sampling technique implied by AC-CS, starts with a small size of frequent single ruleitems and then the ruleitems passes the process of cloning, pruning and mutation. The rules with the higher quality are added in the classifier, and then the rules are used to classify the test dataset.

The studies have revealed that the approach has generated less number of rules than benchmark AC algorithms. The execution time is reduced clearly for all datasets and the accuracy of AC-CS with sampling is almost similar as compared to without sampling. It can be concluded that the sampling approach used in the model is effective and produced better representative sample of the original data set.

2.8.3 Using Associative Classification for Treatment Response Prediction

In the research, a KDD framework is used to predict the outcome of the treatment with ribavirin and interferon in the patients with hepatitis C virus by (Enas et al., 2012). The framework contains main phases of pre-processing and data mining. In the initial phase of pre-processing the data is cleaned and suitable features are selected from the data of the patients and then proposed data mining technique is applied.

The data gathered at Cairo University from the data base of Egyptian patients, 200 cases were selected from the patients of hepatitis C virus genotype 4. The features selected are from the blood test, patient characteristics and response features. The three features like HAI, ALT and fibrosis stage are selected along with features of response and all are grouped together in such a form that data mining approach can be applied. To find the patterns among the selected features an AC technique of adapted PMA (Enas, 2010) is used to generate CARs and then these CARs are used to construct the classifier and the classifier is used to predict the possible response of the patient to treatment.

It is concluded in the paper that an AC algorithm is used to predict the response of the patients to the treatment with hepatitis C virus (HCV). The algorithm is trained on only 200 tuples and so the sample size is not very high and a much larger sample size is desired for future training of the algorithm. The accuracy achieved is 90% with adapted PMA.

2.8.4 Fuzzy Associative Classification Approach for Recommender Systems

A hybrid methodology is proposed for recommender systems by (J. P Lucas et al., 2012) which make use of the content based and filtering approaches. Fuzzy logic is also applied to improve the quality of recommendations. A fuzzy associative classifier called CBA-Fuzzy is developed, which has basics from CBA algorithm.

The CBA algorithm is obtained from the LUCS-KDD repository in the University of Liverpool and a portable CBA-fuzzy algorithm is built in Java using Java2 SDK. The approach consists of rule generator and classifier builder components. A hybrid methodology was build that can generate a set of CARs which are used for the classification. Three components of the proposed methodology are, building of group of users, generation of rule sets and recommendations. The two components that are built online and used to induce the estimated model, also includes the CARs produced by CBA –Fuzzy. At the runtime, the third component provides the user with recommendations and also responsible for classifying.

A case study was conducted for evaluation of proposed framework. One of the two experiments is performed to compare the accuracy of the classifier and the second for the comparative analysis of quantity of false positives. The experiments also take in to consideration the main reason for using AC classifier in recommender system, analyze the effects on associative classifiers performance due to sparsity and general classification based on association compared with fuzzy logic and false positive rate is compared with learning classifiers and proposed CBA-fuzzy algorithm. The performance of the algorithm is analyzed using five datasets, one extracted from MovieLens that contains rating of the movies by users in 200 and others from BookCrossing databases. To estimate the accuracy of the algorithm tenfold cross validation method is used and experiments are conducted using WEKA tool for data preprocessing and transformation.

In the study C4.5, FURIA, BayesNet, CPAR, CBA and CMAR are analyzed. WEKA tool is used to run C4.5, FURIA and BayesNet and rest of the three are taken from LUCS-KDD repository. The purpose of the experiments is to compare the accuracy of the algorithms. And before experiments, the sparsity of the data sets is measured. The sparse datasets are BCrossing World and BCrossing USA and less sparse data sets are BCrossing USA10 and BCrossing World10 and MovieLens is dense data set.

The results have shown higher or similar accuracy of AC algorithms on Book Crossing data except CMAR when compared with classification algorithms. The CBA achieved the highest accuracy on two datasets of BookCrossing. CMAR achieved the highest accuracy on MovieLens data. It is concluded from the experiments that AC methods can be effective in recommender systems than classification techniques. It is also concluded from FURIA results that fuzzy rules have produced better results in recommender systems and hence can be used in development of Fuzzy AC systems.

2.9 Current Pruning Methods

As we know now that in the process of rule generation the rule are generated in AC algorithms by a factor of $2^{(n)}$-1, where ‘n’ is the number of frequent attribute values. If the value of ‘n’ is large the number of rules generated is obviously large. Handling of these huge numbers of rules in large data sets consumes storage and needs longer execution times. In the real world the data is growing on daily basis, so we need some optimization techniques that can reduce the size of the

rules without affecting the predictive capability of the rules generated. Many pruning methods are adopted to shrink the size of the classifier in Associative classification. There are two specific points documented where pruning methods are applied; 1, during the process of classifier building such as pessimistic estimation (Quinlan, 1987) used in decision trees and from statistics like Chi-square testing ($\chi 2$) (Snedecor and Cochran, 1989). At this point the ruleitems that have less support value than *minsupp* are eliminated and frequent ruleitems are produced. Pruning technique like chi-square testing is used during the rule generation stage.2, Post classifier builder pruning methods such as database coverage (Liu et al., 1998) are used when all the rules are generated. In the following section pruning method in AC are explained.

2.9.1 Database Coverage

The main focus in our research is on Database coverage technique as it is used by CBA and is also used by our proposed algorithm for the sake of comparison. Database coverage approach is a post pruning technique and used in AC (Liu et al., 1998) when all the rules have been discovered. This technique starts by picking each rule with highest ranked rule, all training cases that fully covered by the rule and the class is matched are marked for deletion from the training dataset and the rule gets inputted into the classifier. In cases where a rule cannot cover a training case (the rule body does not fully match any training case) then the rule is discarded. The database coverage method ends when either the training dataset gets is totally covered and becomes empty or there are no more rules to be evaluated. In the case when no more rules are left without evaluation, the remaining uncovered training cases are used to generate the default class rule which represents the largest frequency class (majority class) in the remaining unclassified cases.

It should be noted that the default class rule is used during the prediction step in cases when no rule covers that instance of the test data. When all the rules are inserted then *cut-off rule* is identified, it is defined as a rule with minimum number of errors. The rules after the *cut-off* point are removed from the classifier as they will only show errors and will affect the accuracy. Database coverage method has been criticized by (Baralis, et al., 2004) since in some cases it discards some useful knowledge. Alternatively, they urge that rich classifiers often provide useful and rich knowledge during the classification step.

2.9.2 Redundant Rule Pruning

Rule generated in AC algorithms are placed in classifier when they satisfy some conditions. All rules have rule condition that contains combination of different attribute values of the training data. The combination of attribute values in one rule condition can be same in another rule condition. It means that we can have rules in the classifier that are redundant. If the rule contained in the classifier is extremely large, many redundant rules exist. Redundancy of rules is unnecessary in classifier and is a serious issue. The process of removing redundant rules from the classifier is carried out by redundant rule pruning.

Redundant rule pruning removes some particular rules containing low confidence value than remaining other rules was introduced in (Li et al., 2001). The algorithm works as described below: when the rules generation and ranking phase finishes, then these rules are pruned by an evaluation step like $I' \to C$ from the generated rule set, where general rules $I \to c$ of a high confidence are present and $I \subseteq I'$. This redundant pruning approach decreases the size of the final classifier and reduces the rule redundancy (Li et al., 2001).

Algorithms like CACA (Tang and Liao, 2007), ACN (Gourab Kundu et al., 2008), ARC-BC (Antonie and Zaïane, 2003) and CMAR (Li et al., 2001), have used currently discussed technique of redundant rule pruning. The rule is firstly added in the CR-Tree, the algorithm checks the redundancy in the compact data structure, either it removes another already existing rule from the CR-Tree or it does not insert the current rule in hand.

2.9.3 Pessimistic Error Estimation

Two pruning techniques of pre pruning and post pruning (Witten and Frank, 2000) are worth mentioning here used in decision trees. The latter approach is more popular and it uses backward pruning. Backward pruning is adopted by most decision tree methods such as C5 and C4.5 (Quinlan, 1998, 1993). Pessimistic error estimation is mainly used in data mining within decision trees (Quinlan, 1993) in order to decide whether to replace a sub-tree with a leaf node or to keep the sub-tree unchanged. The sub-tree replacement is the phenomenon of a leaf node substitution by the sub-tree. On the basis of training data, error is measured by pessimistic error estimation measure.

The pessimistic error estimation has been exploited successfully in decision tree algorithms including C4.5 and C5.0. In AC mining, the first algorithm which has employed pessimistic error pruning is CBA. For a rule *R*, CBA removes one of the attribute value in its antecedent to make a new rule *R'*, then it compares the estimated error of *R'* with that of *R*. When the estimated error value of *R'* calculated as smaller as compared to *R*, then the original rule *R* gets changed with the new rule *R'*.

It should be mentioned here that CBA employs two pruning methods, pessimistic error and database coverage. Some studies reported that employing several pruning procedures may affect the accuracy rate (Baralis, et al., 2004) (Abumansour et al., 2010) (Thabtah et al., 2011).

2.9.4 Lazy Pruning

Few of the AC techniques hold a strong argument when they say that pruning process should be restricted to "negative" rules only (Baralis et al. 2008 ; 2004) and (Baralis and Torino, 2000), because negative rules lead to inaccurate classification. They also argue that database coverage pruning removes some rules that contain very useful knowledge. Alternatively, Lazy based associative algorithms store those rules discarded by database like methods in a compact-set aiming to use them during the prediction step especially when no primary rules cover a test case. Lazy pruning is initiated when the rules are extracted and stored, it starts by taking each training instance one by one and the instance is checked by the first rule in the ranked rules set. If the rule correctly classifies the training instance, it will be inputted into the primary rule set, and all of its corresponding cases will be deleted from the training dataset. Whereas, if a higher ranked rule covers correctly the selected rule training case(s), the selected rule will be inserted into the secondary rule set (Spare rule-set). Lastly, if the selected rule does not cover correctly any training case, it will be removed. The process is repeated until all discovered rules are tested or the training data set becomes empty. At that time, the output of this lazy pruning will be two rules sets, a primary set which holds all rules that cover correctly a training case, and a secondary set which contains rules that never been used during the pruning since some higher ranked rules have covered their training cases.

The distinguishing feature between the database coverage and lazy pruning is that the secondary rules set which is held in the main memory by the lazy method is completely removed during building the classifier by the database coverage. In other words, the classifier resulting

from CBA based algorithms which employ the database coverage pruning does not contain the secondary rules set of the lazy pruning, and thus it is often smaller in size than that of lazy based algorithms. This is indeed an advantage especially in applications that necessitate a concise set of rules that the end-user can control and maintain.

Experimental findings based on 26 different datasets from (Merz and Murphy, 1996) outlined in (Baralis et al. 2008; 2004) and (Baralis and Torino, 2000) have shown that approaches that uses lazy pruning like L^3G and L^3, depict higher accuracy than other approaches that incorporate database coverage pruning method (Li et al., 2001) and (Liu et al. 1998) on average by 1.63%. The main disadvantage is that lazy pruning may produce a large classifier, which adds up the difficulty level for a human to interpret and understand. The experiments have indicated that AC techniques which uses lazy pruning methods utilizes more memory space in comparison to other AC methods because they generate and keep the rules in main memory and there is always a fear of failure of this approach when the support values are kept at a low value.

2.9.5 Laplace Accuracy

Laplace accuracy by (Clark and Boswell, 1991) is used in classification and AC problems to calculate the accuracy of derived rules. The formula to calculate the accuracy of a rule, *r*, is:

$$\text{Laplace}(r) = \frac{(p_c(r)+1)}{(p_{tot}(r)+m)} \qquad (2.6)$$

where $p_{tot}(r)$ refers to the no. of instances covering *r* antecedent, *m* refers to the no. of classes that occurs in the training data and $p_c(r)$ refers to number of instances matched by *r* .

CPAR (Yin and Han; 2003), uses this approach to prune the rules and calculates the expected accuracy of each rule before it classifies test data. This is to ensure that only those rules with best expected accuracy are participating in the classification phase. One disadvantage in the proposed algorithm is that the rules are of a lower quality than those generated through other AC algorithm. The reason is that CPAR is using greedy algorithm (FOIL) and rule *r* is generated for the remaining cases in training dataset instead of the whole dataset.

Experimental findings on 26 data sets (Merz and Murphy, 1996) have demonstrated that CPAR performed more accurately by an average of +0.48% and +1.83% than CBA and C4.5 algorithms, respectively.

2.9.6 Boosting Weak Association Rules

The Boosting Weak Association Rules algorithm (Yoon and Lee, 2008) uses the principle of boosting and is a modified version of database coverage pruning. The algorithm selects a rule and checks on all the training data .The value of the cover count V_i is incremented when the rule correctly predicts an instance in the training data. This process of updating the cover count V_i continues till it exceeds a coverage threshold. All the instances covered by the rule are deleted from the training dataset.

2.9.7 I-Prune

Item prune is a recent proposed method in (Baralise and Garza, 2012), I-Prune is a pre-pruning method and tends to mark uninteresting items based on interestingness measure (correlation measures e.g "Chi Square", "Lift", "Odd ration") and remove them and use only interesting items to build a high quality rules which will be used in building the classification model. Consequently, such early pruning step will reduce the number of generated rule as well the tame taken for learning the classifier.

Several AC algorithms such as CBA, CPAR, CMAR, and MCAR consider an item interesting according to the support count. Alternatively, I-prune selects only those that are frequent and correlated, Given an item *i* that is correlated to class *c*, an interestingness measure is given as follows: if *interestegness-measuer* (*i*,*c*)> *predefined-threshold* then *i* is selected else items are discarded as soon as detected. Assume *I* is a subset of frequent and correlated items with respect to class *c*, set of rules *R* is generated for *c*.; only the rules that contains interesting items are generated. On the other hand, I-prune may inadvertently discard some items that might produce useful classification rules. Among the set of all measures used to measure, experimental results shows that Chi Square is the best correlation measure with respect to effectiveness. Lastly, I-Prune can be easily and effectively integrated with a number of AC algorithms especially those are Apriority based algorithms such as CBA.

2.9.8 PCBA Based Pruning

Class imbalancing problem has not received big attention in the context of information retrieval and data mining approaches. Classifiers with Imbalance class examples may increase the misclassification ratio (Liu et al., 2003) (Wen-C et al,. 2012).

A new pruning method that consider class imbalancing has been introduced in (wen-C et al,. 2012), PCBA pruning method was proposed in an attempt to deal with imbalanced class when the associative classification is applied. Conventional AC algorithms used one fixed minsupp and minconf which might be working properly when dealing with balanced data but not for imbalanced one. Alternatively, the algorithm uses different minsupp and minconf values, these are defined based on the rule distribution each class through "under-sampling" concept. PCBA is proposed aiming to adjust CBA algorithm by proposing a new pruning method that filters the set of deemed CARs to be suitable to SBA algorithm which will lead to better accuracy in cases where classes are imbalanced. PCBA has improved three aspects in the original CBA, which are: adopting sampling method which usually used to adjust the confidence of the minority classes that lead to enhancement on the ranking effectiveness and decrease the level of class imbalancing that consequently makes minority classes more common. There are two sampling approaches, "under-sampling" and "over-samples", due to the fact that over-sampling might cause over-fitting problems; PCBA adopts under-sampling approach in adjusting the confidence of the rare CARs. When under sampling is used during the ranking procedure it controls the amount of positive rules by decreasing the amount of negative rules.

Second issue addressed in this work is setting multiple minsupps and minconfs thresholds, setting single value is not appropriate when dealing with imbalanced data since two problems might occur; when setting the support to high value, it becomes impossible to locate those rule from the minority classes and on the other hand, setting the support to a low value in order to locate the rule from minority class will lead to a huge number of rules which need high computational cost and might degrade. As for setting the confidence threshold to low value results in having ~~a~~ meaningless rules on the other hand high confidence value will eliminate the possibility of having classes form minority classes.

2.10 Current Prediction Methods in Associative Classification

Prediction measures are used to judge the accuracy of the classifier built at the end of all the rule generation and pruning phases. It is performed on the test data whose class label of each instance is unknown. The test data is actually the training data with hidden class values. The prediction approaches apply the rules generated to predict the class labels of the test data and computes the prediction accuracy percentage at the end of this process. In AC predicting class labels are divided into two main groups, one is single Accurate Rule Prediction and second is Group of Rules Prediction. In the former approach prediction is based on single rule that has highest precedence and is applied to the test data. The latter measures the prediction on the basis of multiple rules. Both the approaches are discussed in the section below.

2.10.1 Single Accurate Rule Prediction

Consider a set of classified rules R and a test data instance *test(i), where i ={1,.....testdata. length},* single rule prediction approaches considers the top ranked rule with high confidence value in R and matches the *test(i)* attribute values or the body of test instance. Where there is no rule in R that can predict any instance of the *test(i)* then the instance is allocated the value of the default class. The majority class in the training data that is not covered during pruning process is called the default class.

The AC algorithms that use the single rule comparison approach for prediction are (Liu et al., 1998; Baralis and Torino, 2000; Wang et al., 2000; Tang and Liao, 2007; Baralis, et al., 2004; Li et al., 2008; Niu et al., 2009; Kundu et al., 2008). It is simple, useful and effective approach for classification. Rules with the high confidence values are used for prediction and these contribute mainly in test instances classification. When the confidence value is high, likelihood of the test instance to be measured correctly is also high and so is the probability measure to predict the test instances correctly (Liu et al., 2003). However this approach can show some drawbacks when data sets have uneven class distribution (Liu et al., 2003; Li et al., 2001) and there could be more than one rule with same confidence in 'R' that can match the test instance. The above shortcoming led to a discovery of more appropriate technique which groups a small subset of the rules having same confidence value but different class labels. Grouping of

rules in a subset have shown better results. The next section will explain different ways in which this approach is used.

2.10.2 Group of Rules Prediction

The process to predict the test instance and the decision to allocate a class value, multiple rules having nearly same confidence values are applied to match the conditions of a test instance. In (Vyas et al., 2008; Yin and Han, 2003; Antonie and Zaïane, 2002; and Li et al., 2001) it is argued that making decision on one rule has shown poor results. Some techniques in AC are reviewed in the next section that incorporated multiple rules in the prediction step.

2.10.2.1 Score based Prediction Methods

A new score based prediction method is proposed in CMAR algorithm (Li et al., 2001) that selects all the rules with high confidences that can be applied to a test instance and calculates the associations between the selected rules. The correlation is measured to estimate the bonding between the rules, which is calculated by considering the support and frequency of the class by a method called weighted chi-square (Li, 2001).

CMAR finds a group of rules 'C_k' from a discovered classifier in '*C'* those cover the test instance *test(i)*. If the same class is predicted by all the rules in subset 'C_k' then this class is obviously assigned to the test instance *test(i)*. But if the rules in 'C_k' points to different classes, the algorithm CMAR divides the classes into groups and then the strength of each group is measured and compared. The strength of the group is determined by the support and correlation among the rules. The class of the group showing the highest value of strength is selected and allocated to the *test(i)*. Weighted (χ^2) analysis (Li, 2001) tries to find the positively of the rules in the group which in turn helps to evaluate the correlation within each group.

A related prediction method is introduced by (Dong, et al., 1999), final decision to select a class label for a test instance is based on all emerging patterns *(ep)* of a class that have the test instance. Suppose a test database (*test*), all the classes that are present are assigned an estimated score. The score is formulated from the emerging patterns *(ep's)* that covers *test(i)* for the related class, and the test instance is linked with class containing highest score value.

Experimental results have shown that (Dong et al., 1999) and (Vyas et al., 2008) classification methods that have used correlation in a group of rules to predict the class label of a test instance increased the prediction accuracy slightly when comparison is drawn with single rule methods.

2.10.2.2 Laplace based Prediction Method

It is also a prediction method implemented by CPAR (Yin and Han; 2003) and uses multiple rules for prediction. The groups are compared by using the Laplace expected error estimates (Clark and Boswell, 1991) and here to classify a test instance, expected accuracy of all rules is calculated in advance. The process to classify a test instance in CPAR works as follows: 1) it selects all the rules from the classifier *'C'* whose attributes values satisfies the test instance. 2) Selection of the best rule is determined for each class from the step 1. 3) Comparison of average expected accuracy is done among the best rule for each class and the one rule is selected with the highest expected accuracy to predict the test instance.

2.11 Phishing in Websites and Emails

Communication through email and addiction to internet has become an effective way of people contacting each other in today's fast-paced world. People access different social, educational, informative and other kind of websites, exchange a huge quantity of email messages daily to share texts, files, photos and video links. Phishing is one of the active social engineering practices used to gain advantage of website users unaware of phishing traps (Toolan and Carthy, 2010).

Shortcoming in email and web security technologies also allows abusers to take advantage of the unaware web/email users. In computing, "phishing" is termed as an activity that uses social engineering techniques to try to get confidential information from the email/web user, such as identity, usernames/passwords, pin codes, bank account details and credit card information, by pretending to be someone else and hiding their real identity (Ma et al.,2009).

Due to the number of the users using the internet are increasing rapidly, the numbers of phishing attacks are also increasing**.** The increase of 59% in phishing attack volumes is reported in 2012 than 2011 and globally the losses due to phishing are estimated at $1.5 billion in 2012 an

increase of 22% than 2011(RSA's Report, 2013). The countries that are mostly affected by attacks are UK, USA, Canada, South Africa, Germany, France, Colombia and Brazil. The phishers attack mostly the financial activities and because of better economic growth of Canada the phishing attacks increased to 400% in 2012.

In 2012, cyber criminals have used the simple hosting method tactics and targeted the hijacked websites to launch the phishing attack. The web shells, smarter web analytical tools and automated toolkits are used to hack huge number of websites (RSA Report, 2013). The RSA analysts have noted that the combined attack schemes are in use to phish users and redirecting them to infection points.

In 2013, the launch of 4-G channels in mobile communication and the growth in the use of the mobile usage in the personal and office life, and the increasing pivotal need of the application in mobiles. It is forecasted in 2013 that the phishing attacks expected to be more directed at the mobile and smart phone users. The expected attacks would be by voice (vishing), Mobile applications, SMS (smishing) and spammed emails that the user will open on their mobiles. The use of social networking, shopping and gaming applications is very common. Only from the Google and Apple stores 25 billion app are downloaded in 2012 and by 2015, the number may grow up to 185 billion (RSA's Report, 2013).

2.11.1 What is Phishing

The idea behind phishing is that bait is thrown out in the web by abusers with the hope that a less aware user will grab it and bite into it just like the fish. In most cases, bait is floated either through an e-mail or disguised web page that requires user to input data that are hosted by a phishing website. The popular methods usually used by fishermen to camouflage are to be a well-known bank, Online tradesman, credit card company, people who say their deceased parents have left huge sums of money and they want help to get the money by offering a part out of the sum, and emails congratulating that a lottery is won by you and require some personal information to redeem the cash. Victims of phishing websites may lose their identity, pin codes, passwords, bank account and credit card details to the phishing email senders.

Phishing is sometimes confused with spam and conventionally spam blocking techniques were tried to block it, but were not as effective due to the structural variance and close resemblance of phishing websites to legitimate websites. The main difference between SPAM

and phishing is that SPAM is used for mass advertisement to sell a product whereas phishing is employed to collect information that can be further used for some illegal activity (Irani et al., 2008).

As the world is becoming a global village, quick access to useful and actionable information has become very vital for accurate decision making and to survive in this very competitive market. Although, phishers are now employing several techniques in creating phishing websites to fool and allure users, they all use a set of mutual features to create phishing websites. Since, without those features they lose the advantage of deception (Sophie et al., 2011). This helps us to differentiate between legitimate and phishy websites based on the features extracted from the visited website.

Overall, two approaches are employed in identifying phishing websites. The first is based on blacklist (Sanglerdsinlapachai and Rungsawang, 2010), in which the requested URL is compared with those in that list. The downside of this approach is that the blacklist usually cannot cover all phishing websites since, within seconds, a new fraudulent website is expected to be launched. The second approach is known as heuristic-based methods (Sophie et al., 2011), where several features are collected from the website to classify it as either phishing or legitimate. In contrast to the blacklist method, a heuristics-based solution can recognize freshly created phishing websites in real-time (Miyamoto et al., 2008). The efficiency of the heuristic-based method, sometimes called features-based method, depends on picking a set of discriminative features that could help distinguish phishing websites from legitimate one's (Guang et al., 2011). The way in which the features are processed also play an extensive role in classifying websites accurately.

Data mining is a process of extracting meaningful information from a large data bank (Fayyad et al., 1998). Data mining and knowledge discovery techniques have been employed in different capacities including financial analysis, market forecasting, retail industry and decision support systems (Toolan and Carthy, 2010). The two important data mining techniques which are discussed in Chapter 3 are association rule mining and classification rule mining. Classification and association rule are alike unless classification rule mining exercises prediction of one attribute, for instance, the class, on the other hand, association rule discovery can describe the relationships among attributes in the data set (Thabtah et al., 2010).

As mentioned in this chapter, AC integrates association rule mining with classification to find additional knowledge missed by traditional classification techniques (Furnkranz, 1999).

Many experimental studies, e.g. (Hao et al., 2009), (Sangsuriyun et al., 2010) and (W-C et al., 2012) showed that AC is a high conceivable technique that developed more predictive and accurate classification systems than traditional classification methods like decision trees (Abu-nimeh et al., 2009). This is axiomatic while AC finds hidden correlations among the different features. Moreover, many of the rules found by AC methods cannot be found by traditional classification techniques such as decision trees, PART (Frank and Witten, 1998), RIPPER (Cohen, 1995), Prism (Cendrowska, 1987) since the AC algorithm discovers all relationship between the class attribute and the other attributes in the training data set.

Through this part of the chapter, the intention is to discuss the intelligent data mining techniques to solve the complex problem of detecting phishing. This will surely show the usability of the developed AC algorithm in practical applications. The detailed review of common approaches in machine learning and data mining that are currently used to detect phishing will be demonstrated and the detailed description of the methods used to extract the features from the website will be discussed in later chapter 5.

2.12 Common Approaches to Detect Phishing

The Internet community in general has a reasonable understanding of the spam problem. Phishing, on the other hand, is a relatively newer and more insidious threat (Irani et al., 2008). While spam and phishing are similar on the surface, phishing attacks are comparatively sophisticated and logistically different from spam in a number of ways. Understanding these differences is important when designing a framework for the protection of users from these attacks. Here we examine some more important differences between spam and phishing.

The messages and techniques used in phishing attacks are complex and often amended than the techniques used in spamming. Messages that phishers send out are usually carefully constructed to impersonate familiar and reliable financial institutions or organizations. Many anti-spam filtering systems are unable to distinguish phish websites or emails from legitimate websites from these organizations due to such close resemblance to the real communication (Toolan and Carthy, 2010). Phishers also systematically exploit software vulnerabilities in web browsers, web servers, and local operating systems in order to trick both filtering software and users, and steal as much information as possible (Ma et al., 2009).

Unlike spam, phishing messages are directed to a more targeted user base in which spammers often send messages in bulk without any specific target in mind in an attempt to increase their response rate. Phishers normally target carefully selected email addresses lists and individual entities by which they try to evade less sophisticated data collection and filtration systems which might alert authorities of their scam.

A major difference between spam and phishing is that spam is dumped on the internet in large batches whereas phishing on the other hand is targeted and has a short life span on the internet to avoid getting too many hits that it eventually becomes noticeable in the community and is detectable by various filtering solutions. Moreover, spammers advertise a specific range of product brand mostly have a static source of emergence whereas phishers who tend to attack specific targets do not have a fixed source of emergence simply to avoid or obfuscate the real source of attack. Phishers dynamically install web server software through Worms and Trojans onto vulnerable or compromised hosts to launch their attacks. Also the phishers keep switching between multiple hosts to avoid detection of source, and mostly the switching is automated. Hereunder common approaches related to phishing are reviewed.

2.12.1 Pilfer Approach

Pilfer is a machine learning and adaptive method that extracts the features in the websites and analyses them to predict an outcome or classify the website as phishing or good also called as 'ham'. For instance, the authors of (Fette et al., 2007) have selected ten features like IP-based URLs, Age of linked-to domain names, HTML emails, number of domains and number of dots etc., from a large set of features, which have significance in detecting phishing in websites. The authors have acknowledged that these features can be used to detect the website phishing attacks in the browser environment. The phishing websites usually lasts for two days or so, making it difficult to extract information from aged emails. WHOIS (i.e., DNS) query is often performed to establish the date when the domain of a phishing website is registered. The success rate of WHOIS query to establish correct registration dates is 505 out of 870 different domain names.

Experiments (Fette et al., 2007) were performed using the ten-fold cross validation method and comparative evaluation was conducted on PILFER and SpamAssassin (Apache Software Foundation, 2006). The open source data sets used in the evaluation are ham corpora (Apache Software Foundation, 2006) containing non phishing 6950 and phishing corpus having

860 websites data. PHILFER showed 96% accurate classification of phishing websites with 0.0013% wrong classification of genuine websites (false positive) and 0.035% mis-classification of phishing websites (false negative). The results have revealed that spam and phishing filter combination proves to be an excellent solution, as the accuracy is improved when the decision of a spam filter is taken as a seed to PILFER, as compared to their performances alone. The authors concluded with thoughts on the future for such techniques to specifically identify deception, specifically with respect to the evolutionary nature of the attacks and information available.

2.12.2 Machine Learning Approaches to Detect Phishing

The predictive accuracy of phishing emails were detected by different Machine Learning approaches like CART (Breiman et al, 1984), SVM (Joachims, 1999), BART (Fette et al., 2007), RF (Breiman, 2001), and Neural Networks (NN) (Marques 2001) in a comparative study of (Abu-nimeh et al., 2007). The comparative study of the above classifiers used 43 features from the phishing email data for training and testing. The phishing data set that represents newer trends in phishing emails was constructed from 1171 phishing, 1718 legitimate emails and contains 2889 emails in total. In the comparison, the authors used a number of evaluation metrics but here the focus is only on two evaluation measures of false positive and false negative. False positives have more significance as users dislike when their important messages are deleted due to misclassification. On the other hand false negative are the email messages that are malicious (phishy) and classified as genuine.

The results of the experiments on number of simulation runs have shown that LR algorithm produced the lowest false positive (*FP*) rate of 04.89% followed by BART algorithm with FP value of 05.82%. The highest FP value is generated by LR algorithm and it was 08.29%. Though, the RF algorithm has the lowest false negative (FN) value of 11.12% and NN has the highest FN value of 21.72%.

2.12.3 Bayesian Additive Regression Trees (BART)

Due to the increase in the use of mobile devices to browse the internet recently, the likelihood of SMishing and Vishing attacks (Abu-nimeh, et al., 2009) have increased, as they are weak in their protective mechanisms. The features of phishing email are changing on a regular basis and the

use of the entire variables in training phase may lead to over fitting and poor prediction results. (Abu-nimeh et al., 2009) introduced a client-server model to detect phishing emails by using automatic selection of variables in Bayesian Additive Regression Trees (BART). The model used by the authors in the classification process is called CBART. The CBART selects the variables in the training phase automatically from a huge set of input features and discovers the relationship among features and response. As variable selection is a tedious and computationally complex process, methods like Bayesian Networks and Naive Bayes (Duda and Hart, 1973) perform this process separately hence classifier performance is poor as compared to CBART technique.

A phishing dataset containing 178 financial and 4974 other legitimate emails, and 1409 phishing emails, in total containing 6561 records with 71 features were used in the experiments. CBART selects frequent variables in Markov chain Monte Carlo (MCMC) simulations. The experiments are performed using 60 features that appear frequently in the phishing emails and TF/IDF (term frequency/inverse document frequency) is calculated by multiplying the number of occurrences of a word in a document to a function that computes the inverse of the total number of documents where that word appeared. The remaining 10 features are extracted from the structural behaviour of phishing emails, and the effectiveness is compared with Kruskal-Wallis (KW) test (Howard and Lee, 2006).

CBART has been compared with common classification data mining techniques such as Support Vector Machines (SVM) (Joachims, 1999), Classification and Regression Trees (CART) (Breiman et al, 1984), NN (Marques 2001), Random Forests (RF) (Breiman, 2001) and Logistic Regression (LR). The results depicted that an AUC (area under the ORC curve) value is lowered for all the classifiers except for the LR algorithm. While when variable selection is done through CBART, false negative rate, false positive rate and error rate showed a decrease in all approaches except SVM. When the CBART is applied the classifier's false positive values are -0.24%,-1.65%, +3.36%, 3.15%, 2.39% for CART, RF, SVM, LR and inNNet respectively.

2.12.4 Multi-tier Classification of Phishing Websites and Emails

Many investigations have used classification algorithms to filter the phishing websites and emails such as (Zhang et al., 2003) and (Sanglerdsinlapachai and Rungsawang, 2010). An approach presented in (Islam et al., 2009) first extracted the features from the suspicious

websites and then three classification methods named as support vector machine (Joachims, 1999), AdaBoost and Naive Bayes (Duda and Hart, 1973) is applied to detect the correlations among the features attribute values. The multi-tier filtering of phishing emails system relies on the spam filtering. The algorithms works by classifying the email messages in a sequence and sending the output to decision fusion process. The emails are sent to their mailboxes according to the class label. In case the email is misclassified by classifiers at any tier, the output will be finalized by the last tier. Different classifier is used at each tier and the results of the experiments have shown a reduction in the false positive and improvement in efficiency.

2.12.5 Hybrid Features using Information Gain

The phishing websites and emails contain many different features and considering all the features for the experimental studies can lead to complexity. The approach used by (Ma et al., 2009) works by generating a classifier while exploring each feature from the phishing data set, using the evaluation metric of information gain (IG) and then a set of high quality features were selected. The author defined three types of features: content, orthographic and derived. Out of these types, 7 features were extracted from each email and were used in the experiments. The model represents four components: Feature generator, machine learning method selection, inductor and feature evaluation (Ma et al., 2009). The model starts by generating a matrix of features from websites or documents by feature selection component. Then few selected machine learning algorithms such as C4.5 (Quinlan, 1993), RF (Breiman, 2001), and SVM (Joachims, 1999) are applied on the feature matrix and their performance evaluations are done. The main purpose of this step is to select the best machine learning algorithm for detecting phishing websites. Then induction step generates IG and lastly feature evaluation step is performed by calculating the accuracy of a feature before and after its removal from a small vector space. The last procedure is repeated several times till a best group of features are generated.

The experiments are conducted using a data set of 659,673, consisting of both phishing and legitimate websites, of which 7% of the websites were phishing. The results demonstrated that decision tree based algorithms such as C4.5 (Quinlan, 1993) have outperformed other machine learning algorithm in terms of prediction accuracy. The prediction values of other algorithms in comparison to C4.5 are, RF (- 0.02%), multi-layer perceptron (-0.72%), and SVM (-1.92%). The accuracy of the short feature selected data of the training and testing phases when

compared with original data came out to be same. So the author concluded that with feature selection few features are used in the classifier building and prediction without affecting the accuracy.

2.12.6 BoosTexter

Profiling is very useful to judge or predict the intention or the possible activity of an individual or a group of phishers (Toolan & Carthy, 2010). The method proposed by (Yearwood et al., 2010) profiles the phishing emails by considering the structural features of websites that are sent to different persons and then gathering information about the hyperlinks from the WHOIS database like APNIC and RIPE NCC. The properties of the structural features are taken as classes and for experimental purposes, three data sets are extracted from the information of hyperlinks by applying the boosting algorithm known as AdaBoost and a classification algorithm based SVM (Joachims, 1999). During the profile building up, the classifier generates the prediction weights for all classes to show the significance and relevance of each class. The main aim of the above process is to extract the classes that are more presentable in profiling and explore useful features.

The results are obtained from 2038 phishing websites that contain hyperlinks using cross validation. AdaBoost was executed 300 times and profile generation is performed. The performance analysis is performed by one-error, coverage and average precision with BoosTexter. The findings of the BoosTexter have shown high values of accuracy with the multi-label classes if contrasted with the other classification algorithms.

2.12.7 Phishing Evolving Clustering Method (PECM)

A new clustering base learning model called Phishing Evolving Clustering Method (PECM) is proposed in (Al-Momani et al., 2011), and is a modification of evolving clustering method for Classification (ECMC) (Song and Kasabov, 2003).The PECM model works in online mode and its functionality is to extract the similarities within the groups of the phishing dataset of emails. The method consumes less memory and generates the classifier from only one pass. The model works in 3 stages, first is the pre-processing of the emails where parsing is done to discover the features of phishing emails and stemming is performed to remove the test data linked with the

phishing email features. In the second stage the ranking and classification of the features is carried out and the highly effective feature is selected followed by creation of crisp values and then the features with the similarities are grouped together like body features and URL features. In the last stage, the ECM (Song and Kasabov, 2003) algorithm is applied on the data pairs (x,y) where 'x' is the group value of the feature and 'y' is the output . The learning phase works in a sequential manner and deals with the input vectors having the knowledge of class label and then the new input vectors are classified. The new email feature values are entered, and depending on the distance of the input vectors with the rule nodes, if the input vector values are present in one of the fields of rule nodes it will belong to the class of that rule node. Otherwise, the input vector 'x' will be placed with the closest rule node.

The results have shown that PECM is very effective to classify websites, while it decreases the false negative and positive values.

2.12.8 Data Mining Classification Methods

A phishing case study is demonstrated and implemented in real world to explain the process of phishing in e-banking by (Aburrous et al., 2010). The data collected clearly indicated that the users are very vulnerable to phishing attacks that can steal the customers' online banking details. Six classification methods were contrasted and comparative analysis was demonstrated for accuracy, performance and rules size. The rules produced from the different classification techniques identifies and characterizes all the correlations between different features in the phishing data and factors in the archive data of e-banking phishing site. The experiments were conducted on classification algorithm like C4.5, PART, PRISM and RIPPER , using an open java source *WEKA,* and comparative analysis is done between two AC algorithms CBA and MCAR. The authors have used APWG and Phishtank web access archive (Phishtank.com) and successfully extracted 27 phishing features and then clustered them into 6 groups.

The results are compared taking into consideration the number of rules formed and predictive accuracy. Further, the number of test cases used for evaluation is 1006 and prediction test method is ten-fold cross validation. Among the classification algorithms considered, the MCAR has shown the lowest error rate of 12.62%. PRISM has produced the highest number of rules i.e., 155 among the classification approaches, as it does not perform pruning. MCAR algorithm has performed well and predicted more accurately test data than the rest of the

classification approaches. The experiments have found interesting relationships and significant impact of *Domain Identity and URL, and Encryption and Security* and non-significant implication of *Content and page style* features with the *Social factor in Humans* to identify phishing in e-banking website.

2.12.9 Detecting Phishing Websites Using Associative Classification

The automated mining techniques are used to investigate the difficult problem of phishing in websites using associative classification algorithms by (Al Ajlouni et al., 2013). The data is collected from 1010 phishing and legitimate banking websites. In total 27 features are selected to test and train classifiers. Short programming scripts are used to extract the features and stored in a excel sheet. Depending on the difference in the strategies for learning, four classification and associative classification approaches used in the experiments are SVM (Vapnic, 1995), NB (Thabtah et al., 2009), MCAR (Thabtah et al., 2005) and CBA (Liu et al., 1998). MCAR is implemented using the java and other three approaches are implemented using the WEKA (www.weka.com). The parameters for experiments are set at 2% minsupp and 40% minconf to calculate the classification accuracy of all four approaches.

The experiments results have shown that MCAR has achieved 6.1%, 5.4% and 6.8% better accuracy than CBA, NB and SVM respectively. The other associative classification algorithm of CBA has also outperformed NB and SVM. It is also demonstrated that associative classification approaches are better than probabilistic and statistical approaches and are much feasible to be used for the large data pools to detect phishing in websites.

2.12.10 Phishing Detection Taxonomy for Mobile Device

The recent trends for the detection of phishing for the mobile devices are studied by (Foozy et al., 2013). The existing applied approaches are analysed and compared and recommendation were made to make necessary improvements in the phishing techniques in mobile devices.About 5% of the users click on a phishing link on an Andriod device every year and detection of phishing is handled differently because of the nature of the attacks on mobile and it is characterized as Voice over IP Phishing or known as vishing, Bluetooth phishing and Short

Message Service (SMS) phishing (Dunham, 2009). The common detection techniques discussed are Content Based Filtering, blacklist and whitelist (Foozy et al., 2013).

The data are collected from various sources like, SpringerLink, ScienceDirect, Yahoo, Google Scholar, Google, IEEE Xplore and ACM Digital Library. The analysis is carried out in two steps, first step is to study the phishing attack on a device category and second step is the detection and classification of phishing attack. The taxonomy of the phishing attack on mobile devices is analyzed and categorized as Bluetooth, SMS, Vishing and Mobile Web application. The comparison between the six detection techniques like Content Based (Peizhou et al., 2008 and Yoon et al., 2010), Blacklist (Singh, 2011) and (Mahmoud and Mahfouz, 2012), Whitelist (Mahmoud and Mahfouz, 2012), Hotspot and Gaussian Mixture Model (Chang and Lee, 2010) are studied in the paper.

The whitelist detection technique is used to detect phishing attacks on SMS and Mobile Web, the technique works by collecting the data from known users and only detects phishing attack from the already known senders. Wireless defense hotspot tools, can spot any change in MAC address default gateway and access point, ESSID and signal fluctuations on network and are used to prevent Phishing attack in Bluetooth. Gaussian mixture model is used to detect vishing attacks and helps in identifying the true and false statements. The study is helpful in understanding the other solutions that can be used in the detection of phishing attacks on mobile devices, one of the modern era technologies.

2.12.11 Anti-Phishing Prevention Technique (APPT)

A novel technique for anti-phishing prevention is introduced by (A. A. Khan, 2013) that uses one time password and encrypted token to access the website. APPT works in two steps, in the first step the user enter his/her information in the retrieval site to get a random onetime password. is generated by the machine when the user go to the one time password retrieval site .This one time password is sent via SMS or email, then an encrypted token is generated which contains the user specific data. In the second step an encrypted token is generated by PKIEncrypt function, it consists of user's machine IP address and its validity is 15 minutes. The primary website is authenticated by using both the one time password and the token.

The attacker will not be able to cause any damage while trying to get the one time password through forged website, because the site is only used to get the password via email or

SMS which is accessed by legitimate user. The prevention against the cookie attack is achieved by transmitting the token on encrypted channel and will expire in 15 minutes. CAPTCHA is the program that is used to protect the user from flooding of emails and SMS messages when in case the phishing attacker is successful in getting the user information via forged website.

It is successfully demonstrated that by using the APPT technique prevention of Phishing attacks can be reduced. But at the same time it is important to get user aware of the kind of danger they are in and use of antivirus/malware/spyware are recommended in the study.

2.12.12 Automated Detection of Phishing using Classification Scheme and Feature Selection

A hybrid approach is proposed by (Hamid et al., 2013) that combines the behaviour and content based approaches for selection the features in websites and emails. The behavioural features are identified because of the fact that these cannot be disguised by attackers. The message-ID tag in the header of the email is studied and if it indicates that the email has come from more than one domain, it is considered as phishing email.

The study was conducted using hybrid feature selection from 6923 datasets from Nazario (www.monkey.org) and SpamAssasin. The results have shown that the approach of hybrid feature selection achieved an accuracy of 94% and effectively identified and classified phishing emails.

2.12.13 Rouge DHCP-Enabled LAN Used in Phishing Attacks

A laboratory based research is conducted by (Purkait, 2013) to demonstrate the use of rouge Dynamic Host Control Protocol (DHCP) server by phishers. The study is conducted in Kharagur Laboratory and shown how a phisher uses rouge DHCP that bypasses all the security filters and toolbars and thus compromising the DHCP- enabled LAN.

Tests were carried out using two main web browsers like Internet Explorer and Mozilla, two security toolbars and Norton Antivrus. All counter measures taken are not able to detect the phishing attack and the toolbars used have given a clean go for the phishing sites tested.

2.13 Summary

In this chapter, a detailed discussion on classification and association rule mining approaches in data mining is conducted. In the first part of the survey some well-known approaches in classification like decision trees, statistical ad rule induction were discussed. The aim is to understand them well enough as they will be used in the comparative study with the proposed work in experimental chapter. In the later part of the chapter the review of some popular association mining algorithms is carried on with special emphasis on Apriori algorithm.

Promising AC techniques have been also discussed in this chapter which started with the first algorithm in AC, CBA and then followed by CPAR, CMAR, MMAC, MCAR, BCAR, ACCA, GARC, ARM, PST etc. Most of the algorithms in the review have shown two main phases in AC, one is the generation of rules and second is the classifier building in all the algorithms. All the work mainly focused on improving the processing time and accuracy of the generated classifiers and prediction. The methods have demonstrated with success that by integrating classification and association rule mining shown better results in terms of effectiveness and built more accurate classifiers when compared with traditional approaches in classification C4.5 and decision tree. The table presented below is taken from the literature survey on AC by (Thabtah F., 2006). In Table 2.6 the summary of the all the steps (rules discovery, rules ranking, rules pruning, and test cases prediction) for nearly all AC algorithm discussed. After all the literature review on AC, the new approach in AC mining called Looking at the Class (LC) will be presented in the next Chapter. The results with the traditional approaches in classification and AC mining are also compared.

Phishing is a serious security issue for the internet users. In the third part of the chapter, a survey of common approaches related to phishing in the literature like CART (Breiman et al, 1984), SVM (Joachims, 1999), BART (Fette et al., 2007), RF (Breiman, 2001), Neural Networks (NN) (Marques, 2001), Pilfer approach (Fette et al., 2007), Boostexter(Toolan and Carthy, 2010), PECM (Al- Momani, 2011), Taxonomy for Mobile devices (Foozy et al., 2013), Hybrid approaches like (Ma et al., 2009 and Hamid et al., 2013) and APPT (A.A Khan., 2013) were discussed. The implementation of some of the approaches in classification and AC mining would be carried out in experimental chapter 5 on phishing datasets.

Table 2.6: Summary of AC algorithms

Name	**Data Layout**	**Rule Discovery**	**Ranking**	**Pruning**	**Prediction Method**	**Reference**
CBA	horizontal	Apriori candidate generation	Support, confidence, rules generated first	Pessimistic error, database coverage	Maximum likelihood	(Liu, et al., 1998)
CMAR	horizontal	FP-growth approach	Support, confidence, rules cardinality	Chi-square, database coverage, redundant rule	CMAR multiple label	(Li, et al., 2001)
CPAR	horizontal	Foil greedy	Support, confidence, rules cardinality	Laplace expected error estimate	CPAR multiple label	(Yin and Han; 2003)
ARC-BC	horizontal	Apriori candidate generation	Support, confidence, cardinality	Redundant rule	dominant factor multiple label	(Antonie and Zaïane, 2003)
MMAC	Vertical	Tid-list intersections and recursive learning	Support, confidence, cardinality, class distribution frequency	Database coverage	Maximum likelihood	(Thabtah et al., 2004)
MCAR	Vertical	Tid-list intersections	Support, confidence, cardinality, class distribution frequency	Database coverage	Maximum likelihood	(Thabtah et al., 2005)
2-PS	horizontal	Apriori candidate generation	Support, confidence, rules cardinality	Database coverage	dominant factor	(Qian et al., 2005)
ACN	horizontal	Apriori candidate generation + Negative Rules	Confidence, rules Correlations, Support, rules cardinality , Positive Rules	redundant rule , pearson's correlation coefficient	Maximum likelihood	(Gourab Kundu et al., 2008)
BCAR	horizontal	Boosting Association Rule	Support, confidence, cardinality	Boosting Weak Association Rule	normalized prediction score model	(Yongwook and Gary Lee, 2008)
ACCR	horizontal	Cluster-based association rule	Support, confidence, cardinality	Pessimistic error, database coverage	Maximum likelihood	(Qiang Niu et al. ,2009)
ACCF	Vertical	Charm	Support, confidence, rules generated first	Pessimistic error, database coverage	Maximum likelihood	(Li X. et al. , 2008)
CACA	Vertical	Class-based associative classification	Support, confidence, rules generated first	Compact set, redundant rule	Maximum likelihood	(Tang and Liao, 2007)

Chapter 3

Classification Based on Association Rule (CBA)

3.1 Introduction

This chapter is included to explain the steps of rule generation and classifier building in the CBA algorithm so as to make the comparative analysis between CBA and proposed LC algorithms easier. The proposed LC algorithm is explained in the next chapter 4. The CBA uses the Apriori candidate generation function to generate rules in each iteration and LC algorithm is also based on the Apriori candidate generation function with modifications. The effects of the modification on the processing time and the number of candidate ruleitems generated at each step will be discussed in chapter 4. The CBA is the first algorithm in Associative classification and pioneer work or the new approach of CBA to build classifiers and to predict classes by using the association rules is presented by (Liu et al., in 1998). It was shown that using association in rule mining to generate all the rules, whose right hand side are bonded with a class attribute, is much more effective and accurate than traditional classification system like C4.5. The algorithm comprises of two main steps, *rules generation* (CBA-RG) and *classifier building* (CBA-CB). Firstly it finds and generates all ruleitems like Apriori algorithm (Agarwal and Srikanth, 1994). The ruleitems that passes the *minsupp* value are called *frequent ruleitems*. In the *classifier building* part the frequent ruleitems are used in building the full set of CARs and these generate the final classifier. The two stages of the algorithm are described in depth in the following sections. It is important to understand the full concept of CBA in this study as the proposed algorithm is based on CBA's working and introduces major improvements in CBA and demonstrates it with a comparative study.

3.2 CBA-RG Basic Concepts

In the main step of CBA-RG, firstly all *ruleitems* are found that contains the support value greater than *minsupp*. The algorithm uses multiple iterations over the dataset to generate *frequent ruleitems*. Figures 3.1 shows the rule or candidate generation. In the first iteration support count of all the items is calculated and then checked whether they are frequent. In the second iteration

```
F 1 = {large 1-ruleitems};
CAR 1 = genRules (F 1 );
prCAR 1 = pruneRules (CAR 1 );
 for (k = 2; F k-1≠ Ø; k++)
{           C k = candidateGen (F k-1 );
            for each data case d∈ D
C d = ruleSubset (C k , d);
 for each candidate c∈ C d
{                   c.condsupCount++;
                    if d.class = c.class then
            c.rulesupCount++;
                    }
            F k = {c∈ C k | c.rulesupCount≥ minsup};
            CAR k = genRules(F k );
            prCAR k = pruneRules(CAR k );
}
CARs = ∪ k CAR k ;
prCARs=∪ k prCAR k
```

Figure 3.1 Frequent itemset generation step in CBA algorithm

these *frequent 1-ruleitems* are used as seed set for more possible *frequent 2-ruleitems.* When the disjoint *frequent 1-ruleitems* are combined together candidate 2-ruleitems are produced. The *candidate 2-ruleitems* that passes the *minsupp* threshold are called *frequent 2-ruleitems*. This process continues till no more frequent ruleitems are left.

3.3 CBA-CB Basic Steps

This section will explain how CBA algorithm builds a classifier. When all the itemsets that are frequent are formed by doing multiple iterations on the dataset, then the main aim is the selection of subset of frequent ruleitems with the perfect rule sequence and minimum errors. All the subsets are evaluated on the training dataset and the best subset is called the classifier. It is a heuristic algorithm and the classifier built performs more accurately than C4.5.

After the CARs are derived the algorithm ranks them according to the procedure described in Figure 3.3. When the CARs are sorted then a rule qualifies to be placed in the classifier if it can cover at least one instance in the training dataset. Then all the instances in the training dataset covered by that rule are deleted from the training dataset. This process will not stop until all the training instances covered by a set of rules or all the rules are used. If all the

```
1   sort(R)
2   For each rule r ∈ R in sequence do
3       temp = Ø;
4       for each case d ∈ D do
5           if d satisfies the conditions of r then
6               store d.id in temp and mark r if it correctly classifies d;
7       if r is marked then
8           insert r at the end of C;
9           delete all the cases with the ids in temp from D;
10      selecting a default class for the current C;
11      compute the total number of errors of C;
12      end
13  end
14  Find the first rule p in C with the lowest total number of errors and drop all the rules after p
in C;
15  Add the default class associated with p to end of C, and return C (our classifier).
```

Figure 3.2 Building a classifier in CBA algorithm

rules are used and still we are left with some part of the training data then the majority class between all instances will be considered as default class.

Consider two rules, $rule_a$ and $rule_b$, $rule_a$ precedes $rule_b$ ($rule_a \rangle rule_b$) if:

1. Confidence value of $rule_a$ is higher than $rule_b$.
2. Confidence value of $rule_a$ and $rule_b$ are same, but support value of $rule_a$ is higher than $rule_b$.
3. Confidence, support of $rule_a$ and $rule_b$ are same, but $rule_a$ is produced before $rule_b$.

Figure 3.3 Rule ranking steps in CBA

When a new data instance or a document is classified, the first ruleitem that correctly satisfies the data instance values or the body of the document classifies them. And if no rule can predict the class label of the document or data instance then it will take the default class (Zaiane and Antonie 2002; Liu et al. 1998).

Now we will explain by an example the complete process of frequent ruleitem generation and building a classifier by CBA. Consider the training data shown in Table 3.1, which represents three attributes Age (senior, youth, junior), income (middle, low, high) and Has_Car (n,y) and a class (yes, no). User defined input values are set as *minimum support* = 15% and *minimum confidence* = 50%, the frequent ruleitems found are presented in Table 3.2, with their confidence and support values.

Next step is the formation of a classifier as described in Figure 3.1. Firstly the process of rule ranking is carried out as seen can be seen in Table 3.3, and then these ranked rules are applied on the training dataset to get the classifier. The process of rule pruning helps to group or find the set of rules that predicts the highest number or the full set of training data.

Table 3.1: Training data

Has_Car	Income	Age	Buy_car
n	middle	senior	Yes
y	low	youth	no
y	high	junior	yes
y	middle	youth	yes
n	high	senior	yes
n	high	junior	no
n	low	senior	no

Table 3.2: Rule discovery in CBA

Pass Count	Candidate/Frequent Itemsets	Rule Itemsets	Support	Confidence
1stPass	Frequent-1	<({Age,senior)},3), ((Buy_car, yes), 4)>	2/7	2/4
		<({(income, middle)}, 2), ((Buy_car, yes), 4)>	2/7	2/4
		<({(income,low)}, 2), ((Buy_car, no), 3)>	2/7	2/3
		<({(income,high)}, 3), ((Buy_car, yes), 4)>	2/7	2/4
		<({(Has_car,n)}, 4), ((Buy_car, yes), 4)>	2/7	3/5
		<({(Has_car,n)}, 4), ((Buy_car, no), 3)>	2/7	2/4
		<({(Has_car,yes)}, 3), ((Buy_car, yes), 4)>	2/7	2/3
2ndPass	Candidate-2	<{(Age,senior), (income, middle)}, (Buy_car,yes)>	1/7	2/3
		<{(Age,senior), (income, middle)}, (Buy_car,no)>	0/7	0
		<{(Age,senior), (income, low)}, (Buy_car,yes)>	0	0
		<{(Age,senior), (income, low)}, (Buy_car,no)>	1/7	1/1
		<{(Age,senior), (income, high)}, (Buy_car,yes)>	1/7	1/1
		<{(Age,senior), (income, high)}, (Buy_car,no)>	0	0
		<{(Age,senior), (Has_car,n)}, (Buy_car,yes)>	2/7	2/3
		<{(Age,senior), (Has_car,n)}, (Buy_car,no)>	1/7	1/3
		<{(income,middle), (Has_car,n)}, (Buy_car,yes)>	1/7	1/1
		<{(income,middle), (Has_car,n)}, (Buy_car,no)>	0	0
		<{(income,middle), (Has_car,y)}, (Buy_car,yes)>	1/7	1/1
		<{(income,middle), (Has_car,y)}, (Buy_car,no)>	0	0
		<{(income,low), (Has_car,n)}, (Buy_car,yes)>	0	0
		<{(income,low), (Has_car,y)}, (Buy_car,yes)>	0	0
		<{(income,low), (Has_car,n)}, (Buy_car,no)>	1/7	1/1
		<{(income,low), (Has_car,y)}, (Buy_car,no)>	1/7	1/1
		<{(income,high), (Has_car,y)}, (Buy_car,yes)>	1/7	1/1
		<{(income,high), (Has_car,n)}, (Buy_car,yes)>	1/7	1/2
		<{(income,high), (Has_car,y)}, (Buy_car,no)>	0	0
		<{(income,high), (Has_car,n)}, (Buy_car,no)>	1/7	1/2
	Frequent-2	<{(Age,senior), (Has_car,n)}, (Buy_car,yes)>	2/7	2/3
	Candidate-3	<{(Age,senior),(income,high),(Has_car,n)}, (Buy_car,yes)>	1/7	1/1
	Frequent -3	None		
CAR1		{(Age,senior)} → (Buy_car, yes) {(income, middle)} → (Buy_car, yes) {(income,low)} → ((Buy_car, no) {(income,high)} → ((Buy_car, yes) {(Has_car,n)} → ((Buy_car, yes) {(Has_car,n)} → ((Buy_car, no) {(Has_car,yes)}→ ((Buy_car, yes),		
CAR2		{(Age,senior), (Has_car,n)} → (Buy_car,yes)		
CARs		CAR1 U CAR2		

Table 3.3 : Ranked rules

Row-id	Rules	Conf	Sup
1	{(Age,senior), (Has_car,n)} → (Buy_car,yes)	2/3	2/7
2	{(income,low)} → ((Buy_car, no)	2/3	2/7
3	{(Has_car,yes)}→ ((Buy_car, yes),	2/3	2/7
4	{(Has_car,n)} → ((Buy_car, yes)	3/5	2/7
5	{(income, middle)} → (Buy_car, yes)	2/4	2/7
6	{(income,high)} → ((Buy_car, yes)	2/4	2/7
7	{(Has_car,n)} → ((Buy_car, no)	2/4	2/7
8	{(Age,senior)} → (Buy_car, yes)	2/4	2/7

The final step of the algorithm is to pick a test instance, apply the first top rank rule and predict the class of the test instance. CBA prediction step matches the test instance body with the rule and if the body matches it will assign the class of that rule to the test instance. We will explain this with an example of the test data in Table 3.4, first test instance with the values *{(Age,senior), (Income,middle)) and (Has_car,n)* is picked from the Table 3.1, to classify this test instance CBA algorithm picks the top ranked rule *{(Age,senior), (Has_car,n)* from Table 3.4. The body of test instance matches the body of the 1st rule selected from the Table 3.3. The class value of the rule that is *'yes'* is assigned to the test instance. Then CBA picks the second test instance *{(Age,youth), (Income,low)) and (Has_car,y)}* is picked from the Table 3.4 and this test instance

Table 3.4: Data for testing

Transaction -id	Age	Has_Car	Income
1	senior	n	middle
2	youth	y	low

tends to be classified by the ranked rules from Table 3.3. Again the top rank rule *{(Age,senior), (Has_car,n)* is selected from the Table 3.3 but this time the rule does not match the body of the test instance, so the second rule *{(income,low)} → ((Buy_car, no)}* from Table 3.3 is selected. The algorithm checks whether this rule matches the test instance value and the answer is yes. The class label of this rule *((Buy_car, no)* will be assigned to the test instance. The above procedure to classify the test data will continue until all the test instances are classified.

3.4 Summary of the Chapter

The chapter explains the working of the first algorithm CBA in the associative classification domain. The detailed working of CBA's rule generation and the classifier building steps with an example is described in section 3.2 and 3.3. In the next chapter the proposed LC algorithm is presented with its implementation on bench mark datasets and the complete analysis of the results are discussed in comparison to CBA and other AC algorithms.

Chapter 4

Looking at the Class Associative Classification Algorithm (LC)

4.1 Introduction

As explained in Chapter 2, two data mining tasks, classification and association rule mining are integrated to form AC algorithms. The AC algorithms build accurate classifiers than classification approaches in data mining. The classifiers produced by AC algorithms contain simple yet effective rules that are understandable and are easily interpreted and used by the domain experts. AC tends to discover rules that cannot be found by other classification algorithms (Antonie and Zaïane, 2002).

Some of the AC approaches use an exhaustive search technique to find the set of rules by scanning the database again and again. This time consuming searching method is proposed in Apriori algorithm (Agrawal and Srikant, 1994) to generate rules and the main drawback is the high execution CPU time (Liu, et al., 2001; Thabtah, et al., 2010). In terms of rule ranking, most of the AC algorithms rank rules on the basis of confidence values, support values, the time when rule is generated and the cardinality of the rule. Another tie breaking criteria is added which is adopted from (Thabtah, et al., 2004) that takes into account the class distribution frequencies among rules in the training data set. This can be utilized beside confidence, support and rule length to discriminate among rules.

Chapter "4" main focus is to investigate and improve the efficiency of the rule generation phase, to reduce the number of merging at each iteration, to decrease the size of frequent ruleitems generated, to cut down the size of produced rules in classifier, and investigate and implement the methods to improve and maintain the prediction accuracy when compared with other classification and AC algorithms. This is all accomplished in a new AC algorithm that deals with the problem of large number of itemsets joining in the training phase within the known CBA algorithm. It tends to find an effective rule generating method that can derive the rules efficiently without negatively affecting the classification rate. As a consequence, a new AC algorithm, called Looking at the Class (LC) AC algorithm is proposed. LC also adds new criteria

during the process of predicting new test data instance and proposes a new effective prediction procedure that improves accuracy.

The LC algorithm while dealing with both types of attributes (categorical and continuous) is an improved version of the CBA algorithm and introduces a novel method of itemset merging in the training phase of CBA, a new rule ranking procedure that includes the cardinality and class distribution criterion to rank rules and a proposes a new prediction method that is based on selecting and assigning the class with highest average of confidence in the group of rules that correctly matches the conditions of the test instance. The results obtained from the implementation of the above proposed changes are demonstrated in the experimental section of this chapter. The comparative analysis of the results with the current well-known AC and classification algorithms are also discussed.

The summary of main differences between LC and CBA are discussed in section 4.2, the development of a new proposed algorithm LC is discussed in Section 4.3 with the rule generation, rule ranking, rule pruning and discusses the new prediction algorithm. Experimental parameters, datasets and findings are demonstrated in Section 4.4. In the final Section 4.5, the summary of the chapter is given.

4.2 Main Differences of LC and CBA Algorithms

CBA algorithm is discussed in the previous chapters; in this section the main differences in CBA and LC algorithm are highlighted, so to get an idea of the modifications made in the CBA algorithm. The main differences in both the approaches to build a classifier are pointed out in the Table 4.1 below which analyses the data layout, rule discovery, rule ranking, pruning and prediction phases of both the approaches.

Table 4.1: Main differences between LC and CBA algorithms

Name	**LC**	**CBA**
Data Layout	Horizontal	Horizontal
Rule Discovery	Apriori Candidate Generation function with looking at the class labels. Use Minority rule, in selecting class with the itemset. Uses highest frequent class in the dataset with the ruleitem, if a ruleitem contains more than single class attribute.	Generation function without looking at the class labels. Don't uses Minority Rule Keeps all rulesitems with different class values.
Ranking	Confidence, Support, Rule length and Class Distribution	Support, confidence and selects rule which generated first
Pruning	Database Coverage	Database Coverage, Pessimistic error Estimation
Prediction Method	Based on Confidence value of Group of Rules	Maximum likelihood

4.3 The Development of New AC Algorithm

An experimental work is conducted and a new algorithm and its experimental evaluations are presented to demonstrate the improvements in the procedure of rule generation in CBA algorithm. The programming language used for the implementation of the algorithm is Visual C # and Visual C++. It was noted in the training or the rule generation phase of CBA and CBA (2) (Liu et al., 1998, Liu et al., 1999), that the joining of frequent itemsets of size '*k*' to generate candidate itemsets of size '*k+1',* is carried out without the consideration of their class labels. The above process of merging is computationally expensive and needs excessive CPU time. At the same time it produces a large number of candidate itemsets in each iteration due to unnecessary itemsets joining. Our main focus and aim is to minimize the CPU time during frequent itemsets generation process of CBA. In the training phase of CBA we will consider the class labels of the frequent itemsets and only merge the itemsets that have same class labels. So, if 'item_A' and 'item_B' are two disjoint itemsets that are found at iteration "0", our proposed training approach

in LC considers merging itemset ‘item_A’,‘item_B’ only if ‘item_A’ and ‘item_B’ have the same class labels. The reduction in the number of itemsets merging in each iteration may improve the searching process and consequently minimizes the CPU time for dense and large datasets significantly.

4.3.1 The Proposed Rule Discovery Algorithm

In the rule discovery phase of the CBA algorithm it uses the Apriori candidate itemsets generation function in which the frequent itemsets are discovered by level wise search. The CBA starts by counting the support values of items of size 1(1-itemset) in the first level, and checks whether they are frequent or not. In each of the subsequent level, process of merging the itemsets that are found frequent in the last level will continue regardless of their class labels, to extract candidate itemsets at all levels. Our basic idea is to enhance the merging process of disjoint frequent itemsets in CBA training phase by considering the class labels of the itemsets in each level. If the selected disjoint itemsets are linked with the common class in the last iteration then join them, else do not join them. A minority class rule is applied in case when the itemset has more than one class, the class with the highest support count in the training data set is selected and the remaining class labels are ignored. If the itemset’s class labels have the same frequency of support count then the class of the itemset is selected randomly.

Data used by LC algorithm contains a header that reads relation name, attribute names, and total number of data instances in the table. The values of all the attributes are comma separated and the last column of the data file must present the class attribute. The missing values are considered as other existing values (Witten and Frank, 2000). For categorical attribute, all the possible values are mapped to a set of integers. For the attributes that are continuous, multi-interval technique for discretization (Fayyad and Irani, 1993) has been used.

The process of discretization starts by selecting a continuous attribute from the training data and sorted in ascending order with the class values of each instance. A break point is placed where the class value changes and information gain (Quinlan, 1979) measure is calculated for all break points. The information gain depicts the quantity of information required to specify the class values. The break point is selected which minimizes the information gain over all other break points. The process starts again on the lower range of that attribute.

The frequent itemsets generation step known as the training phase is explained of the LC algorithm as shown in Figure 4.1:

Line 1: The algorithm begins by reading the training data table from the selected location and takes user defined *minsupp* and *minsconf* values.

Line 2: The candidate 1-ruleitems are generated with the support values.

Line 3-4: The class with the maximum frequency in the data table is determined first and then the minority rule is applied, if the ruleitems contains more than single class values in candidate 1-ruleitems, the ruleitems with the highest frequency class are kept and all other are removed from the list of candidate 1-ruleitems. If the class values are same then the ruleitem is assigned the highest frequency class that is extracted in line 3.

Line 5: The frequent 1-rueitems are generated from the candidate 1-ruleitems that passes the *minsupp* threshold value defined by user.

Line 6: The confidence values of all the frequent 1-rueitems are determined and ruleitems that

Input: Training Data (D), *minsupp* and *minconf* values

Output: Frequent set of rules

1. *read the table from D;*
2. *Candidate_Ruleitem_1* = gen*Candidate_Ruleitem_1(D)*;
3. *Maximum_Class_Frequency = Class_values_Max();//* determines class with highest frequency in data table
4. *Candidate_Ruleitem_1 = minority_Rule(Candidate_Ruleitem_1); //* removal of ruleitems with lowest class frequencies
5. *Frequent_Ruleitem_1* = gen*Frequent Ruleitem_1* (*Candidate_Ruleitem_1*, *minsupp*);
6. *ClassAssociationRules_1 = genRules(Frequent_Ruleitem_1, minconf)*
7. **for** *(k* = 2; $F_{k-1} \neq \emptyset$; k++)
8. {
9. *Candidate_Ruleitems_k* = gen*Candidate_Ruleitems_k* (*Frequent_Ruleitems_(k-1)*); // candidate ruleitems are generated with their number of occurances in data table (support counts)
10. *Candidate_Ruleitem_1 = minority_Rule(Candidate_Ruleitem_1); //* removal of ruleitems with lowest class frequencies
11. *Frequent_Ruleitem_1* = (*Candidate_Ruleitems_k,, minsu*p);
12. *ClassAssociationRules_k = genRules(Frequent_Ruleitem_1, minconf);*
13. **}**
14. *ClassAssociationRule's* = $\cup_k$ *ClassAssociationRules_k* ;
15. *Compute total number of merging;*
16. *Calculate total time needed to generate Ruleitems;*

Figure 4.1 Frequent itemset generation step in LC algorithm

passes the *minconf* value are termed as Class Association Rules (CARs).

Line 7-13: This process of candidate ruleitems generation in each iteration or cycle is repeated until all frequent ruleitems are produced.

Line 14-16: All the CARs are combined together in one file and these rules are then ranked considering their support, confidence and length of rule metrics. The number of merging at each iteration, total number of merging, and the time taken to execute the algorithm is calculated.

The Figure 4.2 represents the candidate ruleitems generation at each step including the support count values of the ruleitems.

Line 1: The function takes as an input the frequent ruleitems from the previous iteration and data table.

Line 3-5: The itemsets are picked from the frequent ruleitems and their class labels are matched, if the class labels are identical then the itemsets are combined together otherwise the process of class label matching continues for all the itemsets in the frequent ruleitems and candidate itemsets are produced.

Line 6-13: The class labels of the joined itemsets are determined and the support counts of the candidate ruleitems are calculated. A complete set of candidate ruleitems are returned with their support counts.

Input: *Frequent_Ruleitems_k*, data Table 'D'

Output: Candidate set of rules at each *kth* iteration, support counts of each ruleset contained in the Candidate set

1. *for(i=0; i < frequent_Ruleitem_(k-1).count; i++)*
2. *{*
3. *if (c.rulesubset_k-1 == c.Rule_item)*
4. *{*
5. *Candidate_Ruleitems_k = rulesubset_k-1* $\cup$ *Rule_item* // candidate ruleitems are generated
6. ***for** each data instance d* $\in$ *D*
7. C_d *= ruleSubset (Candidate_Ruleitems_k, d);*
8. ***for** each candidate c* $\in$ C_d
9. ***{***
10. *condsupCount++; //Calculates the support count of each rule subset*
11. ***if** d.class = c.class*
12. ***then***
13. *c.rulesupCount++;*
14. *}*
15. *else*
16. *break;*
17. *}}}}*
18. *Return Candidate_Ruleitems_k;*

Figure 4.2 Generate candidate ruleitems function and support count calculation in LC algorithm.

4.3.1.1 Example on the New Training Algorithm

To explain our proposed training algorithm, consider for example Table 4.2 (Thabtah, et al., 2005) which contains three attributes (age, income, has_car) and a class label (buy_car) and represents whether an individual will buy a new car. Assume that *minSupp* (support count) is set to 2. In the first iteration, LC generates candidate 1-ruleitems of size "14" as shown in Table 4.2,

Table 4.2: Training data

Age	Has_Car	Income	Buy_car
senior	n	middle	yes
youth	y	low	no
junior	y	high	yes
youth	y	middle	yes
senior	n	high	yes
junior	n	high	no
senior	n	low	no

Table 4.3: LC candidate 1-ruleitems

ITEMSET	CLASS	support
senior	yes	2/7
senior	no	1/7
youth	yes	1/7
youth	no	1/7
junior	yes	1/7
junior	no	1/7
middle	yes	2/7
low	no	2/7
high	yes	2/7
high	no	1/7
y	yes	2/7
y	no	1/7
n	yes	2/7
n	no	2/7

Table 4.4: LC candidate 1-ruleitems when minority rule is applied

ITEMSET	CLASS	support
senior	yes	2/7
youth	yes	1/7
junior	yes	1/7
middle	yes	2/7
low	no	2/7
high	yes	2/7
y	yes	2/7
n	yes	2/7

it is important here to notice the number of ruleitems generated. The minority rule (Figure 4.1, line 4), is applied on the candidate 1-ruleitems. In case when a rule item is found that is linked with more than single class then the minority rule will select the class with the highest frequency with that ruleitem. If the classes' frequencies are same for that ruleitem then the class with the highest support value in the data table is selected with that ruleitem, and if the classes frequencies are same in the data set then a class is selected randomly. Table 4.4 shows the candidate 1-ruleitems after the minority rule is applied and we can notice the number of ruleitems are reduced to "8". For example in the case of <Age=Senior, yes> and <Age=Senior, no> ,the frequency of class "yes" is "2" and "1" respectively, the minority rule has selected <Age=Senior, yes> as seen in Table 4.4. We take another example to explain in more detail, both ruleitems of <has_car= n, yes> and <has_car= n, no> has same frequencies of "2" in Table 4.3 Our approach has selected the ruleitem <has_car= n, yes> (see Table 4.3) because the support value of class "yes" is "4" that is higher than the support value of class value "no" i.e., "3' as in Table 4.2. After the first iteration, frequent 1- ruleitems can be seen in Table 4.6. In the second iteration, disjoint frequent ruleitems are merged based on their classes to generate candidate 2-ruleitems; so in this case, <Age=senior, yes> and <Income=middle, yes> is merged because they have the same class, i.e., "yes", "senior" and "high" are merged in the same way, as in Table 4.4.

Our method did not consider joining "senior" with "low" and "middle" with "low" since

Table 4.5: LC candidate-2 ruleitems

ITEMSET	ITEMSET	CLASS
senior	middle	**yes**
senior	high	**yes**
senior	n	**yes**
middle	n	**yes**
middle	y	**yes**
high	y	**yes**
high	n	**yes**

Table 4.6: LC frequent 1-ruleitems

ITEMSET	CLASS	support
senior	yes	2/7
middle	yes	2/7
low	no	2/7
high	yes	2/7
y	yes	2/7
n	yes	2/7

they have uncommon class. On the other hand, CBA and other AC algorithms such as CBA (2) consider joining itemsets without checking their class labels. Table 4.7 illustrates the itemsets produced by CBA from Table 4.2 using a *minSupp* of 2 in the first iteration, and Table 4.8

displays the possible 2-candidate itemsets obtained after merging frequent 1-itemsets. Now we will examine the total number of merging in the second iteration of both CBA and our approach

In Table 4.5, the number of frequent itemsets from Table 4.2 column "Age" are 3 (senior, junior and youth), 3 frequent itemsets (middle, low and high) from "income" column and 2 frequent itemsets from column "has_car". The total number of possible candidate-2 itemsets can be calculated by a simple formula $(2^{n1} + 2^{n2} + 2^{n3}) + 1 = (2^3 + 2^3 + 2^2) + 1 = (8+8+4) + 1 = 21$. As in comparison the total number of candidate-2 itemsets produced by LC in 2^{nd} iteration is "7". It is obvious from Table 4.7 that the number of merges performed by CBA is larger than LC.

The first iteration of CBA has produced "7" frequent-1 ruleitems (See chapter 3) as compared to LC "6" (see Table 4.9). The second iteration has produced the same number of Frequent-2 ruleitems for both CBA and LC that is 1, and there are no Frequent-3 ruleitems generated in third iteration for both algorithms. So it is very clear that the total numbers of frequent ruleitems generated in both the approaches are not similar for both algorithms, CBA's

Table 4.7: CBA candidate-2 itemsets

ITEMSET	ITEMSET		ITEMSET	ITEMSET
senior	middle		junior	middle
senior	high		junior	high
senior	low		junior	low
senior	y		junior	y
senior	n		junior	n
youth	middle		middle	y
youth	high		middle	n
youth	low		low	y
youth	y		low	n
youth	n		high	y
			high	n

Table 4.8: Frequent 1-itemsets produced by CBA

ITEMSET	support
senior	3/7
junior	2/7
youth	2/7
middle	2/7
low	2/7
high	3/7
y	2/7
n	2/7

Table 4.9: Frequent rule itemsets at each iteration with support and confidence counts

Pass Count	Frequent Itemsets	Rule Itemsets	Support	Confidence
1st Pass	Frequent-1	<({Age,senior)},3), ((Buy_car, yes), 4)>	2/7	2/4
		<({(income, middle)}, 2), ((Buy_car, yes), 4)>	2/7	2/4
		<({(income,low)}, 2), ((Buy_car, no), 3)>	2/7	2/3
		<({(income,high)}, 3), ((Buy_car, yes), 4)>	2/7	2/4
		<({(Has_car,n)}, 4), ((Buy_car, yes), 4)>	2/7	3/5
		<({(Has_car,n)}, 4), ((Buy_car, no), 3)>	2/7	2/4
		<({(Has_car,y)}, 3), ((Buy_car, yes), 4)>	2/7	2/3
2nd Pass	Frequent-2	<{(Age,senior), (Has_car,n)}, (Buy_car,yes)>	2/7	2/3
3rd Pass	Candidate-3	<{(Age,senior),(income,high),(Has_car,n)}, (Buy_car,yes)>	1/7	1/1
		<{(Age,senior),(income,middle),(Has_car,n)}, (Buy_car,yes)>	1/7	1/1
3rd Pass	Frequent -3	None		
CAR1		{(Age,senior)} → (Buy_car, yes) {(income, middle)} → (Buy_car, yes) {(income,low)} → ((Buy_car, no) {(income,high)} → ((Buy_car, yes) {(Has_car,n)} → ((Buy_car, yes) {(Has_car,yes)}→ ((Buy_car, yes)		
CAR2		{(Age,senior), (Has_car,n)} → (Buy_car,yes)		
CARs		CAR1 U CAR2		

"8" and LC's "7", and also the number of candidate ruleitems merge of CBA during the training phase in most iterations is significantly high as compared to LC algorithm. So if the data set is large or highly correlated it would be obvious that there will be an exhaustive search by any AC algorithm during the discovery of the frequent ruleitems and the number of candidate ruleitems join is expected to be massive. Later in the experimental section of this chapter we compare the

results of CBA and LC algorithms in regards to different measures especially training time and memory resources using medium and large datasets that have various size and number of columns. We can assume from here that as the number of iterations increases to generate frequent ruleitems the likelihood is that our approach will significantly reduce the unnecessary ruleitems merging and this will improve the efficiency in terms of execution time and memory usage of the algorithm.

4.3.2 Classifier Construction

The second stage of LC algorithm is the construction of a classifier, the main aim of the process is to extract a number of rules that can be used for the prediction of test data. Rule ranking and pruning of the rules generated in the training phase of LC algorithm, is performed in this step.

4.3.3 Rule Sorting

The number of rules produced by the AC techniques may be large (Baralis, et al., 2004), and extraction of reduced number of rule set is very significant. The rule ranking is the first step to build a classifier in AC and it is mainly utilized to choose the most useful rules for prediction. LC ranks rules on the basis of different criteria mainly rule's confidence, support, length and class distribution in the training data set.

Rule ranking is very helpful in pruning specific rules which have low confidence than general rules and are also redundant. As in rule evaluation step of ADT algorithm (Wang, et al., 2000), when the choice has to be made between two rules, the highest ranked rule is selected.

Mostly rule ranking in AC is carried out on parameters such as confidence, support and rule length (number of items in the rule body (Left Hand Side)). This ordering of parameters was used in a number of AC approaches such as those in (Thabtah, et al., 2010, Antonie, et al. 2003; Liu, et al., 2001; Wang, et al., 2000). In the above techniques if the support, confidence and the cardinality values are same for some rules then these techniques select one of the tied rules randomly which in many cased may degrade the accuracy performance (Thabtah et al., 2011).

In this section, we will present the rule ranking procedure used in our algorithm. We have selected the rule ranking criteria used by MCAR to be used in our study. The LC algorithm uses the general rule ranking procedure that is based on rule's confidence, support and length while

adding another tie breaking parameter that considers the class frequencies in the training data and when required the rule linked with the highest distributed class is preferred. For example if two rules, $rule_1$ and $rule_2$ having the same confidence, support and rule length, but $rule_2$ is linked to the class that have more frequency in the training data than $rule_1$, LC selects $rule_2$ in the process of rule ranking.

In case when we come across a rule that shows same class frequencies, then we will select rule on random basis. The procedure used in LC is presented in Figure 4.3.

input : Class Association Rules(CARs)

output : Ranked rules based on their confidence, support and rule length counts

```
1.  Ranked_Rules = ClassAssociationRules;
2.  for (i=1 ; i= ClassAssociationRules.Count ; i++)
3.  {
4.   if (confidence_value(i-1) >  confidence_value(i) )  //comparing confidence value
5.  { Swap= Ranked_Rules (i-1) & Ranked_Rules (i);}
6.  if (confidence_value(i-1) ==  confidence_value(i) )
7.  {if (support_value (i-1) > support_value(i)) // checking support count
8.  {Swap= Ranked_Rules (i-1) & Ranked_Rules (i); }
9.  elseif (support_value (i-1)== support_value(i))
10. {If(Ranked_Rules (i-1).rowlenght > Ranked_Rules (i).rowlenght)
11. {Swap = Ranked_Rules (i-1) & Ranked_Rules (i);}
12. elseIf(Ranked_Rules (i-1).rowlenght== Ranked_Rules (i).rowlenght)
13. {If(Ranked_Rules(i-1).class_distribution > Ranked_Rules(i).class_distribution)//checking class frequency
14. {Swap = Ranked_Rules (i-1) & Ranked_Rules (i);}
15. elseif (If(Ranked_Rules (i-1).class_distribution == Ranked_Rules(i).class_distribution)
16. {random select a rule }
17. }}}
18. return Ranked_Rules;
```

Figure 4.3 Rule ranking in LC algorithm

4.3.4 Pruning of Rules

A significant rule is a rule that is contained in the classifier and will be used in the step of prediction of accuracy of the test data. In our rule generation and then selection system a rule will be a part of the classifier if it correctly covers at least one training instance during building the classifier. After the rule generation step finished, we usually ends up having very huge number of rules some of which are redundant or contradictory. To minimize the size of the rules

in this step pruning unnecessary rules becomes important. In the LC algorithm, we use database coverage like pruning and its functionality is discussed in Figure 5.4, which can be summarized as follows.

Step 1 (line 1-13): rules are applied against the training data set from the set of ranked rules R which were produced after training phase and rule ranking. LC selects a rule r such that $r \in R$, and then it goes through the training data set T searching for the instances or records covered by r (meaning those satisfy the $r's$ body) (line 7). Rule r is marked if it covers an instance t (line 8). In our classifier building step r is a potential rule and will be added to our final classifier (line 10). The instances that are correctly classified by r will be deleted from the training data T (line 11). If some of the data is not covered by the ranked rules, then we will select a default class from this data. Default class will be the highest class in the remaining training data (line 12). The error made by the ranked rules that wrongly classified the classes of the training data T will be computed and stored along with the default class (line 13). The errors are calculated when there is no rule left in $'R'$ or when all the training data is covered by the selected rules. This will conclude the selection process of rules.

All rules that are marked during evaluation get generated as a classifier and all other unmarked rules are discarded by the LC.

Input: Ranked rule set, Training instance 'T'

Output: A Classifier

```
1   R = Ranked rule set
2   For each rule r ∈ R
3   do
4   temporary = Ø;
5   for each case t ∈ T
6   do
7   if t satisfies the conditions of r then
8   store t.id in temporary and mark r if it correctly classifies t;
9   if r is marked then
10  insert r at the end of Classifier;
11  delete all the cases with the ids in temporary from T;
12          select a default class for the current Classifier;
13          calculate total number of errors of Classifier;
14  end
15  end
16  Find the first rule p in C with the lowest total number of errors and drop all the rules after p in C;
17  Add the default class associated with p to end of C, and return C (our classifier).
```

Figure 4.4 Classifier building in LC algorithm (database coverage pruning method)

The remaining uncovered training instances are utilized to produce a default rule that represents the largest frequency class among them. LC and CBA require that all items in the candidate rule to be contained in the training case during evaluation in order to consider the rule significant.

4.3.5 Prediction Method of LC

Prediction is the final and most important step in classification. It helps in determining the accuracy of the classifiers produced and it uses the classifier's set of to make the decision of assigning the class label to the test instance. The basic structure of the new LC prediction method named as "Group Base Decision Prediction Algorithm" is presented in Figure 4.5. It works by selecting the rules from the classifier (C) that match the conditions of the test instance *($test_i$)*. Then divides these rules into different groups based on class labels and compute the average group confidence to select the class that belongs to the largest group average confidence. This class will be assigned to the test instance *($test_i$)*. The description of the steps in the algorithm is as under:

Inputs for the algorithm are set of rules (Classifier), test data and default class. The output is a prediction error rate generated after assigning the new class labels to test instances.

Line 1: Start of the loop for each test data instance.

Line 3: Another loop starts for each rule in the classifier.

Line 4-6: The test data instance *TestData(i)* conditions are matched with the all the rules in the classifier and all the rule that matches the test instance conditions is placed in an array "*ArrayRule"* rule conditions .

Line 9-12: *ArrayRule* count is checked and if it is not zero then it means that at least one rule is found that matched test data instance condition. The confidence of all rules having similar class is calculated and the group of rules belonging to a specific class with the highest average confidence is selected. If the *ArrayRule* is found empty then the default value is assigned to the test instance. And the value of "assign" variable is set to "true".

Line 15-16: If the a rule or set of rules are found that matches the test instance, the class with the highest average value of confidence is assigned to the test instance *TestData(i).*

Line 20: The total numbers of errors (Pe) are calculated of the instances that are wrongly classified.

Input: Classifier (CRules), default class (dc), test data (*TestData*), training data
Output: Prediction error rate *Pe*

```
1 For (i=0, i<= TestData.count; i++)
2 { Assign=false;
 3 For (j=0 ; j<= CRules.count ; j++)
 4   { verify(TestData(i).conditions , CRules(j).conditions)
 5    If (TestData(i).conditions == CRules(j).conditions)
 6       { ArrayRules.Add(CRules(j));
7          }
8          }
9          if ArrayRules.count ≠ ∅
10         { calculate average_confidence per class;
11         HAC = Select highest average confidence class;}
12         else { TestData(i) == dc;
13                    Assign== true;
14              }
15         If (assign==false)
16         { TestData(i)== class.HAC;//assign highest    //average_confidence  class
17         }
18       ArrayRules=null;
 19        }
 20     Pe= verify (TestData , TrainingData); // error calculation
```

Figure 4.5 Prediction method in LC

Normally, in AC prediction methods, to predict the classes of the test data, generally the techniques work on the principle that, when rule $r's$ in the classifier C matches the test instance (Ts) attribute value(s) or is contained in Ts classifies it. So the test case Ts will be assigned the first rule in the classifier that matches it. This is typically called single rule prediction and it has been criticized for allowing only very limited of rules to classify test data. Instead we have developed a multiple rule prediction approach which employs a collection of rules to predict the outcome of a test instance in LC. Therefore, more than one rule play role in deciding the class label that should be given to the test case and consequently we minimized the dominancy of a single rule predicting the class for most test cases.

4.4 Experimental Environment Setup

4.4.1 Data Sets Used in Experiments

Experiments are conducted using a number of different data sets from the UCI repository (Murphy and Merz, 1996). The data sets of balloon, Sick, Cleve, Vote, Weather, Breast-w, Diabetes are the binary class dataset and multiple class datasets used in the experiments are Glassd, Iris, Led7, Lymph, Zoo and Contact-lenses. For the authenticity of the experiments the small, medium and large sized datasets are considered. The descriptions of the datasets used in the experiments are shown in the table 4.10.

Table 4.10: UCI data sets used in experiments

Data Set Name	No of Classes	Data set Size
Balloon	2	20
Sick	2	3772
Cleve	2	303
Glassd	7	214
Iris	3	150
Led7	10	3200
Lymph	4	148
Vote	2	435
Zoo	7	101
Contact-lenses	3	24
Weather	2	14
Breast-w	2	699
Diabetes	2	768

4.4.2 Experimental Parameters and Setup

The algorithms of LC and CBA are designed using Visual C++ and C# and evaluations are performed on 1.6 GHz processor machine with 1GB memory. The comparison of LC algorithm with CBA with reference to the number of merges for both approaches at each iteration in the training phase or rule generation phase, the number of frequent rules generated, the number of rules contained in the classifiers of both algorithms, CPU time, memory usage and prediction accuracy. The pruning and prediction methods for both algorithms are also implemented. The *minSupp* in our experiments are set at 5% and *minConf* to 40%.

One of the main aims of the experiments is to calculate the total number of merging (joining) of the itemsets in each iteration in both our proposed algorithm LC and CBA. It will also be investigated whether the reduction in the number of joining during all the iterations has any impact on the execution time and memory usage (paged memory, physical memory and virtual memory). It should be noted that the learning of the rules phase (training phase) and the classifier construction step (building a classifier) as well as the prediction phase are thoroughly investigated in this section. In other words, the investigation is done experimentally and a comparative study is carried out with CBA.

4.5 Experimental Results and Discussion

In this section, thorough analysis of the experimental findings will be discussed. Table 4.11 represents a comparison of the rule generation phase for both LC and CBA on different binary and multi-class classification data sets from UCI repository. Specifically, and for all data sets considered, the number of times disjoint itemsets are merged in each iteration are computed and shown in columns 4-9, and the sum of all the itemsets joining (merging) is presented in the last column of the table. The third column of the Table 4.11 represents the merging of a single item with the class attribute values, and it can be seen that the values are same for both LC and CBA for all the data sets. While analysing the results it can be said that a significant change is seen in the number itemsets merging during each iteration in both algorithms. The number of itemsets merging is our proposed algorithm LC is reduced to merely 2%, 4.2%, 5.18%, 4.95%, 7.66% and 3.4% of CBA on the "Zoo", "Breast–w", "Vote", "Lymph", "Glassd" and "Cleaved" data sets respectively. For the remaining small data sets like "balloon", "contact" and "weather" and

"irisd" the number of merging has also reduced for the LC when contrasted with CBA training phase by 48.84%, 24%, 41.97% and 58.04% respectively. This means that LC training phase has improved upon CBA learning by cutting down unnecessary itemsets joining very significantly during the training phase for small, medium and large data sets.

Furthermore, it is notable from Table 4.11 that the differences in the number of joining between LC and CBA in the later iterations of large data sets like "Zoo" and "Lymph" is large. This is because the number of itemsets available before any merging in the later iterations such as iteration (2) is often larger that of the previous iteration, i.e. iteration (1), excluding the last iteration. For instance in the "Zoo" data set, the number of itemsets merged using CBA are 1086, 4374, 10220, 15187, 14875, and 6414 for iterations, 2,3,4,5,6 and 7 respectively. In comparison, the LC algorithm decreases the number of merging for the same iterations significantly to 38, 122, 195, 271, 223 and 0 respectively.

According to Table 4.11 and in general, our new approach has presented a new idea of avoiding so many unnecessary merging for all the data sets whether they are large or small. The reduction in the search space of rule generation process in LC may lead us to the reduction in the execution time because efficiency is always a significant issue when dealing with software related solutions, and it may lead us to decrease in the memory usage.

The processing time of CBA and LC are calculated and shown in Table 4.12. For example for "Zoo" data set, the time required to execute the experiment is significantly decreased from 39515ms in CBA to 6942ms in LC with high margin of 82.43% improvement for the LC algorithm. It is noticed in the data set of "Led7" that iterations from 3-6 have "zero" values in Table 4.11. This was due to fact that "Led7" is a multiclass problem and contains many different classes about '10'and as our algorithm considers only those itemsets for merging that have same class labels, so it explains the zero values in iteration of 3-6, and consequently shown a large saving in execution time of "Led7" data set.

Table 4.11: The number of itemsets joining of LC and CBA algorithms

Approach	Data set	Iteration 1	Itemsets Merging in each iteration						Total Number of merging in all iteration
			Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	
LC*	Balloon	14	14	12	4	0	0	0	44
CBA**		14	24	32	16	0	0	0	86
LC	Contact	23	23	28	0	0	0	0	74
CBA		23	30	44	0	0	0	0	97
LC	Iris-Id	27	18	12	3	0	0	0	60
CBA		27	45	52	19	0	0	0	143
LC	Vote	60	102	423	775	897	791	0	3048
CBA		60	399	2605	8832	17396	21159	8386	58837
LC	Zoo	196	38	122	195	271	223	0	1045
CBA		196	1086	4374	10220	15187	14875	6414	51266
LC	Led7	140	21	24	16	6	1	0	208
CBA		140	84	280	560	669	432	62	2227
LC	Glassd	90	39	49	26	8	1	0	213
CBA		90	152	525	885	759	325	42	2778
LC	Lymph	93	169	631	909	642	212	0	2656
CBA		93	430	2585	8460	16103	18549	7381	53601
LC	Sick	45	86	537	1173	1721	1722	0	5284
CBA		45	228	1103	3136	5650	6669	2829	19660
LC	Cleaved	50	108	380	697	935	796	3	2969
CBA		50	296	2008	8356	21794	35928	16959	85391
LC	Weather	19	23	20	3	0	0	0	65
CBA		19	35	44	14	0	0	0	112
LC	Breast-w	59	172	371	371	513	471	285	1871
CBA		59	387	2565	8479	14493	13816	4519	44318
LC	Breast cancer	82	284	848	993	559	123	0	2889
CBA		82	591	3438	10091	16071	14618	5050	49941
LC	Diabetes	30	62	134	130	36	0	0	392
CBA		30	92	290	486	394	0	0	1292

LC*= Looking at labels of the class CBA **= Not looking at labels of the class

.

It is very obvious from the numbers displayed in Table 4.12 that our proposed algorithm needs less time to execute while generating the rules in the training phase when compared with CBA. The small data sets of “balloon”, “contact” and “weather” and “irisd” showed the reduction in execution time by an average of 15.79% as compared to CBA. On the other hand for the medium and large data sets of “Zoo”, “Breast–w”, “Vote”, “Lymph”, “Glassd” and “Cleaved” , the average decrease in execution time is 71.1% which shows a very significant decrease as

Table 4.12: Execution time (milliseconds) of LC and CBA algorithms

Data set	CBA	LC	Difference (%)
Balloon	97	82	15.46
Contact	115	96	16.52
Cleved	457129	116346	74.54
glassd	1294	530	59.04
Irisd	109	94	13.76
Led7	22948	2917	87.28
lymph	71432	8954	87.46
Vote	152117	63461	58.28
Sick	242706	276683	-13.99
weather	94	78	17.02
Zoo	39515	6942	82.43
Breast cancer	47081	11825	74.88
Breast-w	59452	23712	60
Diabetes	1092	920	15.75

compared to CBA. This is due to the fact that LC does not perform the unnecessary joining of itemsets that have different class labels, in n^{th} iteration that follow the previous $(n\text{-}1)^{th}$ iteration. This has eventually reduced the search space and consequently decreased CPU time (execution time). The execution time findings of our LC approach is better and consistent than CBA on ‘13’ data sets with only one exception the “Sick” data set. When we have analysed the “Sick” data set, we found out that it is a binary classification problem with only two classes and the class frequency of the one class is much higher than the frequency of the other class. It was found that most of the itemsets in “Sick” data set are linked with the dominant “negative” class. In fact, out of “3772” instances or entries in “Sick” data set “3541” contains “negative” class in their consequent, so it constitutes about 93.8% of the data set. Hence the large number of itemsets that passes the *minSupp* threshold at the initial iterations are associated with the same dominant class i.e., “negative’. Our approach has took longer time to execute than CBA because our approach

looks at the classes before merging and here the class associated with most of the itemsets is identical. On the other hand CBA do not look at the class labels and saves an extra effort of checking, this explains the higher CPU time and memory usage in our case for the “Sick” data set.

Table 4.13 shows the memory usage in terms of physical, paged and virtual for both

Table 4.13: Physical, paged and virtual memory (bytes) used by CBA and LC algorithms

Data set	Physical Memory		Paged Memory		Virtual Memory	
	CBA	LC	CBA	LC	CBA	LC
Balloon	3440640	3432448	6307840	6307840	82763776	82763776
Contact	3432448	2719744	7340032	3846144	82763776	69636096
Iris-Id	3432448	3428352	6307840	6307840	82763776	82763776
Vote	12103680	12066816	14462976	14458880	130703360	130703360
Sick	11382784	12070912	13963264	14450688	126967808	130678784
Cleved	12107776	12075008	14458880	14458880	130678784	131203072
Led7	11354112	11313152	13971456	13946880	126967808	126418944
Zoo	12029952	12013568	14446592	14446592	130703360	130678784
Lymph	12038144	12029952	14450688	14454784	130703360	130703360
Weather	3440640	3432448	7340032	6307840	82763776	82763776
Glassd	11476992	10354688	14254080	13750272	129814528	126078976

approaches during the training phase. The memory usage of LC in terms of physical, paged and virtual is also less for all the data sets except the “Sick” if compared with CBA because of the facts described above.

We have generated the results of the number of frequent ruleitems and rules produced in LC and CBA before building the classifier (pruning) and after building the classifier. We have performed our experiments by keeping the *minSupp* to 5%, and the *minConf* value kept at 40%. The results are computed while generating the frequent ruleitems in all the iterations of the training phase in the LC and CBA approaches against the 14 UCI datasets considered. The results depicted in Table 4.14 revealed that the numbers of CARs derived for both LC and CBA

Table 4.14: LC and CBA number of CARS and rules generated after pruning, using minSupp=5% and minconf =40%

Approach	**Data set**	**Total no of CARs generated (w/o) pruning** *minsupp = 5, minconf =40%*	**Total no of Candidate rules generated (with pruning)**
LC*	Balloon	38	3
CBA**		88	3
LC	Contact	51	5
CBA		72	9
LC	Irisd	45	11
CBA		64	15
LC	Vote	1865	79
CBA		3729	83
LC	Zoo	629	7
CBA		1996	8
LC	Led7	82	33
CBA		588	206
LC	Glassd	129	24
CBA		511	30
LC	Lymph	640	44
CBA		2007	50
LC	Sick	2406	16
CBA		2406	16
LC	Cleaved	2211	73
CBA		5513	78
LC	Weather	56	6
CBA		121	6
LC	Breast-w	676	59
CBA		1455	61
LC	Breast cancer	751	62
CBA		829	74
LC	Diabetes	213	52
CBA		315	70

against the 14 data sets without pruning are different 13 datasets and same for only “Sick” data set. In Table 4.14, column 4 we have also shown the results of the number of rules generated after performing pruning on both CBA and LC algorithms.

The dataset of “Lymph” has produced only “640” frequent ruleitems as compared with “2007” with CBA. The medium and large data sets of “Zoo”, “Vote”, “Lymph”, “Glassd”, “Breast–w” and “Cleaved” have produced on an average of 62.97% less number of CARs for LC when evaluated against CBA. For the smaller data sets of “balloon”, “contact” and “weather” and “irisd”, LC has shown a considerable decrease in the number of CARs generated i.e., 42.34% of CBA.

When the comparisons are drawn for the number of candidate rules generated after the pruning is applied on the CARs the amount of candidate rules generated by our approach LC are found less than the CBA for “11” data sets except “Balloon”, “Weather” and “Sick”, these three data sets have shown same number of candidate rules generated of “3’, “6” and “16” respectively for both LC and CBA approaches. The average number candidate rules generated for all “14” UCI data sets for CBA is 50.64 as compared to 30.57 for LC, the average decrease of 39.63% of CBA is calculated when the total number of candidate rules are considered for LC.

It can be concluded after the results evaluation of our experiments for Table 4.14, that the number of CARs generated for LC are much less than the number of CARs of CBA for the data sets considered, due to the fact that LC algorithm implies two conditions before and after merging of itemsets, the first condition is to only merge the disjoint itemsets which have common class labels and second is the removal of the ruleitems with less frequent class if the ruleitem is associated with more than single class in the consequent and keeping only the ruleitems with the frequent class. These two conditions that are not applied in CBA have led to the significant decrease in the number of frequent ruleitems generated in LC. The decrease in the number of CARs has a significant impact on the total number of candidate rules generated, and hence smaller size candidate rules are generated. This may have an impact on the total number of rules in the final classifier. We will demonstrate in our description below the number of rules contained in the final classifier and the prediction accuracy of these rules for both AC algorithms and classification algorithms for better comparisons.

It is noticeable from the results given in Table 4.15 that LC produced smaller size classifiers than CBA and for most of the data sets used during the experiments. The main reasons for the less number of rules generated for the LC algorithm is explained in the above section.

Table 4.15: LC and CBA number of rules generated after pruning,using *minSupp=5% and minconf =40%*

Data Set	**Total No. of Rules in Classifier "C" of LC minsupp = 5%, minconf =40%**	**Total No. of Rules in Classifier "C" of CBA minsupp = 5%, minconf =40%**	**Difference (no of rules)**
Balloon	1	3	2
Contact-lenses	5	6	1
Irisd	5	5	0
Vote	32	34	2
Sick	16	16	0
Cleved	44	70	26
Led7	27	52	25
Zoo	7	7	1
lymph	33	35	2
weather	6	6	0
glassd	19	26	7
Breast-w	51	33	18
Breast cancer	48	49	1
Diabetes	34	42	8

The most important and the final step now to check the prediction power and accuracy of our algorithm and compare it with that of CBA, the *minsupp* is kept at 5% and *minconf* is set to 40%, the results are shown in the Table 4.16. The results are very promising and our proposed algorithm LC has outperformed CBA algorithm on the accuracy measure for almost all the data sets. The results of Table 4.16 revealed that LC algorithm derived an average of 87.45% prediction accuracy on the classification benchmarks we consider, whereas the average prediction accuracy of CBA was 86.15%. In general, LC is +1.3 % better than CBA with respect to classification accuracy on the benchmark problems utilised in this section. We have explored the reasons why the prediction accuracy is low for "Led7" data set; it contains 7 attributes with each having two attribute values, and 10 classes. By calculating the class distribution with the attribute values, it came to our knowledge that these are evenly distributed. Each attribute value

Table 4.16: Comparisons between LC, CBA and C4.5 for prediction accuracy

Data Set Name	**No of Classes**	**Data set Size**	**C4.5**	**CBA**	**LC**
Balloon	2	20	100	100	**100**
Sick	2	3772	93.87	91.56	**93.90**
Cleve	2	303	81.54	83.13	**89.76**
Glassd	7	214	66.82	69.89	**71.02**
Iris	3	150	96	93.25	95.33
Led7	10	3200	73.56	71.74	58.78
Lymph	4	148	85.13	76.38	**87.84**
Vote	2	435	89.19	86.91	**90.57**
Zoo	7	101	93.06	**95.96**	93.06
Contact-lenses	3	24	83.33	91.66	91.66
Weather	2	14	100	100	**100**
Breast-w	2	699	94.56	95.84	**98.14**
Diabetes	2	768	73.85	75.34	**78.13**

is linked with all the ten classes. In our LC algorithm, the minority rule in the training phase tends to remove all the ruleitems that are linked with the minority class values. In case of "Led7", considering that an attribute value is associated with "10" classes, so "9" of the ruleitems with same attribute value and different classes will be removed and only one ruleitem is kept with the majority class. This has led to the removal of significant potential rules in case of "Led7"data set and may have degraded the prediction accuracy. Our LC is working well for multiclass problems of "Zoo", "Glassd", "Lymph" and "Contact–Lenses" except "led7" data set, because of the reasons described above. Our algorithm LC has outperformed C4.5 in "8" data sets, showed same accuracy for "3" data sets and C4.5 have shown better accuracy for "2" data sets considered in our experiments.

Our next step is to demonstrate the effectiveness of our new prediction approach that considers group of rules to make the prediction rather than a single rule. For that purpose we have conducted experiments using the LC algorithm and implementing a method for prediction

using single rule and another method using multiple rules. In single rule prediction method, the class value of the first rule that matches the test instance condition will be assigned to that test case. In the case of the multi rule the decision to assign the class value will be made by a group of rules. All the rules are selected that matches the condition of the test instance, the average confidence of all the classes in the group is calculated and the class with the highest value of confidence is assigned to the test instance. Hereunder is the description of the columns in Table 4.16.

Column 1: Contains the data sets used.

Column 2: Shows the number of records or entries in the data set.

Column 3 and 4: Presents the total number of rules in the classifier that have produced the least number of errors, and the total minimum number of errors using single rule for prediction.

Column 5: Shows the prediction accuracy percentage for single rule prediction method.

Column 6-7: Presents the number of rules from the classifier that has achieved minimum number of errors.

Column 8: the prediction accuracy of using group of rules prediction depending on average confidence for each group

The results in Table 4.17 have shown an improvement in accuracy of four data sets using group of rules prediction including “Cleve”, “Zoo”, “Lymph” and “Glassd” with prediction accuracies of 89.76%, 93.06%, 87.83% and 71.02%. On the other hand, single rule prediction method achieved 86.46%, 91.08%, 85.81% and 69.62% respectively on the same data sets. The prediction accuracy remained the same for single and multiple rules prediction on six data sets including “ Balloon”, “Weather”, “Irisd”, “Contact”, “Diabetes” and “Sick”. The prediction accuracy is decreased by a fraction of 0.14 % for “Breast-w” and 0.39 % for “Led7” data sets when multiple rules based prediction is applied as compared with single rule prediction.

Table 4.27: Comparison of single vs group of rule prediction accuracy

Data Set	No of Records in Data Sets	Single Rule Prediction		Prediction % (Single Rule)	Group of Rules Prediction		Prediction % (Group of Rules)
		No of Rules	No of errors		No of Rules	No of errors	
Balloon	20	1	0	100	1	0	100
Contact	24	5	2	91.66	5	2	91.66
Iris-Id	150	5	7	95.33	5	7	95.33
Vote	435	32	45	89.65	32	41	90.57
Sick	3772	16	230	93.90	16	230	93.90
Cleved	303	44	41	86.46	44	34	89.76
Led7	3200	27	1306	59.18	27	1319	58.78
Zoo	101	10	9	91.08	10	7	93.06
lymph	148	37	21	85.81	37	18	87.83
weather	14	6	0	100	6	0	100
glassd	214	19	65	69.62	19	62	71.02
Breast-w	699	51	12	98.28	51	13	98.14
diabetes	768	34	168	78.13	34	168	78.13

4.6 Summary of Chapter

In this chapter we have discussed in detail the rule generation, rule ranking, rule pruning, classifier builder steps in the proposed algorithm with the experimental results. The proposed algorithm enhanced on CBA main steps especially frequent ruleitems discovery by reducing the candidate ruleitems merging at each iteration and using the minority rule have improved the performance with reference to memory use and training time. The experimental results, against UCI data sets from UCI repository, of proposed algorithm LC and CBA have been produced. The results have shown that the number of merging at each stage of the rule generation has reduced significantly for the LC algorithm in comparison to CBA and for all the data sets considered. The experimental findings have clearly shown a positive impact when the number of

candidate ruleitems merging at each iteration is reduced on the execution time and the memory usage of the LC algorithm for all data sets except “Sick”.

In the experiments of the classifier building step, the number of frequent rules generated before pruning starts have reduced significantly when different values of support and confidence are used if contrasted to CBA. The results have also shown that when pruning of those frequent ruleitems is performed using database coverage method, the numbers of rules generated are less than the CBA. In the final part of the chapter the most important step of checking the accuracy of the classifiers build by our proposed prediction algorithm that uses multiple rules to predict the class labels was shown. The comparative analysis with CBA and C4.5 have shown significant and better prediction accuracy for almost all the test data sets, using the classifiers build by LC algorithm and using our prediction algorithm. The results have shown some improvement in the prediction accuracy when comparisons are presented while using group of rules as compared to single rule prediction.

In the next Chapter 5 we will discuss an example from the real world relating phishing. We will demonstrate and discuss in detail the process of data collection from the real data repositories and findings of our LC approach with comparison to the classification and AC mining techniques.

Chapter 5

Phishing Data Collection Model and Implementation of LC and other Algorithms

5.1 Introduction

In this chapter one of the main problems in information security will be addressed in detail that is phishing. The main focus will be to explain the steps to collect the phishing data of websites, to extract useful or important features from the data and to implement the proposed AC algorithm LC along with other classification and AC algorithms to demonstrate the use of data mining techniques in Phishing. The results of all the approaches on phishing data sets are compared and critical analysis will be discussed in later part of the chapter. The chapter is organized as section 5.2 demonstrates the model for data extraction from phishing and legitimate websites, section 5.3 explains the features selection process and section 5.5 discusses the experiments setup and results.

5.2 Features Selection for Experiments

To execute the experiments, the task of extracting the features from the phishing and legitimate emails is very vital. This task is mutually related to the process of accurately predicting phishing. So it can be said that if the rule used in the feature extracting process from the sample of phishing emails is accurate then the affects will be reflected in the accuracy percentage of detecting phishing. In the next sections, we will describe the procedures of extracting a set of features from a set of phishing, will figure out the features that have significant impact to predict phishing, and will formulate some rules on the basis of some findings for each feature extracted to classify them.

5.2.1 Phishing Data Sources, How and Why Features are Selected

The websites have large domain or matrix of features that can be considered for experimental purposes. How and where these website features are selected has an answer that these set of phishing websites are collected from a free community site of Phishtank archive (Phishtank.com) where phishing data is shared, can be submitted, verified and tracked. The other source of data collection is Millersmiles archive (Millersmile.co.uk), which is the prime source of data for phishing scams and spoof websites. We have collected around 4500 URLs and 250 phishing URLs sample is selected randomly for our experiments. The data is collected in the AI centre at University of Huddersfield.

The preparatory step adopted to construct our data sets are as follows:

- The abnormal base features are extracted by a script as we need to connect with the external website like Who.is (DNS) and results are exported to Excel.
- Data Extractor and HTML extractor tool is used for the extraction of HTML and Java Script base features.
- A Java script is used to extract the domain base features from Who.is (DNS) and Alexa.com.

Initially 27 features are selected from the phishing URLs by the steps explained above. To handle these features and use them together in experiments is not possible and have produced poor results. To select the best features out of the 27 extracted features, frequency analysis and Chisquare testing methods are used. Frequency analysis is based on group base decision and chisquare testing is a mathematical measure used to measure the correlation between the class and feature value and evaluates each feature value. The features that have exceeded the threshold of 3.84 are been selected.

Finally 16 features are selected grouped in four categories. The abnormal base features selected contain three features, address bar features have six, HTML and Java Script base features represents four features and Domain base features have three features. What are the criterion used to extract these features are explained in detail in the following section.

5.3 The Model for Extraction and Evaluation of Chosen Features

It is learnt from the literature that websites and mails have large domain or matrix of features and considering all the features together could be complex and showed poor performance. The model used extracts features from the phishing page that can take at least one input. The features are divided here into four main categories depending on the impact factor in predicting phishing and then are allocated to one of the four categories. The proposed model uses output from higher

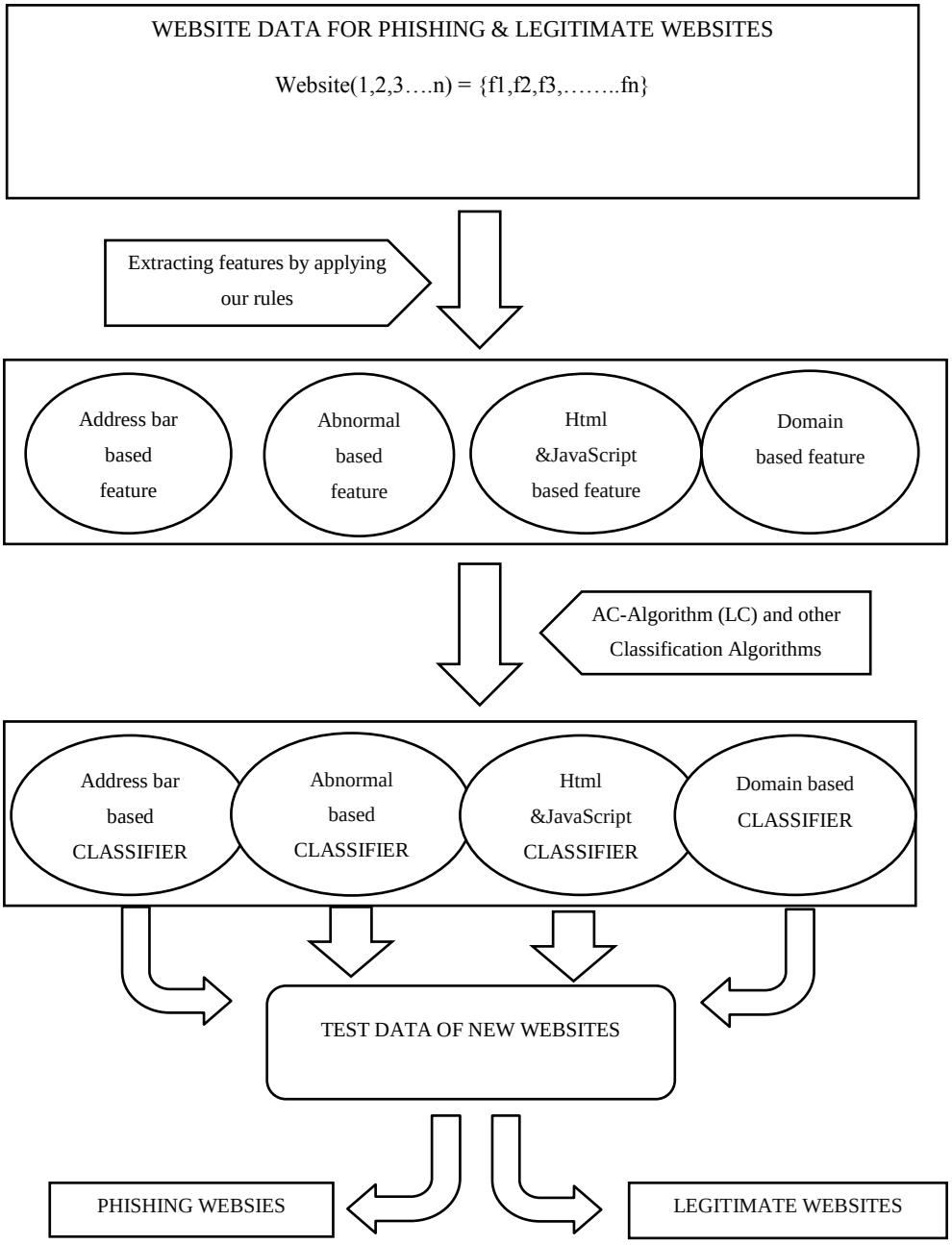

Figure 5.1 Feature extraction and phishing detection model

level as the input to the next lower level, and then different AC mining techniques are applied on the extracted features to generated rules and detection accuracy. These rules will be used for future prediction of the websites. In this sub-section, we describe the categories of the features used.

5.3.1 Abnormal Based Features

5.3.1.1 Request URL

Webpages contains objects that may be externally linked to different domains. Normally in a genuine website nearly all the objects would be linked to the current domain. In the experiments the extraction of the <src=> from the http://www.dft.gov.uk/dvla/ website source code and compared it with the domain <src=> present in the URL, if it is different than the website is "phishing".

The rule condition to classify the above feature is produced by calculating the percentage of URLs that have different domain name in the source code to the entered domain name in the address bar.

5.3.1.2 URL of Anchor Feature

The website is classified as phishing if the domain of the anchor usually defined by <a> tag is different than the website or the anchor that do not show any link to a webpage, e.g. <a href= "# content"> and <a href= "#skip">. This feature is same as the feature of request URL. When we analysed our data set, we found 58 URLs of anchor feature and that refers to 23.2% of the total data set.

5.3.1.3 Feature of Server Form Handler (SFH)

When the domain name in SFH-s is not similar to the domain name of the website then the website is termed as "suspicious", and when the SFH have "about:blank" or have empty string("") then its termed as " phishy" as some action has to be taken. We have found 10 URLs containing SFH-s and presents only 4% of data pool.

5.3.2 Address Bar Based Features

5.3.2.1 Use of IP Address in Domain Name

If a user notices an IP address e.g. 196.25.6.101 in place of a normal domain name in the URL, he should be pretty sure that something is wrong and suspicious. The user may find a hexadecimal in the IP address like: http://0xDD.0xBE.0x25.0xAA/3/ebay.uk/html. This feature is very important and selected in the proposed group, because it is used in most of the previous researches. 22.8% of our collected data has this feature.

5.3.2.2 URL Length

The Phishing emails are often linked with long URLs to cover the suspicious portion of the address box. The exact length of the phishing URL is not established yet and it is difficult to differentiate between a legitimate URL and phishing ones. In our study we calculated a mean of the URL length of the phishing URL collected from the Phishtank archive. The outcome of the calculation is, if the URL length is >= 54, then URL is marked as Phishing. When the datasets are checked 122 URLs of size 54 are found and greater and this makes 48.8% of our data. This feature is classified in our study into three types; 'Low', 'Moderate' and 'High', when length of URL is <54, >= 54 and <=75, and >75 respectively.

5.3.2.3 URL having @ Symbol

The Phishers uses "@"symbol in the URL because the browser may overlook the actual address following the "@" symbol. This "@" symbol is used to dodge the system that URL is genuine. We have found 9 URLs with @symbol in our data set which represents 3.6%.

5.3.2.4 Prefix or Suffix in the Domain Part

Phishers try to hide information in the domain part of the URL as a suffix or a prefix separated by dash (-). Dash is used in the genuine URL but very rarely. Users have no idea whether they are dealing with the phishing website or with legitimate website. In our study 66 URLs are found containing (-) symbol in our data set which comprises about 26.4% of total URLs.

5.3.2.5 URL having Sub Domain

A domain name comprises of mainly two levels: top level domain (TLD) and a second level domain (SLD). We analyse a link like http://www.dft.gov.uk/dvla/, where .uk is TLD, .gov is an abbreviation for government, joined ".gov.uk" is known as SLD and "dft" is main domain name. Thus it can be noted that a genuine URL contains two dots when ignoring the dot with 'www'. The collected data set having 111 URLs that contains >= 3 dots in the domain section represents 44.4% of the data set.

5.3.2.6 SSL (Secure Sockets Layer) Final State

Due to growth in online transactions large quantities of personal and important information are transferred. The websites provide security by using domain names that are secure, and protocols uses SSL to ensure a safe transfer of sensitive data by generating certificates. Presence of HTTPs gives genuineness to the website, but as Netcraft's community have extracted 450 phishing URLs that uses HTTPs since 2005 [6], this fact is not valid any more. Authorities like GoDaddy, Verisign and GeoTrust that generate certificates are trust worthy.

In the pool of data 92.8% of the data found contains 232 URLs not supporting HTTPs or used a fake https. In the past studies, researchers have not considered fake https but the extracted feature does take up these fake providers.

5.3.3 HTML and JavaScript Based Features

5.3.3.1 OnMouseOverfeature

A fake URL is displayed in the status bar when the OnMouseOver function changes the status bar. The extracted dataset have counted 50 URLs and these comprise 20% of whole data pool.

5.3.3.2 Redirect Feature

The phishers are looking for the sites that offer open redirects; these are exploited and phishers create links to phishing sites, this gives an impression of a genuine website. Only one click leads

the unaware user to the phishing site. 10% of the data sets collected contain the redirect page features.

5.3.3.3 PopUp Window Feature

The genuine websites normally do not ask the user to enter their login information via a popup. It is found that 9.2% of the users entering their personal information in the popup window (PW).

5.3.3.4 RightClick Disabled Feature

Right click is disabled by phishers when they use JavaScript in the phishing website, thus preventing the user from viewing source code. In the collected data only 4 URLs of such nature were found.

5.3.4 Domain Based Features

5.3.4.1 Domain's Age Feature

Some of the features from the WHOIS database are extracted and used in our study. Domain's age feature used in our dataset is assigned "Low" value if the age of the domain is more than 1 year. If the domain's age is less than 1 year then it is assigned "High" value. In the data set it is found that 239 URLs, where the domains were originated for less than 1 year and this constitute 95.6% of the data set.

5.3.4.2 Record from DNS Feature

This feature is also collected from the WHOIS repository. The identity of the phishing website domain name cannot be found in the DNS database. In the data set build up, if the hostname's record cannot be found in the DNS entries or the identity of the website is not matched by WHOIS then the website is marked as "phishing" and assigned "high", otherwise the website is genuine and assign it value of "low".

5.3.4.2 Website Traffic

The analysis of the website traffic feature is very significant in recognizing the legitimacy of the websites. When no traffic is recorded in Alexa database, then the website is marked as "phishing". The trusted websites are included in the top 10,000 ranked entries. This feature is the popularity measure of the URL. If the URL ranked inside the top 10,000 then the website is "legitimate and trusted", else "suspicious".

5.4 Experimental Setup and Data Sets Used

The experiments are conducted and evaluations are demonstrated on the phishing data sets of *AddressBarBaseFeatures*, *HTMLandJavaScriptBasedFeatures*, *AbnormalBasedFeatures* and *ALEXA data set* also named as *DomainBaseFeatures* , extracted by the procedures described in section 5.3., for number of merging at each iteration of LC and CBA, the number of rules generated and accuracy of the classification and AC algorithms. All the data sets constructed represent binary classification problem and contains two possible values for the class attribute: "Legitimate" or "Phishing". We will use different classification data mining algorithms including PART (Frank and Witten, 1998), C4.5 (Quinlan, 1993) and Naïve Bayes (Duda and Hart, 1973), and an AC algorithm of CBA (Liu et al., 1998) and proposed LC algorithm in the experiments.

We have performed experiments using WEKA tool (weka.com), an open source Java application, for PART, C4.5 and Naïve Bayes classification algorithms. On the other hand, the experiments are conducted for LC and CBA using the implemented version in Visual C#. The training database used is composed of 564 phishing instances and 450 legitimate ones and has been gathered from Yahoo at the University of Huddersfield.

5.5 Experimental Results and Discussion

A comparative analysis will be conducted for all the above mentioned algorithms in the following sequence.

- Finding the number of disjoint itemsets merging at each iteration for the proposed algorithm LC and CBA.
- Finding the number of CARs and number of candidate rules for rules for the final classifier.

- Finding the number of rules in the final classifier for all classification algorithms including PART, C4.5, CBA, LC and Naïve Bayes
- Deriving the accuracy figures for all classification algorithms to demonstrate the most predictive algorithm that is able to detect phishing accurately.

Table 5.1 and Figure 5.2; demonstrate the number of times itemsets have been merged during each iteration for LC and CBA using *minSupp* and *minConf* of 5%, 40% respectively. The

Table 5.1: Comparison of number of merging for the security data sets using minsupp =5% and minconf = 40%

Website Data set	**Algorithm**	**(single item with class) Cycle 1**	**Cycle 2**	**Cycle 3**	**Cycle 4**	**Cycle 5**	**Cycle 6**	**Total of All Cycles Merging**
Abnormal Base Feature Data Set	LC*	21	11	5	-	-	-	37
	CBA*	21	35	24	-	-	-	80
Address Bar Feature Data Set	LC	30	52	89	75	26	2	274
	CBA	30	92	236	296	177	40	871
Domain Base Data Set	LC	18	11	5	-	-	-	34
	CBA	18	23	14	-	-	-	55
HTML Base Data Set	LC	22	25	27	9	-	-	83
	CBA	22	50	64	25	-	-	161

generated values in have revealed that the number of itemsets merging at each cycle in LC and for all the data sets have decreased significantly as compared to CBA. For example, since the *AddressBarBaseFeatures* data set contains more number of features than the other phishing data sets it has shown larger number of merging in the iterations during the rule generation phase, the

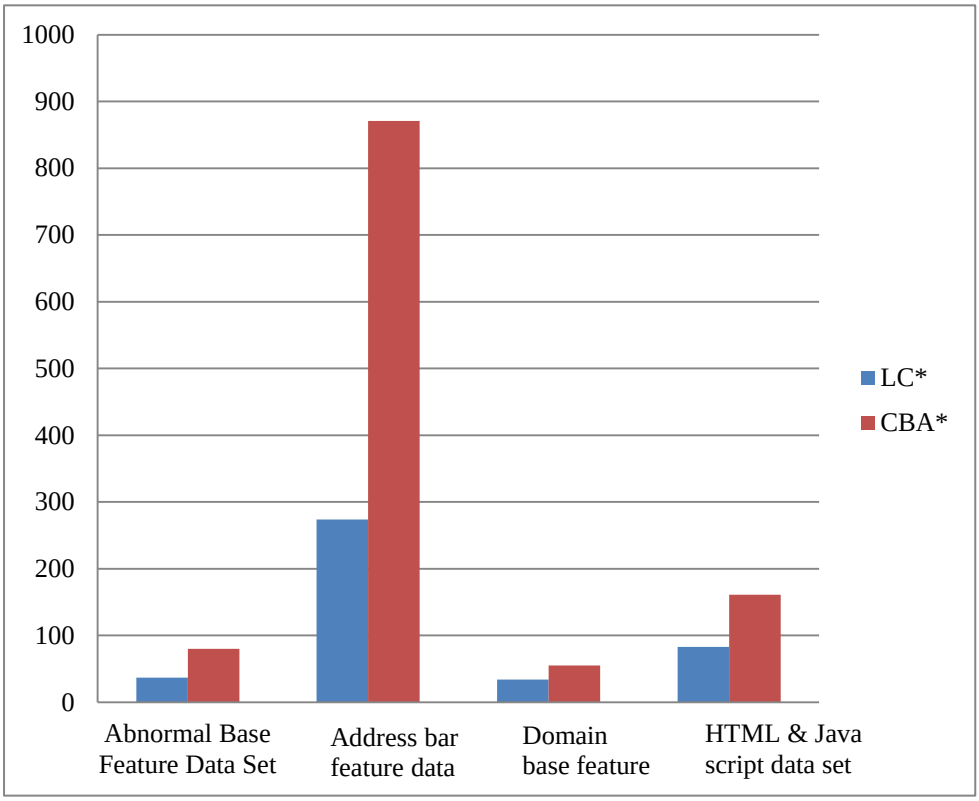

Figure 5.3 Total number of itemsets merging of phishing data sets

total number of itemsets merging in LC and CBA are '274' and '871' respectively. Meaning the number of itemsets merging during the training phase has been reduced in LC to 52, 89, 75, 26 and 2 as compared to CBA 96, 236, 296, 177 and 40 in the 2nd, 3rd, 4th, 5th and 6th iterations, as

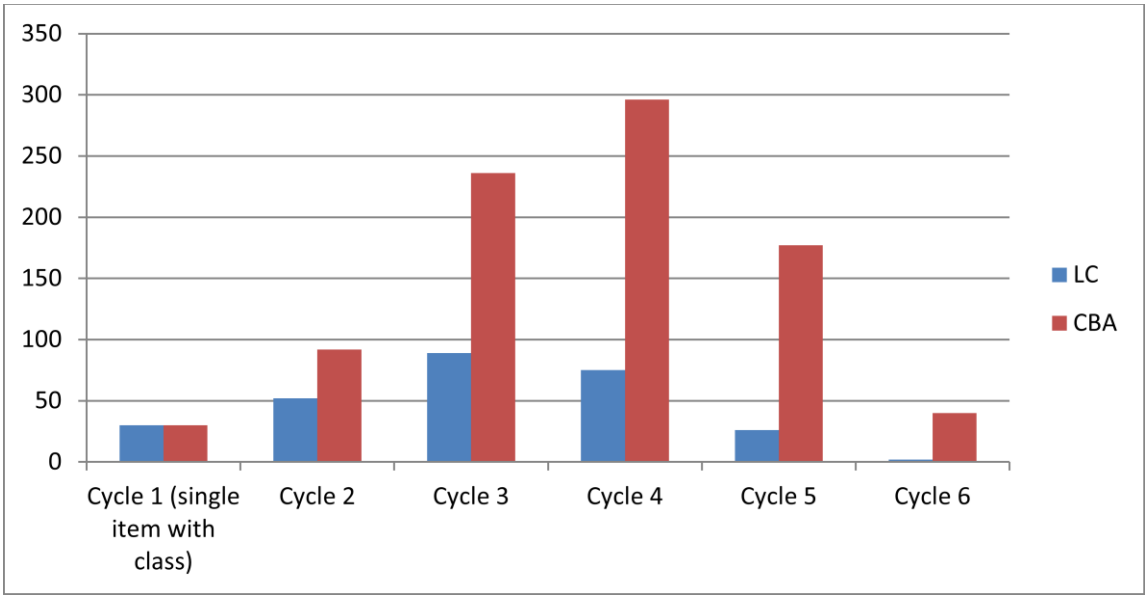

Figure 5.2 No. of merging for all iterations of address bar feature data set of LC and CBA

shown in Figure 5.3. These results are consistent with the UCI results discussed in Chapter 4 which means LC training phase has improved the performance and efficiency of CBA's current training method in particular the training time and memory usage.

Figures 5.2 and 5.3 depict the difference in the number of itemsets merging at each cycle of the AddressBarFeature and total number of candidate itemsets merging for all phishing data sets for LC and CBA algorithms. It can be concluded from Figures 5.2 and 5.3 that the CBA algorithm generates larger number of candidate itemsets than LC algorithm on the AddressBarFeature and HTMLBaseFeature data and for most iterations.

Experiments are also conducted to find the total number of CARs and total number of candidate rules after pruning in the security data sets. Table 5.2 displays the results for the frequent ruleitems and candidate rules for the final classifier for LC and CBA. The results have shown that total number of CARs generated for LC is much less than CBA for all the security data sets. For example the total number of CARs produced for the Address bar feature data set in LC is "148" i.e., 47.88% less than CBA "284", the total number of frequent ruleitems for CBA is "417" in all the "4" security data sets and "244" for LC i.e 58.51% of CBA. This significant decrease is due to the reasons explained in the experimental section of Chapter 4. For example, the number of candidate rules generated from the "Address Bar" data set for the LC algorithm is less than that of CBA. In fact, LC generated '18' rules for *minSupp* value of 5% and *minconf* of 40% whereas CBA produced '24' for the same values of *minSupp* and *minconf* . However, for Abnormal Base feature data set, the number of candidate rules is '17' rules and '15' for CBA and LC algorithm respectively.

The detailed comparative analysis on the number of candidate rules generated by CBA and LC against the four categories data sets are displayed in Figure 5.4-5.7. It clear from the below figures that LC often derives smaller classifiers that CBA algorithm. This surely is an advantage for the decision makers since it increases understand ability and control on behalf of the end-user. The main reasons for generating less number of rules by LC is due to the fact it uses a looking at class condition before merging of itemsets in each iteration and uses a minority rule to remove less frequent associations of ruleitems with different class attribute values.

Table 5.2: Comparisons of number of CARs and candidate rules generated for LC and CBA

Website URL's Data set	**Algorithm**	**Number of Frequent rules (CARs) in all iteration (w/o pruning)**	**Number of Candidate Rules (with pruning)**
		minsupp = 5% *minconf*= 40%	*minsupp* = 5% *minconf* =40%
Abnormal Base Feature Data Set	LC*	21	15
	CBA	32	17
Address Bar Feature Data Set	LC	148	18
	CBA	284	24
Domain Base Data Set	LC	18	9
	CBA	26	11
HTML Base Data Set	LC	57	9
	CBA	75	12

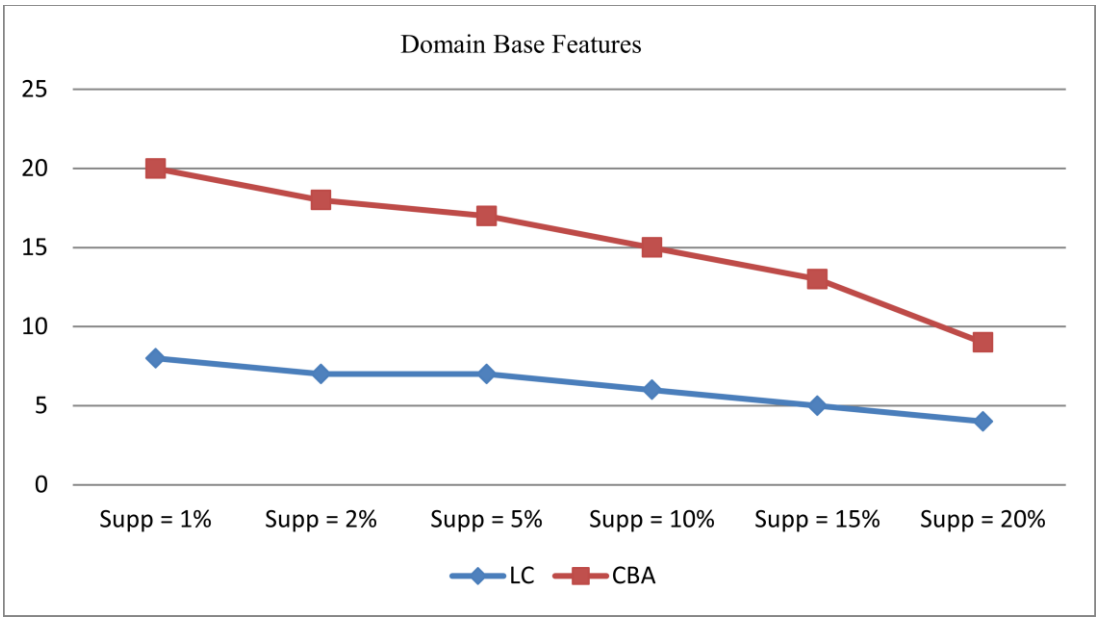

Figure 5.4 No. of candidate rules generated for LC and CBA for domain base feature data set

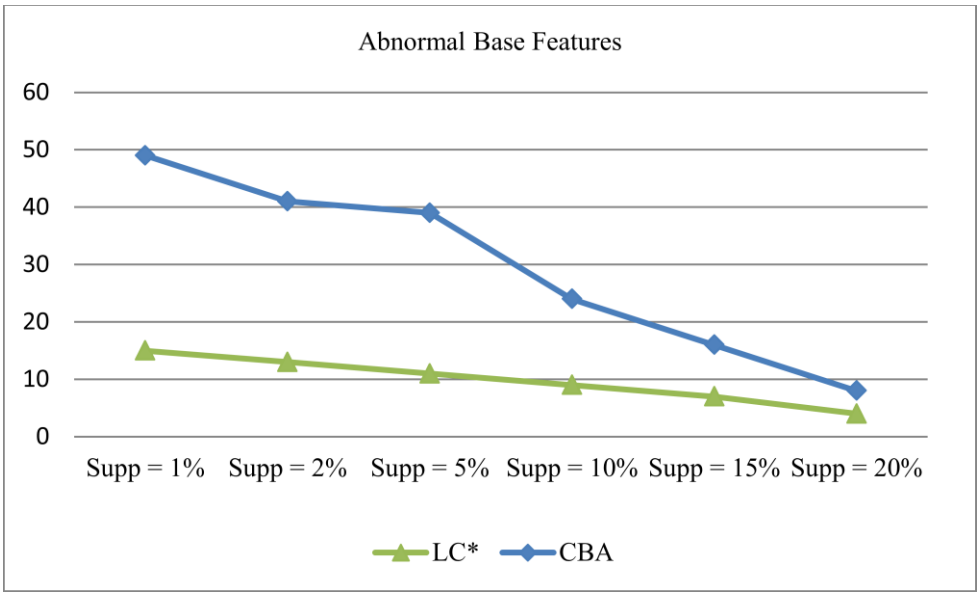

Figure 5.5 No. of candidate rules generated for LC and CBA for Abnormal base features

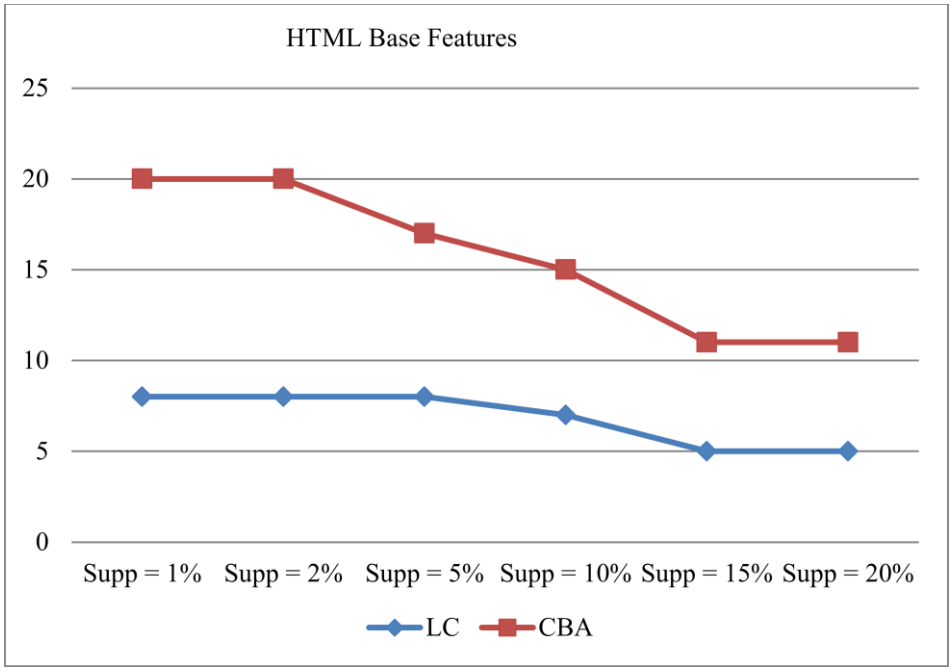

Figure 5.6 No. of candidate rules generated for LC and CBA for HTML base feature data set

The experiments are performed using AC algorithms of LC and CBA and for better comparative study the evaluated is carried out in the well-known classification algorithms of C4.5, PART and Naïve Bayes, for the number of rules in the final classifier. The experiments on classification algorithms are conducted using WEKA tool and for AC algorithms and implemented versions are used for LC and CBA. The results are demonstrated in the Table 5.3 and the sample of number of rules for classification algorithms of C4.5 and PART are shown in Figure 5.7 – 5.10.

Table 5.3: Number of rules in classifier of AC and classification algorithms

	ALGORITHM			
Website Dataset	**PART**	**J48(C4.5)**	**CBA**	**LC**
Address Bar Based Features	7	9	17	**10**
Domain Based Features	4	5	7	**7**
Abnormal Based Features	7	7	13	**12**
HTML and JavaScript Based Features	4	9	12	**8**

J48 (C4.5) pruned tree

SFH = HIGH: Phishing (543.0/21.0)
SFH = LOW
| URL_of_Anchor = HIGH: Ligitemate (44.0/16.0)
| URL_of_Anchor = LOW: Ligitemate (291.0/10.0)
| URL_of_Anchor = MODERATE: Phishing (14.0)
SFH = MODERATE
| Request_URL = LOW: Ligitemate (0.0)
| Request_URL = HIGH: Phishing (2.0)
| Request_URL = MODERATE: Ligitemate (120.0)

Number of Leaves: 7
Size of the tree: 10

Figure 5.7 Sample for no. of rules generated in C4.5 algorithm for Abnormal base data set

---

J48 (C4.5) pruned tree

---

on_mouseover = LOW: Ligitemate (420.0/4.0)

on_mouseover = MODERATE

| Redirect = LOW: Phishing (144.0)

| Redirect = MODERATE

| | RightClick = False: Phishing (79.0/8.0)

| | RightClick = True: Ligitemate (8.0)

| Redirect = HIGH: Phishing (123.0/8.0)

on_mouseover = HIGH

| RightClick = False: Phishing (204.0)

| RightClick = True

| | Redirect = LOW: Ligitemate (8.0)

| | Redirect = MODERATE: Phishing (0.0)

| | Redirect = HIGH: Phishing (24.0)

Number of Leaves: 9

Size of the tree: 14

Figure 5.10 Sample for no. of rules Generated in C4.5 HTML and Java Script Base Features data set from WEKA

PART decision list

---

on_mouseover = LOW: Ligitemate (420.0/4.0)

RightClick = False: Phishing (538.0/16.0)

Redirect = HIGH: Phishing (36.0)

: Ligitemate (16.0)

Number of Rules: 4

Figure 5.8 Sample for no. of rules Generated in PART algorithm for HTML and Java Script Base Features data set

---

PART decision list

SFH = MODERATE AND

Request_URL = MODERATE: Ligitemate (120.0)

SFH = HIGH: Phishing (543.0/21.0)

URL_of_Anchor = LOW: Ligitemate (291.0/10.0)

URL_of_Anchor = HIGH AND

Request_URL = LOW: Ligitemate (28.0/8.0)

URL_of_Anchor = HIGH AND

Request_URL = HIGH: Phishing (16.0/6.0)

URL_of_Anchor = MODERATE: Phishing (14.0)

: Ligitemate (2.0)

Number of Rules: 7

Figure 5.9 Sample for no. of rules Generated in PART for Abnormal Base data set

Table 5.3 shows that the classifiers build by AC algorithms of LC and CBA are larger than classification algorithms of PART and C4.5 (J48) by 65.38%. The total number of rules for all security data sets are ‘86’ and ‘52’ for AC and classification algorithms respectively. The rules in the final classifier for our approach of LC are found 24.48% less than CBA. Our LC has shown the similar results for security data set for generating less number of rules for the classifier as it has shown on the UCI data sets, due to the factors described in Chapter 4. Therefore these findings show the strength of the new algorithm to adapt to different data sets.

Figure 5.11 shows the predictive accuracy of the considered classification data mining algorithms. The algorithm LC has achieved higher accuracy than classification and AC approaches of PART, CBA, C4.5 and Naïve Bayes for ‘2’ data sets, domain base feature and HTML and JavaScript feature. The LC has shown same accuracy percentage for Address bar feature data of 94.07% with PART, CBA, C4.5 and Naïve Bayes. The LC has achieved 0.34%. 0.43%, 0.42% and 1.7% higher average prediction accuracy for all the ‘4’ security data sets, when compared with average accuracy of PART, C4.5, CBA and Naïve Bayes algorithms respectively. Overall the AC approaches of LC and CBA have generated large size classifiers with better accuracy than classification approaches. The prediction results as shown in the Figure 5.11 have shown that LC has performed better against other approaches for Domain based and

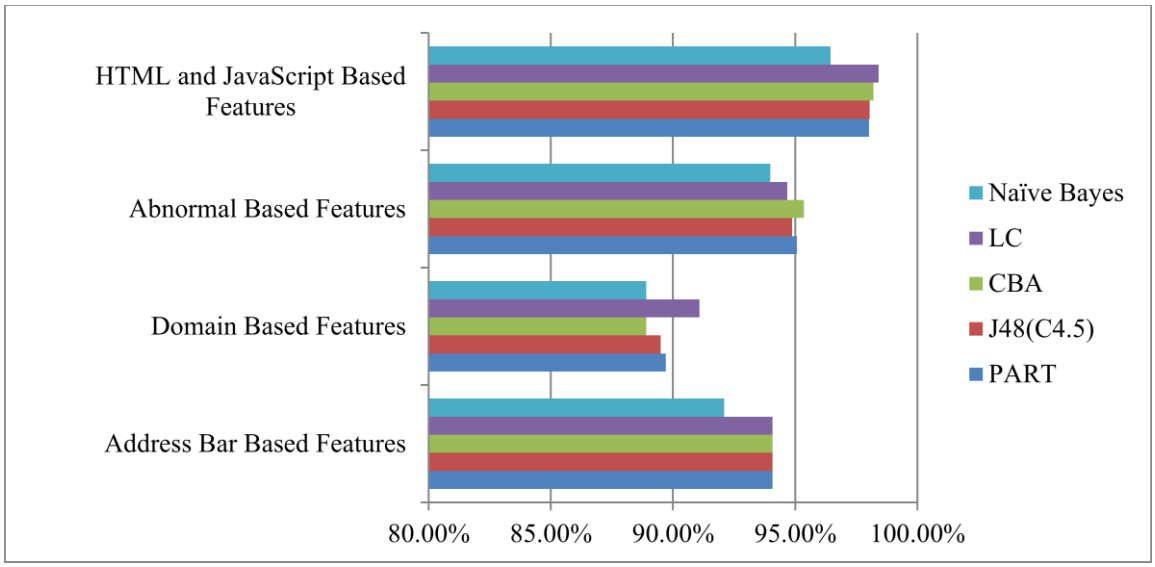

Figure 5.11 Comparison of prediction accuracy in (%) of LC, CBA, PART, C4.5 and Naïve Bayes algorithms

HTML and JavaScript base, and Address bar base feature data sets except Abnormal base feature data set.

In Table 5.4, the results of the number of instances correctly classified by all the AC and classification approaches are demonstrated for better comparative analysis.

Table 5.4: Number of correctly classified instances for security data sets for AC and classification algorithms

	ALGORITHM				
Website Dataset	**PART**	**J48(C4.5)**	**CBA**	**LC**	**Naïve Bayes**
Address Bar Based Features	952	952	952	952	932
Domain Based Features	906	904	898	920	898
Abnormal Based Features	964	962	967	960	953
HTML and JavaScript Based Features	990	990	992	994	974

5.6 Summary of the Chapter

Phishing is a serious security issue for the internet users. In chapter 2, we surveyed common approaches related to phishing in the literature like CART (Breiman et al, 1984), SVM (Joachims, 1999), BART (Fette et al., 2007), RF (Breiman, 2001), Neural Networks (NN) (Marques, 2001), PECM (Al Momani, 2011), APPT (A.A Khan, 2013), and hybrid approaches (Hamid et al., 2013). In this chapter, a comparative analysis is conducted to show the applicability of the proposed LC algorithm on the difficult problem of phishing. Phishing data is collected and different types of features are extracted like HTML and JavaScript, Abnormal, Address Bar and domain base. An experimental section using the proposed algorithm and other known classification techniques like CBA, C4.5, PART and Naïve Bayes against the features data set has been conducted. The main factors considered in the experimentation are the number of itemsets merging of LC and CBA, The number of rules in the final classifier produced by CBA and LC and the prediction accuracy of all considered algorithms. The results revealed that LC algorithm have very promising results and generated less number of frequent ruleitems and

rules in the final classifier when compared with CBA. LC has also outperformed the other classification approaches (PART, C4.5 and Naïve Bayes) and AC approach of CBA in terms of accuracy on most data sets considered.

It can be concluded from the results that LC has an execution time faster than the CBA as it has shown a significant reduction in the number of merging at each iteration and so it also uses less memory space. The prediction accuracy of the approach has shown significant results with less number of mis-classification.

Chapter 6

Critical Analysis of the Experimental Results

The results obtained by all the experiments using LC, CBA and classification algorithm on 14 UCI datasets and phishing datasets will be analyzed and discussed critically in this chapter. The results that are explained are extracted from the experiments conducted on AC algorithms of LC and CBA by using the implementation version in Visual C, and using a statistical algorithm of Naïve Bayes, a decision tree algorithm of C4.5 and a hybrid approach known as PART. The aim is to compare and analyze the results of all the bench mark and well known approaches in classification rule mining and associative classification rule mining for better understanding.

6.1 Reduction in the Number of merging of itemsets in LC and its Impact on the Execution time and Memory Usage

The findings as demonstrated in Table 4.11 and Table 5.1 have clearly indicated that the LC is able to reduce the research space to find the candidate rule itemsets during each iteration. Many researcher as mentioned in literature review chapter 2 are trying to handle the complexity in the training phase of AC where frequent rule itemsets are produced. In this research reduction in the search space is successfully achieved by reducing the number of candidate itemsets generation at each iteration. Table 4.11 and Table 5.1 indicates that the number of merging is reduced significantly for all UCI and phishing data sets used in the experiments because of the modifications made in LC algorithm.

The LC algorithm works well for the small data sets like “Balloon”, “Contact”, “Weather” and “Irisid”, large datasets like “Zoo” and “Lymph” and phishing datasets like “Abnormal Base Feature Dataset”, “Address Bar Feature Dataset”, “Domain Base Feature Dataset” and “HTML Base Dataset”, and decreases the total no of CARs generated without pruning when compared with CBA as in Table 4.14 and Table 5.2.

By digging deep in the results of table 4.11 at each iteration, it is observed that the LC algorithm for rule generation phase terminates before the iteration 7 for all medium and large datasets like “Vote”, “Glassd”, “Lymph”, “Zoo”, and “Led7” etc except the “Breast-w” and “Cleaved” datasets. So the number of mergings at the iteration 7 column in Table 4.11 shows “0” value for the above mentioned datasets. It is noticeable that for the multi class problems like “Zoo”, “Led7”, “lymph” and “glassd” with 7, 10, 4 and 7 numbers of classes, the number of ruleitemsts produced at the end of each iteration is much less than the datasets with binary classes like “Breast-w” and “Cleaved”. As the LC look at the class lables of itemsets before merging and uses minority rule after merging, so the number of merging reduces with each iteration and there are no itemsets left to merge with same class lables after iteration 6.

The “Zoo” dataset has ‘7’ classes and the number of itemset merging at iteration 2, 3, 4, 5 and 6 are 3.49% , 2.78%, 1.9%, 1.78% and 1.49% of CBA respectively which is much less than the number of itemset joining 44%, 14.46%, 4.37%, 3.53% and 3.4% of CBA in “Breast –w” dataset with two classes for the same iterations.The difference in ratio in the number of the itemsets merging at each iteration for multiple class datasets and binary class datasets are consistent in the experimental results of Table 4.11 and 5.2. This shows that the LC is performing efficiently for all type of UCI datasets and also for phishing datasets in the rule generation phase.

Because the number of frequent rule itemsets produced in the rule generation phase is significantly reduced as explained in chapter 4 and 5. The effects of this reduction is studied on the memory usage and processing time. Both are found to be reduced as shown in the Table 4.12 and 4.13 for all the datasets used in the experiements except the dataset of “Sick” because the frequency of the class “negative” constitutes about 93.8% of the dataset and large number of itemsets are associated with the “negative” class. Hence the approach in LC algorithm has taken longer time to execute the training phase as its looks at the class lables to join the itemsets as compared with CBA and so as the physical and page and virtual memory used is also higher as shown in Table 4.13.

6.2 Total Number of CARs Generated Before and After Pruning

Table 4.14 and table 5.2 shows the total number of CARs produced after and before pruning, it is noticeable that the number of CARs produced in LC for all the phishing datsets and UCI datsets

are considerably reduced as compared to CBA. The reduction percentage is the highest for the datasets used with the highest number of classes like in “led7” dataset the number of classes are ‘10’ and the decrease in the total number of CARs is 13.94% of CBA. The datsets with ‘7’ classes like “Glassd” and “Zoo” have the reduction percentage in number of CARs is 25.24% and 31.5% respectively. And the percentage is 40.10% and 46.46% of CBA in datasets of “Cleaved” and “Breat-w” with 2 classes. So it is concluded that the number of CARs generated is dependent on the number of classes in the dataset and the number of attributes. The percentage of reduction in the number of CARs is found to be highest in the datasets with highest number of classes and a linear relationship is found between the number of CARs and number of classes in datasets when the numbers of classes are changing in datasets. There is no reduction in the number of CARs found in “Sick” dataset when compared with CBA in Table 4.14 and 4.15 because of the reason that the “Sick” dataset contains one dominant class which is about 93.8% of the total data. So the numbers of frequent itemsets generated are same as “2406” and “16” before and after pruning respectively for both LC and CBA approaches which show that LC algorithm will generate same number of CARs as CBA algorithm for the datasets with one dominant class with percentage more than 90%.

In the phishing datasets as the number of classes is same for all ‘4’ datasets. The number of CARs generated for LC is found to be dependent on the number of attributes in the phishing datasets. The decrease in the percentage of number of CARs in “Address Bar Feature Dataset” with 6 attributes, “Domain Base Data set” with 3 attributes and “Abnormal Base Feature Data set” with 3 attributes are 52.11%, 69.23% and 65% of CBA respectively.

Analyzing the results of Table 4.15 and Table 5.3, the number of rules in the final classifier is found to be less for LC than CBA but are more than the other classification approaches like PART and C4.5. The reason for the less number of rules produced in C4.5 for phishing datasets is becasuse the algorithm selects one attribute as the root value by the entropy measure and then builds the tree around this attribute. In PART the number of rules generated are found to be minimum in phishing datasets among the other approaches is due to the fact that its uses the partial decision tree approach of C4.5 and separate and conquer approach of RIPPER. The rules generated by PART are simple, less in number and accurate. In AC algorithms of CBA and LC the rules generated are more in number because these approaches tends to explore the correlations between attributes that cannot be found by classification approaches like PART and C4.5 and hence increases the understandability of the data.

6.3 Analysis of the Prediction Accuracy Measure

The most important performance measure is the prediction accuracy of any algorithm. The modifications made in the LC algorithm training and prediction phases have achieved improvements in execution times, memory and also shown better results for prediction accuracy when compared with well-known classification algorithm of C4.5 and CBA as demonstrated in table 4.16. The LC algorithm has also shown promising results on the phishing data sets as in Table 5.4 and Figure 5.11 when compared with CBA, statistical and hybrid approaches.

The main focus and concern is for the results of prediction accuracy of the LC algorithm as it avoids two many merging and in the process may eliminate some important rules and consequently may decrease the prediction accuracy. But mostly the results for all the data sets used for experiements in chapter 4 and 5 are promising except for "led7" dataset where the prediction accuracy has decreased as shown in Table 4.16. The resaon for this low accuracy for the "led7" dataset is that it contains 7 attributes and 10 classes. The class distribution is evenly distributed and each attribute is linked with all 10 classes. When LC algorithm applies the minority rule in rule generation phase, it removes all the ruleitems that are with the minority class values. So "9" out of "10" ruleitems are deleted with same attribute value but differenet classes. This has led to the poor performance in terms of accuracy of LC in case of "led7" dataset because of the removal of some potential significant rules. The critical analysis has found that the LC algorithm did not perform well in term of prediction accuracy for multiple class problems with evenly distributed classes among all the items in the dataset.

The prediction accuracy as shown in Figure 5.11 and Table 5.4 are studied in depth to critically analyze the reasons behind these results. It is found that as C4.5 algorithm works by finding the root value, the root value found in case of "*HTML and JavaScript Base Dataset*" is the attribute value of "*on_mouseover*" and all the leaves are constructed by appending other attribute values and '7' rules are generated. In the PART algorithm the numbers of rules produced are '4' for the same dataset. Referring to Figures 5.8 and 5.9 and looking at the rules produced by LC algorithm it was found that LC is able to explore rules that are not found by the PART and C4.5 like "<*RightClick --False* and *popUpWindow– HIGH* = phishing>" and " <*RightClick—False* and *Redirect--Low* = Phishing>". These different correlations found in LC have contributed in the higher prediction accuracy result in the "*HTML and JavaScript dataset*".

While analyzing the results of the "*Abnormal Base Dataset*" it is found that the prediction accuracy is the lowest for the LC algorithm. The in depth exploring of the results in Figure 5.7 and 5.9 has revealed that both classification approaches of C4.5 and PART has produced '7' rules each. But only two rules, if <*SFH =HIGH* then *Phishing*> and if *SFH = MODERATE* and *Request_URL= MODERATE* then website is *Ligitimate* are found for both approaches and rest of the '5' rules are different. The prediction accuracy is 95.44% that is higher in PART than C4.5's 95.24%. It means that the rest of the '5' rules that are different in composition have contributed in the difference found in the results of prediction accuracy. When the results are analyzed in the LC algorithm, it is noticed that LC algorithm have produced some rules that have generated more errors hence are responsible for the lower accuracy. The rules that are not found in PART and C4.5 are < *URL_of_Anchor –HIGH, SFH –HIGH* → *Phishing*> and < *URL_of_Anchor –HIGH, SFH –HIGH* → Phishing >, have shown '4' errors each and have contributed to the lower prediction accuracy result in *"Abnormal Base Dataset"* for LC algorithm.

The new novel prediction method of LC that uses group of rules prediction method has also contributed to the better results in UCI datsets as indicated in Table 4.17. So it is concluded that in the future application of the algorithm it is advised to use the group of rules prediction method for higher prediction accuracy.

The reduction in the number of frequently generated ruleitems in phishing datasets used is due to the factors of looking at the class labels while merging of itemsets at each iteration and use of minority rule in training phase of LC algorithm. This also reduces the time needed to generate all CARs and it is demonstrated in chapter 4. It shows that the proposed LC algorithm can be used in the datasets that are large and appending and can be effective in the extraction of rules from the data streams. As the data of phishing websites are growing on daily basis, every minute or hour there is a new phishing website emerging and the phishing attacks are proving to be damaging, need for an efficient and effective technique to predict the phishing website is the need of the day and our proposed LC algorithm will serve the purpose. It is concluded from the results as in Figure 5.11 that the LC has shown better prediction accuracy than the other bench-mark algorithms of classification and AC.

6.4 Summary of the Chapter

The in depth analysis of the results is carried out in this chapter considering all the findings of Chapter 4 and 5. The results generated by LC, CBA and other classification approaches are studied in terms of search space, itemset merging, execution time, memory usage and prediction accuracy. The reasons behind all the experimental results are explored in depth and analyzed critically. It was found that the LC approach works well and reduces the search space and execution time for all datasets used except few exceptions and the application of LC algorithm has significant results in terms of prediction accuracy when used to detect phishing and compared with CBA, PART, C4.5 and Naïve Bayes approaches. In the next chapter the conclusions will be drawn and the future directions will be highlighted.

Chapter 7

Conclusions and Future Directions

7.1 Conclusions

In this thesis the training, pruning and prediction phases are investigated in AC data mining. The outcome is two novel methods, one a training method in the rule generation phase and second is a new prediction method. These methods have been implemented in a novel AC algorithm called LC. The LC algorithm can be applied to both binary and multi-class benchmark data sets. The effective use of modified version of CBA algorithm LC is used effectively in data of features selected from the phishing and legitimate websites. The training phase of the LC algorithm enhances the efficiency and performance of AC with reference to decrease in number of rules generated in classifier, training time, and memory usage as demonstrated in Chapter 4.

The rule generation phase of the AC algorithms often do itemsets merging regardless of class labels while training phase of LC only considers the class labels of the disjoint candidate ruleitems in each iteration when performing merging operation and uses a minority rule in the training phase of LC that only selects a ruleitem with the highest frequency class, to reduce the number of frequent rules and candidate rules. The rule ranking phase of the LC implements the criteria of confidence, support, cardinality and the frequency of the class to rank rules. The training, rule ranking and pruning phases of the LC builds the classifier and reduces the number of generated rules in the final classifier solving one important problem in AC approach which is the exponential growth of rules. Moreover, new prediction method has been implemented and evaluated in the LC algorithm that takes the advantage of employing all related rules to the test case in the class assignment decision. This improves the classification accuracy of the classifiers over current AC mining algorithms that utilize single rule for prediction as demonstrated in Chapter 4. Lastly the LC algorithm has been applied to a real world emerging problem called phishing in which a model is designed to extract related and significant features of legitimate and phishing websites. Then, LC has been used to mine important knowledge from the features data sets that can on the fly detect the phishing websites. This has been validated in Chapter 5 where

the contrast is demonstrated of the LC algorithm with well know classification and AC algorithms with respect to accuracy, number of rules produced and training phase efficiency.

In the section below the thesis contributions will be discussed in details.

7.1.1 Issue 1: An Associative Classification Algorithm for Emails and Website Prediction

Phishing is an issue of serious concern all over the world. Stealing of internet user's private information has costed millions of pounds to individuals and banks. Detecting phishing is the need of the time. Many machine learning and classification techniques are been used to detect phishing. Our thesis presents the introduction of an AC approach to be applied on phishing data. Our contribution is not just the application of our LC on phishing data but also the model and formulation of rules to extract the most significant features in the phishing data pool, our thesis demonstrates successfully the extraction process of these features based on some rules.

The problem of automatically categorizing the websites and emails to phishing or legitimate is successfully demonstrated in the thesis. The development of the proposed algorithm LC described in chapter 4 have shown to accurately classify the phishing data sets as compared to other classification approaches like C4.5, PART, Naïve Bayes and AC approaches like CBA. The LC algorithms have shown better execution time as it has produced less candidate itemsets and subsequently produced less no of CARs and classifier size is also less than the traditional and most effective AC algorithm of CBA. The modifications made in the proposed algorithm of LC in its training and prediction phases have contributed to its effective use in information security domain of phishing in websites. The experimental results conducted on '4' extracted data sets from phishing and legitimate websites data pool have demonstrated that our LC new algorithm outperformed classification algorithms like C4.5, PART and Naïve Bayes on prediction accuracy on '2' data sets and showed highly competitive results for other '2' data sets in terms of accuracy measure, as demonstrated in Chapter 5 and at the same time out performed AC approach of CBA in '3' data sets in %age accuracy (see Chapter 5). It is concluded that LC algorithm should be used to detect phishing or to predict the category of a website as legitimate for fake.

7.1.2 Issue 2: Efficiency of Training Phase of AC

There are several AC algorithms like CBA (Liu et al., 1998), CMAR (Li et al., 2001), ARC-BC (Antonie and Zaïane, 2003),2-PS (Qian et al., 2005) and ACN (Gourab Kundu et al., 2008) uses the Apriori candidate generation property or Frequent Pattern Growth technique to generate candidate ruleitems in the training or the rule generation phase of the AC. In these approaches candidates are joined at each iterations irrespective of looking at the class labels of the itemsets. The rule discovery step consumes majority of the computational time of all the AC algorithms. We have introduced a new fast and effective method in our thesis that considers the class labels of the itemsets before merging them at each iteration to generate candidate ruleitems. Our LC algorithm has implemented a minority rule in each iteration which removes the less frequent class labels with the ruleitem if a ruleitem is associated with more than one class and selects only the ruleitem which is associated with the most frequent class.

Our new training method is proposed in LC algorithm, which discovers all the frequent ruleitems 40.06% faster than the CBA algorithm for ‘14’UCI repository data sets while reducing the number of merging at each iteration very significantly as demonstrated in Chapter 4 and 5, when experiments are conducted against CBA for ‘14’ benchmark problems and ‘4’ security data sets we considered. As our LC algorithm generates less number of candidate ruleitems and reduced number of rules in final classifier, as evaluated and demonstrated in the results section of Chapter 4 and 5 and hence experimental results have shown that physical memory used by the new approach is less as compared to CBA for ‘17’ problems considered.

7.1.3 Issue 3 & 4: Prediction Based on Group of Rules and Rule Ranking

Since most of the AC algorithms like CBA, MMAC, MCAR, ACCF, ACCR, CACA and ACN use single rule prediction method to predict the accuracy of the resultant classifier on test data. Our new LC algorithm uses a multi rule approach that considers all rules that matches the test instance in the decision process of assigning a class label. Our new prediction method calculates the confidence values of all classes in the group of rules that have correctly classified a test instance, and selects a class with the highest average confidence value.

The results of our experiments for single rule prediction and our approach that uses multi rule prediction is investigated for our LC algorithm that showed improvement in the accuracy of

relatively large datasets, as described in Chapter 4, that contains more number of rules in their classifier and demonstrated same results for accuracy for small data sets when 13 UCI datasets are considered when compared with CBA.

Our algorithm LC have used a much better rule ranking procedure that takes care of all the possibilities of the counts of support, confidence, cardinality and class frequencies. Hence it has produced better results for all the data sets of UCI and Phishing data sets.

7.1.4 Issue 5: Experimental Study on UCI and Phishing Data Sets

A thorough investigation of our LC and other classification and AC algorithms are carried out using the data sets from UCI repository and also on the data generated by our methods relating phishing in websites. The experiments on both the type of data sets have shown that our LC approach is adaptable to different types of data, as in Chapter 4 and 5.

7.2 Future Directions

7.2.1 Phishing in Mobile Applications

The taxonomy of the phishing attack on mobile devices is analyzed and categorized as Bluetooth, SMS, Vishing and Mobile Web application. The mobile application downloads on smart phones and android phones are in millions. 5% of the Android application users click on the phishing websites. The people are using mobile to access the financial institutions and the banks. Due to the vast availability of free application and games download, mobile users are very susceptible to the phishing attacks. Research work is carried out by (Foozy et al., 2013) has discussed the taxonomy of mobile phishing attacks. The comparison between the six detection techniques like Content Based (Peizhou et al., 2008 and Yoon et al., 2010), Blacklist (Singh, 2011) and (Mahmoud and Mahfouz, 2012), Whitelist (Mahmoud and Mahfouz, 2012), Hotspot and Gaussian Mixture Model (Chang and Lee, 2010) are studied. The mobile user market is growing and so as the need to develop accurate prediction models to detect phishing in mobile technology and applications.

7.2.2 Distributed Learning in AC

The data is collected in most of the applications on daily basis and so the data sets have grown enormously. The data operations of addition, deletion and updation are going on for almost all existing data sets. To extract the desired outcome all the existing AC algorithms scan the complete training data set at least one time to reflect the changes in the data. The cost of I/O operations and CPU time, when the database is scanned at the time of any change in the data set to update the rules generated previously, is very high and the data sets are growing as well. Incremental AC algorithms that can keep track of last results and only uses the records updated may prove to be an effective and efficient approach and can lead to enormous saving in computational times.

It is believed that incremental AC mining is a challenging problem in data mining, which has not carefully studied. Further, the key success to solve this problem is to determine the frequent *ruleitems* that overlap between the original training dataset and the records, which have been updated regardless whether the operation is insert, delete or edit. Finally, incremental mining in AC framework is a promising approach, which can be applied to dynamic real world applications where training datasets are updated frequently.

There has been some research work on incremental association rule mining algorithms, i.e. (Zhou and Ezeife, 2001; Valtchev, et al., 2002; Valtchev, et al., 2003). These can be considered as a starting point for incremental AC mining. For example, an incremental association rule mining algorithm called Maintenance Association Rule with Apriori Property (MAAP) (Zhou and Ezeife, 2001), has been presented in 2001. This algorithm efficiently generates incremental rules from an updated database using Apriori candidate generation step (Agrawal and Srikant, 1994). MAAP computes high level frequent n-itemsets and then starts producing all lower level *n*-1, *n*-2, …, 1 frequent itemsets. This approach decreases the processing overhead for generating some of the low-level frequent itemsets that have no chance of being frequent in the updated database. Thus, the key feature of MAAP is the downward closure property presented in Apriori.

7.2.3 Noise in Source Databases

A classifier in data mining is constructed from labelled data records, and later is used to forecast classes of previously unseen data as accurately as possible. Training data set may contain noise, including, missing or incorrect values inside records. One has to think carefully about the importance of missing or incorrect values in training datasets. As a result, only human experts in the application domains used to generate the datasets can make an implicit assumption about the significance of missing or invalid values.

In data mining and machine learning communities, several classification algorithms have been proposed, where most of them produce classifiers with an acceptable error rate. However, these algorithms assume that all records in the training or even test data collection are complete and no missing data are present. When training/test datasets suffer from missing attribute values or incomplete records, classification algorithms often produce poor classifiers with reference to prediction accuracy. This is due to that these algorithms tend to tailor the training dataset too much.

In real world applications, it is common that a training or test data contains attribute with missing values. For instance, the "labor" and "hepatid" datasets published in the UCI data repository contain many missing records. Thus, it is imperative to build classifiers that are able to predict accurately the classes for test datasets with missing attribute values. These classifiers are normally called robust classifiers. Unlike traditional classifiers, which assume that the test data is complete, robust classifiers deal with existing and non-existing values in test data.

There have been some solutions to avoid noise in the training datasets. Naïve Bayes algorithm for instance ignores missing values during the computation of probabilities, and thus missing values have no effect on the prediction since they have been omitted. Although omitting missing values may be not the ideal solution since these unknown values may provide a good deal of information. Other classification techniques like CBA assume that the absence of missing values may be of some importance, and therefore they treat them as other existing known values in the training data. However, if this is not the case, then missing values should be treated in a special way rather than just consider them as other possible values that the attribute might take (Witten and Frank, 2000). Decision tree algorithms such as C4.5 and C5 deal with missing values using probabilities, which are calculated from the frequencies of the different values for an attribute at a particular node in the decision tree.

The problem of dealing with unknown values inside datasets has not yet been explored well in traditional classification or AC approach.

Appendix A

Class Form.cs

Main form class for the programme from where to control all the input and options and click the Process button.

Main Methods:

DoMining(double support_min, int support_count, double confidence_min, ArrayList Table_Data_Main, long start)

Purpose:

Do mining of the main file for processing

Input:

Support value

Support count

Confidence value

Test data

DoPrediction(DataTable dtTrainingData, string fileRuleItemSet)

Purpose:

Do prediction data coverage of the data

Input:

Training data

Name of file to do prediction on

DoPruning(string fileRuleItemSet, DataTable trainingData)

Purpose:

Do prunning of the data

Input:

Name of file for prunning

Trainning data

Helper Functions

ArrayList ConvertList2ArrayList(List<List<string>> lsArray)

Purpose:

Helping Function: Convert the list of string to arraylist

Input:

List of strings

Output:

Converted arraylist

List<List<string>> ConvertArrayList2List(ArrayList tmpArray)

Purpose:

Helping Function: Convert the array list to list of strings

Input:

Array list to convert

Output:

Converted list of strings

List<T>[] Partition<T>(List<T> list, int totalPartitions)

Purpose:

Dividing the list of data to partitions for 10 Fold cross validation

Input:

List for partition

Number of partitions

Output:

Partitioned data after processing

Class Candidate_Items_Generation.cs

Class to handle operations for item set generation, frequent items, etc.

Main Methods:

ArrayList Candidate_itemset_Gen(ArrayList Table_Columns)

Purpose:

This generates the itemset with the input of array list of the data

Input:

The array list of the input data

Output:

Array list of output data

ArrayList Candidate_1_Rule_item_generation(ArrayList Table_Columns)

Purpose:

Function for candidate rule item generation with the array list of data

Input:

Array list of data

Output:

Array list of output data

ArrayList Candidate_1_Rule_Prunning(ArrayList Candidate_1_Rule_itemset_Pruned)

Purpose:

Rule prunning of the data input from the array list

Input:

Array list of data to rule prune

Output:

Array list of output data

ArrayList **Confidence_support_calculations(ArrayList Prunned_Candidate_1_array, double conf_percentage, double supp_percentage, int s_count)**

Purpose:

Calculate support and confidence of the prunned input data

Input:

Array list of prunned data

Confidence percentage value

Support percentage value

Support count value

Output:

Array list of output data

ArrayList Candidate_1_Rule_Itemset_Rest(ArrayList frequent_Rule_itemset, ArrayList Table_Columns)

Purpose:

Calculate rules itemset rest data

Input:

Array list of rule itemset data

Columns of the data in array list form

Output:

Array list of output data

Class DataCoverageAlgo

Class for data coverage algo and other operations.

Main Methods:

void dataCoverage(DataTable rulesTbl, DataTable trainTbl)

Purpose:

This function implements the data coverage algorithm and also picks one rule from the rule table and search all the items verified by it in the training data table and take action acoordingly

Input:

Table for the rules data

Table for trainning data

Boolean verifyTrainingData(Array arrRuleItems, Array arrTrainItems, out bool isLastRowChecked)

Purpose:

This function compares a rule and data row item by item to check the validity of the rule for the s pecified data row

Input:

Array of rules data

Array of trainning data

To check last row or not

Output:

True if everthing runs fine, False if there is error for verifying data

Class DataCoverageAlgoPrediction

Main Methods:

DataTable dataCoverage(DataTable rulesTbl, DataTable testTbl, string defaultClass, Boolean bSingleRule)

Purpose:

This function implements the Prediction Method algorithm. This algo picks one rule
from the rule table and search all the items verified by it in the test data table
and take action accordingly

Input:

Rules table data

Test table data

Default class name

Single rule or All rules

Output:

Returns data table after processing the input data

int computeErrors(DataTable testTbl, DataTable trainTbl)

Purpose:

Calculate errors of the test data abd the trainning data

Input:

Test table data

Trainning table data

Total number of errors counted

string verifyTrainingData(Array arrRuleItems, Array arrTestItems)

Purpose:

This function compares a rule and data row item by item to check the validity of the rule for the s pecified data row

Input:

Rules data array

Test data array

Output:

Returns null if failed, else the string of the failed array rule

Bibliography

[1] Abu-nimeh S., Nappa D., Wang X. and Nair S. (2007). A Comparison of Machine Learning Techniques for Phishing Detection. *Neural Networks*.

[2] Abu-nimeh S., Nappa D., Wang X. and Nair S. (2009). Hardening Email Security via Bayesian Additive Regression Trees. *Machine Learning*, (February).

[3] Aburrous M., Hossain M. A., Dahal K. and Thabtah F. (2010). Intelligent phishing detection system for e-banking using fuzzy data mining. *Expert Systems with Applications: An International Journal*, (pp. 7913-7921).

[4] Aburrous M., Hossain M. A., Dahal K. and Thabtah F. (2010). Predicting Phishing Websites using Classification Mining Techniques with Experimental Case Studies. *In proceedings of the 7th International Conference on Information Technology*. (pp. 176-181). Las Vegas, Nevada, USA.

[5] Agrawal R., Amielinski T. and Swami A. (1993). Mining association rule between sets of items in large databases. Proceedings of the ACM SIGMOD International Conference on Management of Data, (pp. 207-216). Washington, DC.

[6] Agrawal R. and Srikant R. (1994). Fast algorithms for mining association rule. Proceedings of the 20th International Conference on Very Large Data Bases (pp. 487-499), Santiago, Chile.

[7] Ajlouni M., Hadi E., and Alwedyan J. (2013). Detecting Phishing Websites Using Associative Classification. *European Journal of Business and Management*, *5*(15), 36-40.

[8] Ali K., Manganaris S. and Srikant R. (1997). Partial classification using association rules. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, (pp. 115-118), Newport Beach, CA.

[9] Al-Momani A. A. D., Wan T., Al-Saedi K., Altaher A., Ramadass S., Manasrah Ahmad., Melhiml L. and Mohammad A. (2011). An Online Model on Evolving Phishing E-mail Detection and Classification Method. *Journal of Applied* Sciences, 11: 3301-3307.

[10] Antonie M. and Zaïane O. (2004). An associative classifier based on positive and negative rules. Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (pp. 64 - 69), Paris, France.

[11] Antonie M., Zaïane O. R. and Coman A. (2003). Associative Classifiers for Medical Images, Lecture Notes in Artificial Intelligence 2797, Mining Multimedia and Complex Data, (pp. 68-83), Springer-Verlag.

[12] Antonie M. and Zaiane O. (2002). Text Document Categorization by Term Association, Proceedings of the IEEE International Conference on Data Mining (ICDM '2002), (pp.19-26), Maebashi City, Japan, December 9 - 12.

[13] Apache Software Foundation. (2006) Spamassassin public corpus, http://spamassassin.apache.org/publiccorpus/.

[14] Breiman L. (2001) Random forests. Mach. Learn., 45(1):5–32.

[15] Baralis E., Chiusano S. and Garza P. (2008). A Lazy Approach to Associative Classification. IEEE Trans. Knowl. Data Eng. 20(2): 156-171.

[16] Baralis E., Chiusano S. and Garza P. (2004). On support thresholds in associative classification. Proceedings of the 2004 ACM Symposium on Applied Computing, (pp. 553-558). Nicosia, Cyprus.

[17] Baralis E. and Torino P. (2000). A lazy approach to pruning classification rules. Proceedings of the 2002 IEEE ICDM'02, (pp. 35). Maebashi City, Japan.

[18] Bayardo, R., and Agrawal, R. (1999) Mining the most interesting rules. *Proceedings of the 5th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, (pp. 145-154). August, 1999.

[19] Blackmore K. and Bossomaier T. (2002) Comparison of C5 and J48.PART algorithms for missing persons profiling. *Proceedings of the ICITA '2*. Bathurst, NSW, Australia.

[20] Breiman L., Friedman J., Olshen R., and Stone C. (1984) Classification and regression trees. *Wadsworth International Group*, Belmont, CA.

[21] Brin S., Motwani R., Ullman J. and Tsur S. (1997). Dynamic itemset counting and implication rules for market basket data. Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, (pp. 265-276). Tucson, Arizona, USA.

[22] Cendrowska J. (1987). PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, Vol.27, No.4, (pp. 349-370).

[23] Chang J. and Lee H. (2010). Voice phishing detection technique based on minimum classification error method incorporating codec parameters. *Signal Processing, IET,* vol. 4, (pp502-509).

[24] Clark P. and Boswell R. (1991). Rule induction with CN2: Some recent improvements. Proceedings of the Fifth European Working Session on Learning, (pp. 151-163). Berlin, Germany.

[25] Cohen W. (1995). Fast effective rule induction. Proceedings of the 12th International Conference on Machine Learning, (pp. 115-123). CA, USA.

[26] Dong G., Zhang X., Wong L. and Li J. (1999). CAEP: Classification by aggregating emerging patterns. Proceedings of the Second Imitational Conference on Discovery Science, (pp. 30-42). Tokyo, Japan.

[27] Duda R. and Hart P. (1973). Pattern classification and scene analysis. John Wiley & son.

[28] Elhalees A. (2006). Mining Arabic Association Rules for Text Classification In the proceedings of the first international conference on Mathematical Sciences. Al-Azhar University of Gaza, Palestine, (pp.15 -23).

[29] Elmasri R. and Navathe S. (1999). Fundamentals of database systems, Fourth Edition, Addison-Wesley.

[30] Enas M., Houby E., and Marwa S. (2012). Using Associative Classification for Treatment Response Prediction. *Journal of Applied Sciences Research*, 8(10): 5089-5095, 2012 ISSN 1819-544X.

[31] Fayyad U. and Irani K. (1993). Multi-interval discretisation of continues-valued attributes for classification learning. Proceedings of IJCAI, (pp. 1022-1027). Chambéry, France.

[32] Fayyad U., Piatetsky-Shapiro G., Smith G. and Uthurusamy R. (1998). Advances in knowledge discovery and data mining. AAAI Press.

[33] Fette I. Sadeh and Tomasic A. (2007). Learning to detect phishing emails. In Proceedings of the 16th International World Wide Web Conference (WWW 2007).

[34] Foozy, C., Rabiah A., and Mohd F. (2013). Phishing Detection Taxonomy for Mobile Device. *IJCSI International Journal of Computer Science Issues*, Vol. 10, Issue 1, No 3.

[35] Frank, E., and, Witten, I. (1998) Generating accurate rule sets without global optimisation. *Proceedings of the Fifteenth International Conference on Machine Learning*, (pp. 144–151). Morgan Kaufmann, Madison, Wisconsin.

[36] Friedman N., Geiger D. and Goldszmidt M. (1997) Bayesian network classifiers. In *Machine Learning* 29: (pp. 131—163).

[37] Freitas A. (2000) Understanding the crucial difference between classification and association rule discovery. ACM SIGKDD Explorations Newsletter, 2(2000): 65 -69.

[38] Freund Y. and Schapire R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci., 55(1):119–139.

[39] Furnkranz J. (1999) Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3-54.

[40] Furnkranz J. and Widmer G. (1994). Incremental reduced error pruning. Proceedings of the Eleventh International Machine Learning Conference, (pp. 70-75). New Brunswick, NJ.

[41] Guang X., Jason O., Carolyn R., Lorrie P and C. (2011). CANTINA+: A Feature-rich Machine Learning Framework for Detecting Phishing Web Sites. *ACM Transactions on Information and System Security*, pp. 1-28.

[42] Hamid I., Abawajy J., and Kim Y. (2013). Using Feature Selection and Classification Scheme for Automating Phishing Email Detection. *Studies in Informatics and Control*. ISSN 1220-1766, vol. 22 (1), (pp. 61-70).

[43] Han J., Pei J. and Yin Y. (2000). Mining frequent patterns without candidate generation. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, (pp. 1-12). Dallas, Texas.

[44] Hao J. and Hamiez J. (2001). Solving the sports league scheduling problem with Tabu search. *Lecture Notes in Artificial Intelligence 2148*: 24-36. Springer 2001.

[45] Henderson M., Henderson P., and Keirnan C. (2000). Missing persons: incidence, issues and impacts. Trends & Issues in Crime and Criminal Justice, vol. 144.

[46] Holsheimer M., Kersten M., Mannila H. and Toivonen H. (1995). A prospective on databases and data mining. Proceedings of First International Conference on Knowledge Discovery and Data Mining (KDD'95), (pp. 150-155). Montreal, Canada.

[47] Holte R. (1993) Very simple classification rules perform well on most commonly used datasets. Machine Learning, vol. 3, (pp. 63-91).

[48] Irani D., Webb S., Giffin J., and Pu C. (2008). Evolutionary Study of Phishing. *Security*.

[49] Islam M. R., Abawajy J., and Warren M. (2009). Multi-tier Phishing Email Classification with an Impact of Classifier Rescheduling. *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, 789-793. IEEE. doi:10.1109/I-SPAN.2009.142.

[50] Jabbar M., Deekshatulu L., and Chandra P. (2013). Heart Disease Prediction System using Associative Classification and Genetic Algorithm. *arXiv preprint arXiv:1303.5919*.

[51] Jiang Z., and Karypis G. (2013). AREM: A novel associative regression model based on EM algorithm. *In Proceedings of the 17th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. PAKDD'13, Springer-Verlag.

[52] Joachims, T. (1998) Text categorisation with support vector machines: Learning with many relevant features. *Proceedings of Tenth European Conference on Machine Learning*, (pp. 137-142).

[53] Kasabov N. and Song Q. (2002). Denfis: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. Fuzzy Syst., 10:144-154.

[54] Khan A. (2013). Preventing Phishing Attacks using One Time Password and User Machine Identification. *International Journal of Computer Applications* (0975 – 8887). Volume 68-No.3.

[55] Kundu G., Islam M., Munir S. and Bari M. (2008). ACN: An Associative Classifier with Negative Rules, Computational Science and Engineering, IEEE International Conference on, vol. 0, no. 0, (pp. 369-375), 2008 11th IEEE International Conference on Computational Science and Engineering.

[56] Lakshmi K. and Reddy C. (2012) Compact Tree for Associative Classification of Data Stream Mining. *IJCSI International Journal of Computer* Science Issues, Vol. 9, Issue 2, No 2.

[57] Lewis D. (1998b). Naive (Bayes) at Forty: the independence assumption in information retrieval. Proceedings of the 10th European Conference on Machine Learning, (pp. 4-15). Chemnitz, Germany.

[58] Li B., Li H., Wu M. and Li P. (2008). Multi-label Classification based on Association Rules with Application to Scene Classification, icycs, (pp.36-41), 2008 The 9th International Conference for Young Computer Scientists.

[59] Li W. (2001). Classification based on multiple association rules. M.Sc. Thesis. Simon Fraser University.

[60] Li W., Han J. and Pei J. (2001). CMAR: Accurate and efficient classification based on multiple-class association rule. Proceedings of the ICDM'01, (pp. 369-376). San Jose, CA.

[61] Li J., Zhang X., Dong G., Ramamohanarao K. and Sun Q. (1999). Efficient mining of high confidence association rules without support thresholds. In Zytkow, J., and Rauch, J. editors, *PKDD99*, volume 1704 of *LNAI*, (pp. 406- 411). Prague, Czech Republic.

[62] Li X., Qin D. and Yu C. (2008). ACCF: Associative Classification Based on Closed Frequent Itemsets. FSKD (2) 2008: 380-384.

[63] Lin J., and Dunham M. (1998). Mining association rules: Anti-Skew algorithms. *Proceedings of the Fourteenth International Conference on Data Engineering*, (pp. 486-493).

[64] Liu B., Hsu W. and Ma Y. (1999). Mining association rules with multiple minimum supports. Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (pp.337-341). San Diego, California.

[65] Liu B., Hsu W. and Ma Y. (1998). Integrating classification and association rule mining. Proceedings of the KDD, (pp. 80-86). New York, NY.

[66] Liu B., Ma Y. and Wong C-K. (2001). Classification using association rules: weakness and enhancements. In Vipin Kumar et al., (eds) Data mining for scientific applications. (2001): 591.

[67] Liu B., Ma Y. and Wong C-K. (2000). Improving an association rule based classifier. Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, (pp. 504-509). Lyon, France.

[68] Liu B., Ma Y., Wong C-K. and Yu. P. (2003). Scoring the data using association rules. Applied Intelligence, 18(2003): 119-135.

[69] Liu Y., Yang Y. and Carbonell J. (2002). Boosting to correct inductive bias in text classification. *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, (pp. 348-355). McLean, VA.

[70] Ma L., Ofoghi B., Watters P., and Brown S. (2009). Detecting Phishing Emails Using Hybrid Features. *Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, 493-497. IEEE. doi:10.1109/UIC-ATC.2009.103.

[71] Mahmoud T., and Mahfouz A.M. (2012). SMS Spam Filtering Technique Based on Artificial Immune System. *IJCSI International Journal of Computer Science Issues,* vol. 9, 2012.

[72] Meretakis D., Fragoudis D., Lu H., and Likothanassis S. (2000) Scalable association-based text classification. *Proceedings of the Ninth International Conference on Information and Knowledge Management*, (pp. 5-11). McLean, Virginia, United States.

[73] Meretakis D., and Wüthrich B. (1999) Extending naïve Bayes classifiers using long itemsets. *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 165 – 174). San Diego, California.

[74] Merz C. and Murphy P. (1996). UCI repository of machine learning databases. Irvine, CA, University of California, Department of Information and Computer Science.

[75] Mitchell M. (1997). Machine Learning, chapter IV, Artificial Neural Networks, (pp. 81-127). WCB/McGraw-Hill, New York, New York.

[76] Miyamoto D., Hazeyama H. and Kadobayashi Y. (2008). "An Evaluation of Machine Learning-based Methods for Detection of Phishing Sites," *Australian Journal of Intelligent Information Processing Systems,* pp. 54-63.

[77] Niu Q., Xia S. and Zhang L. (2009). Association Classification Based on Compactness of Rules, wkdd, (pp.245-247), 2009 Second International Workshop on Knowledge Discovery and Data Mining.

[78] Park J., Chen M. and Yu P. (1995). An effective hash-based algorithm for mining association rules. *Proceedings of the ACM SIGMOD*, (pp. 175-186). San Jose, CA.

[79] Purkait S. (2013). DHCP-Enabled LAN Prone to Phishing Attacks. *The IUP Journal of Information Technology*, Vol. IX, No. 1, (pp. 24-40).

[80] Peizhou H. (2008). A Novel Method for Filtering Group Sending Short Message Spam in Convergence and Hybrid Information Technology. ICHIT '08. International Conference (pp. 60-65).

[81] Qian T., Wang Y., Long H. and Feng J. (2005). 2-PS based associative text classification, in: Proceedings of the Seventh International Conference on Data Warehousing and Knowledge Discovery, (pp. 378-387).

[82] Quinlan J. (1998). Data mining tools See5 and C5.0. Technical Report, RuleQuest Research.

[83] Quinlan J. (1993). C4.5: Programs for machine learning. San Mateo, CA: Morgan Kaufmann.

[84] Quinlan J. and Cameron-Jones R. (1993). FOIL: A midterm report. Proceedings of the European Conference on Machine Learning, (pp. 3-20), Vienna, Austria.

[85] Quinlan J. (1988). Decision trees and multi-valued attributes. In Hayes, J., Michie, D., and Richards J., (eds), Machine Intelligence, 11(1988): 305-318.

[86] Quinlan J. (1987). Simplifying decision trees. International journal of man-machine studies, 27(1987): 221-248.

[87] Quinlan J. (1986). Induction of decision trees. Machine Learning, 1(1986): 81 – 106.

[88] Quinlan J. (1979). Discovering rules from large collections of examples: a case study. In D. Michie, editor, *Expert Systems in the Micro-electronic Age*, (pp.168—201). Edinburgh University Press, Edinburgh.

[89] Rissanen J. (1985). The minimum description length principle**.** In: Kotz S., Johnson N. (Eds.), *Encyclopedia of Statistical Sciences*. Vol. 5. John Wiley and sons, New York, pp. (523–527).

[90] Samir E. (2013). Artificial Immune System for Associative Classification. *Doctoral Dissertations.* Paper 22. http://digitalcommons.uconn.edu/dissertations/22.

[91] Sánchez-Moreno D., Gil B., and Moreno M. (2013). TV-SeriesRec: A Recommender System Based on Fuzzy Associative Classification and Semantic Information. In *Trends in Practical Applications of Agents and Multiagent Systems* (pp. 201-208). Springer International Publishing.

[92] Sanglerdsinlapachai N. and Rungsawang A.(2010). "Using Domain Top-page Similarity Feature in Machine Learning-based Web," in *Third International Conference on Knowledge Discovery and Data Mining.*

[93] Savasere A., Omiecinski E. and Navathe S. (1995). An efficient algorithm for mining association rules in large databases. Proceedings of the 21st conference on Very Large Databases (VLDB '95), (pp. 432 - 444). Zurich, Switzerland.

[94] Schapire R., Singer Y. and Singhal A. (1998). Boosting and rocchio applied to text filtering. In Proc. 21st Annual Intl. ACM SIGIR Conf. on R&D in Information Retrieval.(pp. 215-223)

[95] Shenoy P., Haritsa J., Sudarshan S., Bhalotia G., Bawa M. and Shah D. (2000). VIPER: A vertical approach to mining association rules. Proceedings of the ACM SIGMOD International Conference on Management of Data, (pp 22-33). Dallas, Texas.

[96] Singh D. (2011). Telephony Fraud Prevention US Patent.

[97] Snedecor W. and Cochran W. (1989). Statistical Methods, Eighth Edition, Iowa State University Press.

[98] Sophie G. P., Gustavo G. and Maryline L. (2011) "Decisive Heuristics to Differentiate Legitimate from Phishing Sites," in *2011 Conference on Network and Information Systems Security*

[99] Sparck K. (1972). A statistical interpretation of term specificity and its application in retrieval," Journal of documentation, Vol.28, No.1, (pp. 11-21).

[100]Tang Z. and Liao Q. (2007). A New Class Based Associative Classification Algorithm. IMECS 2007: 685-689.

[101]Tanbeer S. K., Ahmed C. F., Jeong B. and Lee Y. (2008). CP-tree: a tree structure for single-pass frequent pattern mining. *In Proc. of PAKDD*, Lect Notes Artif Int, 1022-1027.

[102]Thabtah F., Cowling P. and Hamoud S. (2006): Improving Rule Sorting, Predictive Accuracy and Training Time in Associative Classification. Journal of Expert Systems with Applications, Volume 31, Issue 2, Pages 414-426. Elsevier.

[103]Thabtah F., Cowling P. and Peng Y. (2005) MCAR: Multi-class classification based on association rule approach. Proceeding of the 3rd IEEE International Conference on Computer Systems and Applications (pp. 1-7).Cairo, Egypt.

[104]Thabtah F., Cowling P. and Peng Y. (2004) MMAC: A new multi-class, multi-label associative classification approach. Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM '04), (pp. 217-224). Brighton, UK. (Nominated for the Best paper award).

[105]Toolan F. and Carthy J. (2010). Feature Selection for Spam and Phishing Detection *Group*.

[106]Utgoff P. (1989) Machine Learning. Volume 4, Issue 2, (pp.161- 86). Kluwer Academic Publishers Hingham, MA, USA.

[107]Vapnik V. (1995). The Nature of Statistical Learning Theory, chapter 5. Springer-Verlag, New York.

[108]Vyas R., Sharma L., Vyas O. and Scheider S. (2008). Associative Classifiers for Predictive Analytics: Comparative Performance Study, ems, (pp.289-294), 2008 Second UKSIM European Symposium on Computer Modeling and Simulation, 2008.

[109]Wang K., Zhou S. and He Y. (2000). Growing decision tree on support-less association rules. Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (pp. 265–269). Boston, Massachusetts.

[110]WEKA (2000): Data Mining Software in Java: http://www.cs.waikato.ac.nz/ml/weka.

[111] Wen-Chen C., Chiun-Chich H. and Yu-Chun C. (2012). Increasing the effectiveness of associative classification in terms of class imbalance by using a novel pruning algorithm. Expert System with Applications, Volume 39, Issue17, (pp 12841-12850).

[112]Witten I. and Frank E. (2000). Data mining: practical machine learning tools and techniques with Java implementations. San Francisco: Morgan Kaufmann.

[113]Xu X., Han G. and Min H. (2004). A novel algorithm for associative classification of images blocks. Proceedings of the fourth IEEE International Conference on Computer and Information Technology, (pp. 46-51). Lian, Shiguo, China.

[114]Yearwood J., Mammadov M. and Banerjee A. (2010). Profiling Phishing Emails Based on Hyperlink Information. *2010 International Conference on Advances in Social Networks Analysis and Mining*, 120-127. IEEE. doi:10.1109/ASONAM.2010.56.

[115]Yin X. and Han J. (2003). CPAR: Classification based on predictive association rule. Proceedings of the SDM (pp. 369-376). San Francisco, CA.

[116]Yoon Y. and Lee G. (2008). Text Categorization Based on Boosting Association Rules, icsc, pp.136-143, 2008 IEEE International Conference on Semantic Computing.

[117]Yoon J.W. (2010). Hybrid spam filtering for mobile communication. *Computers & Security*, Vol. 29, (pp. 446-459).

[118] Zaïane O. and Antonie A. (2002). Classifying text documents by associating terms with text categories. Proceedings of the Thirteenth Australasian Database Conference (ADC'02), (pp. 215 - 222), Melbourne, Australia.

[119] Zaki M. and Hsiao C-J. (1999). Charm: An efficient Algorithm for closed Association Rules Mining, Technical Report, TR99-10.Computer Science Dept. Rensselaer Polytechnic Institute.

[120] Zaki M. and Gouda K. (2003). Fast vertical mining using diffsets. Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (pp. 326 – 335). Washington, D.C.

[121] Zaki M., Parthasarathy S., Ogihara M. and Li W. (1997). New algorithms for fast discovery of association rules. Proceedings of the 3rd KDD Conference. (pp. 283-286). Menlo Park, CA.

[122]Zheng Z., Kohavi R., and Mason L. (2001). Real world performance of association rule algorithms. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 401 – 406). San Francisco, California.

[123]"PhishTank," October 2006. [Online]. Available: http://www.phishtank.com/.

[124]"Millersmiles," [Online]. Available: http://www.millersmiles.co.uk/.

[125]"Yahoo Directory," [Online]. Available: http://dir.yahoo.com/.

[126]"Starting Point Directory," [Online]. Available: http://www.stpt.com/directory/.

[127]"WhoIS," [Online]. Available: http://who.is/.

[128]"Alexa the Web Information Company," [Online]. Available: http://www.alexa.com/.