

UNIVERSITY OF PRIMORSKA
FACULTY OF MATHEMATICS, NATURAL SCIENCES AND
INFORMATION TECHNOLOGIES

DOKTORSKA DISERTACIJA
(DOCTORAL DISSERTATION)

UPORABA RAZVRŠČANJA V SKUPINE NA
KLASIFIKACIJSKIH ASOCIACIJSKIH PRAVILIH ZA
TVORBO SMISELNIH IN NATANČNIH KLASIFIKATORJEV
(CLUSTERING CLASS ASSOCIATION RULES TO FORM A
MEANINGFUL AND ACCURATE CLASSIFIER)

JAMOLBEK MATTIEV

KOPER, 2020

UNIVERSITY OF PRIMORSKA
FACULTY OF MATHEMATICS, NATURAL SCIENCES AND
INFORMATION TECHNOLOGIES

DOKTORSKA DISERTACIJA
(DOCTORAL DISSERTATION)

UPORABA RAZVRŠČANJA V SKUPINE NA
KLASIFIKACIJSKIH ASOCIACIJSKIH PRAVILIH ZA
TVORBO SMISELNIH IN NATANČNIH KLASIFIKATORJEV
(CLUSTERING CLASS ASSOCIATION RULES TO FORM A
MEANINGFUL AND ACCURATE CLASSIFIER)

JAMOLBEK MATTIEV

KOPER, 2020

MENTOR: DOC. DR. BRANKO KAVŠEK
SOMENTOR: IZR. PROF. DR. JERNEJ VIČIČ

Acknowledgement

Firstly, I would like to express my sincere gratitude to my mentor Assist. Prof. Branko Kavšek for the continuous support of my Ph.D study, related research, personal life and administrative issues, for his patience, motivation, and immense knowledge. His guidance helped me during all the time of my research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my mentor, I would like to thank my co-mentor Assoc. Prof. Jernej Vičič and the rest of my thesis committee: Prof. Dunja Mladenić, Assoc. Prof. Kljun Matjaž, and Assist. Prof. Klen Čopič Pucihar for their insightful comments and encouragement, but also for the hard questions which motivated me to widen my research from various perspectives.

I would like to acknowledge the European Commission for funding the InnoRenewCoE project (Grant Agreement #739574) under the Horizon 2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Union of the European Regional Development Fund). I would like also to acknowledge the “El-Yurt-Umidi” foundation under the Cabinet of Ministers of the Republic of Uzbekistan for the financial support during my mobility time.

I am also very grateful and extend my sincere thanks to Prof. Klavdija Kutnar and Prof. Ademir Hujdurović for their valuable support. A special thanks goes to the other colleagues Aleš Oven, Tina Franca, Tanja Labus who supported me in administrative issues. I would also like to thank all my best friends, namely Cuauhtli Campos Mijangos, Sead Jahić, Leon Kopitar and others. You helped me a lot in making my time in Slovenia enjoyable both in research and personal life.

Last but not least, I would like to thank my family: my father, brother, sister and my wife for supporting me spiritually throughout the writing of this thesis and my life in general. Furthermore, I would like to express my gratitude to Prof. Gorazd Drevenšek and his wife Martina for spending their valuable time for us to organize family activities.

Dedicated to my family!

Abstract

Clustering class association rules to form a meaningful and accurate classifier

Huge amounts of data are being collected and analyzed nowadays. By using the popular rule-learning algorithms, the number of rules discovered on “big” datasets can easily exceed thousands. Thus, building accurate and compact classifiers in real-world applications is nowadays one of the crucial tasks in data mining. To produce compact, understandable and accurate classifiers, such rules have to be grouped and pruned, so that only a reasonable number of them are presented to the end user for inspection and further analysis.

Existing classification rule-learning algorithms use mainly greedy heuristic search to find regularities in datasets for classification. In recent years, extensive research on association rule mining is being done in the machine learning community. The main objective of this research is to find all rules in data that satisfy some user-specified constraints. Although the whole set of rules may not be used directly for accurate classification, effective and efficient classifiers have been built using the so-called classification association rules or class association rules (CARs) for short.

In this research work, we propose new methods that are able to reduce the number of class association rules (CARs) produced by “classical” class association rule classifiers, while maintaining an accurate classification model that is comparable (in terms of classification accuracy) to the ones generated by state-of-the-art classification algorithms.

In the preliminary step of our research, we propose two simple associative classifiers (ACs): SA (Simple Associative classifier) and J&B (Coverage-based Associative Classifier, named using the initials of the authors). The first one - SA - is used to investigate the effect of confidence (the main constraint for choosing “good” CARs) and class distribution on pruning the initial associative classifier, while the other - J&B - uses CARs coverage of examples for the same purpose. Both simple ACs prove that it is possible to reduce the number of CARs, while maintaining the classification accuracy and size of the models comparable to state-of-the-art rule-learning algorithms.

Since neither of our simple AC methods significantly outperforms state-of-the-art rule-learning algorithms in terms of model size, we propose new associative classifiers, called DC (which is built based on Direct distance measure and the method for extracting a representative class association rules (CARs) is based on cluster Center),

DDC (which is formed based on Direct distance measure and the method for extracting a representative CAR is based on Dataset Coverage) and CDC (which is formed based on Combined distance measure and the method for extracting a representative CAR is based on Dataset Coverage), in the main part of our research.

DC, DDC and CDC firstly generate "strong" class association rules - we use the term "strong" referring to CARs exceeding the two user-defined constraints: minimum support and minimum confidence. Then, the distance-based agglomerative hierarchical clustering algorithm is utilized as a post-processing step to reduce the number of rules and in the rule-selection step, we use different strategies to extract representative class association rules (based on dataset coverage and cluster center) for each cluster to produce the final compact and understandable associative classifier for each algorithm. We also discuss how overall coverage of such classifiers affects their classification accuracy.

Experimental results performed on selected datasets from the UCI ML repository show that our algorithms are able to learn classifiers containing significantly fewer rules (DC: 22, DDC: 22, CDC: 36 rules on average) than state-of-the-art rule-learning algorithms (DTNB: 457, DT: 494, c4.5: 300, PT: 302, FR: 71, RDR: 61, CBA: 74, SA: 108). On the other hand, the classification accuracy (DC: 82.5%, DDC: 83.3%, CDC: 83.8% on average) of our proposed algorithms is not significantly different from state-of-the-art rule-learners on most of the datasets.

The advantage of our proposed method is that the size of the produced classification models (measured in number of classification rules) is 2–4 times smaller on average compared to the other (selected) rule-learners, while this ratio is even bigger on datasets with higher number of examples.

Key words: Frequent Itemset, Class Association Rule (CAR), Classification, Statistical Significance Testing, Overall Coverage, Associative Classification, Agglomerative Clustering, Representative Rule.

Izvleček

Uporaba razvrščanja v skupine na klasifikacijskih asociacijskih pravilih za tvorbo smiselnih in natančnih klasifikatorjev

Dandanes se zbirajo in analizirajo ogromne količine podatkov. Z uporabo priljubljenih algoritmov za učenje pravil lahko število pravil, odkritih na teh "velikih" naborih podatkov, zlahka presežejo tudi nekaj tisoč pravil. Gradnja natančnih in kompaktnih klasifikatorjev v resničnih aplikacijah je zato v današnjem času ena ključnih nalog pri podatkovnem rudarjenju. Za izdelavo kompaktnih, razumljivih in natančnih klasifikatorjev je treba takšna pravila razvrstiti v skupine in jih porezati tako, da je končnemu uporabniku v pregled in nadaljnjo analizo predstavljeno le razumno število pravil.

Obstoječi klasifikacijski algoritmi za učenje pravil uporabljajo predvsem požrešna heuristična iskanja pri odkrivanju zakonitosti v naborih podatkov za potrebe napovedovanja. V zadnjih letih so bile izvedene obsežne raziskave na področju rudarjenja asociacijskih pravil z uporabo izčrpnih heuristik preiskovanja. Glavni cilj teh raziskav je najti vsa pravila v podatkih, ki izpolnjujejo neke omejitve, določene s strani uporabnika. Čeprav se celotne množice pravil ne uporablja neposredno za natančno klasifikacijo, so ta t.i. klasifikacijska asociacijska pravila bila uporabljena za gradnjo uspešnih in učinkovitih klasifikatorjev.

V tem raziskovalnem delu predlagamo nove metode za zmanjšanje števila klasifikacijskih asociacijskih pravil, ki jih proizvajajo "klasični" asociacijski klasifikatorji, hkrati pa naše metode ohranjajo natančen klasifikacijski model, primerljiv s tistimi modeli, ki jih generirajo najsodobnejši klasifikacijski algoritmi.

V preliminarni raziskavi najprej predlagamo dva preprosta asociativna klasifikatorja: SA (Simple Associative Classifier) - preprosti asociativni klasifikator in J&B - asociativni klasifikator, ki temelji na pokritju primerov (poimenovan je po začetnicah imen prvih avtorjev). Prvega - SA - uporabimo za preučitev učinka zaupanja (glavna omejitev pri izbiri "dobrih" klasifikacijskih asociacijskih pravil) in porazdelitve razreda na rezanje začetnega asociativnega klasifikatorja. Drugi - J&B - za isti namen uporablja pokritost primerov s klasifikacijskimi asociacijskimi pravili. Oba preprosta asociativna klasifikatorja dokazujeta, da je mogoče zmanjšati število klasifikacijskih asociacijskih pravil, hkrati pa ohraniti klasifikacijsko točnost in velikost modelov, ki je primerljiva z najsodobnejšimi algoritmi za učenje pravil.

Ker noben od predlaganih asociativnih klasifikatorjev iz preliminarne raziskave signifikantno ne prekaša najsodobnejših klasifikacijskih algoritmov glede velikosti naučenih modelov, v glavni raziskavi predlagamo nove asociativne klasifikatorje imenovane DC

(ki pri razvrščanju v skupine klasifikacijskih asociacijskih pravil uporablja direktno oddaljenost pravil, metoda za identifikacijo reprezentativnega klasifikacijskega asociacijskega pravila pa temelji na centroidih skupin), DDC (ki pri razvrščanju v skupine klasifikacijskih asociacijskih pravil prav tako uporablja direktno oddaljenost pravil, metoda za identifikacijo reprezentativnega klasifikacijskega asociacijskega pravila pa temelji na pokritosti primerov) in pa CDC (ki pri razvrščanju v skupine klasifikacijskih asociacijskih pravil uporablja kombinacijo direktne in indirektno oddaljenosti pravil, metoda za identifikacijo reprezentativnega klasifikacijskega asociacijskega pravila pa temelji na pokritosti primerov).

DC, DDC in CDC najprej generirajo “močna” klasifikacijska asociacijska pravila, nato je, v koraku postprocesiranja množice pravil, uporabljen algoritem aglomerativnega hierarhičnega združevanja v skupine za zmanjšanje števila pravil, kjer se pri izluščanju reprezentativnih klasifikacijskih asociacijskih pravil iz skupin pravil uporabljajo različne strategije (ki temeljijo tako na pokritosti podatkov kot na centroidih skupin). Prav ta reprezentativna klasifikacijska asociacijska pravila na koncu tvorijo končni, kompaktni in razumljiv asociativni klasifikator. Razpravljamo tudi o tem, kako skupna pokritost primerov takšnih klasifikatorjev vpliva na njihovo klasifikacijsko točnost.

Eksperimentalni rezultati na izbranih naborih podatkov iz repozitorija UCI ML kažejo, da se naši algoritmi lahko naučijo klasifikatorjev, ki vsebujejo manj pravil (DC: 22, DDC: 22, CDC: 36 pravil v povprečju) od najsodobnejših algoritmov za učenje pravil. Po drugi strani pa se njihova klasifikacijska točnost (DC: 82,5%, DDC: 83,3%, CDC: 83,8% v povprečju) bistveno ne razlikuje od najsodobnejših algoritmov za učenje pravil na večini podatkovnih zbirk.

Prednost predlaganih klasifikatorjev je v tem, da so števila pravil, ki jih le-ti vsebujejo, v povprečju manjša za 2 do 4 krat v primerjavi z ostalimi “dobro poznanimi” klasifikatorji, medtem ko je to razmerje še večje pri naborih podatkov z večjim številom primerov.

Ključne besede: podosta množica postavk, klasifikacijska asociacijska pravila, klasifikacija, test statistične značilnosti, skupna pokritost, asociativna klasifikacija, združevanje v skupine, reprezentativno pravilo.

Contents

List of Figures	viii
List of Algorithms	ix
List of Tables	x
1 Introduction	1
1.1 Scientific Background	1
1.1.1 Association Rules	3
1.1.2 Classification Rules	3
1.1.3 Cluster Analysis	7
1.2 Literature Review	9
1.3 Contribution to Science and Methodology	12
1.4 Content Guide	14
2 Discovery of Class Association Rules	17
2.1 Frequent Itemsets Mining Algorithms	17
2.1.1 Brute-Force (Naive) Method	18
2.1.2 APRIORI (Level-Wise) Approach	19
2.1.3 ECLAT algorithm	20
2.1.4 Frequent Pattern Tree Approach: FP-Growth algorithm	21
2.2 Class Association Rules	23
3 Associative Classification	25
3.1 Simple Associative Classification Approach (SA)	25
3.1.1 Experimental Evaluations of SA Approach	28
3.2 J&B Associative Classification Approach	31
3.2.1 Experimental Evaluations of J&B Approach	39
4 Distance Metrics	47
4.1 Indirect Distance Metrics	47
4.2 The New “Direct” Distance Metric	50
4.3 The New “Combined” Distance Metric	51

5	Identifying the Clusters of CARs	53
5.1	Partitional clustering algorithms	53
5.2	Hierarchical clustering algorithms	55
6	Identifying the Cluster Representative Class Association Rule	59
6.1	Representative CAR based on cluster center (RCC)	59
6.2	Representative CAR based on dataset coverage (RDC)	59
6.3	Final associative classifier	60
7	Experimental Evaluations and Discussion	64
7.1	Discussion of Results	71
8	Conclusion and Future Work	75
	Bibliography	76
	Index	86
	Povzetek v slovenskem jeziku	87
8.1	Uvod	87
8.2	Znanstvena izhodišča	88
8.2.1	Klasifikacijska pravila	88
8.2.2	Pogoste podmnožice in asociacijska pravila	88
8.2.3	Klasifikacijska asociacijska pravila	89
8.3	Opredelitev problema	89
8.4	Raziskovalna vprašanja, hipoteze in cilji	89
8.4.1	Definicija mere podobnosti klasifikacijskih asociacijskih pravil	90
8.4.2	Ugotavljanje skupin za klasifikacijska asociacijska pravila . . .	90
8.4.3	Ugotavljanje reprezentativnega klasifikacijskega asociacijskega pravila za posamezno skupino pravil	90
8.4.4	Eksperimentalna evalvacija	90
8.5	Rezultati	91
8.5.1	Diskusija	94
8.6	Zaključek in nadaljnje delo	97
8.7	Prispevki k znanosti	99
	Kazalo	102
	Stvarno kazalo	104

List of Figures

1.1	The data classification process.	5
1.2	A decision tree for the concept <code>buys_computer</code>	6
1.3	Represented classification models in various forms, as (a) IF-THEN rules, (b) a decision tree, (c) a neural network.	7
1.4	2-D plot of customers with respect to customer locations in a city.	8
3.1	The effect of confidence and <i>numrules</i> thresholds to the accuracy on “Nursery” dataset.	31
3.2	Associative classification working procedure	32
3.3	Confusion matrix for binary class classification	38
3.4	Confusion matrix for multi-class classification problems	39
3.5	Comparison of rule-based classification methods on the average number of rules.	44
3.6	Comparison of the J&B classifier on accuracy, precision, recall and f-measure.	45
5.1	Clustering of a set of objects based on the k-means method. (The mean of each cluster is marked by a “+”.)	55
5.2	Agglomerative and divisive hierarchical clustering on data objects {a, b, c, d, e}	56
5.3	Dendrogram representation for hierarchical clustering of data objects {a, b, c, d, e}	57
7.1	Comparison of performance of our proposed associative classification models	70
7.2	Comparison between our proposed associative classification models on accuracy for selected datasets	72
7.3	Comparison between our proposed associative classification models on size for selected datasets	73
8.1	Primerjava uspešnosti našega predlaganega asociativnega klasifikatorja.	96
8.2	Primerjava med našim predlaganim asociacijskim klasifikatorjem in ostalimi modeli glede velikosti na izbranih naborih podatkov.	98

List of Algorithms

1	“BruteForce” algorithm for finding the frequent itemsets	18
2	“Compute_support” algorithm for calculating the support of an item-set	19
3	“APRIORI” algorithm for finding the frequent itemsets	20
4	ECLAT algorithm	21
5	FP-Growth Algorithm	22
6	Simple and accurate classification algorithm	26
7	Building compact and accurate classifier	33
8	Classification process of J&B approach	34
9	Overall coverage and accuracy of the classifier	35
10	The k-means algorithm for partitioning, where each cluster’s center is represented by the mean value of the objects in the cluster.	54
11	Agglomerative Hierarchical Clustering with Complete Linkage (AHC-CLH: Heights AHCCLC: Clusters)	58
12	Computing the optimal number of clusters	58
13	A Representative CAR based on Cluster Center (RCC)	60
14	A Representative CAR based on Dataset Coverage (RDC)	61
15	Learning the proposed associative classifier	61
16	Classification process of proposed associative classifiers	62

List of Tables

3.1	List of CARs generated from input dataset.	27
3.2	Ordered class association rules.	27
3.3	Grouped CARs based on class value.	28
3.4	Final simple associative classifier.	28
3.5	The comparison between SA method and some classification algorithms.	29
3.6	The results of second experiment (with different default parameters).	30
3.7	Generated CARs from training dataset	36
3.8	Training dataset	36
3.9	Sorted CARs based on support and confidence thresholds	37
3.10	Final compact and accurate classifier	37
3.11	Description of datasets and parameters of AC algorithms	40
3.12	The comparison between J&B method and other classification algorithms on accuracy.	41
3.13	Statistically significant wins/losses counts of J&B method on accuracy	42
3.14	The number of classification rules generated by the classifiers	42
3.15	Statistically significant wins/losses counts of J&B method on rules	42
7.1	Default parameters of associative classifiers and datasets description.	65
7.2	Overall accuracies with standard deviations.	66
7.3	Statistically significant wins/losses counts of DC method on accuracy	67
7.4	Statistically significant wins/losses counts of DDC method on accuracy	67
7.5	Statistically significant wins/losses counts of CDC method on accuracy	67
7.6	number of CARs	67
7.7	Statistically significant wins/losses counts of DC and DDC method on rules	68
7.8	Statistically significant wins/losses counts of CDC method on rules	68
7.9	Overall Coverage	69
8.1	Privzeti parametri asociativnih klasifikatorjev in opis naborov podatkov.	91
8.2	Klasifikacijske točnost s standardnimi odkloni - skupno.	93
8.3	Število KAP	94
8.4	Skupna pokritost	94

Chapter 1

Introduction

In this chapter, we will introduce the scientific background and motivation of our proposed research in Section 1.1. Then, we will introduce the existing associative classifiers that are related to our research in Section 1.2. Section 1.3 discusses the contribution of the dissertation and methodology. Finally, the content guide of the dissertation will be given in Section 1.4.

1.1 Scientific Background

Huge amounts of data are being generated and stored every day in corporate computer database systems. Mining association rules [1] from transactional data is becoming a popular and important knowledge discovery technique. For example, association rules (ARs) of retail data can provide valuable information on customer buying behavior. The number of rules discovered in a real-life dataset can easily exceed thousands of rules. To manage this knowledge, rules have to be pruned and grouped, so that only a reasonable number of rules have to be inspected and analyzed.

Data mining is the process of extracting hidden regularities from large data sets, which are novel knowledge. This can be achieved by combining methods from statistics and artificial intelligence. Data mining plays an important role in the analysis of real-life datasets as well as collections of observations.

Classification rule mining aims to discover a small set of rules in the dataset that forms an accurate classifier, while association rule mining finds all the rules existing in the dataset that satisfy user-defined minimum support (support is an indication of how frequently the items appear in the dataset described in Section 2.2) and confidence (confidence indicates the number of times the IF-THEN statements have been found to be true) constraints (described in Section 1.1.1). Various types of classification approaches (supervised learning) have been proposed (e.g., KNN [31], Bayesian classifiers [2], decision trees [25], neural networks [32] and others [43, 44]).

Associative Classification (AC) is a combination of these two important data mining techniques, namely, classification and association rule mining. AC aims to build accurate and efficient classifiers based on association rules. Recently, researchers have proposed several classification algorithms (described in section 1.2) based on association rules called associative classification methods such as CBA: Classifica-

tion based Association [18], CMAR: Classification based on Multiple Association Rules [17], MCAR: Mining Class Association Rules [76], and CPAR: Classification based on Predicted Association Rules [30], MMSCBA: Associative classifier with multiple minimum support [45], CBC: Associative classifier with a small number of rules [46], MMAC: Multi-class, multi-label associative classification [47], ARCID: Association Rule-based Classification for Imbalanced Datasets [48].

Experimental evaluations [5, 18, 30, 45–48] show that associative classification approaches achieve higher accuracy than some traditional classification algorithms produced by decision trees [3, 24, 25, 37–40] and non-rule-based approaches [2, 31, 32, 43, 44]. Most of the associative classification methods [17, 19, 49–55] tend to build accurate and compact classifiers that include a limited number of effective rules by using rule selection or pruning techniques. In comparison with some traditional rule-based classification approaches, associative classification has two main characteristics. Firstly, it generates a large number of association classification rules. Secondly, support and confidence thresholds are applied to evaluate the significance of classification association rules. However, associative classification has some drawbacks. Firstly, it often generates a very large number of classification association rules in association rule mining that is computationally expensive, especially when the training dataset is large. It takes great effort to select a set of high-quality classification rules among them. Secondly, the accuracy of associative classification depends on the setting of the minimum support and the minimum confidence constraints, that is, imbalanced datasets [94] may heavily affect the accuracy of the classifiers. Furthermore, the efficiency of associative classification depends on the minimum support (if minimum support is applied low, larger number of rules are generated and analyzing the larger number of rules to produce AC may lead a low efficiency) and the training dataset (if the learning dataset is large and dense). Although associative classification [17, 18, 30] has some drawbacks, it can achieve higher accuracy than rule and tree-based classification algorithms on some real-life datasets (such as “Breast cancer”, “Hayes-root”, “Lymp”, “Tic-Tac-Toe” shown in Table 8.2).

Another important approach is clustering methods (unsupervised learning) studied in [16, 22, 23]. These clustering techniques are split into two main parts: partitional and hierarchical clustering. In partitional clustering [20, 26, 28], objects are grouped into disjoint clusters such that objects in the same cluster are more similar to each other than objects in other cluster. Hierarchical clustering [27], on the other hand, is a nested sequence of partitions. In the bottom-up method, larger clusters are built by merging smaller clusters, while it starts with the one cluster containing all objects and divides into smaller clusters in the top-down method.

Researchers have proposed associative classifiers based on clustering approaches [12, 57, 77, 81, 82]. The main goal of cluster-based approaches is to reduce the number of classification rules and increase the accuracy of the classifiers. The advantage of such classifiers compared to associative or traditional classifiers is that ACs based on clustering produce more compact rules and those approaches build more understandable, descriptive and accurate models.

1.1.1 Association Rules

Association rule mining is one important data mining task that is nowadays applied to solve different kinds of problems in weak formalization fields. The main goal of association rule mining is to find all rules in datasets that satisfy some basic requirements such as minimum support and minimum confidence. It was initially proposed by Agrawal [1] to solve the market basket problem in transactional datasets and it has now been developed to solve many other problems such as classification, clustering et al. The discovery of correlations and associations between items in large datasets is performed by frequent itemset mining. The discovery of interesting correlation relationships between real-life transactions can simplify the decision-making processes in many business activities, such as cross-marketing, catalog design, medical diagnosis and customer shopping behavior analysis.

Frequent patterns mean that patterns occur frequently in a dataset. There are many kinds of frequent patterns, including itemsets, subsequences, and substructures. A *frequent itemset* is a set of items that frequently appear together in a transactional dataset. A *frequent sequential pattern* means the itemset that customers tend to buy first some items followed by other items, for example, first a *cheese-burger*, followed by *french-fries*, and then a *coca cola*. Mining frequent patterns leads to the discovery of interesting associations and correlations within data.

Example: Let us suppose that we are working at a fast food restaurant as marketing manager and we would like to report to the director about the items that are bought frequently within the same transactions. One example of such rules that can be mined from the *fast food restaurant* transactional database is as follows:

$(Trans_id: product="hamburger") \rightarrow product = "coca\ cola")$, [support = 5%; confidence = 75%]

The above single-dimensional association rule contains a single attribute (*buys*), where *Trans_id* is a variable representing a transaction's id which is done by a customer, and *product* expresses the single attribute. A 5% support of a rule means that in 5% of all transactions, *hamburger* and *coca cola* were purchased together. A 75% confidence of a rule means that if a customer buys a *hamburger*, there is a 75% chance of buying *coca cola* as well. Simply, the above rule can be defined as "*hamburger* \rightarrow *coca cola* [5%, 75%]".

Next, we are given the fast food restaurant relational dataset related to purchases. The following association rules may be generated:

$(Trans_id: age = "18..28" \text{ and } status = "student") \rightarrow (buys = "student\ menu")$
[support = 3%, confidence = 60%]

Since each attribute is referred to as a dimension, we have given the multi-dimensional dataset. The multi-dimensional association rule (age, status and buys attributes) indicates that 3% of customers are ages between 18-28 years with student status and have purchased a student menu. There is a 60% probability that a student in the above age group will buy a student menu. Typically, interesting association rules must satisfy both minimum support and minimum confidence thresholds.

1.1.2 Classification Rules

Classification and prediction are two important tasks of data mining that can

be used to extract the hidden regularities from datasets (data objects whose class labels are known) to form accurate models (classifiers); classification approaches aim to build classifiers (models) to predict the class label of a future data object. Such analysis can help us to understand the data comprehensively. **Data classification** is a two-step process, consisting of a learning step (where a classification model is constructed) and a classification step (where the model is used to predict class labels for given data). Classification methods have been widely used in many real-world applications, such as customer relationship management [34], medical diagnosis [35], and industrial design [36].

The data classification process is shown for the sample loan application data in Figure 1.1.

(a) defines the learning part of the classification: Training data are analyzed by a classification algorithm. Here, the class attribute is “loan decision”, and the learned model or classifier is represented in the form of classification rules. (b) illustrates the classification part: Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data examples.

For example: A bank loans officer needs analysis of her data to learn which loan applicants are “safe” and which are “risky” for the bank or a marketing manager at “AllElectronics” needs data analysis to help guess whether a customer with a given profile will buy a new computer or a medical researcher wants to analyze “Breast cancer” data to predict which one of three specific treatments a patient should receive. In each of these examples, the data analysis task is **classification**, where a model or classifier is constructed to predict class (categorical) labels, such as “safe” or “risky” for the loan application data; “yes” or “no” for the marketing data; or “treatment A,” “treatment B,” or “treatment C” for the medical data.

Classification rule-learning algorithms are basically divided into two parts: tree-based and rule-based.

Decision tree induction is the learning of decision trees from class-labeled training examples. A decision tree is a flowchart-like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node. Classification rules are extracted easily from the decision tree: Each path from the root to the leaves is considered as a rule, so, the number of rules extracted from the decision tree is equal to the number of leaves. A typical decision tree is shown in Figure 1.2. It represents the concept `buys_computer`, that is, it predicts whether a customer at “AllElectronics” is likely to purchase a computer. Internal nodes are denoted by rectangles, and leaf nodes are denoted by ovals.

Tree-based algorithms [3, 24, 25, 37–40] adopt a greedy approach in which decision trees are constructed by top-down and divide-and-conquer approaches which start with a training dataset, then, the training dataset is recursively partitioned into smaller subsets while a tree is being built. The attribute selection procedure employs measures (such as *information_gain*, *information_gain_ratio* or the *gini_index*) [95]. In the rule-based classification [4, 6, 7, 11, 13, 14, 41, 42], learned models are represented as a set of IF-THEN rules, that is, IF *condition* THEN *conclusion*. An example is rule *R1*,

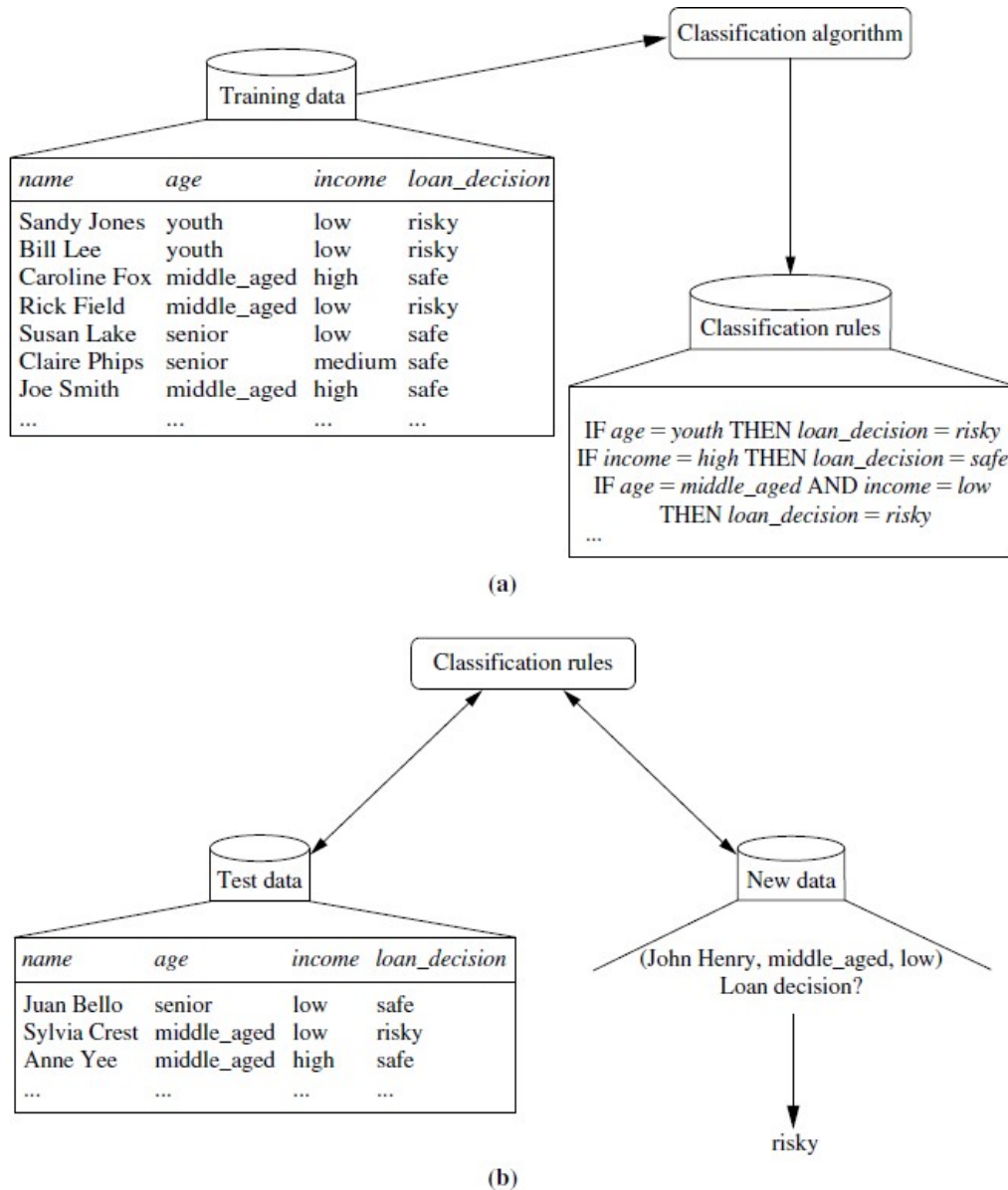


Figure 1.1: The data classification process.

$R1$: IF *age* = *youth* AND *student* = *yes* THEN *buys_computer* = *yes*.

The “IF” part (or left side) of a rule is known as the **rule antecedent** or **pre-condition**. The “THEN” part (or right side) is the **rule consequent**. In the rule antecedent, the condition consists of one or more *attribute tests* (e.g., *age* = *youth* and *student* = *yes*) that are logically AND. The rule’s consequent contains a class prediction (in this case, we are predicting whether a customer will buy a computer). If the condition (i.e., all the attribute tests) in a rule antecedent holds true for a given example, we say that the rule antecedent is **satisfied** (or simply, that the rule is satisfied) and that the rule **covers** the example.

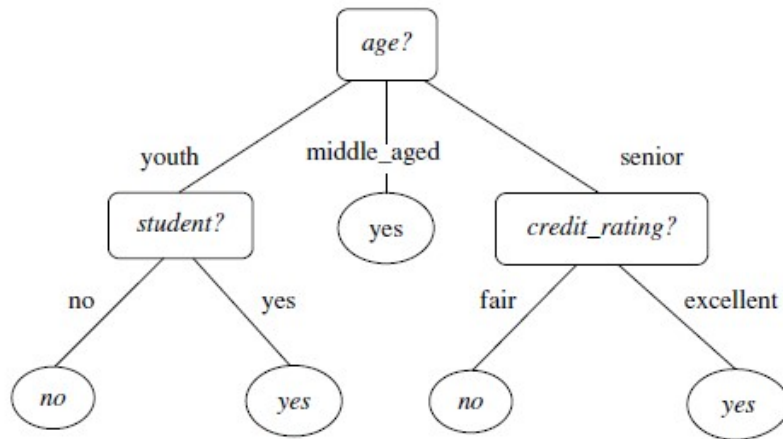


Figure 1.2: A decision tree for the concept `buys_computer`.

We first examine how such rules are used for classification, then we study ways in which they can be generated, either from a decision tree or directly from the training data by using a sequential covering approach.

Extracting classification rules from a decision tree: The decision tree of Figure 1.2 can be converted to classification IF-THEN rules by tracing the path from the root node to each leaf node in the tree. The rules extracted from Figure 8.2 are as follows:

- R1* : IF *age* = *youth* AND *student* = *no* THEN *buys_computer* = *no*
- R2* : IF *age* = *youth* AND *student* = *yes* THEN *buys_computer* = *yes*
- R3* : IF *age* = *middle_aged* THEN *buys_computer* = *yes*
- R4* : IF *age* = *senior* AND *credit_rating* = *excellent* THEN *buys_computer* = *yes*
- R5* : IF *age* = *senior* AND *credit_rating* = *fair* THEN *buys_computer* = *no*

A disjunction (logical OR) is implied between each of the extracted rules. Because the rules are extracted directly from the tree, they are **mutually exclusive** and **exhaustive**. Mutually exclusive means that we cannot have rule conflicts here because no two rules will be triggered for the same example. (We have one rule per leaf, and any example can map to only one leaf.) Exhaustive means there is one rule for each possible attribute–value combination, so that this set of rules does not require a default rule. Therefore, the order of the rules does not matter—they are unordered. IF-THEN rules can be extracted directly from the training data (i.e., without having to generate a decision tree first) using a sequential covering algorithm. The name comes from the notion that the rules are learned sequentially (one at a time), where each rule for a given class will ideally cover many of the class’s examples (and hopefully none of the examples of other classes). Sequential covering algorithms are the most widely used approach to mining disjunctive sets of classification rules. The general strategy is as follows: rules are learned one at a time. Each time a rule is learned, the tuples covered by the rule are removed, and the process repeats on the remaining examples. This sequential learning of rules is in

contrast to decision tree induction, because the path to each leaf in a decision tree corresponds to a rule, we can consider decision tree induction as learning a set of rules simultaneously.

The classification models can be expressed in different ways, such as *classification rules (IF-THEN)*, *decision trees* or *neural networks* (shown in Figure 1.3). A neural network is typically a collection of neuron-like processing units with weighted connections between the units when it is used for classification. There are many other methods for constructing classification models, such as naïve Bayesian classification, support vector machines, and *k*-nearest neighbor classification.

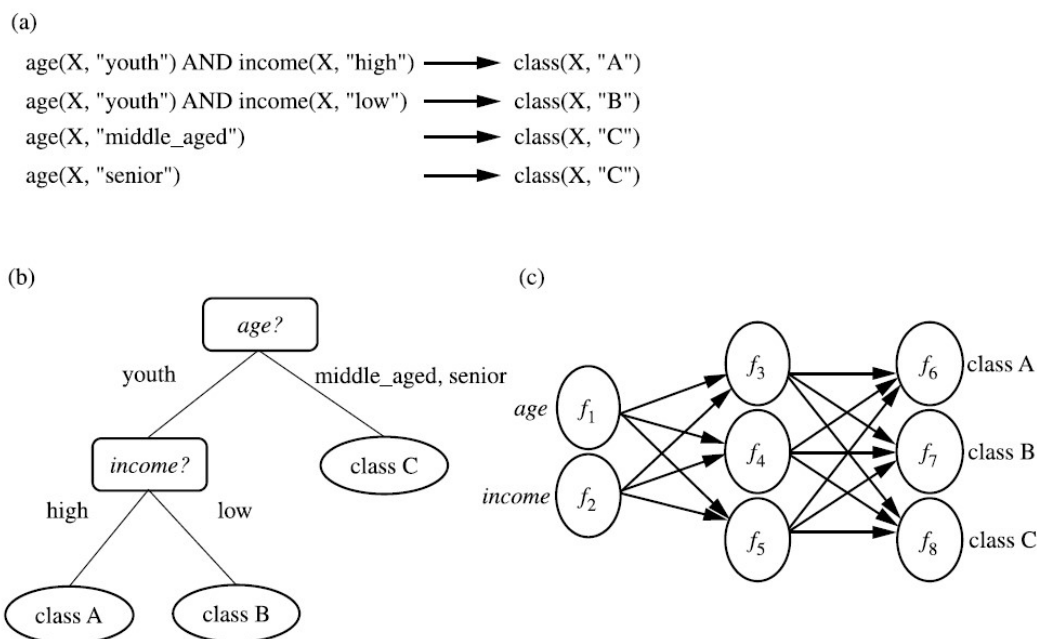


Figure 1.3: Represented classification models in various forms, as (a) IF-THEN rules, (b) a decision tree, (c) a neural network.

1.1.3 Cluster Analysis

Cluster analysis [78–80, 84] is different than classification and prediction. Clustering analyzes (unsupervised) data objects without class labels, while classification and prediction analyze class-labeled data objects. In general, the class labels are not presented in the training dataset, because they are initially unknown. Clustering can be used to generate such labels. The main goal of clustering techniques is maximizing the intraclass similarity and minimizing the interclass similarity inside the clusters when clustering or grouping the data objects. That is, objects within the same cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters. Each cluster that is formed can be viewed as a class of objects, from which rules can be derived. **Example of cluster analysis:** Cluster analysis can be performed on *AllElectronics* customer data in order to

identify homogeneous sub-populations of customers. These clusters may represent individual target groups for marketing. Figure 1.4 shows a 2-D plot of customers with respect to customer locations in a city. Three clusters of data points are evident, Each cluster “center” is marked with a “+”.

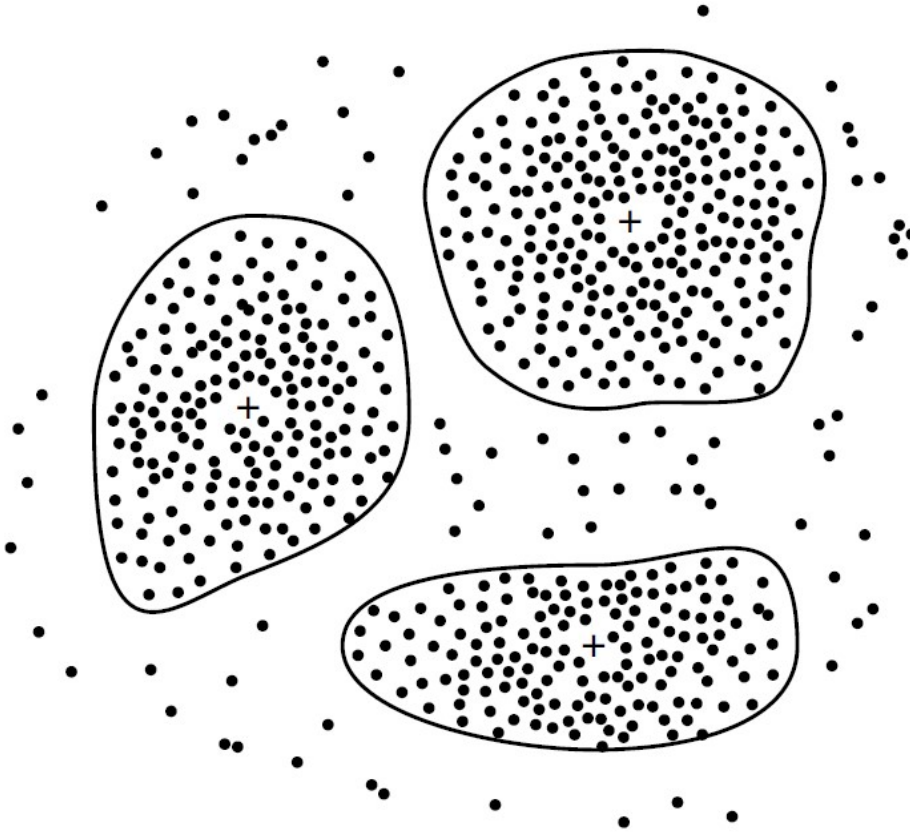


Figure 1.4: 2-D plot of customers with respect to customer locations in a city.

We propose new associative classification algorithms (by using classification, association and clustering techniques) based on hierarchical agglomerative clustering (complete linkage) method. We define the new normalized distance metrics based on “direct” and “indirect” measures to measure the similarities between CARs, which we later use to cluster CARs in a bottom-up hierarchical agglomerative fashion (firstly, we group the class association rules based on their class label and then rules that are in the same group are clustered together). Once we cluster the rules, the optimal number of clusters is identified for each group of CARs by cutting the dendrogram from the point that achieves the maximum difference between two consecutive cluster heights.

Once CARs are clustered, we define a “representative” CAR within each cluster. We propose two methods of extracting the “representative” CAR for each cluster, (1) we choose the CAR based on dataset coverage and (2) based on cluster center.

1.2 Literature Review

In computer science literature, there are a lot of rule-based, tree-based and probabilistic classification models proposed by prominent researchers. Since we propose associative classification methods based on dataset coverage and clustering in our PhD thesis, we discuss the performed research works in the associative classification field that are related to our proposed approaches in this section. We first describe some associative classification methods based on dataset coverage.

The CPAR [30] (Classification based on Predictive Association rules) algorithm proposed by Yin and Han combines the advantages of associative classification and traditional rule-based classification. CPAR utilizes a greedy algorithm and inherits the basic idea of First-Order Inductive Learner (FOIL) [69] method to generate rules, and it generates the rules directly from the training dataset, instead of generating a large candidate rules from frequent itemsets as in other associative classification methods. To avoid overfitting, CPAR uses expected accuracy to evaluate each rule and the classification process is also different than traditional classification approaches: First, CPAR selects all rules whose bodies match the testing example; second, it extracts the best k rules for each class among the rules selected in step 1; and finally, CPAR compares the average expected accuracy of the best k rules for each class found in step 2 and predicts the class label which achieves the highest expected accuracy.

CMAR, an associative classification method, also uses multiple association rules for classification. This method first extends FP-growth [73] (an efficient frequent pattern mining described in section 2.1.4) to mine large datasets. Next, CMAR employs a novel data structure to improve the overall accuracy and efficiency, called a CR-tree. The main goal of using a CR-tree is to store and retrieve a large number of rules compactly and efficiently. A *CR-tree* is a prefix tree structure to explore the sharing among rules, which achieves substantial compactness. A *CR-tree* itself is also an index structure for rules and serves rule retrieval efficiently. In the rule-selection process, CMAR extracts the highly confident, highly related rules based on dataset coverage and analyzes the correlation between those rules. More precisely, for each rule R , all the examples are found covered by rule R and if R correctly classifies one example, then R will be included to the final classifier and cover count (cover count threshold C is applied at the beginning) of those examples covered by R is increased by 1. If the cover count of an example is exceeded by C , then that example is removed. This process continues while both the training data set and rule set are not empty.

CBA [18] (Classification Based on Associations), a heuristic approach proposed by Liu, Hsu and Ma in 1998, had two main steps as in all other associative classification algorithms, rule-generation and rule-selection parts. CBA utilizes the iterative approach that is similar to the APRIORI algorithm [1] (described in section 2.1.2) in the rule-generation part. CBA generates all the frequent rule-items by making multiple passes over the data and then strong class association rules are generated through those frequent itemsets. The pruning method is also used in the rule-generation step based on pessimistic error rate. In the rule-selection phase, rules are extracted based on dataset coverage. More precisely, if a rule can classify at least one example cor-

rectly, that rule is considered as a potential rule for the classifier. At the end, rules are included into the final classifier based on total error.

The MCAR [76] (Mining Class Association Rules) approach is also aimed to produce an accurate associative classifier. In this method, the dataset only needs to be scanned once because the dataset is transformed into the vertical layout format. The cross-support operation and vertical data format are used for easily computing the support of the frequent items. The concept of class frequent items is taken into consideration to remove the rules, not only using the minimum support and minimum confidence thresholds to prune the candidate set as in other associative classification approaches. MCAR achieved some improvements: first, instead of scanning a dataset several times, MCAR just constructs the rule set at once. Second, it has developed a criterion of redundant class-frequent items to prune weak rules before they are actually generated. Third, to improve the complex structure such as frequent-pattern tree (FP-tree) (FP-tree is a prefix tree. For each example in the training data set, attributes values appearing in FP-tree are extracted and sorted according to FP-tree, the class label is attached to the last node in the path. All nodes with same attribute value are linked together as a queue) or CR-tree (CR-tree is also a prefix tree structure which is built for the set of rules. CR-tree is also used as index structure for rules and serves rule retrieval efficiently.), MCAR transforms the data format from horizontal to vertical by scanning the dataset once and counting the support by intersecting the C-Tidset (Data can be presented in item-CTID format (that is, {item: Class-Tidset}), where item is an item name, class is a finite set of class labels, and TID-set is the set of example identifiers containing the item).

The authors in [55] argue that when training examples are discarded after a rule is evaluated, this removal affects other lower-ranked rules since rules normally have common training data examples. Specifically, this method aims to build a classifier based on rule-pruning, and applies the vertical mining [61] (TidList) approach to discover the class association rules based on minimum support and confidence constraints. In the rule-selecting process, if any rule is able to cover at least one training example, it is inserted into the final classifier. When a rule is being evaluated to be a potential candidate for a classifier, ranking of the remaining untested rules is also updated constantly after removing the training examples covered by that rule.

The MMSCBA [45] (Associative classifier with multiple minimum support) classification method is based on multiple minimum supports (MMSs). The proposed method supplies the user-specified minimum support for each item and each class label. MMSCBA is proposed to discover a complete set of classification rules with multiple minimum supports to form a MMSCBA-based classifier. Once all the classification rules are generated, they are used them for prediction of uncertain data example. The prediction phase is split into two parts in AC methods: prediction based on the highest ranked single-rule item and multiple-rule item. MMSCBA utilizes the four measurements in classification process, namely, Maximum likelihood, Max X^2 , Laplace and Scoring.

A recently proposed associative classifier called ARCID [48] (Association Rule-based Classification for Imbalanced Datasets) aims to extract hidden regularities and knowledge from imbalanced datasets by emphasizing information extracted from minor classes. For managing the overwhelming number of CARs generated from

real-life datasets and removing the redundant rules that have the same information, ARCID applies the informative generic base (IGB) [60] algorithm. ARCID starts by scanning the training dataset to create groups of instances based on their class labels. Each resulting group contains instances belonging to the same class. Then, for each group, the support measure is applied to reduce the search space and generate frequent rules. In the rule-selection process, ARCID selects the best k rules based on the combination of two different criteria (measures): first, rules that have higher accuracy (rules that have higher supports can have higher predictive accuracy) and belong to major classes; second, rare rules that have lower support and belong to minor classes. In the class prediction phase, ARCID employs two methods. The first method selects the class label of the highest-ranked rule (based on Lift or Laplace measures) which matches the test instance. The second method selects the class label by using an aggregate value (the highest). This value corresponds to the average of a defined measure (Lift or Laplace) for each group of rules predicting the same class and matching the test instance. If any rule cannot classify the test instance, the minor class is predicted.

In the associative classification method [68], researchers proposed a new measure, called “condenseness”, to evaluate the infrequent rule items. Authors argued that infrequent rule items which are not satisfied with the minimum support threshold could produce strong association rules for classification. “Condenseness” of infrequent rule items is the average of *lift* of all association rules that can be generated from those rule items. A rule item with a high condenseness means that its items are closely related and can produce potential rules for AC although it does not have enough support. A New associative classifier, (CARC)- Condensed Association Rules for Classification is presented based on the concept of “condenseness”. Association rules are generated by using the modified “APRIORI” algorithm and develop the new strategy of rule inference with the condenseness.

The following associative classification methods are mainly constructed based on some clustering techniques, the main goal of these methods is to produce compact and meaningful associative classifiers while maintaining the classification accuracy. There are two main branches of clustering that can be applied to associative classification models, namely partitional and hierarchical (distance based) clustering.

In [59], researchers have proposed a new method to cluster the association rules by the K-means (partitional) clustering algorithm. The main goal of this research is clustering of discovered association rules to make it easy for users choosing the best rules. The algorithm is divided into 4 steps: (1) association rules generated from frequent patterns by the APRIORI algorithm are extracted; (2) Interestingness measures [62,63] such as Lift, Cosinus, Conviction and Information Gain are computed for all rules generated in step 1; (3) A set of association rules are partitioned into disjoint clusters by using the K-means algorithm, they try to cluster the rules which have the smallest similarities degree between them. Euclidian and Degree of similarity distances are used to apply the K-means algorithm; (4) finally, they classify the group of rules from the best to the worst by using a centroid of each cluster.

Another approach is distance-based clustering of association rules proposed in [12]. In hierarchical agglomerative clustering methods, choosing the best and appropriate distance metric is very important. Therefore, they propose new distance

metrics (similarity measures) called conditional market basket difference (CMBP) and conditional market basket log-likelihood methods based on indirect measures such as support and confidence to group the association rules. Based on the distances, agglomerative clustering is applied to cluster the rules. The rules are further embedded in a vector space with the use of multi3-dimensional scaling and clustered using self-organizing maps.

Mining clusters with association rules [29] is another related approach. Here the rules are first generated using the APRIORI algorithm, but the Hamming distance metric is later used to find the similarities between rules. Association rules having high confidence are then clustered using a top-down hierarchical clustering method for finding clusters in a population of customers, where the list of products bought by the individual clients is given. Once the rules are clustered, a specific distance metric is introduced to measure the quality of the clustering. Experimental results show that the success of this approach depends on the structure of the items in the datasets. If the set of possible products is large, association rules tend to have low confidence and this may affect badly to make informative clusters, however, clustering makes sense in this case. On the other hand, if the number of products is small, more reliable association rules are generated which produce an implied clustering.

Another interesting clustering-based approach [57] is “Tightness” which quantifies the strength of binding between the items of an association rule. The idea is that certain items in an application domain might get bound together because they are so strongly correlated that they often occur together in transactions. This tightness of binding is not covered by traditional measures like support or confidence. They build their indirect distance function on the basis of “tightness”, that is, the items in association rules that obtain the maximum and minimum support. The association rules are clustered by applying the proposed distance function and average-linkage method of agglomerative hierarchical clustering.

1.3 Contribution to Science and Methodology

The results of this PhD thesis will contribute to knowledge in Computer Science, more precisely in the field of data mining and machine learning. The expected novel contributions of the dissertation work to the science are as follows:

- We develop new meaningful distance metrics (similarity measure) to compute the similarity between class association rules. Since there are not enough similarity measure functions (distance metrics) of class association rules, this will be an important contribution to this field. In this research work, we propose two normalized distance metrics: first, direct distance metric (based on direct measures); second, combined (based on direct and indirect measures) distance metric;
- We identify the cluster of class association rules by using normalized similarity measure and automatically determine the optimal number of clusters for each class value;

- We define two methods (based on cluster center and dataset coverage) of extracting a representative CAR for each cluster to produce the final, compact and meaningful classifier;
- We implement a new application to solve real-world problems, that is, we apply our proposed classifiers to real-life and well-known datasets.

Furthermore, we produce classification models based on association rules (simple and accurate associative classifiers) within proposed scientific research. To achieve the above-mentioned goals, various methods from the data mining field are used to develop new associative classifiers, namely, classification, association, clustering and evaluation. We use “WEKA” software [33] to generate class association rules and to perform statistical significance testing. WEKA is appropriate open-source software to perform experimental evaluations because it has all necessary algorithms in the field of data mining and is easy to implement.

We use also R software [83] to execute the recently proposed classification algorithms and to find relationships between association rules or attributes as well as visualizing the clusters. We chose this software because it is easy to visualize two-dimensional and three-dimensional spaces, that is, all the necessary packages for visualization are supplied by them. Furthermore, R software has most of the recent works (such as CMAR, MCAR and CPAR) that are not included by WEKA.

The proposed research work consists of three steps. First, we generate the strong class association rules. In this step, we apply the APRIORI algorithm to generate the frequent itemsets, since APRIORI is the most widely-used algorithm for mining association rules in large databases, we used this approach to make our results as comparable to other existing results and the time complexity of APRIORI is better than other association rule generation algorithms. Second, we produce a compact and descriptive associative classifier by using Java programming language. Once our proposed classifier is built, we perform the experimental evaluations in the final step. We compare our algorithms with well-known rule-based and tree-based classification methods (Naïve Bayes, OneR [13], Decision Table [14], Simple Associative Classifier [64], PART [11], C4.5 [25], FURIA [65], Decision Table and Naïve Bayes [66], Ripple Down Rules [67], CBA, Prism [4], Random Forest [3] and JRip [7]) on accuracy, the number of classifications rules and some other relevance measures (described in section 3.2). To analyze the result, we use the experimenter tool of WEKA software. The visualization part and comparison of the result with recent works are performed by R software. We perform our experiments on well-known and real-world datasets taken from the UCI Machine Learning Repository [9].

The results of this PhD Thesis are published in the following articles:

- Mattiev, J., Kavšek, B.: Coverage based classification using association rule mining. *Applied Sciences* 10(20), Basel, Switzerland (2020).
- Mattiev, J., Kavšek, B.: Distance based Clustering of Class Association Rules to Build a Compact, Accurate and Descriptive Classifier *Computer Science and Information Systems*, Serbia (2020).

- Mattiev, J., Kavšek, B.: CMAC: Clustering class association rules to form a descriptive and Meaningful Associative Classifier, *Machine Learning, Optimization, and Data Science, LOD 2020*. Nicosia G., Pardalos P., Umeton R., Giuffrida G., Sciacca V. Eds. vol.11943, Springer, Siena, Italy (2020).
- Mattiev, J., Kavšek, B.: Simple and Accurate Classification Method Based on Class Association Rules Performs Well on Well-Known Datasets, *Machine Learning, Optimization, and Data Science, LOD 2019*. Nicosia G., Pardalos P., Umeton R., Giuffrida G., Sciacca V. Eds. vol.11943, Springer, Siena, Italy (2019), pp. 192–204.
- Mattiev, J., Kavšek, B.: A compact and understandable associative classifier based on overall coverage, *The 11th International Conference on Ambient Systems, Networks and Technologies, Procedia computer science, Vol.170*, Warsaw, Poland (2020), pp. 1161-1167.
- Mattiev, J., Kavšek, B.: How overall coverage of class association rules affects the accuracy of the classifier?, *Data Mining and Data Warehouses - SiKDD : proceedings of the 22nd International Multiconference Information Society, IS 2019*, Ljubljana, Slovenia (2019), pp. 49–52.
- Mattiev, J., Kavšek, B.: Using constrained exhaustive search vs. greedy heuristic search for classification rule learning, *Proceedings of the 5th Student Computer Science Research Conference, StuCoSReC-2018*, Koper, Sloveniya (2018), pp. 35-38.

1.4 Content Guide

In this PhD thesis, we focus on developing new associative classification methods based on hierarchical agglomerative clustering (complete linkage). The rest of the thesis is structured as follows:

Chapter 2 discusses how to generate the class association rules. To discover the CARs, we first need to generate the frequent itemsets, therefore we also discuss the well-known frequent itemset mining algorithms in the second chapter.

Our preliminary approaches are illustrated in Chapter 3. The first part of this chapter is about the SA (Simple Associative Classifier) method, and the J&B (Coverage based Associative Classifier) approach is introduced in the second part of the chapter. Experimental evaluations and examples based on those methods are also defined within this chapter.

Chapter 4 emphasizes the distance metrics to measure the similarity between class association rules. This chapter is also divided into three parts: distance metric based on “direct”, “indirect” and “combined” measures. In this chapter, new distance metrics are introduced and existing distance metrics will be discussed as well.

The Chapter 5 identifies the cluster of class association rules. Partitional and Hierarchical clustering algorithms are highlighted and the new algorithm is expressed to identify the optimal number of clusters in this chapter.

Extracting the representative class association rule (for each cluster) is discussed in Chapter 6. Two new approaches of the representative CAR selection are introduced in this chapter, based on cluster center and dataset coverage. Experimental

results of the main research work and the discussion of results are analyzed in Chapter 7, and the Chapter 8 concludes the PhD thesis.

Chapter 2

Discovery of Class Association Rules

It is interesting to know how often two or more objects of interest co-occur in many applications. For example, consider a laptop seller company, which enters all records into its database in relational form. Each record includes the features of a laptop such as CPU, memory, screen resolution and so on categorized according to their price. Analyzing such a dataset, one might be interested in finding out what kind of laptops are cheaper or more expensive. Such “frequent” sets of laptop features give clues to laptop prices and can be used to build classification models. The quest to mine frequent patterns appears in many other domains. The popular application is *market basket analysis*, that is, mining the sets of items that are frequently bought together at a supermarket by analyzing customer shopping carts (the so-called “market baskets”). Once we mine the frequent sets, they allow us to extract *association rules* among the item sets, where we make some statement about how likely two sets of items are to co-occur or to conditionally occur. For example, in the laptop seller company scenario frequent sets allow us to extract rules like, “CPU=Core i3, Memory=1TB, Screen resolution=Full-HD Laptop price=expensive”; by these rules, we will be able to find out the features of expensive, cheaper or medium-range price laptops. In the case of market baskets, we can find rules such as “Customers who buy milk and bread also tend to buy butter”. We begin this chapter with algorithms to mine frequent itemsets, and then show how they can be used to extract association rules.

Association rule generation is usually split up into two main steps:

1. In the first step, Minimum support threshold is applied to find all frequent itemsets from the training dataset.
2. Minimum confidence constraint is applied to generate strong class association rules (CAR) from the frequent itemsets.

2.1 Frequent Itemsets Mining Algorithms

The step of finding frequent itemsets in associative classification is the most important and computationally expensive step [30, 45]. Several different approaches to discover frequent rule-items from a data set have been adopted from association

rule discovery. For example, some AC methods [5, 70] employ the APRIORI candidate generation method. Other AC methods [17, 71, 72] mainly apply the FP-growth approach [73] to generate association rules. The CPAR algorithm utilizes a greedy strategy produced in FOIL. “Tid-lists” methods of vertical data layout [74, 75] are used in [47, 76]. In this section, we discuss the frequent itemset mining algorithms.

2.1.1 Brute-Force (Naive) Method

We begin by describing an exhaustive or brute-force algorithm. The brute-force method generates all possible itemsets $X \subseteq I$, and support of each such subset is computed in the input dataset D . The method is divided into two main steps: (1) candidate generation and (2) support computation.

Since each itemset is potentially a candidate frequent pattern, all the subsets (candidates) of I are generated in the *candidate generation* step. The candidate itemset search space is clearly exponential because there are $2^{|I|}$ potentially frequent itemsets. The structure of the itemset search space is instructive; the set of all itemsets forms a lattice structure where any two itemsets X and Y are connected by a link if X is an *immediate subset* of Y , that is, $X \subseteq Y$ and $|X| = |Y| - 1$. In terms of a practical search strategy, the itemsets in the lattice can be enumerated by using either a breadth-first (BFS) [85] or depth-first (DFS) [86] search on the *prefix tree*, where two itemsets X, Y are connected by a link if X is an immediate subset and prefix of Y . This allows one to enumerate itemsets starting with an empty set, and adding one more item at a time.

In the support computation step, the support of each candidate pattern X is computed and frequent candidates are determined. For each record $\langle t, i(t) \rangle$ in the database, we determine if X is a subset of $i(t)$. If so, we increment the support of X . The pseudo-code of brute-force method is highlighted in Algorithm 1. It enumerates each itemset $X \subseteq I$, and then computes its support by checking if $X \subseteq i(t)$ for each $t \in T$.

Algorithm 1 “BruteForce” algorithm for finding the frequent itemsets

Input: D : a dataset of records, I : itemsets and min_sup : the minimum support threshold

Output: L , frequent itemsets in D

```

1:  $L = \emptyset$ ;
2: for all  $i \in I$  do
3:    $supp(i) = Compute\_support(i, D)$  :
4:   if  $supp(i) \geq min\_sup$  then
5:      $L = L \cup \{(i, supp(i))\}$ ;
6:   end if
7: end for
8: return  $L$ 

```

The brute-force algorithm finds all the frequent itemsets in an exhaustive way. It checks all possible itemsets (line 2-7) as a potential candidate. Support of each itemset is found in line 3 described in Algorithm 2 and store those (line 4-6) that

satisfy the user-defined minimum support threshold.

Algorithm 2 “Compute_support” algorithm for calculating the support of an itemset

Input: i , itemset and D , a dataset of records

Output: S , support of itemset i

```

1:  $S = 0$ ;
2: for all  $d \in D$  do
3:   if  $i \subseteq d$  then
4:      $S = S + 1$ ;
5:   end if
6: end for
7: return  $S$ 

```

We shall see next how to systematically improve on the brute-force approach, by improving both the candidate generation and support counting steps.

2.1.2 APRIORI (Level-Wise) Approach

The generation of frequent itemsets by Apriori algorithm is accomplished in levels, wherein on each level APRIORI uses itemsets found to be frequent in the previous level to produce new candidate itemsets. APRIORI utilizes the ‘downward-closure’ property to speed up the searching process by reducing the number of candidate itemsets at any level. The main feature of ‘downward-closure’ is that all subsets of a frequent itemset must be frequent. Infrequent itemsets found at any level are removed, because it is not possible to make frequent itemset from an infrequent one. For example: if a itemset "abc" is infrequent (itemset "abc" is not satisfied by the minimum support threshold), then, itemset "abcd" (generated from infrequent itemset "abc") will be infrequent as well. Because support of a itemset "abcd" becomes lower or equal (it cannot be higher) to the support of a itemset "abc". APRIORI does this process to prune infrequent itemsets before computing their support at any level. This should reduce the time complexity to produce and compute the support for all items combinations in the datasets. The APRIORI algorithm applies an iterative approach called “*level-wise*” search, where k -itemsets are used to generate $(k+1)$ -itemsets. Let L be a set of frequent itemsets, the set of frequent 1-itemsets is found by scanning the dataset, that is, it calculates the support count for each item and collects those items that satisfy minimum support threshold. The resulting set is denoted as L_1 . Next, L_1 is used to generate L_2 , the set of frequent 2-itemsets, which is used to generate L_3 , and so on, until no more frequent k -itemsets can be found. The dataset needs to be fully scanned once to find each frequent itemset. It can be seen from the algorithm definition that time complexity of generating all frequent itemsets from the given dataset is reasonably high since it exhaustively searches (all possible combinations) the entire example space to find all possible itemsets. The set of all possible itemsets has the size of $2^n - 1$. Although the combinatorial complexity grows exponentially by depending on the number of items n in L , method search efficiently by using the “downward-closure” (support based pruning) property of support

which guarantees that all subsets of the frequent itemset must be also frequent and, all supersets of the infrequent itemset must be infrequent. The APRIORI algorithm is defined in Algorithm 3.

Algorithm 3 “APRIORI” algorithm for finding the frequent itemsets

Input: D , a dataset of records, min_sup , the minimum support count threshold

Output: L , frequent itemsets in D

```

1:  $L_1 = \text{find\_frequent\_1-itemsets}(D)$ ;
2: for ( $k = 2; L_{k-1} \neq \emptyset; k++$ ) do
3:    $C_k = \text{apriori\_gen}(L_{k-1})$ ;
4:   for each record  $t \in D$  do ▷ scan D for counts
5:      $C_t = \text{subset}(C_k, t)$ ;
6:     for each candidate  $c \in C_t$  do
7:        $c.count++$ ; ▷ Increment the count of all candidates in  $C_k$  that are
contained in  $t$ 
8:     end for
9:      $L_k = \{c \in C | c.count \geq min\_sup\}$ ;
10:  end for
11: end for
12: return  $L = \cup_k L_k$ ;

```

The frequent 1-itemsets, are found by the APRIORI algorithm from the given dataset in line 1. Lines 2-12 represent that candidate C_k is generated by using L_{k-1} to find L_k for $k \geq 2$. Candidate itemsets are generated by the *apriori_gen* function and it uses them to eliminate those having an infrequent subset (line 3). The fourth line scans the dataset after all of the candidates are generated. A subset function in line 5 finds all subsets of the record that are candidates for each record, and the count for each of those candidates that satisfy minimum support threshold (lines 6-9) to form the set of frequent itemsets, L in line 12.

2.1.3 ECLAT algorithm

The ECLAT algorithm [87] is a more efficient and scalable version of the APRIORI algorithm. The support counting step can be improved significantly by indexing the database in such a way that it allows us to compute the frequency faster. In the level-wise approach, we have to generate subsets of each record and check whether they exist in the prefix tree for computing the support. This can be computationally expensive because we may generate many subsets that do not exist in the prefix tree.

The ECLAT algorithm leverages the Transaction Id Sets (tidsets) directly for support computation. The basic idea is that the support of a candidate itemset can be computed by intersecting the tidsets of suitably chosen subsets. In general, given $\mathbf{t}(X)$ and $\mathbf{t}(Y)$ for any two frequent itemsets X and Y , we have $t(XY) = t(X) \cap t(Y)$. The support of candidate XY is simply the cardinality of $\mathbf{t}(XY)$, that is, $sup(XY) = |\mathbf{t}(XY)|$. ECLAT intersects the tidsets only if the frequent itemsets share a common prefix, and it traverses the prefix search tree in a DFS-like manner, processing a group of itemsets that have the same prefix. The pseudo-code of ECLAT is given in

Algorithm 4.

Algorithm 4 ECLAT algorithm

Initialization: $F = \emptyset, P = \{\langle i, t(i) \rangle \mid i \in I, |t(i)| \geq \text{minsup}\}$

Input: P : potential (assumed) frequent itemsets, F : frequent itemsets and min_sup : the minimum support threshold

Output: L , frequent itemsets

```

1: for  $\langle X_a, t(X_a) \rangle \in P$  do
2:    $L = L \cup \{(X_a, \text{sup}(X_a))\}$ ;
3:    $P_a = 0$ ;
4:   for  $\langle X_b, t(X_b) \rangle \in P$  and  $X_b > X_a$  do
5:      $X_{ab} = X_a \cup X_b$ ;
6:      $t(X_{ab}) = t(X_a) \cap t(X_b)$  ;
7:     if  $\text{sup}(X_{ab}) \geq \text{minsup}$  then
8:        $P_a = P_a \cup \{(X_{ab}, t(X_{ab}))\}$ ;
9:     end if
10:  end for
11:  if  $P_a \neq 0$  then
12:     $\text{ECLAT}(P_a, \text{minsup}, L)$ ;
13:  end if
14: end for
15: return  $L$ ;

```

ECLAT employs a vertical representation of the binary database D . Thus, the input is the set of tuples $\langle i, t(i) \rangle$ for all frequent items $i \in I$, which comprise an equivalence class P (they all share the empty prefix); it is assumed that P contains only frequent itemsets. In general, given a prefix equivalence class P , for each frequent itemset $X_a \in P$ (line 1), we try to intersect its tidset with the tidsets of all other itemsets $X_b \in P$ (line 2-10). The candidate pattern is $X_{ab} = X_a \cup X_b$ (line 5), and we check the cardinality of the intersection $\mathbf{t}(X_a) \cap \mathbf{t}(X_b)$ to determine whether it is frequent in line 7. If so, X_{ab} is added to the new equivalence class P_a that contains all itemsets that share X_a as a prefix (line 8). A recursive call to ECLAT (line 11-13) then finds all extensions of the X_a branch in the search tree. This process continues until no extensions are possible over all branches.

2.1.4 Frequent Pattern Tree Approach: FP-Growth algorithm

Another frequent pattern approach is FP-Growth that indexes the database for fast support computation by using an augmented prefix tree called the *frequent pattern tree* (FP-tree). Each node in the tree is labeled with a single item, and each child node represents a different item. Each node also stores the support information for the itemset comprising the items on the path from the root to that node. The FP-tree is constructed as follows: initially the root node of the tree contains null item \emptyset . Next, for each tuple $\langle t, X \rangle \in D$, where $X = \mathbf{i}(t)$, we insert the itemset X into the FP-tree, incrementing the count of all nodes along the path that represents X . If X shares a prefix with some previously inserted record, then X will follow the same

path until the common prefix. For the remaining items in X , new nodes are created under the common prefix, with counts initialized to 1. The FP-tree is complete when all records have been inserted.

The FP-tree can be considered as a prefix compressed representation of the database D . Because the main goal is to build the tree as compact as possible and the most frequent items to be at the top of the tree. FP-Growth therefore sorts the items in support descending order. It first computes the support of all single items $i \in I$ from the initial database, then it discards the infrequent items, and sorts the frequent items by support descending order. Finally, each tuple $\langle t, X \rangle \in D$ is inserted into the FP-tree after reordering X by decreasing item support. Once the FP-tree has been constructed, it is used as an index instead of the original database. All frequent itemsets can be mined from the tree directly via the FP-Growth method. Pseudo-code of the FP-Growth is shown in Algorithm 5.

Algorithm 5 FP-Growth Algorithm

Initialization: $F = \emptyset, P = \emptyset, R = FP - tree(D)$;

Input: R : FP-tree that is constructed from the input dataset, P : current itemset prefix, F : frequent itemsets and min_sup : the minimum support threshold

Output: L , frequent itemsets

```

1: Remove infrequent items from  $R$ 
2: if  $isPath(R)$  ▷ insert subset of  $R$  into  $F$  then
3:   for each  $Y \subseteq R$  do
4:      $X = P \cup Y$ ;
5:      $sup(X) = \min_{x \in Y} \{cnt(x)\}$ ;
6:      $L = L \cup \{(X, sup(X))\}$ ;
7:   end for
8: else ▷ process projected FP-trees for each frequent item  $i$ 
9:   for each  $i \subseteq R$  in increasing order of  $sup(i)$  do
10:     $X = P \cup \{i\}$ ;
11:     $sup(X) = sup(i)$ ; ▷ sum of  $cnt(i)$  for all nodes labeled  $i$ 
12:     $L = L \cup \{(X, sup(X))\}$ ;
13:     $R_x = \emptyset$ ; ▷ projected FP-tree for  $X$ 
14:    for each  $path \in PathFromRoot(i)$  do
15:       $cnt(i) = countof\ i\ in\ path$  :
16:       $insert\ path\ excluding\ i\ into\ FP - tree\ R_x\ with\ count\ cnt(i)$ ;
17:    end for
18:    if  $R_x \neq \emptyset$  then
19:       $FP - Growth(R_x, X, L, min\_sup)$ ;
20:    end if
21:  end for
22: end if
23: return  $L$ ;
```

The method gets as input an FP-tree R constructed from the input database D , and the current itemset prefix P , which is initially empty. Given a FP-tree R , projected FP-trees are built for each frequent item i in R in increasing order of

support, therefore, we remove infrequent items from R in line 1. To project R on item i , we find all the occurrences of i in the tree, and for each occurrence, we determine the corresponding path from the root to i (line 14). The count of item i on a given path is recorded in $cnt(i)$ (line 15), and the path is inserted into the new projected tree R_X , where X is the itemset obtained by extending the prefix P with the item i . While inserting the path, the count of each node in R_X along the given path is incremented by the path count $cnt(i)$. We omit the item i from the path, as it is now part of the prefix. The resulting FP-tree is a projection of the itemset X that comprises the current prefix extended with item i (line 10). We then call FP-Growth recursively with projected FP-tree R_X and the new prefix itemset X as the parameters (lines 18-19). The base case for the recursion happens when the input FP-tree R is a single path. FP-trees that are paths are handled by enumerating all itemsets that are subsets of the path (lines 2-6).

2.2 Class Association Rules

An association rule has two parts, an antecedent (*if*) and a consequent (*then*). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent. In CARs, the antecedent (left-hand side of the rule) is an itemset and the consequent (right-hand side of the rule) is a class label. CARs are created by analyzing data for frequent IF-THEN patterns and using the criteria *support* and *confidence* to identify the most important relationships. *Support* is an indication of how frequently the items appear in the dataset. *Confidence* indicates the number of times the IF-THEN statements have been found to be true. In data mining, association rules are useful for analyzing and predicting customer behavior. They play an important role in shopping basket data analysis, product clustering, catalog design and store layout.

Let D be a dataset with n attributes $\{A_1, A_2, \dots, A_n\}$ that are classified into M known classes and $|D|$ objects (cases). Let $Y = \{y_1, y_2, \dots, y_m\}$ be a list of class labels. A specific value of an attribute A_i and class Y is denoted by lower-case letters a_{im} and c_j respectively.

Definition 1. An itemset is a set of some pairs of attributes and a specific value, denoted $\{(A_{i1}, a_{i1}), (A_{i2}, a_{i2}), \dots, (A_{im}, a_{im})\}$.

Definition 2. A CAR R has the form $\{(A_{i1}, a_{i1}), \dots, (A_{im}, a_{im})\} \Rightarrow y_j$ where $\{(A_{i1}, a_{i1}), \dots, (A_{im}, a_{im})\}$ is an itemset and $y_j \in Y$ is a class label.

Definition 3. The actual occurrence $ActOcc(R)$ of a rule R in D is the number of records of D that match R 's antecedent (left-hand side).

Definition 4. The support of rule R , denoted by $Supp(R)$, is the number of records of D that match R 's antecedent and are labeled with R 's class.

Definition 5. The confidence of rule R , denoted by $Conf(R)$, is defined as follows: $Conf(R) = Supp(R)/ActOcc(R)$.

In this research work, we use the APRIORI algorithm to find frequent itemsets described in 2.1.2, APRIORI is widely-used algorithm, hence we used this approach to make our results as comparable to other existing results. Once we found all frequent itemsets from learning datasets, it is a straightforward approach to generate strong

CARs that satisfy both minimum support and minimum confidence constraints from the frequent itemsets. This can be done using following equation for confidence, we define for the rule: $X \rightarrow C$,

$$\text{Confidence}(X \rightarrow C) = \frac{\text{support_count}(X \cup C)}{\text{support_count}(X)} \quad (2.1)$$

The equation 2.1 is expressed in terms of itemsets support count, where X is antecedent (itemsets that is, left-hand side of the rule), C is consequence (class label that is, right-hand side of the rule), $\text{support_count}(X \cup C)$ is the number of records containing the itemsets $X \cup C$, and $\text{support_count}(X)$ is the number of records containing the itemsets X . Based on this equation, CARs can be generated as follows:

- For each frequent itemsets L and class label C , generate all nonempty subsets of L ;
- For every nonempty subset S of L , output the rule R in the form of $S \rightarrow C$, if $\frac{\text{support_count}(R)}{\text{support_count}(S)} \geq \text{min_conf}$, where min_conf is the minimum confidence threshold.

Frequent patterns and their corresponding CARs characterize interesting relationships between attribute condition and class labels. Thus, it has been used widely for effective and accurate classification models. The general idea is that we can search for strong associations between frequent patterns and class labels, because class association rules explore highly confident associations among multiple attributes.

Chapter 3

Associative Classification

Results of this chapter are published in the following articles:

- Mattiev, J., Kavšek, B.: Simple and Accurate Classification Method Based on Class Association Rules Performs Well on Well-Known Datasets, *Machine Learning, Optimization, and Data Science, LOD 2019. Nicosia G., Pardalos P., Umeton R., Giuffrida G., Sciacca V. Eds. vol.11943, Springer, Siena, Italy, (2019)*, pp. 192–204.
- Mattiev, J., Kavšek, B.: Coverage based classification using association rule mining. *Applied Sciences.*, Basel, Switzerland, (2020), 485–490.
- Mattiev, J., Kavšek, B.: QA compact and understandable associative classifier based on overall coverage, *ANT 2020, The 11th International Conference on Ambient Systems, Networks and Technologies, Procedia computer science, Vol.170, Warsaw, Poland, (2020)*, pp. 1161-1167.

In this chapter, we discuss our proposed associative classification models in preliminary step. We assume that we have been given a normal relational table which consist of N examples described by L distinct attributes and all examples are classified into M known classes. Attribute type can be categorical (nominal) or continuous (numeric). Our proposed research supports only categorical (nominal) attributes, therefore, we treat all the attributes uniformly. We map the categorical attribute's values to consecutive positive integers. Numeric attributes are discretized into intervals (bins), and the intervals are also mapped to consecutive positive integers. Since association rule learning does not support numeric attributes, all numeric attributes (in all datasets) were pre-discretized with “class-dependent” discretization method (that is able to automatically determine the number of bins for each numeric attribute) of WEKA software.

3.1 Simple Associative Classification Approach (SA)

In the first approach, we propose a simple and accurate classification method based on class association rules. We extract the reasonable number of strong class association rules for each class value by simple pruning to form a simple classifier. A reasonable number is selected by analyzing the dataset (it depends on the size of

the dataset and class distribution), because there are no any methods to select this number. SA method differs from other AC algorithms by its simplified rule-selection phase and class-imbalanced problem is avoided manually in SA, while this step is missed on most of ACs. This is a preliminary study that aims to show that an adequate choice of the “right” class association rules by considering the dependent (class) attribute distribution of values can produce a compact, understandable and relatively accurate classifier. Our proposed method is outlined in Algorithm 6.

Algorithm 6 Simple and accurate classification algorithm

Input: a set of CARs with their *support* and *confidence* constraints, *test_example*

Output: a *predicted_class*

```

1:  $F = frequent\_itemsets(D)$ ;
2:  $R = genCARs(F)$ ;
3:  $R = sort(R, minconf, minsup)$ ;
4:  $G = Group(R)$ ;
5: for  $k = 1 : k \leq numClass; k++$  do
6:    $X = extract(class[k], numrules)$ ;
7:    $Y = Y.add(X)$ ;
8: end for
9: for each  $y \in Y$  do
10:  if  $y.classifytest\_example$  then
11:     $class\_count[y.class]++$ ;
12:  end if
13: end for
14: if  $max(class\_count) == 0$  then
15:   $predicted\_class = majority\_class(D)$ ;
16: else
17:   $predicted\_class = index\_of\_max(class\_count)$ ;
18: end if
19: return  $predicted\_class$ 

```

The First line finds all frequent itemsets in the dataset by APRIORI algorithm described in 2.1.2, strong class association rules that satisfy the minimum support and confidence constrains are generated from frequent itemsets in line 2. Once the class associations rules are generated, then, we sort them by confidence and support descending order in line 3 as follow: Given two rules R_1 and R_2 , R_1 is said to have higher rank than R_2 , denoted as $R_1 > R_2$,

- If and only if, $conf(R_1) > conf(R_2)$; or
- If $conf(R_1) = conf(R_2)$ but, $supp(R_1) > supp(R_2)$: or
- If $conf(R_1) = conf(R_2)$ and $supp(R_1) = supp(R_2)$, R_1 has fewer attribute values in its left-hand side than R_2 does;
- if all the parameters of the rules are equal, we choose any of them randomly.

Line 4 illustrates the grouping of class association rules by their class labels (for example, if the class has three values, then rules are clustered into three groups).

From lines 5-8, we extract the reasonable number of rules per class that are equal to *numrules* to form a simple and accurate classifier. These set of rules become our final classifier. In lines 9-12, classification is performed by extracted CARs in lines 5-8, if the rule can classify the example correctly, then we increase the corresponding class count by one and store that rule. By lines 13-16, if none of the rules can classify the example correctly, then the algorithm returns the majority class value in the training dataset. Otherwise, it returns the majority class value of correctly classified rules.

Example. Let us assume that we have generated (shown in Table 3.1) the following class association rules (satisfied the user specified minimum support and confidence thresholds) described in the previous section.

Table 3.1: List of CARs generated from input dataset.

CARs	Support	Confidence
$\{a1=1, a3=2, a4=1, a5=3\} \rightarrow \{1\}$	4	72%
$\{a1=1, a2=5, a5=2\} \rightarrow \{3\}$	7	100%
$\{a1=1, a3=5, a4=4\} \rightarrow \{3\}$	9	95%
$\{a2=3, a3=4, a4=4, a5=4\} \rightarrow \{1\}$	3	100%
$\{a1=2, a2=3, a3=1\} \rightarrow \{2\}$	11	85%
$\{a3=1, a4=1, a5=2\} \rightarrow \{1\}$	8	90%
$\{a1=1, a2=5, a3=5, a5=2\} \rightarrow \{2\}$	11	85%
$\{a1=1, a2=2, a3=1, a4=3, a5=1\} \rightarrow \{3\}$	4	64%
$\{a2=2, a5=5\} \rightarrow \{2\}$	6	69%

In the first step, we obtain the following result (presented in Table 3.2) by sorting the CARs in *confidence* and *support* descending order:

Table 3.2: Ordered class association rules.

CARs	Support	Confidence
$\{a1=1, a2=5, a5=2\} \rightarrow \{3\}$	7	100%
$\{a2=3, a3=4, a4=4, a5=4\} \rightarrow \{1\}$	3	100%
$\{a1=1, a3=5, a4=4\} \rightarrow \{3\}$	9	95%
$\{a3=1, a4=1, a5=2\} \rightarrow \{1\}$	8	90%
$\{a1=2, a2=3, a3=1\} \rightarrow \{2\}$	11	85%
$\{a1=1, a2=5, a3=5, a5=2\} \rightarrow \{2\}$	11	85%
$\{a1=1, a3=2, a4=1, a5=3\} \rightarrow \{1\}$	4	72%
$\{a2=2, a5=5\} \rightarrow \{2\}$	6	69%
$\{a1=1, a2=2, a3=1, a4=3, a5=1\} \rightarrow \{3\}$	4	64%

In the next step (3.3), CARs are distributed into groups that are equal to the number of classes. We can achieve this result by sorting the CARs on class label.

Last but not least, we extracted the highly qualitative rules (already sorted by support and confidence) that are equal to the intended user specified *numrules* parameter for our classifier. Let us apply *numrules* is equal to 2, the final associative classifier is shown in Table 3.4.

Table 3.3: Grouped CARs based on class value.

CARs	Support	Confidence
$\{a2=3, a3=4, a4=4, a5=4\} \rightarrow \{1\}$	3	100%
$\{a3=1, a4=1, a5=2\} \rightarrow \{1\}$	8	90%
$\{a1=1, a3=2, a4=1, a5=3\} \rightarrow \{1\}$	4	72%
$\{a1=2, a2=3, a3=1\} \rightarrow \{2\}$	11	85%
$\{a1=1, a2=5, a3=5, a5=2\} \rightarrow \{2\}$	11	85%
$\{a2=2, a5=5\} \rightarrow \{2\}$	6	69%
$\{a1=1, a2=5, a5=2\} \rightarrow \{3\}$	7	100%
$\{a1=1, a3=5, a4=4\} \rightarrow \{3\}$	9	95%
$\{a1=1, a2=2, a3=1, a4=3, a5=1\} \rightarrow \{3\}$	4	64%

Table 3.4: Final simple associative classifier.

CARs	Support	Confidence
1. $\{a2=3, a3=4, a4=4, a5=4\} \rightarrow \{1\}$	3	100%
2. $\{a3=1, a4=1, a5=2\} \rightarrow \{1\}$	8	90%
.....
3. $\{a1=2, a2=3, a3=1\} \rightarrow \{2\}$	11	85%
4. $\{a1=1, a2=5, a3=5, a5=2\} \rightarrow \{2\}$	11	85%
.....
5. $\{a1=1, a2=5, a5=2\} \rightarrow \{3\}$	7	100%
6. $\{a1=1, a3=5, a4=4\} \rightarrow \{3\}$	9	95%

Now, we classify the following example;

$$\{a1 = 1, a2 = 5, a3 = 5, a4 = 4, a5 = 2\} \rightarrow ?$$

So, this example is classified by rules 4, 5, 6. Class label of the rules that classified the example correctly is 2, 3, 3. Our classifier predicts that the class of the new example is 3 (because the third class value has max count).

3.1.1 Experimental Evaluations of SA Approach

We evaluated the accuracy of our model (SA) by performing two experiments on 12 datasets (only nominal datasets were selected, because SA cannot handle the numeric datasets) taken from the UCI Machine Learning Database Repository and compared the results with 8 well-known classification algorithms, namely, Naive Bayes (NB), Decision Table (DT), C4.5, PART (PT), 1R, Prism (PR), Random Forest (RF) and JRip (JR). The WEKA software is used to explore all 8 classification algorithms with default parameters. Two widely-used (standard) percentage split methods are utilized in the experiment, namely, 75% (learning):25% (testing) and 66% (learning):34% (testing).

Experiment 1: In the first experiment we set up the same default parameters for all datasets and all classification algorithms. For all classification algorithms and our method, we used the percentage split (75% is for learning, 25% is for testing,

preserved order of split) method to split the dataset into learning and test set. All classification algorithms parameters are default values. For our method, we set up the following default values for the first experiment: $minsup=0.5\%$, $minconf=50\%$, $numrules=10$. The results of the first experiment are shown in Table 3.5.

Table 3.5: The comparison between SA method and some classification algorithms.

Dataset	#attr	#Cls	#recs	Accuracy(%)								
				NB	DT	C4.5	PT	1R	JR	PR	RF	SA
Breast.Can	10	2	286	78	79	69	68	69	78	62	71	81
Balance	5	3	625	89	67	64	76	57	69	59	80	76
Car.Evn	7	4	1728	84	59	66	77	59	62	59	64	59
Vote	17	2	435	91	96	95	94	98	97	95	99	95
Tic-Tac	10	2	958	73	77	86	93	74	99	95	96	86
Contact.Len	5	3	24	83	83	83	83	66	83	83	83	83
Nursery	9	5	12960	92	94	96	99	71	96	96	98	80
Mushroom	23	2	8124	94	100	100	100	98	100	100	100	92
Hayes-root	6	3	160	81	54	75	78	54	84	72	84	72
Lymp	19	4	148	89	64	72	81	72	70	72	75	72
Monks	7	2	554	73	100	100	100	73	100	99	100	90
Spect.H	23	2	267	80	93	93	89	93	93	80	97	89
Average Accuracy (%)				83.9	80.5	83.3	86.5	73.7	85.9	81	87.3	81.3

Table 3.5 shows that in the first experiment, our proposed method achieved better average accuracy than Decision Table, OneR and Prism algorithms with 81.3%, while the Random Forest method gained the highest average accuracy..

Experiment 2: All classification algorithms parameters are also set up as default values in the second experiment and the percentage split (66% is for learning, 34% is for testing, preserved order of split) method is used to split the dataset into learning and test sets; the randomize function is also used on some datasets to apply the percentage split method with preserved order function. We test the decision tree and rule-based models on accuracy and obtained results are reported in Table 3.6.

For our method, we set up different default parameters for different datasets to obtain the intended number of rules per class (at least 5-10 rules for each class value, this situation mainly happens with imbalanced datasets) and to increase the accuracy. We also take into consideration the class distribution to keep balance when we are extracting the rules, for example, if the class has three values that distributed 60%, 20%, 20% respectively, then our classifier should have 20, 5, 5 rules approximately for each class and we apply $numrules$ higher to achieve the intended number of rules for each class. $\#attr$ (number of attributes), $\#cls$ (number of class values), $\#recs$ (number of records) are the same as Table 3.5. Results are shown in Table 3.6.

As can be seen from the results, our proposed method outperforms Naive Bayes, OneR and Prism on accuracy, more specifically, it wins more than 50% of datasets among these three algorithms and our method has similar results with the Decision Table method, winning half of the datasets on accuracy. Even though our method gained lower average accuracy than Naive Bayes and C4.5 algorithms (81.3, 83.9 and 83.3 respectively) in the first experiment, it got higher average accuracy than these algorithms in the second experiment (82.2, 81.8 and 81.7 respectively). Although

Table 3.6: The results of second experiment (with different default parameters).

Dataset	Min Sup	Min Conf	#Rules per Class	Accuracy (%)								
				NB	DT	C4.5	PT	1R	JR	PR	RF	SA
Breast.Can	1%	75%	10	77	74	75	71	71	70	60	69	80
Balance	0.3%	60%	15	90	65	65	76	57	71	60	76	74
Car.Evn	0.5%	60%	72	72	62	65	72	59	61	61	64	63
Vote	1%	80%	15	92	94	95	97	96	96	91	97	94
Tic-Tac	1%	90%	30	70	73	79	95	73	98	92	94	92
Contact.Len	1%	80%	6	75	75	75	75	63	75	75	75	75
Nursery	2%	60%	55	89	93	95	98	71	93	96	98	89
Mushroom	25%	90%	100	94	100	100	100	98	100	100	100	94
Hayes-root	0.5%	50%	20	86	57	71	64	51	91	73	82	71
Lymp	5%	80%	18	81	76	76	74	72	74	68	74	74
Monks	1%	70%	30	74	100	92	100	74	100	94	98	92
Spect.H	3%	60%	25	81	92	92	89	92	92	70	95	88
Average Accuracy (%)				81.8	80.1	81.7	84.3	73.1	85.1	78.3	85.2	82.2

our algorithm loses around 70% of the datasets to C4.5, PART, JRip and Random Forest algorithms, it achieves the best accuracy on the “Breast.Can” dataset in both experiments. We obtained slightly higher accuracy in the second experiment compared to the first one with 82.16 and 81.25 respectively, because we modified the default parameters and *minsup*, *minconf* constraints.

We tested our method with different *confidence* and *numrules* thresholds on all datasets and here, the experiments on the “Nursery” dataset are reported to show how we analyzed and selected the best *numrules* and *confidence* thresholds which achieve the highest accuracy. On some datasets, higher confidence and a higher number of rules achieve higher accuracy (most probably in larger datasets). On some datasets (mainly on imbalanced datasets), lower confidence plays an important role to achieve higher accuracy because we can generate the intended number of class association rules by applying lower minimum confidence constraint on imbalanced datasets. Results are shown in Figure 3.1.

Figure 3.1 illustrates that the peaks of accuracy are achieved at *numrules* ≥ 50 of all curves. The best accuracy (89%) in “Nursery” dataset is gained when *numrules* is 55 and *confidence* is 60%, that is, there is optimal settings for both thresholds.

Accuracy and efficiency are crucial factors in classification tasks in data mining. Several research studies [88–92] showed that associative classification gets higher accuracy than some traditional rule-based classification approaches in some cases. However, it generates a large number of association classification rules. Therefore, the efficiency of associative classification is not high when the minimum support is set to be low and the training dataset is large.

Our experiments on 12 datasets from the UCI machine learning repository show that our method is consistent, accurate, and comparative with other well-known classification methods. It achieved the fourth highest average accuracy (82.2%) among all classification methods on the selected UCI datasets. This is just a preliminary research and a step to our final goal of producing a more comprehensive and inter-actively explorable classifier using class association rules and clustering.

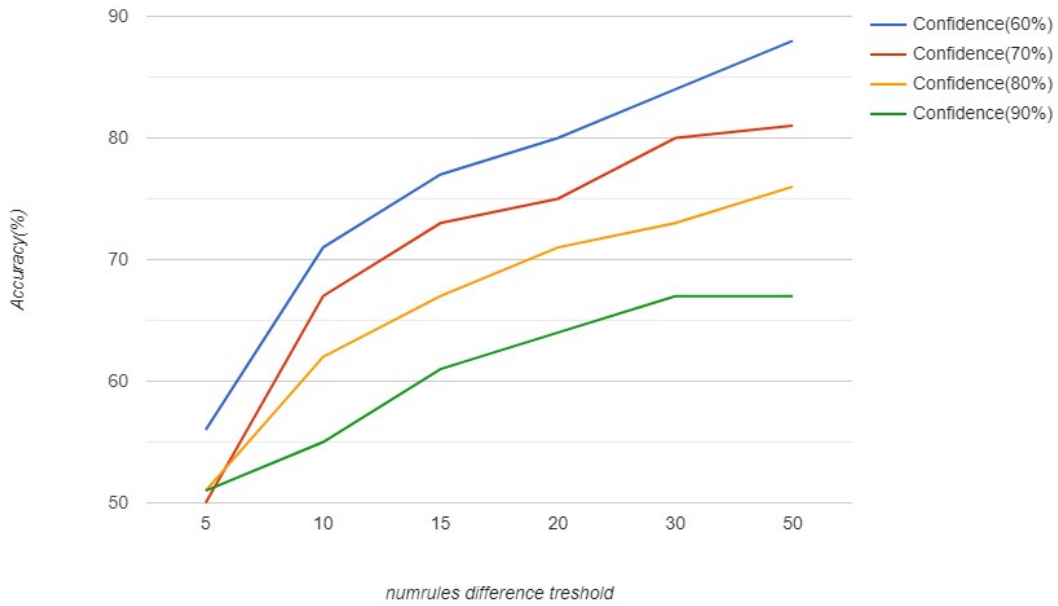


Figure 3.1: The effect of confidence and *numrules* thresholds to the accuracy on “Nursery” dataset.

3.2 J&B Associative Classification Approach

In the second approach, we propose a relatively simple, descriptive and accurate method to produce a compact and understandable classifier by exhaustively searching the entire example space. More precisely, we select the strong class association rules according to their contribution for improving the overall coverage of the learning set. Our proposed classifier has an advantage that it produces smaller classifier on bigger datasets compared to traditional classifiers, because our method has a stopping criterion in the rule-selection process based on the training dataset’s coverage, that is, our algorithm does not depend on the size of the dataset, while other classical classification methods are sensitive to the dataset’s size. Once we build the classifier, we compute the relevance measures such as “Precision”, “Recall” and “F-measure” and also overall coverage of the classifier to show how overall coverage affects the accuracy of the classifier, because overall coverage also plays an important role in forming compact and accurate classifiers that we showed in our previous research [56]. We then perform the statistical significance testing by using the paired T-test method [93] between our method and other classification methods. We have performed experiments on 12 datasets from the UCI Machine Learning Database Repository and compare the experimental result with 8 well-known classification algorithms (Decision Table (DT), Simple Associative Classifier (SA), PART (PT), C4.5, FURIA (FR), Decision Table and Naïve Bayes (DTNB), Ripple Down Rules (RDR), CBA).

Our goals are stated below:

- Our first goal is to generate the complete set of strong class association rules

that satisfy the user-specified minimum support and minimum confidence constraints.

- The second and main goal is to build a simple and accurate classifier by straightforward pruning, more precisely, we select the strong rules that have the higher contribution to the improvement of overall coverage.
- The third goal is to find the overall coverage, the number of rules, and accuracy of the intended classifier.
- The fourth goal is to compute relevance measures such as “Precision”, “Recall” and “F-measure” to compare performance of our approach and other classical classification approaches.
- Finally, we aim to perform the statistical significance testing between our intended classifier and other well-known classifiers on accuracy and the number of rules by using the paired T-test method.

We can define the building of an accurate and compact classifier by using AC in the following steps:

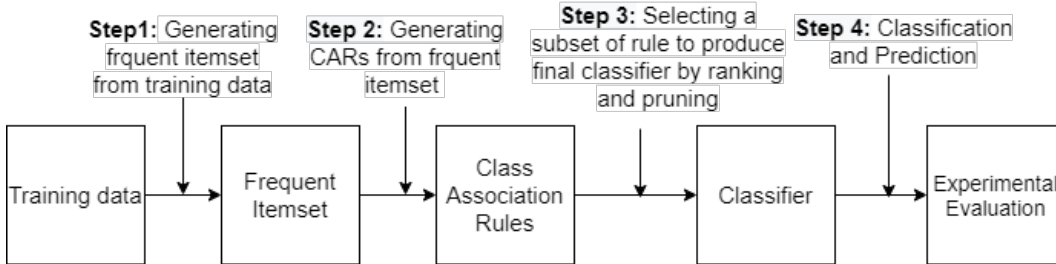


Figure 3.2: Associative classification working procedure

Figure 3.2 shows the general steps used in the AC approach. The first step is computationally expensive because it is similar to the association rule discovery which exhaustively searches the entire example space to discover the frequent itemsets (Step 1). In the second step, we generate strong class association rules from frequent itemsets that satisfy a user-defined minimum confidence threshold. A single rule is represented in AC by implication $X \rightarrow C$, where C is the consequence (class label) and X is the antecedent (Itemset) of the rule, that is, the highest frequency of class associated with its antecedent X in the learning dataset. Once the strong class association rules are generated, building an accurate and compact classifier is a straightforward approach, because no support counting or no scanning of the training data set is required. The most important task is to select the most effective ordered subset of rules to form an intended classifier. In that step, we select the strong class association rules that highly contribute to increase the training dataset coverage until we meet the stopping criterion to build an accurate, compact and understandable classifier. In the last step, we measure the quality of the classifier on some real-life

datasets. Since we use the existing approaches in the first and second steps, the key innovation steps are the third and last steps in this research work.

Our intended classifier

We propose our classifier by using the class association rules described in Chapter 2. Our proposed method is outlined in Algorithm 7.

Algorithm 7 Building compact and accurate classifier

Input: A training dataset D , minimum *support* and *confidence* constraints, intended coverage

Output: A subset of strong class association rules for classification

```

1:  $D = \text{training\_dataset}$ ;
2:  $F = \text{frequent\_itemsets}(D)$ ;
3:  $Rule = \text{genCARs}(F)$ ;
4:  $Rule = \text{sort}(R, \text{minconf}, \text{minsup})$ ;
5:  $\text{Fill}(\text{classified\_traindata}, \text{false})$ ;
6: for ( $i = 1; i \leq Rule.length; i++$ ) do
7:   for ( $j = 1; j \leq D.length; j++$ ) do
8:     if  $\text{classified\_traindata}[j] = \text{false}$  then
9:       if  $Rule[i].premise$  classifies  $D[j].premise$  then
10:         $\text{classified\_traindata}[j] = \text{true}$ ;
11:         $\text{contribution} = \text{contribution} + 1$ ;
12:      end if
13:    end if
14:  end for
15:  if  $\text{contribution} > 0$  then
16:     $\text{overall\_coverage} = \text{overall\_coverage} + \text{contribution}$ ;
17:     $Classifier = Classifier.add(rule[i])$ ;
18:  end if
19:  if  $(\text{overall\_coverage}/D.length) \geq \text{min\_coverage}$  then
20:    break;
21:  end if
22: end for
23: return  $Classifier$ 

```

1-2 lines find all frequent itemsets in the training dataset by using the APRIORI algorithm. The third line generates the strong class association rules that satisfy the minimum support and confidence constraints from frequent itemsets defined in 2.2. Class association rules (generated in line 3) are sorted by *confidence* and *support* in descending order in fourth line by the following criteria: Given two rules R_1 and R_2 , R_1 is said having higher rank than R_2 , denoted as $R_1 > R_2$,

- If and only if, $\text{conf}(R_1) > \text{conf}(R_2)$; or
- If $\text{conf}(R_1) = \text{conf}(R_2)$ but, $\text{supp}(R_1) > \text{supp}(R_2)$; or

- If $conf(R_1) = conf(R_2)$ and $supp(R_1) = supp(R_2)$, R_1 has fewer attribute values in its left-hand side than R_2 does;
- If all the parameters of the rules are equal, we can choose any of them.

Initial values of the *classified_traindata* array are filled with **false** value in line 5, this array is used later to compute the overall coverage of the training data. Basically, the classifier is built in lines 6-23. We scan the training dataset for each rule (lines 6-7), if the rule classifies a new unclassified example (that is, the rule's body matches the new example's body), then, we increase the contribution (this is, the rule's contribution to increase the overall coverage) in lines 8-12. If the *contribution* is higher than θ , that is, the rule classified new example(s), then, we increase the overall coverage by *contribution* and we add that rule to our final classifier in lines 15-18. If the rule can classify training examples, but cannot contribute to the improvement of the overall coverage, then we do not evaluate that rule as a potential rule for our classifier. Line 19 stops the procedure if the classifier achieves the intended overall coverage that is higher than the user-defined *intended_coverage* threshold. The classification process of our method is shown in algorithm 8.

Algorithm 8 Classification process of J&B approach

Input: A Classifier and test dataset

Output: Predicted class

```

1: for each rule  $y \in Classifier$  do
2:   if  $y$  classify test_example then
3:     class_count[ $y.class$ ]++;
4:   end if
5: end for
6: if  $\max(class\_count) == 0$  then
7:   predicted_class = majority_classifier;
8: else predicted_class = max_index(class_count);
9: end if
10: return predicted_class

```

Algorithm 8 predicts the class label of the test example by using the classifier. For each rule in the classifier (line 1), if the rule can classify the example correctly, then we increase the corresponding class count by one and store it (lines 2-4). In lines 6-9, if none of the rules can classify the new example correctly, then the algorithm returns the majority class value. Otherwise, it returns the majority class value of correctly classified rules.

Overall coverage and accuracy

After our classifier is built, it is straightforward to compute the overall coverage and accuracy of the classifier that is defined in Algorithm 9. To compute the overall coverage, we count the examples that are covered by the classifier and divide them by the total number of examples in the dataset. For accuracy, we count all the examples that are classified by classifier and divide it by the total number of examples in the dataset.

Algorithm 9 Overall coverage and accuracy of the classifier**Input:** Dataset and classifier**Output:** Overall coverage and accuracy of the classifier

```

1:  $n = D.length;$ 
2:  $C = Classifier;$ 
3: Fill(classified_testdata, false):
4: Fill(classified_testdata_withclass, false):
5: for ( $i = 1; i \leq C.length; i++$ ) do
6:   for ( $j = 1; j \leq n; j++$ ) do
7:     if  $C[i].premise$  classifies  $D[j].premise$  then
8:       if classified_testdata[ $j$ ] = false then
9:         classified_testdata[ $j$ ] = true;
10:      end if
11:    end if
12:    if  $C[i]$  classifies  $D[j]$  then
13:      if classified_testdata_withclass[ $j$ ] = false then
14:        classified_testdata_withclass[ $j$ ] = true;
15:      end if
16:    end if
17:  end for
18: end for
19: for ( $i = 1; i \leq n; i++$ ) do
20:   if classified_testdata[ $i$ ] then
21:     testcoverage++;
22:   end if
23:   if classified_testdata_withclass[ $i$ ] then
24:     accuracy++;
25:   end if
26: end for
27:  $Overallcoverage\_testdata = testcoverage/n;$ 
28:  $Overallaccuracy\_testdata = accuracy/n;$ 
29: return  $Overallaccuracy\_testdata, Overallcoverage\_testdata$ 

```

The first line finds the length of the dataset. We form our classifier from the training dataset in line 2. In the third and fourth lines, we fill all initial values of *classified_testdata* (to compute the overall coverage) and *classified_testdata_withclass* (to compute the overall accuracy) arrays as *false*. Lines 5-18 generally switch the status of the test example, without class label to compute the overall coverage in lines 7-11 and with class label to calculate the accuracy in lines 12-16. The number of correctly classified examples by classifier without class labels and with class labels is counted in lines 19-26 to calculate the overall coverage of the classifier and overall accuracy is computed in lines 27-28. The last line returns the obtained results.

Example. Let us assume that we have the following class association rules (shown in Table 3.7) (satisfied the user specified minimum support and confidence thresholds) generated from a dataset. We apply here the minimum coverage thresh-

old as 80%, that is, when our intended classifier covers at least 80% of the training examples, then we stop.

Table 3.7: Generated CARs from training dataset

CARs	Support (%)	Confidence (%)
{a1=3, a3=2, a4=1, a5=3} → {1}	4	72
{a1=1, a2=5} → {3}	5	100
{a1=3, a4=1} → {1}	8	100
{a1=1, a3=5, a4=4} → {3}	5	100
{a1=2, a2=2, a3=2} → {2}	5	63
{a2=3, a3=1, a4=4, a5=2} → {1}	3	80
{a1=2, a2=3, a3=4} → {2}	11	85
{a3=1, a5=2} → {1}	8	90
{a1=1, a2=5, a5=5} → {3}	6	65
{a1=1, a3=4, a5=2} → {2}	10	85
{a1=1, a2=2, a3=1, a4=3, a5=1} → {3}	4	64
{a2=2, a5=5} → {2}	6	100

And, learning dataset (Table 3.8) is to build the model

Table 3.8: Training dataset

#	a1	a2	a3	a4	a5	Class
1	3	4	2	1	3	1
2	5	3	1	4	2	2
3	1	5	2	4	2	1
4	3	2	3	1	2	3
5	?	2	?	3	5	1
6	1	3	5	4	3	2
7	2	2	1	2	2	2
8	1	4	4	2	2	1
9	1	?	2	4	3	2
10	1	5	1	1	5	3
11	1	2	1	1	5	2
12	1	4	4	4	2	1
13	1	3	?	?	?	2
14	1	4	5	4	4	3
15	3	2	2	1	2	1

In the first step, we sort the class association rules by *confidence* and *support* descending order, the result is shown in Table 3.9.

In the next step, we form our classifier by selecting the strong rules. We select strong rules which contribute to improve the overall coverage, we continue until achieving the intended training dataset coverage. Table 3.10 illustrates our final classifier.

Table 3.9: Sorted CARs based on support and confidence thresholds

CARs	Support (%)	Confidence (%)
$\{a1=3, a4=1\} \rightarrow \{1\}$	8	100
$\{a2=2, a5=5\} \rightarrow \{2\}$	6	100
$\{a1=1, a2=5\} \rightarrow \{3\}$	5	100
$\{a1=1, a3=5, a4=4\} \rightarrow \{3\}$	5	100
$\{a3=1, a5=2\} \rightarrow \{1\}$	8	90
$\{a1=2, a2=3, a3=4\} \rightarrow \{2\}$	11	85
$\{a1=1, a3=4, a5=2\} \rightarrow \{2\}$	10	85
$\{a2=3, a3=1, a4=4, a5=2\} \rightarrow \{1\}$	3	80
$\{a1=3, a3=2, a4=1, a5=3\} \rightarrow \{1\}$	4	72
$\{a1=1, a2=5, a5=5\} \rightarrow \{3\}$	6	65
$\{a1=1, a2=2, a3=1, a4=3, a5=1\} \rightarrow \{3\}$	4	64
$\{a1=2, a2=2, a3=2\} \rightarrow \{2\}$	5	63

Table 3.10: Final compact and accurate classifier

N	Classifier Rules	# Classified examples
1.	$\{a1=3, a4=1\} \rightarrow \{1\}$	3
2.	$\{a2=2, a5=5\} \rightarrow \{2\}$	1
3.	$\{a1=1, a2=5\} \rightarrow \{3\}$	2
4.	$\{a1=1, a3=5, a4=4\} \rightarrow \{3\}$	2
5.	$\{a3=1, a5=2\} \rightarrow \{1\}$	2
6.	$\{a1=1, a3=4, a5=2\} \rightarrow \{2\}$	2

Our classifier includes 6 rules. In this example, intended coverage is equal to 80% and 6 classification rules in the classifier cover 80% of the learning set. Since our training dataset has some examples with missing values, our classifier covered the whole training dataset (examples without missing values). Other rules also may cover unclassified examples, but we cannot exceed the user-defined training dataset coverage threshold. This is our stopping criterion and we cannot include other rules into our classifier. We also cannot include classification rules which cover only classified examples (this means it does not contribute to the improvement of overall coverage). Now, we classify the following unseen example:

$$\{a1=1, a2=5, a3=5, a4=4, a5=5\} \rightarrow ?$$

So, this example is classified by third and fourth classification rules. The class value of the rules which correctly classified the new example are 3 and 3. So, our classifier predicts that the class value of the new example is 3 (because the majority class value is 3).

Relevance measures of the classifier. Once we have built our model, another important task is to evaluate the performance of our model, this means how good is our model and predictions. In this research, we evaluate the performance of our model (classifier) by relevance measures such as Precision, Recall and F-measure. We compute all the relevance measures by constructing the confusion matrix for binary

class classification as shown in Figure 3.3.

		Predicted Class	
		Yes	No
Actual Class	Yes	True Positive	False Negative
	No	False Positive	True Negative

Figure 3.3: Confusion matrix for binary class classification

True positive and true negative statements shown in green are correctly predicted. Our main goal in building a classifier is to minimize the false positive and false negative remarks shown in red.

True Positives (TP) - are the correctly classified positive values which means both predicted and actual class values are yes. E.g. if the actual class value indicates that this person has cancer and the predicted class tells you the same.

True Negatives (TN) - are the correctly classified negative values which means both predicted and actual class values are yes. E.g. if the actual class says this passenger does not have cancer and the predicted class tells you the same.

False positives and false negative statements mean that the predicted class value opposes the actual class value.

False Positives (FP) - are the incorrectly classified positive values which means the predicted class values are yes and actual class values are no. E.g. if the actual class says this person is not cancer but the predicted class tells you that this person is cancer.

False Negatives (FN) - are the incorrectly classified negative values which means the predicted class values are no and actual class values are yes. E.g. if the actual class value indicates that this person is cancer and predicted class tells you that this person is not cancer.

Once we construct the confusion matrix and define all parameters, Precision, Recall and F-measure relevance measures are computed according to those four parameters.

Precision (Positive predictive values) is the fraction of correctly classified positive statements to the total number of predicted positive statements.

$$Precision = \frac{TP}{(TP + FP)} \quad (3.1)$$

Recall (Sensitivity) is the fraction of correctly classified positive statements to the total number of statements that actual class values are yes.

$$Recall = \frac{TP}{(TP + FN)} \quad (3.2)$$

F-measure is the harmonic mean of Precision and Recall.

$$F - measure = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{3.3}$$

For multi-class classification problems, we construct the confusion matrix with multi-classes (A, B, C , and so on) as shown in Figure 3.4. TP_A is the number of true positive examples in class A , i.e., the number of correctly classified examples from class A , and D_{AB} is the examples from class A that are incorrectly classified as class B . Thus, the false negative in the A class (FN_A) is the sum of D_{AB}, D_{AC} , etc. ($FN_A = D_{AB} + D_{AC} + \dots$) which means the sum of all examples belong to class A that are incorrectly classified as other class B, C , etc. False positive in Class A (FP_A) is the sum of D_{BA}, D_{CA} , etc. ($FP_A = D_{BA} + D_{CA} + \dots$) which means the sum of all examples belong to another class values B, C , etc. that are incorrectly classified as class A .

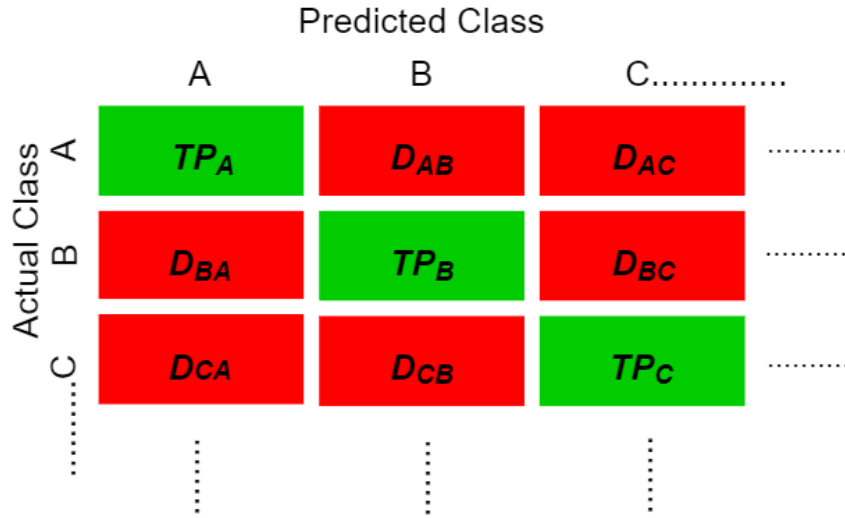


Figure 3.4: Confusion matrix for multi-class classification problems

3.2.1 Experimental Evaluations of J&B Approach

We tested our model on 12 real-life datasets taken from the UCI Machine Learning Repository. We evaluate our classifier (J&B) by comparing it with 8 well-known classification methods on the accuracy, relevance measures and the number of classification rules. All differences were tested for statistical significance by performing a paired t-test (with a 95% significance threshold).

J&B was run with default parameters *minimum support* = 1% and *minimum confidence* = 60% (on some datasets (marked with bold in Table 3.11), however, *minimum support* was lowered to 0.5% or even 0.1% to ensure enough CARs to be generated for each class value). For all other 8 rule learners we used their WEKA workbench implementation with default parameters. We applied 90% as a required coverage (training dataset) threshold that is the stopping point of selecting rules to

our classifier. The description of the datasets and input parameters is shown in Table 3.11.

Table 3.11: Description of datasets and parameters of AC algorithms

Dataset	# of attributes	# of classes	# of records	Min support	Min confidence
Breast.Can	10	2	286	1%	60%
Balance	5	3	625	1%	50%
Car.Evn	7	4	1728	1%	50%
Vote	17	2	435	1%	60%
Tic-Tac-Toe	10	2	958	1%	60%
Nursery	9	5	12960	0.5%	50%
Hayes-root	6	3	160	0.1%	50%
Lymp	19	4	148	1%	60%
Spect.H	23	2	267	0.5%	50%
Adult	15	2	45221	0.5%	60%
Chess	37	2	3196	0.5%	60%
Connect4	43	3	67557	1%	60%

Furthermore, all experimental results were produced by using a 10-fold cross-validation evaluation protocol.

Experimental results on classification accuracies (average values over the 10-fold cross-validation with standard deviations) are shown in Table 3.12 (**OC**: Overall Coverage).

Our observations (Table 3.12) on selected datasets show that our proposed classifier achieved better performance than Decision Table, C4.5, DTNB, RDR and SA (84.9% and 79.3%, 83.6%, 81.5%, 83.6, 82.3%) on average accuracy. Our proposed method achieved the best accuracy on “Breast.Can”, “Hayes-root” and “Connect4” datasets.

Standard deviations ranged around 4.0 in all classification methods and were higher for all methods on “Breast.Can”, “Hayes-root”, “Lymp” and “Connect4” datasets (above 4), that is, the differences between accuracies fluctuated and were reasonably high in 10-fold cross-validation experiments. When the overall coverage is above 90%, our proposed method tends to get reasonably high accuracy on all datasets. On the other hand, overall coverage of J&B was lower than its accuracy on “Vote” and “Nursery” datasets. This fact is not surprising since uncovered examples get classified by the majority classifier.

Statistically significant testing (wins/losses counts) on accuracy between J&B and other classification models is shown in Table 3.13. **W**: our approach was significantly better than compared algorithm; **L**: the selected rule-learning algorithm significantly outperformed our algorithm; **N**: no significant difference has been detected in the comparison.

Table 3.13 illustrates that the performance of the J&B method on accuracy was better than DTNB, DT, RDR and SA methods. Although J&B obtained similar results to FR and CBA (there is no statistical difference on 8 datasets out of 12), it

Table 3.12: The comparison between J&B method and other classification algorithms on accuracy.

Dataset	DTNB	DT	C4.5	PT	FR	RDR	CBA	SA	J&B	OC
Breast.Can	70.4±4.1	69.2±6.7	75.0±6.9	74.0±4.0	75.1±5.3	71.8±5.7	71.9±9.8	79.3±4.4	80.5±4.7	88.7
Balance	81.4±8.1	66.7±5.0	64.4±4.3	76.2±5.6	77.5±7.6	68.5±4.3	73.2±3.8	74.0±4.1	74.1±2.6	86.3
Car.Evn	95.4±0.8	91.3±1.7	92.1±1.7	94.3±1.0	91.8±1.1	91.0±1.8	91.2±3.9	86.2±2.1	89.4±1.4	94.2
Vote	94.7±3.4	94.9±3.7	94.7±4.4	94.8±4.2	94.4±2.8	95.6±4.1	94.4±2.6	94.7±2.3	94.1±1.8	92.8
Tic-Tac-Toe	69.9±2.7	74.4±4.4	85.2±2.7	94.3±3.3	94.1±3.1	94.3±2.9	100.0±0.0	91.7±1.5	95.8±2.0	100.0
Nursery	94.0±1.5	93.6±1.2	95.4±1.4	96.7±1.7	91.0±1.4	92.5±1.5	92.1±2.4	91.6±1.2	89.6±1.1	85.6
Hayes-root	75.0±7.2	53.4±8.3	78.7±8.4	73.1±9.7	77.7±8.7	74.3±7.1	75.6±10.9	73.1±6.0	79.3±5.9	80.7
Lymp	72.9±9.0	72.2±8.3	76.2±8.7	81.7±9.0	80.0±8.2	78.3±7.3	79.0±9.7	73.7±5.1	80.6±5.7	90.1
Spect.H	79.3±2.7	79.3±1.6	80.0±9.0	80.4±5.6	80.4±2.2	80.4±2.2	79.0±1.6	79.1±2.1	79.7±3.1	94.8
Adult	73.0±4.1	82.0±2.3	82.4±4.7	82.1±4.7	75.2±3.2	80.8±2.7	81.8±3.4	80.8±2.6	80.8±2.9	91.7
Chess	93.7±3.0	97.3±3.1	98.9±3.6	98.9±3.1	96.4±2.1	95.8±3.3	95.4±2.9	92.2±3.8	94.6±2.7	100.0
Connect4	78.8±5.9	76.7±7.7	80.0±6.8	81.1±7.9	80.6±7.1	80.0±6.4	80.9±8.1	78.7±6.0	81.2±5.2	90.6
Average(%)	81.5±4.4	79.3±4.5	83.6±5.2	85.6±4.9	84.5±4.4	83.6±4.1	84.5±4.9	82.3±3.4	84.9±3.3	91.3

Table 3.13: Statistically significant wins/losses counts of J&B method on accuracy

	DTNB	DT	C4.5	PT	FR	RDR	CBA	SA
W	6	6	4	2	2	4	2	6
L	3	2	3	4	2	1	2	1
N	3	4	5	6	8	7	8	5

is beaten by the PT algorithm according to win/losses counts. However, on average, the classification accuracies of J&B are not much different from those of the other 8 rule-learners.

Results show (Table 3.14) that J&B generates a reasonably smaller number of compact rules which is not sensitive to the size of the training datasets while the number of rules produced by traditional classification methods such as “DT”, “C4.5”, “PT”, “DTNB” algorithms depend on the dataset’s size. Even though not achieving the best classification accuracies on “Nursery” and “Adult” datasets, it produced the lowest number of rules on those datasets among all classification models.

Table 3.14: The number of classification rules generated by the classifiers

Dataset	DTNB	DT	C4.5	PT	FR	RDR	CBA	SA	J&B
Breast.Can	122	22	10	20	13	13	63	20	47
Balance	31	35	35	27	44	22	77	45	79
Car.Evn	144	432	123	62	100	119	72	160	41
Vote	270	24	11	8	17	7	22	30	13
Tic-Tac-Toe	258	121	88	37	21	13	23	60	14
Nursery	1240	804	301	172	288	141	141	175	109
Hayes-root	5	8	22	14	11	10	34	45	34
Lymp	129	19	20	10	17	11	23	60	29
Spect.H	145	2	9	13	17	12	4	50	11
Adult	737	1571	279	571	150	175	126	130	97
Chess	507	101	31	29	29	30	12	120	24
Connect4	3826	4952	3973	3973	403	341	349	600	273
Average(%):	618	674	409	411	93	75	79	125	64

Statistically significant counts (wins/losses) of the J&B against to other rule-based classification models on classification rules is shown in Table 3.15.

Table 3.15: Statistically significant wins/losses counts of J&B method on rules

	DTNB	DT	C4.5	PT	FR	RDR	CBA	SA
W	10	7	6	6	8	5	7	10
L	2	5	4	6	4	5	3	2
N	0	0	2	0	0	2	2	0

Table 3.15 proves that J&B produced statistically smaller classifiers than DTNB, SA methods on 10 datasets and DT, FR, CBA methods on 7 (or more than 7) datasets out of 12. Most importantly, J&B generated statistically smaller classifiers than all other models on bigger datasets, which was our main goal in this research. Experimental evaluations on bigger datasets (over 10000 samples) are shown in Figure 3.5.

Figure 3.5 proves the advantage of the J&B method, it produced the smallest classifier among all rule-based classification models on selected datasets.

Our experiment on relevance measure (average results) such as “Precision”, “Recall” and “F-measure” is highlighted in Figure 3.6. The detailed result for each dataset can be found in appendix A.

J&B produces classifiers that have on average far fewer rules than those produced by the “classical” rule-learning methods and a slightly smaller number of rules compared to associative classification models included in the comparison. J&B generated a reasonably smaller number of rules on bigger datasets compared to all other classification methods. J&B achieved the best accuracy on “Breast.Can”, “Hayes-root” and “Connect4” datasets among all methods (80.5%, 79.3%, and 81.2% respectively), while it got slightly worse results on “Nursery” and “Car.Evn” datasets. On other datasets, J&B achieved almost similar results with all rule-based classification methods.

We used the WEKA software to generate class association rules and to explore other classification algorithms because WEKA is open-source software that includes all the necessary classification approaches and it is relatively easy to use and program. Our proposed algorithm has some advantages and limitations:

- In this research work, one of the novel approaches is to apply the intended coverage constraint which becomes the stopping criterion for selecting rules for our classifier. We aim here to show that it is possible to achieve good accuracy without covering the whole dataset (especially when the dataset has some missing or noisy examples). This criterion becomes an advantage of our method that produces significantly smaller number of classification rules on bigger datasets.
- The J&B produced an accurate and compact classifier on datasets which have a large number of attributes and records (especially with balanced datasets on class-distribution), while the performance of J&B is slightly worse with datasets having a small number of attributes and a large number of records (proved on “Nursery” and “Car.Evn” datasets).
- To get “enough” class association rules (“enough” means at least 5-10 rules for each class value, this situation mainly happens with imbalanced datasets) for each class value and achieve a reasonable overall coverage, we need to apply appropriate minimum support and minimum confidence thresholds for J&B method.
- We tried to solve the outer-class overlapping problem (that means some samples from different classes have very similar characteristics) by grouping the CARs

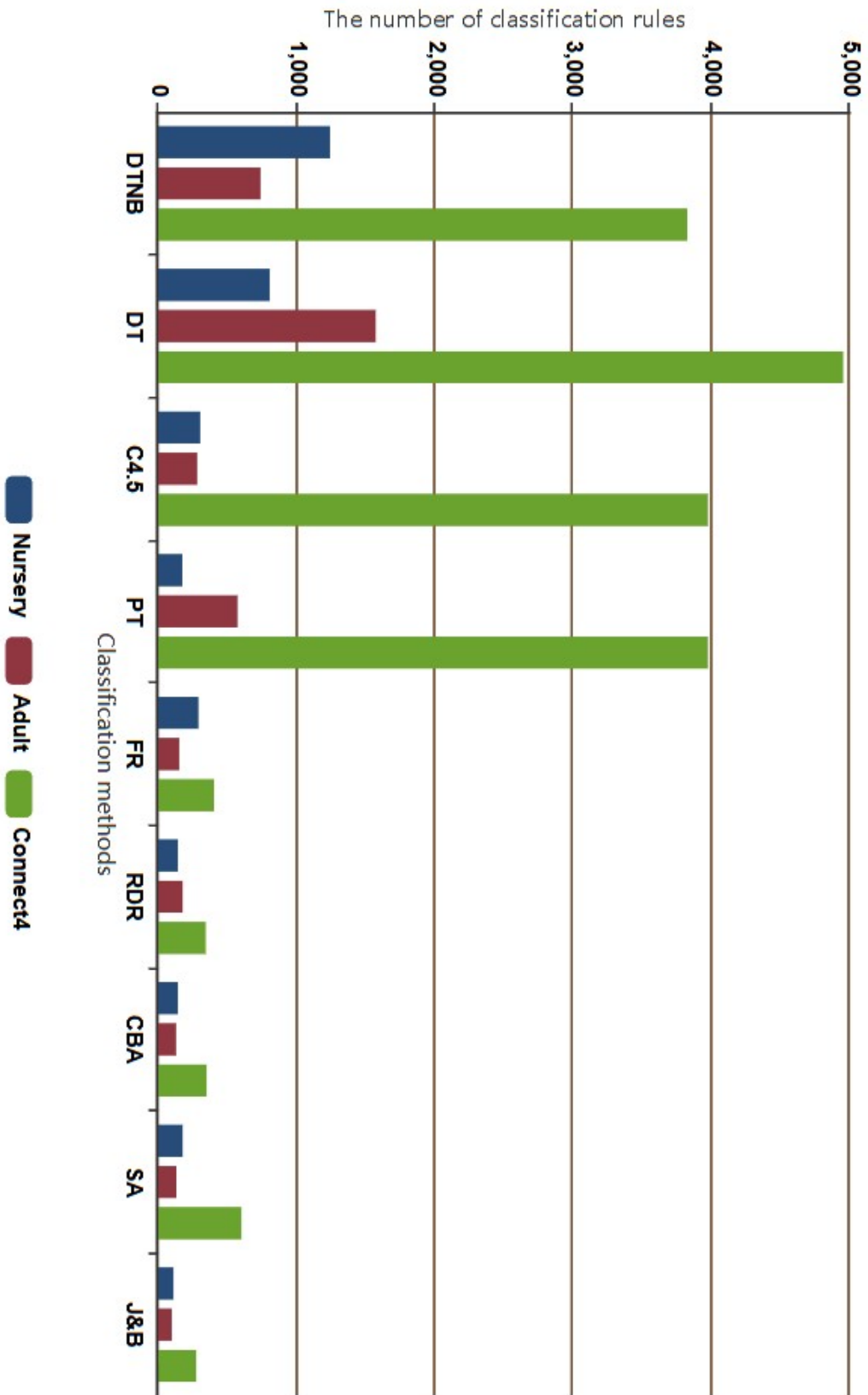


Figure 3.5: Comparison of rule-based classification methods on the average number of rules.

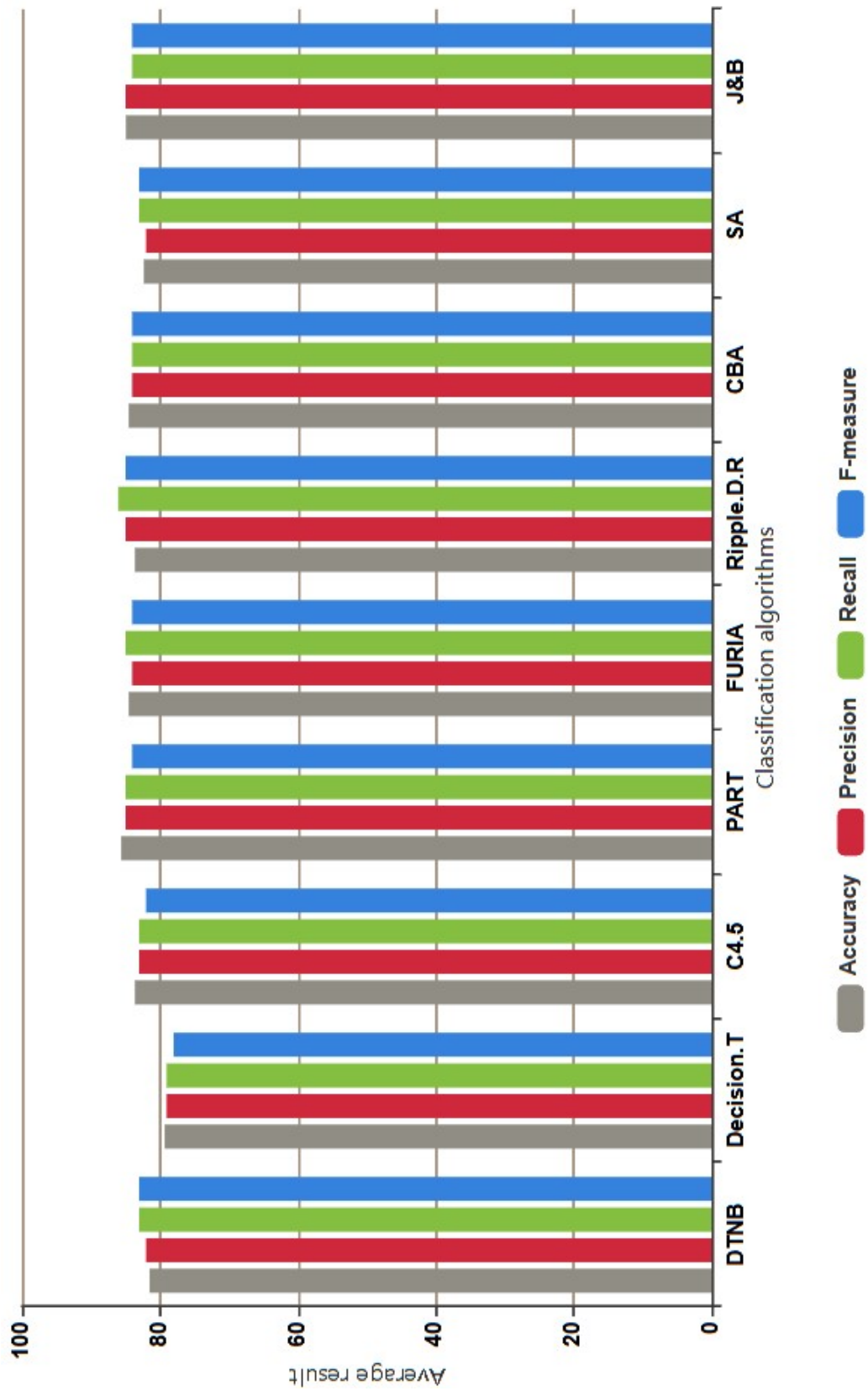


Figure 3.6: Comparison of the J&B classifier on accuracy, precision, recall and f-measure.

according to their class value, and the inter-class overlapping problem (several rules that belong to the same class may cover the same samples) by selecting the rules based on dataset coverage.

- Our method may get slightly worse results in terms of classification accuracy than other traditional classification methods on imbalanced datasets, because unbalanced class distribution may affect the rule-generation part of our algorithm.

Our experiments on accuracy and number of rules show that our method is compact, accurate and comparable with other 8 well-known classification methods. Although it did not achieve the best average classification accuracy, it produced significantly smaller number of rules on bigger datasets compared to other classification algorithms. Our proposed classifier achieved reasonably high average coverage with 91.3%.

Statistical significance testing shows that our method was statistically better than or equal to other classification methods on some datasets, while it obtained worse results than those methods on some other datasets. The most important achievement in this research was that J&B got significantly better results in terms of an average number of classification rules than all other classification methods while it had comparable results to those methods on accuracy.

Chapter 4

Distance Metrics

Distance metrics play an important role in clustering purposes, especially when the hierarchical clustering methods are used. Since we intend to apply hierarchical agglomerative clustering, we must define a way of measuring the similarity between CARs, that is, how far two rules are apart. Unfortunately, there are no distance metrics for class association rules. However, researchers have proposed some indirect distance metrics [12] for association rules. In this section, we discuss the existing distance metrics and our proposed (novel) distance metrics. Distance metrics for class association rules can be divided into two parts: based on indirect measures (obtained from the data) and direct measures (such as support, confidence, lift and so on).

4.1 Indirect Distance Metrics

To cluster rules, we first need a way of quantifying how far two rules are apart. Indirect distance metrics are basically built based on indirect measures that are computed from the data. Unfortunately, there is no intuitive distance metric for class association rules. A Euclidean distance could be defined on rule features such as support, confidence, lift or the bit-vector representation of both sides of the rule (BS). These direct features are very limited in capturing the interaction of rules on the data and characterize only a single rule. However, for a deeper analysis of the relationship of two rules, we have to resort to their common origin: the data. We call rule distances that are obtained from the data Indirect Distance Metrics. An indirect distance is defined as a function of the market basket (MB) sets that support the two considered rules. For the purpose of distance computation, we assign all rules with the same set of supporting market-baskets the same association rule identifier. For example, the following six rules are considered equivalent, because all involve the item-set $\{A, B, C\}$ and all have the same set of supporting market-baskets.

$$A \Rightarrow B, C$$

$$A, B \Rightarrow C$$

$$A, C \Rightarrow B$$

$$B \Rightarrow A, C$$

$$B, C \Rightarrow A$$

$$C \Rightarrow A, B$$

The following four indirect distance metrics for association rules are discussed in this section, namely, Absolute Market Basket Difference (AMBD), Conditional Market basket Probability (CMBP), and Conditional Market basket Log-likelihood (CMBL) and Tightness.

A. Absolute Market basket Difference Distance

In the context of association rule mining [58] introduced their approach to alleviate the rule quantity problem through pruning and grouping techniques. Their pruning technique is based on the creation of so-called rule covers which are related to the pruning of redundant rules. In short, rules are discarded when more general rules exist which cover the same database entries. However, after pruning their initial rule sets, [58] still faced a large number of remaining rules, therefore, they decided to cluster them. They introduced the simple distance measure between association rules with the same consequent. Let $rule1 : A \Rightarrow C$ and $rule2 : B \Rightarrow C$ be two association rules. They defined the distance between rules in terms of the number of market-baskets that they differ in (meaning one rule is supported, but not the other). Based on the number of non-overlapping market-baskets, a distance metric d^{AMBD} between $rule1$ and $rule2$ can be defined by the following equation:

$$d_{rule1,rule2}^{AMBD} = |m(BS_{rule1})| + |m(BS_{rule2})| - 2 * |m(BS_{rule1}, BS_{rule2})| \quad (4.1)$$

Where BS is the both side of the rule, that is, the itemset for the association rule and $m(BS)$ is the support of that rule.

Equation 4.1 illustrates that rules valid for exactly the same baskets have a distance of zero. Rules applying to disjoint sets of baskets have a distance equal to the sum of the numbers of transactions for which each rule is valid. There are several problems with this measure. One such problem is that it grows as the number of market-baskets in the database increases. This can be corrected by normalizing (divide the measure by the size of the database) and it is appropriate for rules only with the same consequent while this approach is intuitive. However, the measure is still strongly correlated with support. High support rules will on average tend to have higher distances to everybody else. This is an undesired property. For example, two pairs of rules, both pairs consisting of non-overlapping rules, may have different distances. High support pairs have a higher distance than low support pairs.

B. Conditional Market basket Probability Distance

Based on the previous approach, another indirect distance measure [12] is proposed as an improvement of [58] using the support values of two association rules. They tried to solve two problems occurring in [58], that it grows as the database grows and that, due to the focus on support values, rules with a high support will on average tend to have higher distances to everybody else. To solve the above-mentioned problems they proposed the new indirect distance metric based on conditional probabilities. Using a probability estimate for distance computation has many advantages. Probabilities are well understood, are intuitive, and a good measure for further processing. The distance d^{CMBP} between two rules $rule1$ and $rule2$ is the

(estimated) probability that one rule does not hold for a market basket, given at least one rule holds for the same basket. This distance is defined in equation 4.2:

$$d_{rule1,rule2}^{CMBP} = 1 - \frac{|m(BS_{rule1}, BS_{rule2})|}{|m(BS_{rule1})| + |m(BS_{rule2})| - |m(BS_{rule1}, BS_{rule2})|} \quad (4.2)$$

With this metric, rules having no common MBs are at a distance of 1, and rules valid for an identical set of baskets are at a distance of 0. This metric can also be interpreted as a normalized AMBD. The CMBP is computed as the AMBD divided by the number of baskets in the joint set of each rule’s supporting basket set. The CMBP does not suffer from the support correlation problem of AMBD. Let us call a distance interesting if it is neither 0 nor 1. Rule pairs with an interesting distance are called good neighbors. In most real databases, the majority of all rule pairs are not good neighbors. Manual exploration of a rule’s good neighbors showed that intuitive relatedness was captured very well by this metric. For example, rules involving different items but serving equal purposes were found to be close good neighbors. Super-set relationships of the item-sets associated to the rules often lead to very small distances.

C. Conditional Market basket Log-likelihood Distance

Due to its probabilistic nature, the range of the CMBP distance space is limited between 0 and 1. Natural distances usually range from 0 to infinity. Also, most of the interesting distances are in the open interval between 0.9 and 1 and the majority of the distances are exactly 1. This property may induce a high embedding dimensionality and, hence, cause problems in the clustering stage. A better-behaved distance d^{CMBL} between two rules *rule1* and *rule2* is given by the log-likelihood distance, which is defined by equation 4.3:

$$d_{rule1,rule2}^{CMBL} = -\log\left(\frac{|m(BS_{rule1}, BS_{rule2})|}{|m(BS_{rule1})| + |m(BS_{rule2})| - |m(BS_{rule1}, BS_{rule2})|}\right) \quad (4.3)$$

Equation 4.3 defines that non-overlapping rules are at a distance of infinity. Rules with identical support have a distance of 0. This metric monotonically spreads the CMBP range of 0 to 1 onto 0 to infinity. Thus, at any point the distance-based ordering or ranking will remain the same as for the CMBP. However, the area of good neighborhood is ‘magnified’. Thus, interesting patterns in this region may be easier to identify for the user when visualized. Also, this seems better suitable for the clustering discussed later.

D. Tightness

For the purpose of clustering association rules [57] presented a new measure called tightness which quantifies the strength of binding between the items of an association rule. The idea is that certain items in an application domain might get bound together because they are so strongly related that they often occur together in transactions. This tightness of binding is not covered by traditional measures like support or confidence. Support, on the one hand, does not consider transactions that contain only some of the bound items and confidence only describes the predictive ability of a rule.

Let $\{x_1x_2\dots x_m\} \Rightarrow \{x_{m+1}\dots x_n\}$ be an association rule and the x_i , $i = 1\dots n$ its respective items. S_{x_i} denotes the support of item x_i . Support values for the most and least frequent items of an association rule r are given by $S_{r_{\max}} = \max(S_{x_1}, \dots, S_{x_n})$ and $S_{r_{\min}} = \min(S_{x_1}, \dots, S_{x_n})$. An increase in support of a rule S_r tightens the binding between its constituent items, therefore an increase in S_r also increases the tightness measure. On the other hand consider the expression $\left(\frac{S_{r_{\max}}+S_{r_{\min}}}{2}\right) - S_r$. It roughly estimates the presence of items of r in other transactions that are not covered by r . That is if this value increases, items of r occur more often separately in different transactions and therefore the tightness between those items decreases. Combining these two effects yields the tightness measure:

$$T_r = \frac{S_r}{\left(\frac{S_{r_{\max}}+S_{r_{\min}}}{2}\right) - S_r} \quad (4.4)$$

T_r reaches its maximum (i.e. ∞) when $S_{r_{\max}} = S_{r_{\min}} = S_r$. Based on the notion of tightness the following distance measure has been introduced:

$$d(r_i, r_j) = \frac{|T_{r_i} - T_{r_j}|}{T_{r_i} + T_{r_j}} \quad (4.5)$$

As [57] have shown in the context of MB analysis, equation (9) is able to discover similar purchasing behavior in different item domains. That is, it groups association rules across several item domains.

4.2 The New “Direct” Distance Metric

In 4.1, we discussed the existing indirect distance metrics. In this section, we propose a new distance metric based on direct measures (such as support, confidence, lift, rule items and so on).

Our main goal of proposing the direct distance metrics is to cluster class association rules and reduce the number of rules. When we apply indirect distance measures described in previous section to cluster the CARS, we get a larger number of clusters, that is, our proposed classifier includes a larger number of rules, and this was a conflict to our main goal. Therefore, we propose a new normalized Item Based Distance Metric (IBDM) in this research work by considering the differences in rule items.

Let $R = \{r_1, r_2, \dots, r_n\}$ be a set of class association rules found from relational dataset D that are defined by $A = \{a_1, a_2, \dots, a_m\}$ distinct items (attribute’s value) classified into $C = \{c_1, c_2, \dots, c_l\}$ known classes. Each rule is denoted as follows:

$$r = \{x_1, x_2, \dots, x_k\} \rightarrow \{c\}, \text{ where, } \{x_1, x_2, \dots, x_k\} \subseteq A \text{ and } c \in C \text{ for } \forall r \in R.$$

Given two rules $rule1, rule2 \in R$:

$$rule1 = \{y_1, y_2, \dots, y_s\} \rightarrow \{c\}$$

$$rule2 = \{z_1, z_2, \dots, z_t\} \rightarrow \{c\}$$

where $\{y_1, y_2, \dots, y_s\} \subseteq .A$, $\{z_1, z_2, \dots, z_t\} \subseteq .A$, and $c \in C$. We compute the similarity between $rule1$ and $rule2$ as follows:

$$\text{distance}_q(\text{rule1}, \text{rule2}) = \begin{cases} \text{if } y_q = z_q \mid y_q = \emptyset \ \& \ z_q = \emptyset, & 0; \\ \text{else if } y_q = \emptyset \ \& \ z_q \neq \emptyset \mid y_q \neq \emptyset \ \& \ z_q = \emptyset, & 1; \\ \text{else } 2 \ (y_q \neq z_q). & \end{cases} \quad (4.6)$$

Where q is the index of rule items that cannot exceed *border* value (computed below).

Equation 4.6 expresses how close two rules are one from another. If rules have similar items, then the distance function has a low value. An empty rule item is considered closer than a different rule item.

$$\text{border} = \text{Max}(s, t); \quad (4.7)$$

border is the length of the longest rule, equation 4.7 is used to normalize the distance metric. The distance between two rules is denoted as follows:

$$d_{\text{rule1}, \text{rule2}}^{\text{IBDM}} = \frac{\sum_{i=1}^{\text{border}} \text{distance}_i}{2 \times \text{border}} \quad (4.8)$$

Distance (4.8) ranges between 0 and 1. CARs having the same items and the same size are at a distance of 0, CARs containing the different items are at a distance of 1.

4.3 The New ‘‘Combined’’ Distance Metric

We analyzed both direct and indirect distance metrics by experiments (we applied those distance measures for clustering the rules) and found that our final classifier tends to get a larger number of rules with higher coverage when we utilize the indirect distance metrics. On the other hand, we got slightly lower coverage with smaller number of rules (comparing to the experiment where we applied indirect distance metric) when we applied the direct distance metric. That is why we decided to propose a new distance metric by combining the direct and indirect distance metrics for producing the compact classifier while maintaining the overall coverage.

Since conditional market-basket distance is appropriate for the rules having the same consequent, we decided to propose a new Weighted and Combined Distance Metric (WCDM) by combining direct (IBDM) and indirect distance (CMBP) measures. When we applied CMBP distance to our proposed method, we got a larger number of clusters on some datasets. WCDM combines direct measure (rule items) and indirect measure (rule coverage). Both distance metrics (IBDM and WCDM) have their advantage: on some datasets IBDM produces the lower number of rules with higher accuracy while WCDM achieves better results on some other datasets. The weighted distance d^{WCDM} between two rules *rule1* and *rule2* is defined as follows:

$$d_{\text{rule1}, \text{rule2}}^{\text{WCDM}} = \alpha \times d_{\text{rule1}, \text{rule2}}^{\text{IBDM}} + (1 - \alpha) \times d_{\text{rule1}, \text{rule2}}^{\text{CMBP}} \quad (4.9)$$

where, α is a weighted measure. We set $\alpha = 0.5$ parameter, the final weighted and combined distance measure is described as follows:

$$d_{rule1,rule2}^{WCDM} = 0.5 \times d_{rule1,rule2}^{IBDM} + 0.5 \times d_{rule1,rule2}^{CMBP}. \quad (4.10)$$

Equation 4.10 describes that if both direct and indirect distances are 0 (that means if two rules have the same items or cover the same examples), then the proposed distance d^{WCDM} gets 0. d^{WCDM} gets 1 when two rules have dissimilar items or cover the non-overlapping examples.

Chapter 5

Identifying the Clusters of CARs

Once the class associations rules are generated and the distance metrics are proposed, our next goal is to cluster class association rules discussed widely in [10, 15, 21, 29] by using some clustering method. Clustering is a machine learning technique that involves the grouping of data points, that is, grouping of CARs. Given a set of CARs, we can use a clustering algorithm to classify each CAR into a specific group. In theory, CARs that are in the same group should have similar properties and/or features, while CARs in different groups should have highly dissimilar properties and/or features. Clustering is a method of unsupervised learning and is a common technique for statistical data analysis used in many fields. Clustering algorithms are basically split into two parts, partitional and hierarchical. We discuss both methods in the following subsections.

5.1 Partitional clustering algorithms

Given D , a data set of n objects, and k , the number of clusters to form, a partitioning algorithm organizes the objects into k partitions ($k \leq n$), where each partition represents a cluster. The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are “similar”, whereas the objects of different clusters are “dissimilar” in terms of the data set attributes.

k-Means [8] and k-Medoids [16] are the most well-known partitional clustering algorithms. It is taught in a lot of introductory data science and machine learning classes. K-Means is easy to understand and implement in code. The K-Means algorithm takes k as an input parameter and partitions the given n objects into k clusters. So, resulting intra cluster similarity is high but the inter-cluster similarity is low. Cluster similarity is calculated by the *mean* value of the objects in a cluster, that can be viewed as the cluster’s centroid.

The k-means algorithm proceeds as follows: First, it randomly selects k of the objects, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean. It then computes the new mean for each cluster. This process iterates until the criterion function

converges. Typically, the **square-error criterion** is used, defined as

$$E = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2 \quad (5.1)$$

where E is the sum of the square error for all objects in the data set; p is the point in space representing a given object; and m_i is the mean of cluster C_i (both p and m_i are multidimensional). In other words, for each object in each cluster, the distance from the object to its cluster center is squared, and the distances are summed. This criterion tries to make the resulting k clusters as compact and as separate as possible. The k-means procedure is summarized in algorithm 10.

Algorithm 10 The k-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input: k : the number of clusters, D : a data set containing n objects

Output: A set of k clusters

- 1: Arbitrarily choose k objects from D as the initial cluster centers;
 - 2: **repeat**
 - 3: (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
 - 4: update the cluster means, i.e., calculate the mean value of the objects for each cluster;
 - 5: **until** no change.
-

Example of clustering by k-means partitioning: Let us suppose that there is a set of objects located in space as depicted in the rectangle shown in Figure 5.1 (a). Let $k = 3$; that is, the user would like the objects to be partitioned into three clusters.

According to the Algorithm 10, we arbitrarily choose three objects as the three initial cluster centers, where cluster centers are marked by a “+”. Each object is distributed to a cluster based on the cluster center to which it is the nearest. Such a distribution forms silhouettes encircled by dotted curves, as shown in Figure 5.1 (a). Next, the cluster centers are updated. That is, the mean value of each cluster is recalculated based on the current objects in the cluster. Using the new cluster centers, the objects are redistributed to the clusters based on which cluster center is the nearest. Such a redistribution forms new silhouettes encircled by dashed curves, as shown in Figure 5.1 (b). This process iterates, leading to Figure 5.1 (c). The process of iteratively re-assigning objects to clusters to improve the partitioning is referred to as iterative relocation. Eventually, no redistribution of the objects in any cluster occurs, and so the process terminates. The resulting clusters are returned by the clustering process.

The algorithm attempts to determine k partitions that minimize the square-error function. It works well when the clusters are compact clouds that are rather well separated from one another.

The K-Means algorithm is sensitive to outliers because an object with large value may substantially deform the distribution of data. We can treat this problem with K-

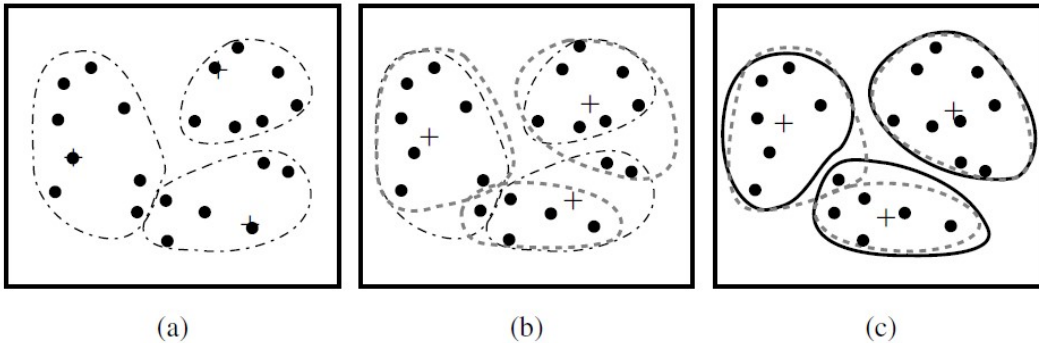


Figure 5.1: Clustering of a set of objects based on the k-means method. (The mean of each cluster is marked by a “+”.)

medoids algorithm. Instead of taking mean value of the objects as a reference point, we can pick actual objects to represent the clusters by using one representative object per cluster. Each remaining object is clustered with the representative object. The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object and its corresponding reference point. That is, an absolute-error criterion is used, defined as

$$E = \sum_{j=1}^k \sum_{p \in C_j} |p - o_j| \quad (5.2)$$

where E is the sum of the absolute error for all objects in the data set; p is the point in space representing a given object in cluster C_j ; and o_j is the representative object of C_j . In general, the algorithm iterates until, eventually, each representative object is actually the medoid, or most centrally located object, of its cluster. This is the basis of the k-medoids method for grouping n objects into k clusters.

K-Means has some advantages in that it is pretty fast, easy to compute the distances between points and group centers, and utilizes very few computations. On the other hand, K-Means has a couple of disadvantages: firstly, you have to select how many groups/classes there are in advance. This is not always trivial and ideal with a clustering algorithm. Secondly, K-means starts with a random choice of cluster centers and therefore it may yield different clustering results on different runs of the algorithm, thus, the results may not be repeatable and lack consistency.

5.2 Hierarchical clustering algorithms

Hierarchical clustering algorithms [27] are split into 2 parts: top-down (*divisive hierarchical clustering*) and bottom-up (*hierarchical agglomerative clustering*). Bottom-up algorithms initially assume each example as a single cluster and then merge the two closest clusters in every iteration until all clusters have been merged into a unique cluster that contains all examples. The resulting hierarchy of clusters is represented

as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the examples; the leaves are considered as clusters with only one sample. The top-down approach is the opposite of the hierarchical agglomerative clustering method. It considers all examples in a single cluster, and then it splits the clusters into smaller parts until each example forms a cluster or until it satisfies the stopping condition.

Example of agglomerative divisive hierarchical clustering. Figure 7.6 shows the application of agglomerative nesting, an agglomerative hierarchical clustering method, and divisive analysis, a divisive hierarchical clustering method, to a data set of five objects, $\{a, b, c, d, e\}$. Initially, each object is placed into unique cluster in agglomerative nesting method. The clusters are then merged step-by-step according to some criterion. For example, clusters C_1 and C_2 may be merged if an object in C_1 and an object in C_2 form the minimum Euclidean distance between any two objects from different clusters. This is a single-linkage approach in that each cluster is represented by all of the objects in the cluster, and the similarity between two clusters is measured by the similarity of the closest pair of data points belonging to different clusters. The cluster merging process repeats until all of the objects are eventually merged into one cluster. In divisive analysis, all of the objects are used to form one initial cluster. The cluster is split according to some principle, such as the maximum Euclidean distance between the closest neighboring objects in the cluster. The cluster splitting process repeats until, eventually, each new cluster contains only a single object.

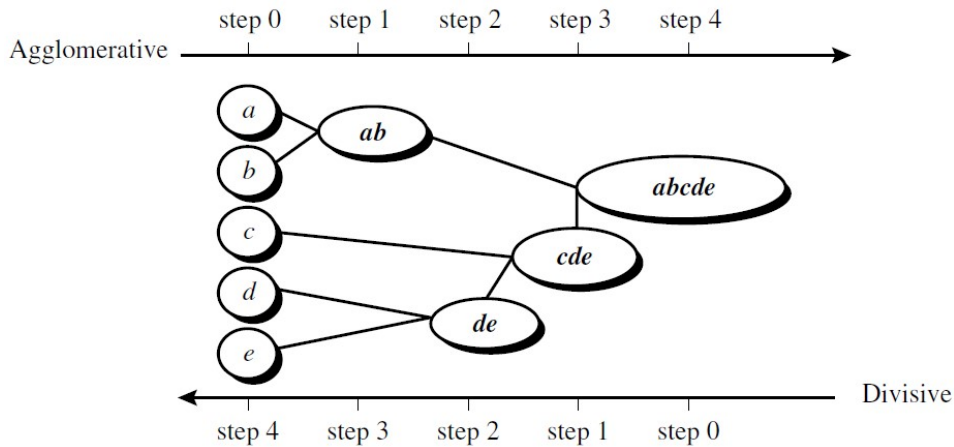


Figure 5.2: Agglomerative and divisive hierarchical clustering on data objects $\{a, b, c, d, e$

}

In either agglomerative or divisive hierarchical clustering, the user can specify the desired number of clusters as a termination condition.

A tree structure called a dendrogram is commonly used to represent the process of hierarchical clustering. It shows how objects are grouped together step by step. Figure 5.3 shows a dendrogram for the five objects presented in Figure 5.2, where $l = 0$ shows the five objects as singleton clusters at level 0. At $l = 1$, objects a and b are grouped together to form the first cluster, and they stay together at all

subsequent levels. We can also use a vertical axis to show the similarity scale between clusters. For example, when the similarity of two groups of objects, $\{a, b\}$ and $\{c, d, e\}$, is roughly 0.16, they are merged together to form a single cluster.

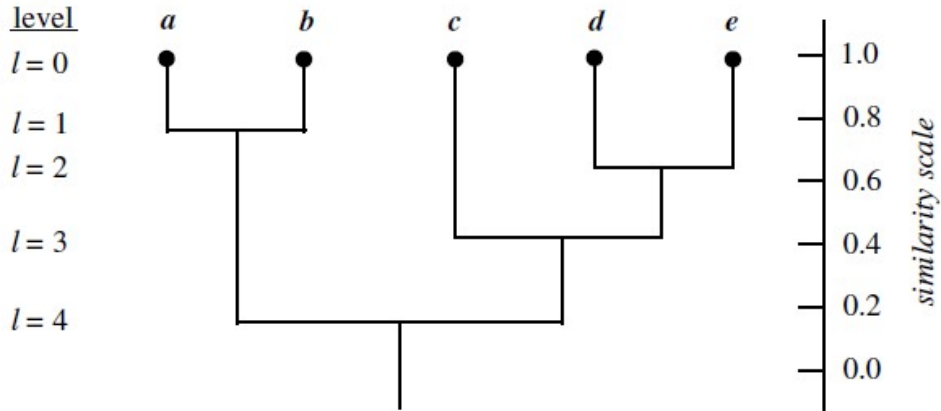


Figure 5.3: Dendrogram representation for hierarchical clustering of data objects $\{a, b, c, d, e\}$

Hierarchical clustering does not require us to specify the number of clusters and we can even select which number of clusters looks best since we are building a tree. Additionally, the algorithm is not sensitive to the choice of distance metric; all of them tend to work equally well whereas with other clustering algorithms, the choice of distance metric is critical.

Since hierarchical clustering algorithms are more consistent than partitional algorithms, we apply the complete linkage method of hierarchical agglomerative clustering in our research work. In the complete linkage (farthest neighbor) method, the similarity of two clusters is the similarity of their most dissimilar examples, therefore, the distance between the farthest groups is taken as an intra-cluster distance. We assume that we have given $N \times N$ distance matrix d , where N is the total number of rules (that is, total number of clusters). The clusters are numbered $0, 1, \dots, (N-1)$ and m is the sequence number of clusters. $L[k]$ is the level of the k -th clustering and the distance between two clusters $cl1$ and $cl2$ is defined as $d[cl1, cl2]$. Complete linkage of the hierarchical agglomerative clustering algorithm is outlined in Algorithm 11.

We need to apply the hierarchical clustering algorithm twice: first, we apply the *AHCCLH* algorithm to find the cluster heights that we will use later to identify the optimal number of clusters. In this case, the number of cluster $S=1$ and distance matrix are defined as input parameters. Because if $S=1$, then, *AHCCLH* iterates $N-1$ times to find the heights of all cluster. Second, *AHCCLC* is utilized to identify the cluster of class association rules. In *AHCCLC*, we provide the number of cluster S (found by using the cluster heights) and distance matrix to identify the clusters of CARs.

When we cluster the rules, we need to find the number of clusters. We get the optimal number of clusters by cutting the dendrogram at the point that represents

Algorithm 11 Agglomerative Hierarchical Clustering with Complete Linkage (AHCCLH: Heights || AHCCLC: Clusters)

Input: a distance matrix d and number of clusters S

Output: Cluster heights (AHCCLH), Cluster of CARs (AHCCLC)

- 1: **Initialization:** Each rule is a unique cluster C at level 0 ($L[0]=0$), sequence number $m=0$ and the optimal number of cluster S is identified, so, to get the intended number of clusters (S), the algorithm should iterate K times $K=N-S$;
 - 2: **Compute:** Find the most similar pair of clusters, $cl1$ and $cl2$ and merge them into a single cluster C to form the next clustering sequence m . Increase the sequence number by one: $m=m+1$ and set the new level $L[m]=d[cl1,cl2]$;
 - 3: **Update:** Update the distance matrix D , by removing the rows and columns corresponding to $cl1$ and $cl2$ and adding a new row and column corresponding to the new cluster. The distance between the new cluster ($cl1,cl2$) and old cluster k is calculated as $d[(cl1, cl2),k]=\max\{d[k,cl1], d[k,cl2]\}$;
 - 4: **Stopping condition:** if $m=K$ then **return** L (AHCCLH) || C (AHCCLC) and stop, otherwise go to step 2.:
-

the maximum distance between two consecutive cluster merges. The algorithm that identifies the optimal number of clusters is presented in Algorithm 12.

Algorithm 12 Computing the optimal number of clusters

Input: an array of cluster heights

Output: Optimal number of cluster

- 1: $Max_height_difference = cluster_height[1] - cluster_height[0]$;
 - 2: $Opt_number_of_cluster = 1$;
 - 3: $N = cluster_height.length$;
 - 4: **for** ($k = 2; k \leq N; k++$) **do**
 - 5: **if** $cluster_height[k] - cluster_height[k - 1] > Max_height_difference$ **then**
 - 6: $Max_height_difference = cluster_height[k] - cluster_height[k - 1]$;
 - 7: $Opt_number_of_cluster = N - k$;
 - 8: **end if**
 - 9: **end for**
 - 10: **return** $Opt_number_of_cluster$
-

The input to Algorithm 12 is a set of cluster distances that are calculated during the building of the dendrogram (so-called cluster “heights”). The output is the optimal number of clusters. In lines 1-3 the total number of clusters generated by hierarchical clustering is stored. Lines 4-7 outline the main part of the algorithm, $Opt_number_of_clusters$ gets to the point where the difference between two consecutive cluster heights will be maximum. Since we start from 0, $Opt_number_of_clusters$ is equal to $(N-k)$. The last line returns the obtained result.

Chapter 6

Identifying the Cluster Representative Class Association Rule

Once we have found all clusters, our final goal is to extract the representative class association rule (CAR) for each cluster to form our meaningful, compact and descriptive associative classifier. In this research work, we propose two methods of extracting the representative CAR: based on cluster center and based on dataset coverage.

6.1 Representative CAR based on cluster center (RCC)

In this method, we choose the CAR which is closer to the center of the cluster as a representative, that is, the representative CAR must have the minimum average distance to all other rules. Algorithm 13 defines the procedure.

The first line gets the number of CARs. We use the distance array “*Dist*” (line 2) to compute the distance from the selected CAR to all other CARs (we use the direct distance metric (IBDM) or combined distance metric (WCDM) described in Section 4). The initial value of *min_avg_distance* in line 3 is the maximum value of the integer and it is used to store the minimum average distance in line 10. Lines 4-9 find the index of the representative CAR that has the minimum average distance to all other rules and the last line returns the representative CAR.

6.2 Representative CAR based on dataset coverage (RDC)

We decided to propose this method to improve the overall coverage and classification accuracy, while the first method (representative CAR based on cluster center) (RCC) struggles to achieve reasonable coverage on some certain datasets. Since we are clustering similar rules having the same class value, it is unnecessary to think about the outer class overlapping problem (that means some samples from different classes

Algorithm 13 A Representative CAR based on Cluster Center (RCC)

Input: a set of class association rules in *CARs* array

Output: A representative class association rule

```

1:  $N = \text{CARs.length}$ ;
2:  $\text{Fill}(\text{Dist}, 0)$ ;
3:  $\text{min\_avg\_distance} = \text{Integer.Max.value}$ ;
4: for ( $i = 0; i \leq N; i++$ ) do
5:   for ( $j = 0; j \leq N; j++$ ) do
6:      $\text{Dist}[i] = \text{Dist}[i] + \text{IBDM}(\text{CARs}[i], \text{CARs}[j]) | \text{WCDM}(\text{CARs}[i], \text{CARs}[j])$ ;
7:   end for
8:    $\text{avg\_distance} = \text{Dist}[i] / N$ ;
9:   if  $\text{avg\_distance} < \text{min\_avg\_distance}$  then
10:     $\text{min\_avg\_distance} = \text{avg\_distance}$ ;
11:     $\text{representative\_CAR\_index} = i$ ;
12:   end if
13: end for
14: return  $\text{CARs}[\text{representative\_CAR\_index}]$ 

```

have very similar characteristics), but we should avoid the inter class overlapping problem (several rules that belong to same class may cover the same samples). We bypass this problem by selecting the representative CAR based on dataset coverage. First, we find a rule that has maximum dataset coverage, then we check if the first CAR classifies at least one new example, then we get it as a representative CAR, otherwise we continue. Once we find the representative CAR, we remove all the examples covered by it. The procedure is outlined in Algorithm 14.

In this approach (RDC), we first sort (line 2) the class association rules in coverage descending order, and we start checking rules from first to last (line 4). If a rule classifies at least one new example (line 13), we return that rule as a representative (line 13-18), otherwise we continue. If any rule cannot be a representative, then, the algorithm returns the first rule (line 3) which has the highest coverage as a representative.

6.3 Final associative classifier

After extracting the representative class association rules, we produce our meaningful, compact and descriptive model (classifier). Since we present two different distance metrics (direct and combined) and two methods of extracting the representative CAR, we produce three distinct models (associative classifiers) in our research work. Our proposed approach is represented in Algorithm 15.

Lines 1-2 generate the strong CARs that satisfy the user-specified minimum support and minimum confidence constraints from training dataset D by using the APRIORI algorithm. The third line sorts the CARs in confidence and supports descending order according to the following criteria:

R_1 and R_2 are two CARs, R_1 is said to have a higher rank than R_2 , denoted as

Algorithm 14 A Representative CAR based on Dataset Coverage (RDC)

Input: a set of class association rules in $CARs$ array, a training dataset D and $classified_traindata$ array

Output: A representative class association rule

```

1:  $N = CARs.length$ ;
2:  $CARs = \text{sort}(CARs, \text{coverage})$ ;
3:  $Representative\_CAR = CARs[1]$ ;
4: for ( $i = 1; i \leq N; i++$ ) do
5:   for ( $j = 0; j \leq D.length; j++$ ) do
6:     if  $classified\_traindata[j] = \text{false}$  then
7:       if  $CARs[i].premise \text{ classifies } D[j].premise$  then
8:          $classified\_traindata[j] = \text{true}$ ;
9:          $contribution = contribution + 1$ ;
10:      end if
11:    end if
12:  end for
13:  if  $contribution > 0$  then
14:     $Representative\_CAR = CARs[i]$ ;
15:    break;
16:  end if
17: end for
18: return  $Representative\_CAR$ 

```

Algorithm 15 Learning the proposed associative classifier

Input: A training dataset D , minimum support and minimum confidence

Output: Associative classifier

```

1:  $F = \text{frequent\_itemsets}(D, \text{minsup})$ ;
2:  $R = \text{genCARs}(F, \text{minconf})$ ;
3:  $R = \text{sort}(R, \text{minsup}, \text{minconf})$ ;
4:  $G = \text{Group}(R)$ 
5: for ( $i = 0; i \leq \text{number\_of\_class}; i++$ ) do
6:    $Distance = \text{IBDM}(G[i]) \mid \text{WCDM}(G[i])$ ;
7:    $Cluster\_heights = \text{AHCCLH}(Distance, 1)$ 
8:    $N = \text{optimal\_number\_of\_cluster}(Cluster\_heights)$ ;
9:    $Cluster = \text{AHCCLC}(N, Distance)$ ;
10:   $\text{Fill}(classified\_traindata, \text{false})$ ;
11:  for ( $j = 0; j \leq N; j++$ ) do
12:     $Y = \text{RDC}(Cluster[j], D, classified\_traindata) \mid \text{RCC}(Cluster[j])$ ;
13:     $\text{Associative\_Classifier.add}(Y)$ ;
14:  end for
15: end for
16: return  $\text{Associative\_Classifier}$ 

```

$$R_1 > R_2,$$

- If and only if, $conf(R_1) > conf(R_2)$; or
- If $conf(R_1) = conf(R_2)$ but, $supp(R_1) > supp(R_2)$; or
- If $conf(R_1) = conf(R_2)$ and $supp(R_1) = supp(R_2)$, R_1 has fewer attribute values in its left-hand side than R_2 does;
- If all the parameters of the rules are equal, we can choose any of them.

CARs are grouped according to their class label in line 4. For each group of CARs (lines 5-15), the distance matrix is constructed by using one of the distance measures defined in chapter 4 (line 6), the hierarchical clustering algorithm complete linkage method (*AHCCLH*) computes the cluster heights (distances between clusters) by using the distance matrix in line 7 and these heights (distances) are used to find the optimal number of clusters (line 8). Then, we apply the hierarchical clustering algorithm (*AHCCLC*) again to identify the cluster of CARs (a *Cluster* array stores the list of clustered CARs). Since we are clustering the class association rules class by class, we need “*classified_traindata*” array to store the information about classified examples, that is, we update this array for the same class only. When we start the clustering of CARs for a new class, we need to initialize the “*classified_traindata*” array. In lines 11-14, the representative CAR is extracted by using one of the methods described in section 6 for each cluster and added to our final classifier. The last line returns the descriptive, compact and meaningful classifier. The classification process of proposed methods is shown in Algorithm 16.

Algorithm 16 Classification process of proposed associative classifiers

Input: A Classifier and A test_example

Output: Predicted class

```

1: Fill(class_count, 0);
2: for each rule  $y \in$  Classifier do
3:   if  $y$  classify test_example then
4:     class_count[ $y$ .class]++;
5:   end if
6: end for
7: if  $\max(\textit{class\_count}) == 0$  then
8:   predicted_class = majority_classifier;
9: else predicted_class = max_index(class_count);
10: end if
11: return predicted_class

```

Algorithm 16 predicts the class label of the test example by using the classifier. The first line files the *class_count* array with 0 (the size of *class_count* array equals to the number of classes). For each rule in the classifier (line 1), if the rule can classify the example correctly, then we increase the corresponding class count by one and store it (lines 2-4). In lines 6-9, if none of the rules can classify the new example correctly, then the algorithm returns the majority class value. Otherwise, it returns the majority class value of correctly classified rules.

We built the following different classifiers: the first method (DC) is built based on Direct distance measure (IBDM) and the method for extracting a representative CAR is based on cluster Center (RCC), the second method (DDC) is formed based on Direct distance measure (IBDM) and the method for extracting a representative CAR is based on Dataset Coverage (RDC), and the third method (CDC) is formed based on Combined distance measure (WCDM) and the method for extracting a representative CAR is based on Dataset Coverage (RDC).

Chapter 7

Experimental Evaluations and Discussion

The achievement of the scientific objectives was validated by experimental evaluation. We tested our models on 17 real-life datasets taken from the UCI Machine Learning Database Repository. We evaluated our classifiers (DC, DDC and CDC) by comparing them with eight well-known rule-based classification algorithms (DTNB, DT, C4.5, PT, FR, RDR and CBA) on classification accuracy and the number of rules. All differences were tested for statistical significance by performing a paired t-test (with a 95% significance threshold).

Associative classifiers were run with default parameters *minimum support* = 1% and *minimum confidence* = 60% (on some datasets (marked with bold in the Table 7.1), however, *minimum support* was lowered to 0.5% or even 0.1% and confidence was lowered to 50% to ensure “enough” CARs (“enough” means at least 5-10 rules for each class value- this situation mainly happens with imbalanced datasets)). For all other classification models we used their WEKA workbench implementation with default parameters. The description of the datasets and input parameters are shown in Table 7.1.

All experimental results were produced by using a 10-fold cross-validation evaluation protocol. Experimental results on classification accuracies (average values over the 10-fold cross-validation with standard deviations) are shown in Table 7.2.

We can observe from Table 7.2 that our proposed associative classifiers achieved comparable accuracies (DC: 82.5%, DDC:83.3% and CDC:83.8% respectively) with other classification models on selected datasets. Interestingly, CDC significantly outperforms all rule-learners on the “Breast.Can” (except DDC), “Hayes-root” and “Lymp” datasets, while on the “Car.Evn”, “Nursery” and “Monks” datasets, our proposed methods obtained worse accuracy than all other algorithms. Standard deviations of accuracy results decreases with an increasing number of examples in a dataset, which is expected behavior. Standard deviation of all classification methods was a bit higher (above 4) on “Breast.Can”, “Hayes-root”, “Lymp” and “Connect4” datasets, that is, differences of accuracies over 10-fold cross-validation experiment fluctuated.

Statistically significant testing (wins/losses counts) on accuracy between DC and

Table 7.1: Default parameters of associative classifiers and datasets description.

Dataset	# of attributes	# of classes	# of records	Min support	Min confidence	# of analyzed rules
Breast.Can	10	2	286	1%	60%	1000
Balance	5	3	625	1%	50%	218
Car.Evn	7	4	1728	1%	50%	1000
Vote	17	2	435	1%	60%	500
Tic-Tac-Toe	10	2	958	1%	60%	3000
Nursery	9	5	12960	0.5%	50%	3000
Mushroom	23	2	8124	1%	60%	3000
Hayes-root	6	3	160	0.1%	50%	1000
Lymp	19	4	148	1%	60%	1500
Monks	7	2	432	1%	50%	800
Spect.H	23	2	267	0.5%	50%	3000
Abalone	9	3	4177	1%	60%	1000
Adult	15	2	45221	0.5%	60%	5000
Insurance	7	3	1338	1%	50%	722
Laptop	11	3	1303	1%	50%	1480
Chess	37	2	3196	0.5%	60%	3000
Connec4	43	3	67557	1%	60%	5000

other classification models is shown in Table 7.3. **W**: our approach was significantly better than the compared algorithm; **L**: selected rule-learning algorithm significantly outperformed our algorithm; **N**: no significant difference has been detected in the comparison.

Table 7.3 illustrates that the performance of the DC method on accuracy was better than DTNB and DT methods. Although DC obtained similar result to CBA and DDC (there is no statistical difference on 8 and 13 datasets out of 17), it is beaten by all other methods according to win/losses counts. However, on average, the classification accuracy of DC are not much different from those of the other 8 rule-learners.

The same experiment on DDC is shown in Table 7.4. Since DC is compared with DDC in Table 7.3, it is not included in Table 7.4.

It can be seen from the table that DDC’s performance on accuracy is better than DC. It outperformed the DTNB, DT and DC methods (by win/losses counts). Even though DDC got statistically worse results in terms of win/losses counts than RDR, PT and FR, there is no statistical difference on 8, 9 and 11 datasets out of 17 between them. DDC obtained almost the same result with C4.5 and CBA algorithms (4/4/9).

DC achieved (shown in Table 7.5) statistically comparable results in terms of average classification accuracy with “classical” and “associative” classification approaches. DC statistically lost to C4.5 and FR methods on 7 datasets out of 17, while it outperformed the rest of the algorithms except PT.

The comparison between our methods and other classification methods on the number of classification rules is shown in Table 7.6. Since DC and DDC differ in the representative CAR selection process, the number of classification rules generated by both methods stays the same. Thus, DC and DDC methods are merged in Table

Table 7.2: Overall accuracies with standard deviations.

Dataset	DTNB	DT	C4.5	PT	FR	RDR	CBA	DC	DDC	CDC
Breast Can	70.4±4.1	69.2±6.7	75.0±6.9	74.0±4.0	75.1±5.3	71.8±5.7	71.9±9.8	81.2±4.0	81.9±4.1	82.6±4.5
Balance	81.4±8.1	66.7±5.0	64.4±4.3	76.2±5.6	77.5±7.6	68.5±4.3	73.2±3.8	72.8±2.4	73.2±2.9	73.2±3.0
Car.Evn	95.4±0.8	91.3±1.7	92.1±1.7	94.3±1.0	91.8±1.1	91.0±1.8	91.2±3.9	85.8±1.4	88.5±1.2	87.1±1.6
Vote	94.7±3.4	94.9±3.7	94.7±4.4	94.8±4.2	94.4±2.8	95.6±4.1	94.4±2.6	92.9±2.5	93.2±2.8	90.6±2.3
Tic-Tac-Toe	69.9±2.7	74.4±4.4	85.2±2.7	94.3±3.3	94.1±3.1	94.3±2.9	100.0±0.0	87.3±1.3	91.8±1.0	92.4±1.1
Nursery	94.0±1.5	93.6±1.2	95.4±1.4	96.7±1.7	91.0±1.4	92.5±1.5	92.1±2.4	88.5±1.1	89.3±1.1	92.3±0.9
Hayes-root	75.0±7.2	53.4±8.3	78.7±8.4	73.1±9.7	77.7±8.7	74.3±7.1	75.6±10.9	79.9±5.7	77.8±5.2	82.7±6.1
Lymph	72.9±9.0	72.2±8.3	76.2±8.7	81.7±9.0	80.0±8.2	78.3±7.3	79.0±9.7	78.4±6.7	80.0±6.1	84.0±6.4
Spect.H	79.3±2.7	79.3±1.6	80.0±9.0	80.4±5.6	80.4±2.2	80.4±2.2	79.0±1.6	81.5±0.7	81.3±1.1	82.8±1.3
Abalone	62.1±1.3	61.8±1.5	62.3±1.2	62.3±1.1	61.7±1.6	60.8±0.8	61.1±1.0	61.0±1.1	60.7±1.0	60.7±1.2
Adult	73.0±4.1	82.0±2.3	82.4±4.7	82.1±4.7	75.2±3.2	80.8±2.7	81.8±3.4	81.9±2.4	82.0±2.6	82.8±3.0
Insurance	74.2±1.1	75.7±1.6	75.8±1.4	75.0±1.8	75.8±1.4	73.4±1.7	75.5±2.0	74.0±1.1	74.2±1.1	73.2±1.5
Monks	98.9±0.9	98.9±0.9	98.9±0.9	98.9±0.9	98.9±0.9	97.1±0.7	97.8±1.4	92.5±0.8	93.6±0.9	91.1±0.8
Laptop	75.7±2.6	72.9±2.9	75.3±2.3	74.5±2.9	75.4±2.1	73.2±1.8	75.4±2.0	71.6±2.1	73.8±2.3	72.6±1.7
Mushroom	98.8±0.3	100.0±0.0	100.0±0.0	100.0±0.0	99.0±0.3	100.0±0.0	92.9±1.6	95.6±0.7	97.2±1.0	96.5±0.5
Chess	93.7±3.0	97.3±3.1	98.9±3.6	98.9±3.1	96.4±2.1	95.8±3.3	95.4±2.9	97.0±2.0	97.3±2.0	97.8±2.0
Connect4	78.8±5.9	76.7±7.7	80.0±6.8	81.1±7.9	80.6±7.1	80.0±6.4	80.9±8.1	80.0±5.9	80.7±5.9	81.7±5.9
Average(%):	81.6±3.5	80.0±3.6	83.3±4.0	84.6±3.9	83.8±3.5	82.8±3.2	83.4±4.2	82.5±2.5	83.3±2.5	83.8±2.6

Table 7.3: Statistically significant wins/losses counts of DC method on accuracy

	DTNB	DT	C4.5	PT	FR	RDR	CBA	DDC	CDC
W	7	7	4	2	3	3	4	1	1
L	6	5	6	8	6	6	5	3	4
N	5	5	7	7	8	8	8	13	12

Table 7.4: Statistically significant wins/losses counts of DDC method on accuracy

	DTNB	DT	C4.5	PT	FR	RDR	CBA	CDC
W	6	6	4	2	2	3	4	2
L	4	4	4	6	4	6	4	3
N	7	7	9	9	11	8	9	12

Table 7.5: Statistically significant wins/losses counts of CDC method on accuracy

	DTNB	DT	C4.5	PT	FR	RDR	CBA
W	8	7	6	4	5	6	6
L	6	5	7	6	7	4	6
N	3	5	4	7	5	7	5

7.6.

Table 7.6: number of CARs

Dataset	DTNB	DT	C4.5	PT	FR	RDR	CBA	DC&DDC	CDC
Breast.Can	122	22	10	20	13	13	63	8	9
Balance	31	35	35	27	44	22	77	34	79
Car.Evn	144	432	123	62	100	119	72	32	32
Vote	270	24	11	8	17	7	22	6	6
Tic-Tac-Toe	258	121	88	37	21	13	23	24	17
Nursery	1240	804	301	172	288	141	141	79	80
Hayes-root	5	8	22	14	11	10	34	19	80
Lymp	129	19	20	10	17	11	23	5	7
Spect.H	145	2	9	13	17	12	4	8	5
Abalone	165	60	49	71	20	57	131	14	14
Adult	737	1571	279	571	150	175	126	13	88
Insurance	23	48	21	49	22	22	84	18	20
Monks	12	36	14	8	12	10	40	14	14
Laptop	101	101	72	60	28	32	41	19	18
Mushroom	50	50	26	12	11	8	8	7	11
Chess	507	101	31	23	29	30	12	12	17
Connect4	3826	4952	3973	3973	403	341	349	59	102
Average(%):	457	494	300	302	71	61	74	22	36

Experimental evaluations on the number of classification rules show that DC and

DDC significantly outperform all other rule-learners on 8 datasets out of 17 (except CDC) and it produces classifiers that have on average far fewer rules than those produced by the other 8 rule-learning methods included in the comparison. More precisely, our proposed methods achieved almost two times better result than FR, RDR and CBA on average number of rules, while this ratio was even bigger (more than 10 times) with DTNB, DT, C4.5 and PT algorithms,

CDC also achieved statistically best results on 8 datasets out of 17 on rules, although it got slightly worse result than DC&DDC in terms of average classification rules.

Our proposed methods generated a reasonably smaller number of rules on bigger datasets compared to other classification methods. Even though our approaches could not achieve the best classification accuracies on “Car.Evn”, “Nursery” and “Laptop” datasets, it produced the statistically smallest classifier on those datasets.

CDC got an unexpected larger number of rules (this is mainly imbalanced and discretized datasets) on “Hayes-root” and “Balance” datasets.

Detailed information on statistical significant win/losses counts of our methods against other classification models on the number of rules is shown in Table 7.7 and Table 7.8.

Table 7.7: Statistically significant wins/losses counts of DC and DDC method on rules

	DTNB	DT	C4.5	PT	FR	RDR	CBA	CDC
W	14	14	13	13	14	10	13	3
L	2	2	0	3	2	4	1	2
N	1	1	4	1	1	3	3	9

Table 7.7 shows that C4.5 method could not produce statistically smaller classifiers than DC and DDC methods on any datasets. Most importantly, DC and DDC generated statistically smaller classifiers than all other models on bigger datasets (over 1000 examples), which was our main goal in this research. DTNB, DT, FR, CBA and CDC algorithms produced statistically larger classifiers only on 2 (or less) datasets.

Table 7.8: Statistically significant wins/losses counts of CDC method on rules

	DTNB	DT	C4.5	PT	FR	RDR	CBA
W	14	14	12	12	12	10	12
L	2	3	2	3	2	5	3
N	1	0	3	2	3	2	2

Unexpectedly, CDC statistically got the worse result among all methods on “Balance” (except CBA) and “Hayes-root” datasets in terms of classification rules. However, the CDC method could achieve statistically better results in terms of rules than all classification models (except DC&DDC) on more than (or equal to) 10 datasets.

Table 7.9: Overall Coverage

Dataset	DC	DDC	CDC
Breast.Can	65.2	72.0	72.7
Balance	74.5	82.8	86.3
Car.Evn	88.7	100.0	100.0
Vote	88.4	86.9	85.1
Tic-Tac-Toe	89.0	92.0	86.0
Nursery	90.4	98.1	100.0
Hayes-root	100.0	100.0	100.0
Lymp	81.0	90.0	88.4
Spect.H	80.9	80.7	79.4
Abalone	74.1	87.6	78.9
Adult	100.0	100.0	100.0
Insurance	81.5	89.5	100.0
Monks	82.4	86.7	90.6
Laptop	86.1	99.0	100.0
Mushroom	96.1	96.2	98.1
Chess	92.5	96.0	96.8
Connect4	91.4	98.4	98.7
Average(%):	86.0	91.5	91.8

Our main goal in proposing the DDC and CDC methods is to improve the overall coverage (shown in Table 7.9) and accuracy achieved by the DC method. Experimental results show that we could achieve our goal: DDC and CDC gained better average classification accuracy with 83.3% and 83.8% (this is still not the best result in terms of average accuracy, but 0.8% and 1.3% higher than the average accuracy of the DC method). Average coverage of DDC (91.5%) and CDC (91.8%) increased to around 5.5% compared to DC (86.0%). More precisely, the overall coverage of DDC and CDC was improved on 12 datasets and they achieved better classification accuracies on 9 datasets out of 17 compared to DC. However, DC produced a comparable associative classifier with all other classical and associative classifiers.

On the other hand, accuracy of DC, DDC and CDC was higher than their coverage on “Breast.Can”, “Vote” and “Monks” datasets. This fact is not surprising, since uncovered examples get classified by the majority class value. When the overall coverage is above 85%, the proposed methods tend to get a reasonably high accuracy on all datasets. Empirical evaluation of our proposed classifiers is shown in Figure 7.1.

Figure 7.1 illustrates that all three methods obtained similar average accuracy (DC:82.5%; DDC: 83.3%; CDC: 83.8%) and average coverage (DC:86.0%; DDC: 91.5%; CDC: 91.8%). But DC&DDC obtained slightly better results than CDC in terms of classification rules with 22 and 36 respectively. Although CDC gained better coverage than DC, it got worse results in terms of classification rules than that method.

The most important advantage of our proposed methods was to generate a smaller

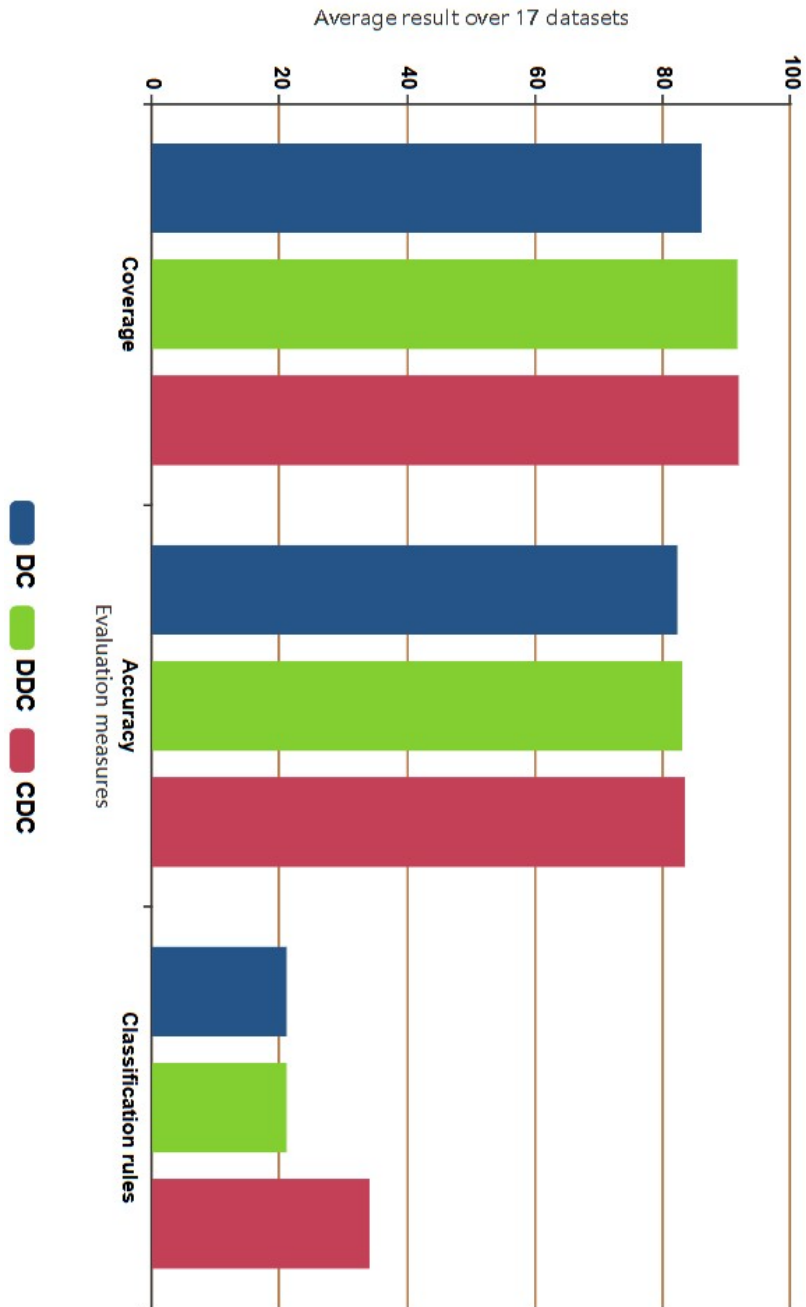


Figure 7.1: Comparison of performance of our proposed associative classification models

classifier on bigger datasets, therefore, we analyze the results obtained by DC, DDC and CDC methods on some bigger datasets which have over 4000 examples. Comparison on accuracy is illustrated in Figure 7.2.

We can observe from Figure 7.2 that all three methods obtained comparable results on all selected datasets except the “Nursery” dataset (the CDC method achieved fairly higher accuracy than DC and DDC). Figure 7.3 presents the comparison between proposed associative classifiers on the number of classification rules.

Since the size of DC and DDC methods is the same, we merged them in the resulting figure. We can see from the result that the CDC method produced larger classifier than DC and DDC on bigger datasets. More precisely, when the size of the dataset increases, CDC tends to generate larger classifiers than DC and DDC.

7.1 Discussion of Results

In this section, we discuss the pros and cons of the proposed associative classifiers within the scientific research.

We used the WEKA software to generate the class association rules for all associative classifiers and to explore other classification algorithms because WEKA is open-source software that includes all necessary classification approaches and it is relatively easy to use and program. Our proposed algorithms namely, SA (Simple Associative classifier), J&B (coverage based associative classifier), DC, DDC and CDC has some limitations:

- To get “enough” class association rules (“enough” means at least 5-10 rules for each class value, this situation mainly happens with imbalanced datasets) for each class value and achieve a reasonable overall coverage, we need to apply appropriate minimum support and minimum confidence thresholds. That is, we need to take into consideration the class distribution of each dataset.
- It is unnecessary to remove the missing values from the dataset because all classical classification algorithms and our method can handle missing values directly.
- Numeric attributes need to be discretized because associative classification models can handle only the nominal attributes.
- Number of rules per class threshold should be applied for SA method according to analysis of the dataset, because this threshold depends on the class distribution.
- We need to apply the intended coverage constraint which becomes the stopping criteria of selecting rules for J&B classifier. This novel parameter is not sensitive to the size of the dataset or imbalanced class distribution.
- We tried to solve the outer-class overlapping problem (that means some samples from different classes have very similar characteristics) by grouping the CARs according to their class value, and the inter-class overlapping problem (several

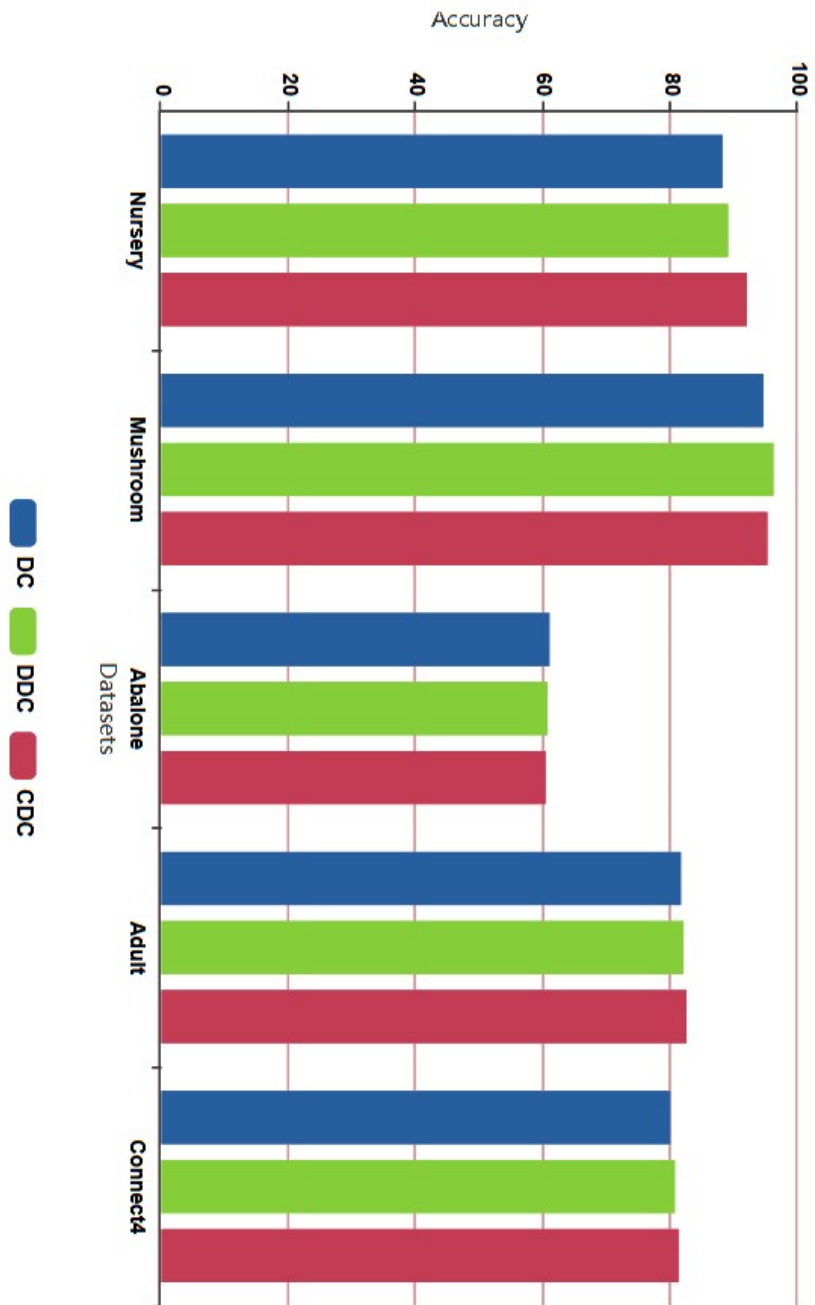


Figure 7.2: Comparison between our proposed associative classification models on accuracy for selected datasets

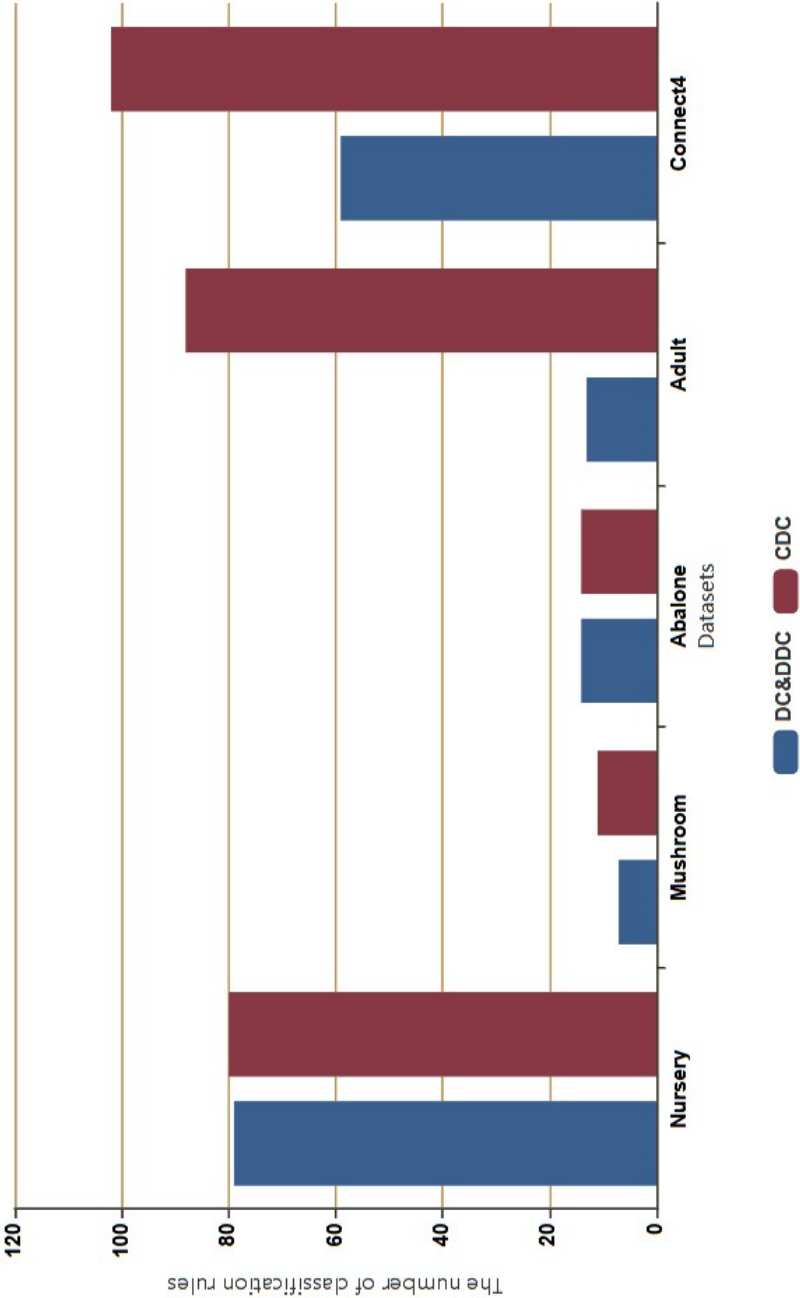


Figure 7.3: Comparison between our proposed associative classification models on size for selected datasets

rules that belong to same class may cover the same samples) by selecting the rules based on database coverage.

To begin with, SA method was the preliminary research in this thesis. We intended to develop a simple classification model that can be comparable to other rule-based and non rule-based classification models (such as OneR, C4.5, Decision Table, Naive Bayes, Random Forest and son on) on accuracy. Reducing the number of rules was not our main goal in SA, therefore, we included also non rule-based models into the comparison. SA is sensitive to the size of the dataset on generating the classification rules, on bigger datasets such as "Adult", "Connect4", "Nursery", SA produced larger classifiers.

The J&B produced an accurate and compact classifier on datasets which have a large number of attributes and records (especially with balanced datasets on class-distribution), while the performance of J&B is slightly worse with datasets having a small number of attributes and a large number of records (proved on "Nursery" and "Car.Evn" datasets).

Since SA and J&B algorithms were discussed in Chapter 3 comprehensively and the number of rules generated by these algorithms depend on the user-defined parameters (*number_of_rules_per_class* in SA and *intended_coverage*), we did not include them into the main comparison table.

DC and DDC algorithms produce classifiers that have on average far fewer rules than those produced by the classical and associative rule-learning methods included in the comparison. DC and DDC generated a reasonably smaller number of rules on bigger datasets compared to all other classification methods while CDC generated a slightly higher number of classification rules compared to DC and DDC, but on average all of our proposed methods achieved the best results in terms of classification rules.

DC, DDC and CDC classifiers achieved the best accuracy on "Breast.Can" and "Spect.H" datasets among all rule-learner approaches.

The accuracy of the CDC method on "Hayes-root", "Adult", "Lymp" and "Connect4" was the best one compared to all other rule-based classification models including DC and DDC.

The main drawback of the DC method is achieving a higher coverage on some smaller datasets. This fact is not surprising since the representative CAR is selected based on cluster center and not checking the coverage. DDC and CDC do not suffer from that problem, because a novel approach for improving the coverage and accuracy is proposed in the representative CAR selection process.

Chapter 8

Conclusion and Future Work

In this thesis, we presented five associative classifiers. The first two – SA: simple associative classifier (pre-determined number of strong CARs are selected for each class value) and J&B: coverage based associative classifier (rules are selected based on dataset coverage) – were introduced as part of preliminary research to show that decreasing the minimum confidence of CARs, increasing their number and overall dataset coverage results in an increase of classification accuracy of the final models. Using this information, the last three proposed associative classifiers, as part of our main research, are novel methods that aim at significantly reducing the size of produced models, while maintaining their classification accuracy.

The three novel associative classifiers are: DC (the DC method is built based on direct distance measure (IBDM) and the method for extracting a representative CAR is based on cluster center (RCC)), DDC (the DDC method is formed based on direct distance measure (IBDM) and the method for extracting a representative CAR is based on dataset coverage (RDC)), and CDC (the CDC method is formed based on combined distance measure (WCDM) and the method for extracting a representative CAR is based on dataset coverage (RDC)). They are all associative classifiers based on clustering.

Experimental evaluations show that we achieved our intended research goal of producing a compact and meaningful, yet accurate, classifier by exhaustively searching the entire example space using constraints and clustering. Our DC, DDC, and CDC classifiers were able to significantly reduce the number of classification rules while maintaining a classification accuracy that was not significantly different from that of state-of-the-art rule-learning classification algorithms. Moreover, experiments show that the number of learnt rules by our classifiers, compared to state-of-the-art classification rule-learners, was 2–4 times lower on average. This ratio is even greater on datasets with higher numbers of examples.

The main drawback of our proposed methods is their time efficiency. While some classification algorithms (e.g. C4.5 or PART) use the divide-and-conquer greedy approach, which is very fast, others use the separate-and-conquer, covering and/or other more exhaustive approaches, that are much slower. Our proposed methods fall into the latter category of methods that use exhaustive search to find “good” CARs. Moreover, after finding the initial set of CARs, they use clustering on top of that to further reduce the number of CARs, which adds additional computation time to the

algorithms.

In future work, we plan to optimize DC, DDC, and CDC to bring their time complexities at least a bit closer to state-of-the-art divide-and-conquer rule-learning algorithms and make our methods adequate for analyzing big data. Moreover, a time-complexity analysis of our methods is needed first to identify the parts of our algorithms that can be optimized or parallelized.

Another promising direction for future research on associative classifiers is to investigate ways of including numeric attributes into the models. There is already ongoing research in this area, but using clustering on CARs may open new potentially interesting perspectives on the matter.

In our research, we used the hierarchical agglomerative clustering method with default parameters to cluster CARs. Using different sets of clustering parameters or even different clustering algorithms and/or different distance metrics are all possible directions for future research on learning accurate, efficient, meaningful and compact classification models.

The results of this PhD Thesis are published in the following articles:

- Mattiev, J., Kavšek, B.: Coverage based classification using association rule mining. *Applied Sciences* 10(20), Basel, Switzerland (2020).
- Mattiev, J., Kavšek, B.: Distance based Clustering of Class Association Rules to Build a Compact, Accurate and Descriptive Classifier *Computer Science and Information Systems*, Serbia (2020).
- Mattiev, J., Kavšek, B.: CMAC: Clustering class association rules to form a descriptive and Meaningful Associative Classifier, *Machine Learning, Optimization, and Data Science, LOD 2020. Nicosia G., Pardalos P., Umeton R., Giuffrida G., Sciacca V. Eds. vol.11943, Springer, Siena, Italy* (2020).
- Mattiev, J., Kavšek, B.: Simple and Accurate Classification Method Based on Class Association Rules Performs Well on Well-Known Datasets, *Machine Learning, Optimization, and Data Science, LOD 2019. Nicosia G., Pardalos P., Umeton R., Giuffrida G., Sciacca V. Eds. vol.11943, Springer, Siena, Italy*, (2019), pp. 192–204.
- Mattiev, J., Kavšek, B.: A compact and understandable associative classifier based on overall coverage, *ANT 2020, The 11th International Conference on Ambient Systems, Networks and Technologies, Procedia computer science, Vol.170, Warsaw, Poland*, (2020), pp. 1161-1167.
- Mattiev, J., Kavšek, B.: How overall coverage of class association rules affects the accuracy of the classifier?, *Data Mining and Data Warehouses - SiKDD : proceedings of the 22nd International Multiconference Information Society, IS 2019, Ljubljana, Slovenia* (2019), 49–52.
- Mattiev, J., Kavšek, B.: Using constrained exhaustive search vs. greedy heuristic search for classification rule learning, *Proceedings of the 5th Student Computer Science Research Conference, StuCoSReC-2018, Koper, Sloveniya* (2018), 35-38.

Bibliography

- [1] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487-499. Chile (1994).
- [2] Baralis, E., Cagliero, L., Garza, P.: A novel pattern-based Bayesian classifier. *IEEE Transactions on Knowledge and Data Engineering* 25(12), pp. 2780–2795 (2013).
- [3] Breiman, L.: Random Forests. *Machine Learning* 45(1), pp. 5-32 (2001).
- [4] Cendrowska, J.: PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies* 27(4), pp. 349-370 (1987).
- [5] Chen, G., Liu, H., Yu, L., Wei, Q., Zhang, X.: A new approach to classification based on association rule mining. *Decision Support Systems* 42(2), pp. 674–689 (2006).
- [6] Clark, P., Niblett, T.: The CN2 induction algorithm. *Machine Learning* 3(4), pp. 261–283 (1989).
- [7] Cohen, W., W.: Fast Effective Rule Induction. In: ICML'95 Proceedings of the Twelfth International Conference on Machine Learning, pp. 115-123, Tahoe City, California (1995).
- [8] David, A., K., Sergei, V.: K-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Louisiana, USA (2007).
- [9] Dua, D., Graff, C.: UCI Machine Learning Repository. Irvine, CA: University of California (2019).
- [10] Eui-Hong, S., H., George, K., Vipin, K., Bamshad, M.: Clustering Based on Association Rules Hypergraphs. Workshop on Research Issues on Data Mining and Knowledge Discovery. December (1999).
- [11] Frank, E., Witten, I.: Generating Accurate Rule Sets Without Global Optimization. In: Fifteenth International Conference on Machine Learning, pp. 144-151, USA (1998).

-
- [12] Gupta, K., G., Strehl, A., Ghosh, J.: Distance based clustering of association rules. Proceedings of artificial neural networks in engineering conference, pp. 759-764, USA (1999).
- [13] Holte, R.: Very simple classification rules perform well on most commonly used datasets. *Machine Learning* 11(1), pp. 63-91 (1993).
- [14] Kohavi, R.: The Power of Decision Tables. In: 8th European Conference on Machine Learning, pp. 174-189, Greece (1995).
- [15] Lent, B., Swami, A., Widom, J.: Clustering association rules. In: Proceedings of the Thirteenth International Conference on Data Engineering, pp. 220-231, England (1997).
- [16] Leonard, K., Peter, J., R.: Finding Groups in Data: An Introduction to Cluster Analysis. New Jersey, USA (1990).
- [17] Li, W., Han, J., Pei, J.: CMAR: accurate and efficient classification based on multiple class-association rules. In Proceedings of the 1st IEEE International Conference on Data Mining, pp. 369-376, California, USA (2001).
- [18] Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, pp. 80-86, New York, USA (1998).
- [19] Man, Z., Xiu, Ch., Qianzhou, H.: An Algorithm of Mining Class Association Rules. In: Z. Cai, Z. Li, Z. Kang, Y. Liu. (editors), Advances in Computation and Intelligence. ISICA 2009. Lecture Notes in Computer Science, Vol. 5821, pp. 269-275, Heidelberg, Germany (2009).
- [20] Martin, E., Hans-Peter, K., Xiaowei, X.: Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. In Proceedings of the 4th International Symposium on Large Special Databases, pp. 67-82, Portland (1995).
- [21] Michael, H., Radoslaw, K.: Visualizing Association Rules in Hierarchical Groups. *Journal of Business Economics* 87(3), pp. 317-335 (2017).
- [22] Mohamed, Z., Hammou, M.: A Comparative Study of Clustering Methods. *Future Generation Computer Systems*, 13(2-3), pp. 149-159 (1997).
- [23] Phipps, A., Lawrence, J., H.: An Overview of Combinatorial Data Analysis. In: Phipps, A., Lawrence, J., H., Geert, D., S (editors), Clustering and Classification, pp. 5-63, New Jersey, USA (1996).
- [24] Quinlan, J.: Induction of Decision trees. *Machine Learning* 1(1), pp.81-106 (1986).
- [25] Quinlan, J.: C4.5: Programs for Machine Learning. *Machine Learning* 16(3), pp. 235-240 (1993).

-
- [26] Raymond, T., Ng, Jiawei, H.: Efficient and Effective Clustering Methods for Spatial Data Mining. In: J.B. Bocca, M. Jarke, C. Zaniolo (editors), Proceedings of the 20th Conference on Very Large Data Bases, pp. 144-155, Santiago, Chile (1994).
- [27] Sergios, T., Konstantinos, K.: Hierarchical Algorithms. *Pattern Recognition* 4(13), pp. 653-700 (2009).
- [28] Tian, Z., Raghu, R., Miron, L.: BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: H.V. Jagadish, I.S. Mumick (editors), Proceedings of the 1996 ACM-SIGMOD International Conference on Management of Data, pp.103-114, Montreal, Canada (1996).
- [29] Kosters, W., Marchiori, E., Oerlemans, A.: Mining Clusters with Association Rules. Lecture Notes in Computer Science (1999), DOI:10.1007/3-540-48412-4_4.
- [30] Xiaoxin, Y., Jiawei, H.: CPAR: Classification based on Predictive Association Rules. Proceedings of the SIAM International Conference on Data Mining, pp. 331-335, San Francisco, USA (2003).
- [31] Zhang, M., Zhou Z.: A k-nearest neighbor based algorithm for multi-label classification. In: Proceedings of the 1st IEEE International Conference on Granular Computing, Vol. 2, pp. 718-721, China (2005).
- [32] Zhou, Z., Liu, X.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering* 18(1), pp. 63-77 (2006).
- [33] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I, H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations. Vol 11(1), (2009).
- [34] Eric, N., Xiu, L., Chau, D.: Application of data mining techniques in customer relationship management, a literature review and classification. *Expert Systems with Applications* 36(2), pp. 2592-2602 (2009).
- [35] Jyoti, S., Ujma, A., Dipesh, S., Sunita, S.: Predictive data mining for medical diagnosis: an overview of heart disease prediction. *International Journal of Computer Applications* 17(8), pp. 43-48 (2011).
- [36] Yoon, Y., Lee, G.: Two scalable algorithms for associative text classification. *Information Processing and Management* 49(2), pp. 484-496 (2013).
- [37] Bayardo, R, J.: Brute-force mining of high-confidence classification rules. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pp. 123-126, USA (1997).

-
- [38] Shafer, J. C., Agrawal, R., Mehta, M.: SPRINT: A scalable parallel classifier for data mining. In Proceedings of 22nd International Conference in Very Large databases, pp. 544-555, Italy (1996).
- [39] Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. Sixteenth International Joint Conference on Artificial Intelligence, pp. 307-335, Sweden, (1999).
- [40] Ahmed, A. M., Rizaner, A., Ulusoy, A. H.: A novel decision tree classification based on post-pruning with Bayes minimum risk. *PLoS ONE* 13(4), e0194168, (2018)
- [41] Qabajeh, I., Thabtah, F., Chiclana, F.: A dynamic rule-induction method for classification in data mining, *Journal of Management Analytics*, vol. 2(3), pp. 233-253, (2015).
- [42] Pham, D., Bigot, S., Dimov, S.: RULES-5: A rule induction algorithm for classification problems involving continuous attributes. *The Institution of Mechanical Engineers Part C-journal of Mechanical Engineering Science* 217(12). pp. 1273-1286 (2003).
- [43] Vapnik, V. N.: An overview of statistical learning theory. *IEEE Transactions on Neural Networks* 10(5), pp. 988-999 (1999).
- [44] Donato, M., Michelangelo, C., Annalisa, A.: A relational approach to probabilistic classification in a transductive Setting. *Engineering. Applications of AI* 22(1). pp. 109-116 (2009).
- [45] Hu, L. Y., Hu, Y. H., Tsai, C. F., Wang, J. S., Huang, M. W.: Building an associative classifier with multiple minimum supports. *SpringerPlus* 5(528), (2016).
- [46] Deng, H., Runger, G., Tuv, E., Bannister, W.: CBC: an associative classifier with a small number of rules. *Decision Support Systems* 50(1), pp. 163-170 (2014).
- [47] Thabtah, F. A., Cowling, P., Peng, Y.: MMAC: a new multi-class, multi-label associative classification Approach. In: Proceedings of the fourth IEEE international conference on data mining, pp. 217-224, Brighton, UK (2004).
- [48] Abdellatif, S., Ben Hassine, M. A., Ben Yahia, S., Bouzeghoub, A.: AR-CID: A New Approach to Deal with Imbalanced Datasets Classification. 44th International Conference on Current Trends in Theory and Practice of Computer Science, Lecture Notes in Computer Science. Vol 10706, pp. 569-580, Austria, (2008).
- [49] Thabtah, F. A., Cowling, P. I.: A greedy classification algorithm based on association rule. *Applied Soft Computing Journal* 7(3), pp. 1102-1111 (2007).

- [50] Liu, Y., Jiang, Y., Liu, X., Yang, S.-L.: CSMC: a combination strategy for multi-class classification based on multiple association rules. *Knowledge-Based Systems* 21(8), pp. 786–793, (2008).
- [51] Dua, S., Singh, H., Thompson, H., W.: Associative classification of mammograms using weighted rules. *Expert Systems with Applications* 36(5), pp. 9250–9259 (2009).
- [52] Song, J., Ma, Z., Xu, Y.: DRAC: a direct rule mining approach for associative classification. In Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence, pp. 150–154, Sanya, China, (2010).
- [53] Lin, M., Li, T., Hsueh, S.-C.: Improving classification accuracy of associative classifiers by using k-conflict-rule preservation. In Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, Malaysia (2013).
- [54] Wang, W., Zhou, Z.: A review of associative classification approaches. *Transaction on IoT and Cloud Computing* 2(2), pp. 31–42 (2014).
- [55] Khairan, D, R.: New Associative Classification Method Based on Rule Pruning for Classification of Datasets. *IEEE Access* 7(1), pp. 157783-157795 (2019).
- [56] Mattiev, J., Kavšek, B.: How overall coverage of class association rules affect affects the accuracy of the classifier?. Proceedings of 22th International Multi-conference on Data Mining and Data Warehouse, pp. 49-52, Slovenia (2019).
- [57] Natarajan, R., Shekar, B.: Tightness: A novel heuristic and a clustering mechanism to improve the interpretation of association rules. In Proceedings of the IEEE International Conference on Information Reuse and Integration, pp. 308-313, USA (2008).
- [58] Toivonen, H., Klemettinen, M., Ronkainen, P., Hatonen, K., and Manilla, H.: Pruning and grouping discovered association rules. In ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases, pp. 47-52, Greece (1995).
- [59] Dahbi, A., Mouhir, M., Balouki, Y., Gadi, T.: Classification of association rules based on K-means algorithm. 4th IEEE International Colloquium on Information Science and Technology, pp. 300-305, Tangier, Morocco (2016).
- [60] Gasmi, G., Yahia, S.B., Nguifo, E.M., Slimani, Y.: IGB: A new informative generic base of association rules. In: Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference, pp. 81–90, Hanoi, Vietnam (2005).

- [61] Zaki, M.J., Parthasarathy, S, Ogihara, M, Li, W.: New algorithms for fast discovery of association rules. In Proceedings of the third International conference on Knowledge Discovery and Data Mining, pp. 283-286, USA (1997).
- [62] Lenca, P., Meyer, P., Vaillant, B., Lallich, S.: On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. *European journal of Operational Research* 184(2), pp. 610–626 (2008).
- [63] Tan, P.-N., Kumar, V., Srivastava, J., Selecting the Right Interestingness Measure for Association Patterns. Proceedings of eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Vol. 2, pp. 32–41, USA (2002).
- [64] Mattiev, J., Kavšek, B.: Simple and Accurate Classification Method Based on Class Association Rules Performs Well on Well-Known Datasets. In: Machine Learning, Optimization, and Data Science. Nicosia G., Pardalos P., Umeton R., Giuffrida G., Sciacca V. Eds. Vol. 11943, pp. 192–204, Italy (2019).
- [65] Hühn, J., Hüllermeier, E.: FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery* 19(1), pp. 293–319 (2019). doi.org/10.1007/s10618-009-0131-8
- [66] Richards, D.: Ripple down rules: a technique for acquiring knowledge. Decision making support systems: achievements, trends and challenges for. IGI Global, pp. 207–226, USA (2002).
- [67] Hall, M., Frank, E.: Combining Naive Bayes and Decision Tables. Proceedings of Twenty-First International Florida Artificial Intelligence Research Society Conference, D.L. Wilson, H. Chad. Editors. pp. 318-319, USA (2008).
- [68] Wu, CH-H., Wang, J.Y.: Associative classification with a new condenseness measure. *Journal of the Chinese Institute of Engineers* 38(4), pp. 458-468 (2015).
- [69] Quinlan, J.R., Cameron-Jones, R.M.: FOIL: A midterm report. Proceedings of the 6th European conference on Machine Learning. Brazdil, P.B. Editors. pp. 1-20, Austria (1993).
- [70] Meretakis, D., Wüthrich. B.: Classification as Mining and Use of Labeled Itemsets. Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Fayyad, U., Chaudhuri, S., Madigan, D Editors. USA (1999).
- [71] Baralis, E., Garza, P.: A lazy approach to pruning classification rules. Proceedings of IEEE International Conference on Data Mining. pp. 35-42, Japan (2002).

- [72] Baralis, E., Chiusano, S., Garza, P.: On support thresholds in associative classification. Proceedings of the ACM Symposium on Applied Computing, pp. 553-558, Cyprus, (2004).
- [73] Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM-SIGMID International Conference on Management of Data. pp. 1-12, New York, USA (2000).
- [74] Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New algorithm for fast discovery of association rules. Proceedings of the 3rd International conference on Knowledge discovery and Data Mining. Heckerman, D., Mannila, H. Editors. pp. 286-289, USA (1997).
- [75] Zaki, M., Gouda, K.: Fast vertical mining using difsets. Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Pedro, D. Editors. pp. 326-335, USA (2003).
- [76] Thabtah, F.A., Cowling, P., Peng, Y.: MCAR: multi-class classification based on association rule. In: Proceedings of the 3rd ACS/IEEE international conference on computer systems and applications. pp. 127-133, Egypt (2005).
- [77] Bui-Thi, D., Meysman, P., Laukens, K.: Clustering association rules to build beliefs and discover unexpected patterns. Applied Intelligence. Vol. 50, pp. 1943-1954 (2020).
- [78] Raykov, Y.P., Boukouvalas, A., Baig, F., Little, M.A.: What to Do When K-Means Clustering Fails: A Simple yet Principled Alternative Algorithm. *PLoS ONE* 11(9), pp. 1-28 (2016).
- [79] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 226-231, Portland, USA (1996).
- [80] Rodriguez, M.Z., Comin, C.H., Casanova, D., Bruno, O.M., Amancio, D.R., Costa, L.F.: Clustering algorithms: A comparative approach. *PLoS ONE*. 14(1) (2019).
- [81] Yildirim, N., Uzunoglu, B.: Association Rules for Clustering Algorithms for Data Mining of Temporal Power Ramp Balance. International Conference on Cyberworlds, pp. 224-228, Sweden (2015).
- [82] Dechang, P., Xiaolin, Q.: A New Fuzzy Clustering Algorithm on Association Rules for Knowledge Management. *Information Technology Journal* 7(1), pp. 119-124 (2008).
- [83] RStudio Team. RStudio: Integrated Development for R (2020). RStudio, PBC, Boston, USA.

-
- [84] Mahamed, O., Andries, E., Ayed, S.: An overview of clustering methods. *Intelligence. Data Analysis* 11(1), pp. 583-605 (2007).
- [85] Thomas, C.H., Charles, L.E., Ronald, R.L., Clifford, S.: 22.2 Breadth-first search. *Introduction to Algorithms (2nd ed.)*. MIT Press and McGraw-Hill. pp. 531–539 (2001).
- [86] Shimon, E.: Depth-First Search. *Graph Algorithms (2nd ed.)*, Cambridge University Press, pp. 46–48 (2011).
- [87] Urvashi, B.: ECLAT Algorithm for Frequent Item sets Generation. *International Journal of Computer System* 1(3), pp. 82-84 (2014).
- [88] Suzan, W.: Review and Comparison of Associative Classification Data Mining Approaches. *International Journal of Industrial and Manufacturing Engineering* 8(1), pp. 34-45 (2014).
- [89] Veloso, A., Meira, W., Gonçalves, M., Zaki, M.: Multi-label Lazy Associative Classification. Proceedings of the Principles of Data Mining and Knowledge Discovery - PKDD, pp. 605-612, Warsaw, Poland (2007).
- [90] Niu, Q., Xia, S., Zhang, L.: Association Classification Based on Compactness of Rules. Proceedings of the Second International Workshop on Knowledge Discovery and Data Mining - WKDD, pp.245-247, Chicago, USA (2009).
- [91] Li, X., Qin, D., Yu, C.: ACCF: Associative Classification Based on Closed Frequent Itemsets. Proceedings of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery - FSKD. pp. 380-384, China (2008).
- [92] Ye, Y., Jiang, Q., Zhuang, W.: Associative classification and post-processing techniques used for malware detection. Proceedings of the second International Conference on Anti-counterfeiting, Security and Identification –ASID, pp. 276-279, USA (2008).
- [93] Hsu, H., Lachenbruch, P.A.: Paired t test. *Wiley Encyclopedia of Clinical Trials* (2008).
- [94] Chawla, Nitesh: Data Mining for Imbalanced Datasets: An Overview. *Data Mining and Knowledge Discovery Handbook* 1(5), pp. 853-867 (2005).
- [95] Han, J., Kamber, M., Pei, J.: Data mining: Concepts and techniques, third edition. Waltham, Mass.: Morgan Kaufmann Publishers (2012).

Index

- ActOcc*(R), 23
- Conf*(R), 23
- Supp*(R), 23
- d^{CMBP} , 48
- α , 51
- d^{WCDM} , 51
- d^{AMBD} , 48
- d^{CMBL} , 49
- AHCCLC*, 57
- border*, 51
- numrules*, 27
- “class-dependent” discretization, 25

- AHCCLH*, 57

- accuracy, 2
- antecedent, 23
- APRIORI, 18
- artificial intelligence, 1
- association rules (ARs), 1
- Associative Classification (AC), 1
- attribute, 3

- bottom-up, 2

- C-Tidset, 10
- CDC, 63
- class, 12
- class association rules, 8
- Classification rule mining, 1
- classifier, 25
- clustering, 2
- Combined Distance Metric (WCDM), 51
- compact, 26
- complete linkage (farthest neighbor), 57

- consequent, 23
- cross-validation, 40
- customer relationship management, 4
- cutting the dendrogram, 8

- Data mining, 1
- dataset coverage, 9
- DC, 63
- DDC, 63
- decision trees, 7
- decision-making processes, 3
- direct distance metric, 12

- Euclidean distance, 47
- exhaustively searching, 75

- F-measure, 39
- FP-growth, 18
- FP-tree, 22
- frequent itemset mining, 3

- hierarchical clustering, 2

- imbalanced datasets, 2
- indirect measures, 12
- inter class overlapping problem, 60
- Interestingness measures, 11
- intra-cluster similarity, 53
- Item Based Distance Metric (IBDM), 50

- L, 65

- majority class value, 69
- market basket problem, 3
- medical diagnosis, 4
- minimum confidence, 2

- minimum support, 2
- missing values, 71
- multi-dimensional dataset, 3
- N, 65
- non-overlapping rules, 48
- objects, 53
- optimal number of clusters, 8
- outer-class overlapping problem, 59
- outliers, 54
- overall coverage, 31
- overfitting, 9
- paired T-test, 31
- Precision, 38
- prefix tree, 20
- pruning techniques, 2
- RCC, 59
- RDC, 60
- real-life dataset, 1
- Recall, 38
- record, 17
- reduce the number of classification
 - rules, 2
- reference point, 55
- relevance measures, 31
- representative CAR, 13
- rule-based, 2
- SA, 26
- similarity measure, 12
- simple associative classifier, 75
- statistical significance testing, 31
- statistically comparable, 65
- statistics, 1
- stopping criterion, 31
- supervised learning, 1
- test set, 29
- tightness, 12
- time efficiency, 75
- top-down, 12
- transactional, 1
- unsupervised learning, 2
- vertical layout format, 10
- W, 65

Povzetek v slovenskem jeziku

Uporaba razvrščanja v skupine na klasifikacijskih asociacijskih pravilih za tvorbo smiselnih in natančnih klasifikatorjev

8.1 Uvod

Ogromne količine podatkov se vsakodnevno generirajo in shranjujejo v najrazličnejših oblikah in sistemih za upravljanje s podatki, zato potrebujemo načine za obvladovanje in analizo teh podatkov. Eden od načinov za spopadanje s tem problemom, ki v zadnjih letih vedno bolj pridobiva na popularnosti, je podatkovno rudarjenje.

Podatkovno rudarjenje je proces odkrivanja skritih povezav v podatkih, ki predstavljajo novo znanje. Je pa tudi znanstveno področje, ki črpa dognanja iz področij matematičnega modeliranja, statistike, umetne inteligence in upravljanja podatkovnih baz. Dandanes je podatkovno rudarjenje eno ključnih področij, ko gre za reševanje realnih problemov, ki temeljijo na zbiranju in obdelavi velikih količin podatkov. Ena glavnih prvin v procesu podatkovnega rudarjenja so algoritmi strojnega učenja, s pomočjo katerih lahko odkrivanje znanja v podatkih do velike mere avtomatiziramo.

Glede na namen podatkovnega rudarjenja ter vrsto in obliko znanja, ki ga v podatkih želimo odkriti, lahko algoritme strojnega učenja uporabljamo za: klasifikacijo, regresijo, razvrščanje v skupine, iskanje pogostih vzorcev, iskanje osamelcev, podporo odločanju. . . Najpogosteje uporabljani in najbolj preučevani so zagotovo klasifikacijski algoritmi med katerimi so najbolj poznani: k-NN [31], naivni Bayesov klasifikator [2], odločitvena drevesa [25], nevronske mreže [32], idr. Cilj klasifikacijskih algoritmov je odkriti relativno majhno množico pravil, ki skupaj tvorijo karseda natančen klasifikacijski model in pri tem pokrivajo čim večji del vseh primerov v podatkih.

Odkrivanje asociacijskih pravil v transakcijskih bazah podatkov postaja v zadnjem času ena priljubljenjših metod podatkovnega rudarjenja [1,6]. Pri tej metodi govorimo o odkrivanju pogostih vzorcev, katerega cilj je odkriti vsa obstoječa (asociacijska) pravila v neki bazi podatkov, ki ustrezajo določenim omejitvam (minimalne podpore in zaupanja). Tako odkrita pravila so tudi bolj splošna od klasifikacijskih.

V zadnjih letih so klasifikacijska asociacijska pravila predmet številnih raziskav [17,18,19,30]. Klasifikacijska asociacijska pravila združujejo metodologijo asociacijskih pravil s klasifikacijo. Eksperimenti na podatkih [5,17,18] kažejo, da klasifikacijska asociacijska pravila lahko dosegajo celo višje klasifikacijske točnosti kot klasični klasifikacijski algoritmi kot npr. odločitvena drevesa [25].

8.2 Znanstvena izhodišča

Obstoječi algoritmi za učenje klasifikacijskih pravil temeljijo predvsem na požrešnem preiskovanju prostora možnih rešitev, v glavnem po principu deli-in-vladaaj. V zadnjih letih potekajo raziskave v smeri izčrpnih preiskovalnih metod, ki po principu razveji-in-omeji preiščejo celoten prostor rešitev, a pri tem upoštevajo določene omejitve, ki omogočajo prihranek časa in prostora pri samem preiskovanju. Slednje ima za posledico generiranje klasifikatorjev, ki vsebujejo zelo velike količine »pravil«, ki pa so veljavna le na zelo majhnih podmnožicah celotnega prostora ter se med seboj prekrivajo. Posledično je takšne klasifikatorje potrebno naknadno poenostaviti in v doktorski disertaciji bomo skušali razviti metodo za poenostavitev tovrstnih klasifikatorjev – v našem primeru so to klasifikacijska asociacijska pravila, ki so kombinacija »klasičnih« klasifikacijskih pravil in pa asociacijskih pravil.

8.2.1 Klasifikacijska pravila

Klasifikacijska ali odločitvena pravila so vrsta logični pravil tipa ČE ... POTEM ..., kjer ČE delu pravimo tudi pogojni del ali pogoj, POTEM delu pa posledični del ali posledica. Pogoj je, v tipičnih aplikacijah podatkovnega rudarjenja, sestavljen iz konjunkcije neodvisnih spremenljivk (atributov) in njihovih vrednosti, posledica pa predstavlja odvisno spremenljivko (razred) in njeno vrednost.

Klasifikacijska pravila delimo v dve skupini: odločitvena pravila in odločitvena drevesa. V prvi skupini je klasifikator predstavljen z množico (ne)urejenih klasifikacijskih pravil. V skupini odločitvenih dreves pa je moč klasifikatorje zapisati v obliki dreves, kjer vsaka pot od korena do lista drevesa predstavlja eno klasifikacijsko pravilo (pravil je torej toliko, kot je listov v drevesu).

Tipični predstavniki prve skupine so algoritmi PART [11], RIPPER [7], CN2 [6], PRISM [4], odločitvene tabele [14], OneR [13], ipd. V drugo skupino pa bi lahko uvrščali algoritme, kot so ID3 [24], C4.5 [25], naključni gozdovi [3], ipd.

8.2.2 Pogoste podmnožice in asociacijska pravila

Asociacijsko pravilo sestoji iz dveh delov: pogojnega dela (če) in posledice (potem). Tako pogojni del kot posledica lahko vsebujeta več postavk (items). Postavke v posledičnem delu se vedno pojavljajo skupaj s postavkami v pogojnem delu. Ko v podatkih odkrivamo asociacijska pravila, pravimo tudi, da iščemo če/potem vzorce. Pri iskanju si največkrat pomagamo z dvema merama: podporo (support) in zaupanjem (confidence). Podpora pove koliko pogosto se neka podmnožica postavk pojavlja v podatkih, zaupanje pa meri pogojno verjetnost posledičnega dela pravila glede na njegov pogojni del. Pri podatkovnem rudarjenju se asociacijska pravila največkrat uporabljajo analizo nakupovalnih navad strank (market basket analysis). Odkrivanje asociacijskih pravil sodi med nenadzorovane metode strojnega učenja.

Osnovne definicije podpore, zaupanja, pogostih podmnožic ter ostalih pojmov, ki se nanašajo na asociacijska pravila so podane v [1].

8.2.3 Klasifikacijska asociacijska pravila

Klasifikacijska asociacijska pravila predstavljajo kombinacijo klasifikacijskih in asociacijskih pravil. Naučena so na enak način kot asociacijska pravila, le da desno stran (posledico) pravila vedno omejimo le na odvisno spremenljivko (razred) in njeno vrednost.

8.3 Opredelitev problema

Glavna omejitev algoritmov, ki pri učenju uporabljajo izčrpno preiskovanje prostora rešitev po principu razveji-in-omeji, kamor sodijo tudi (klasifikacijska) asociacijska pravila, je njihova odvisnost od parametrov, ki pogojujejo »omeji« del. Izbira letih lahko zelo vpliva na zahtevnost algoritma ter velikost naučenih (napovednih) modelov.

V doktorski disertaciji se ne bomo pretirano ukvarjali s samo izbiro parametrov, posvetili se bomo predvsem velikosti modelov ter njihovi »poenostavitvi«. V ta namen bomo naučena klasifikacijska asociacijska pravila najprej poskusili združiti v skupine glede na podobnost med njimi ter nato v vsaki skupini poiskati reprezentativno pravilo, ki naj bi predstavljalo celotno skupino. Namesto celotne množice pravil tako na koncu »prikažemo« le množico reprezentativnih pravil, kar zelo poenostavi celoten model in ga naredi bolj razumljivega. Hkrati lahko končni uporabnik, po potrebi, interaktivno razišče vsako množico pravil, ki pripada posameznemu reprezentativnemu pravilu, če želi natančnejši vpogled v informacije. Za razvoj ustreznega »poenostavitvenega« algoritma so bistvenega pomena: definicija ustrezne mere podobnosti med klasifikacijskimi asociacijskimi pravili, kar omogoča kasnejše razvrščanje v skupine, uporaba ustreznega algoritma razvrščanja v skupine, ustrezen način izbire reprezentativnega klasifikacijskega asociacijskega pravila za vsako skupino ter nenazadnje zagotovitev ustrezne klasifikacijske moči končnih modelov.

Pri razvrščanju v skupine bomo v doktorski disertaciji preizkusili tako partijske metode, kamor sodita metodi k-means in njena izpeljanka k-medoids, kot tudi hierarhične metode. Ostale prvine predlaganega algoritma (mera podobnosti, izbira reprezentativnega klasifikacijskega asociacijskega pravila za vsako skupino ter zagotovitev ustrezne klasifikacijske moči končnih modelov) so opisane v naslednjem poglavju »Raziskovalna vprašanja, hipoteze in cilji«.

8.4 Raziskovalna vprašanja, hipoteze in cilji

Kot je bilo že opisano v prejšnjem poglavju »Opredelitev problema«, so za razvoj novega algoritma za razvrščanje v skupine klasifikacijskih asociacijskih pravil bistveni 4 koraki, ki hkrati predstavljajo cilje te disertacije:

- definicija ustrezne mere podobnosti med klasifikacijskimi asociacijskimi pravili (kar omogoča kasnejše razvrščanje v skupine),
- uporaba ustreznega algoritma razvrščanja v skupine,

- ustrezen način izbire reprezentativnega klasifikacijskega asociacijskega pravila za vsako skupino ter
- zagotovitev ustrezne klasifikacijske moči naučenih modelov.

Vsak od zgoraj naštetih ciljev je v nadaljevanju podrobneje predstavljen.

8.4.1 Definicija mere podobnosti klasifikacijskih asociacijskih pravil

Pri klasifikacijskih asociacijskih pravilih gre, kot pri »navadnih« klasifikacijskih pravilih, za ČE ... POTEM ... tip pravil, kjer POTEM del predstavlja razred ter vrednost razreda. V doktorski disertaciji se bomo ukvarjali izključno z diskretnimi razredi. Prvi korak k definiciji mere podobnosti bo torej ureditev klasifikacijskih asociacijskih pravil glede na vrednost razreda. Znotraj množice pravil, ki napovedujejo isto vrednost razreda pa lahko nato merimo podobnost pravil glede na njihov pogoj (ČE del).

Ker so pogoji pravil vedno predstavljeni v obliki konjunkcije atribut-vrednosti, jih lahko poenostavljeno predstavimo kot množice atribut-vrednosti. Sedaj nam preostane le, da definiramo mero podobnosti med posameznimi množicami.

V disertaciji bomo to mero podobnosti definirali glede na 3 postopke merjenja razdalje med množicami: razdalja na podlagi absolutnih razlik med elementi množic, razdalja na podlagi verjetnosti ter razdalja na podlagi logaritma sličnosti (log-likelihood).

8.4.2 Ugotavljanje skupin za klasifikacijska asociacijska pravila

Sem sodi preizkušanje različnih algoritmov za razvrščanje v skupine, kot je to opisano v prejšnjem poglavju »Opredelitev problema«. Razvrščanje v skupine je dobro raziskana metoda strojnega učenja, ki pokriva paleto možnih algoritmov, med katerimi so najbolj poznani hierarhični in pa k-ta povprečja [10,15,21,29]. V disertaciji bomo preizkusili hierarhično združevanje v skupine, združevanje v skupine na podlagi k-tih povprečij ter združevanje v skupine na podlagi k-tih median.

8.4.3 Ugotavljanje reprezentativnega klasifikacijskega asociacijskega pravila za posamezno skupino pravil

Po fazi razvrščanja v skupine bo vsako klasifikacijsko asociacijsko pravilo pripadalo natanko eni skupini pravil. Da bi zmanjšali število pravil v končnem modelu, si želimo končnemu uporabniku, namesto pravil, na nem način prikazati skupine pravil. V ta namen bomo poskušali za vsako skupino poiskati tisto pravilo, ki jo najbolje opisuje = reprezentativno pravilo. Na koncu bo naš model sestavljen le iz reprezentativnih pravil, ostala pravila v skupinah bodo končnemu uporabniku zakrita, a jih bo lahko (po potrebi) še vedno prikazal.

8.4.4 Eksperimentalna evalvacija

Čeprav bo tako dobljen končni napovedni model bolj kompakten in tako tudi razumljivejši za končnega uporabnika, moramo zagotoviti, da bo njegova napovedna

moč primerljiva z ostalimi »boljšimi« klasifikacijskimi algoritmi. To lahko zagotovimo le tako, da z ustreznimi eksperimentalnimi primerjavami parametre naše metode izberemo na način, ki zagotavlja ohranitev napovedne moči klasifikatorja. V ta namen bomo razvit algoritem primerjali z danes najbolj poznanimi klasifikacijskimi algoritmi: CBA, RIPPER, naivni Bayes, odločitvene tabele, C4.5, naključni gozdovi, SVM, idr.

8.5 Rezultati

Doseganje znanstvenih ciljev je bilo preverjeno z eksperimentalno oceno. Svoje modele smo preizkusili na 17 resničnih naborih podatkov, ki so bili vzeti iz repozitorija baze podatkov UCI MachineLearning. Naše klasifikatorje smo ovrednotili tako, da smo jih primerjali z 8 znanimi algoritmi klasifikacije, ki temeljijo na pravilih, glede natančnosti klasifikacije in števila pravil. Vse razlike so bile testirane na statistično pomembnost z izvedbo parnega t-testa (s 95-odstotnim pragom pomembnosti).

Asociativni klasifikatorji so se izvajali s privzetimi parametri najmanjša podpora = 1% in minimalna zanesljivost = 60% (pri nekaterih naborih podatkov (označena s krepkim tiskom v Tabeli 8.1, pa je bila minimalna podpora znižana na 0,5% ali celo 0,1% in zaupanje znižano na 50% da bi zagotovili, da je bilo za vsako vrednost razreda ustvarjenih dovolj CAR-ov.) Za vse druge klasifikacijske modele smo uporabili njihovo izvedbo delovne mize WEKA s privzetimi parametri. Ker učenje pravil pridruževanja ne podpira številskih atributov, so bili vsi numerični atributi (v vseh naborih podatkov) predhodno diskretizirani z WEKA-jevo metodo diskretizacije, ki je odvisna od razreda (ki lahko samodejno določi število zabojev za vsak številski atribut). Opis naborov podatkov in vhodnih parametrov je prikazan v Tabeli 8.1.

Table 8.1: Privzeti parametri asociativnih klasifikatorjev in opis naborov podatkov.

Podatki	# atributov	# razredov	# primerov	Min podpora	Min zaupanje	# analiziranih pravil
Breast Can	10	2	286	1%	60%	1000
Balance	5	3	625	1%	50%	218
Car.Evn	7	4	1728	1%	50%	1000
Vote	17	2	435	1%	60%	500
Tic-Tac-Toe	10	2	958	1%	60%	3000
Nursery	9	5	12960	0.5%	50%	3000
Mushroom	23	2	8124	1%	60%	3000
Hayes-root	6	3	160	0.1%	50%	1000
Lymp	19	4	148	1%	60%	1500
Monks	7	2	432	1%	50%	800
Spect.H	23	2	267	0.5%	50%	3000
Abalone	9	3	4177	1%	60%	1000
Adult	15	2	45221	0.5%	60%	5000
Insurance	7	3	1338	1%	50%	722
Laptop	11	3	1303	1%	50%	1480
Chess	37	2	3196	0.5%	60%	3000
Connec4	43	3	67557	1%	60%	5000

Vsi eksperimentalni rezultati so bili pridobljeni z uporabo 10-kratnega prečnega preverjanja. Eksperimentalni rezultati o natančnosti razvrščanja (povprečne vred-

nosti pri 10-kratnem prečnem preverjanju s standardnimi odkloni) so prikazani v Tabeli 8.2.

Iz Tabele 8.2 lahko opazimo, da so predlagani asociativni klasifikatorji dosegli primerljivo natančnost (DC: 82,5%, DDC: 83,3% in CDC: 83,8%) z drugimi modeli klasifikacije na izbranih naborih podatkov. Zanimivo je, da CDC bistveno presega vse tiste, ki se učijo pravil pri naborih podatkov »Rak dojke« (razen DDC), »Hayes.R« in »Lymp«, medtem ko je v naborih podatkov »Car.Evn«, »Nursery« in »Monks« naš strokovnjak Predlagane metode so dosegle slabšo natančnost kot vsi drugi algoritmi (razen SA). Standardna odstopanja rezultatov natančnosti se zmanjšujejo z naraščajočim številom primerov v naboru podatkov, kar je pričakovano vedenje. Standardni odmik vseh metod razvrščanja je bil nekoliko višji (nad 4) pri naborih podatkov »Rak dojk«, »Hayes.R«, »Lymp« in »Connect4«, to pomeni, da so razlike v natančnosti pri 10-kratnem navzkrižnem preverjanju nihale.

Primerjava med našimi metodami in drugimi metodami razvrščanja glede na število klasifikacijskih pravil je prikazana v Tabeli 8.3. Ker se DC in DDC razlikujeta v reprezentativnem postopku izbire CAR, število pravil razvrščanja, ustvarjenih z obema metodama, ostane enako. Tako sta v Tabeli 8.3 združeni metodi DC in DDC.

Eksperimentalne ocene števila pravil razvrščanja kažejo, da DC in DDC bistveno presegata vse druge, ki se učijo pravil, na osmih nizih podatkov od 17 (razen CDC) in da klasifikatorje, ki imajo v povprečju veliko manj pravil kot tista, ki jih pripravijo ostalih 8 metod učenja pravil vključena v primerjavo.

CDC je dosegel tudi statistično najboljše rezultate na osmih nizih podatkov od 17 o pravilih, čeprav je dosegel nekoliko slabši rezultat glede povprečnih pravil razvrščanja.

Predlagane metode so ustvarile razmeroma manjše število pravil o večjih naborih podatkov v primerjavi z drugimi metodami razvrščanja. Čeprav naši pristopi niso mogli doseči najboljše natančnosti razvrščanja v naborih podatkov "Car.Evn", "Nursery" in "Laptop", je na teh naborih podatkov ustvaril statistično najmanjši razvrščevalc.

CDC je dobil nepričakovano večje število pravil (to so predvsem neuravnoteženi in diskretizirani nabori podatkov) za nabore podatkov "Hayes-root" in "Balance".

Naš glavni cilj pri predlaganju metod DDC in CDC je izboljšati splošno pokritost (prikazano v Tabeli 8.4) in natančnost, doseženo z DC metodo. Izkušeni rezultati kažejo, da bi lahko dosegli svoj cilj: DDC in CDC sta dosegla natančnost med povprečjem klasifikacije z 83,3% in 83,8% (to še vedno ni najboljši rezultat glede povprečne natančnosti, vendar 0,8% in 1,3% višja od povprečne natančnosti DC metoda). Povprečna pokritost DDC (91,5%) in CDC (91,8%) se je povečala na približno 5,5% v primerjavi z DC (86,0%). Natančneje, splošna pokritost DDC in CDC je bila izboljšana na 12 naborih podatkov in dosegli so boljšo natančnost razvrščanja na 9 naborih podatkov od 17 v primerjavi z DC. Vendar je DC izdelal primerljiv asociativni klasifikator z vsemi drugimi klasičnimi in asociativnimi klasifikatorji.

Po drugi strani pa je bila natančnost DC, DDC in CDC višja od njihovega obsega v podatkovnih nizih "Rak dojke", "Vote" in "Monks". To dejstvo ni presenetljivo, saj se odkriti primeri razvrstijo po vrednosti večinskega razreda. Ko je splošna pokritost nad 85%, predlagane metode ponavadi dobijo razmeroma visoko natančnost vseh nizov podatkov.

Table 8.2: Klasifikacijske točnost s standardnimi odkloni - skupno.

Dataset	DTNB	DT	C4.5	PT	FR	RDR	CBA	DC	DDC	CDC
Breast.Can	70.4±4.1	69.2±6.7	75.0±6.9	74.0±4.0	75.1±5.3	71.8±5.7	71.9±9.8	81.2±4.0	81.9±4.1	82.6±4.5
Balance	81.4±8.1	66.7±5.0	64.4±4.3	76.2±5.6	77.5±7.6	68.5±4.3	73.2±3.8	72.8±2.4	73.2±2.9	73.2±3.0
Car.Evn	95.4±0.8	91.3±1.7	92.1±1.7	94.3±1.0	91.8±1.1	91.0±1.8	91.2±3.9	85.8±1.4	88.5±1.2	87.1±1.6
Vote	94.7±3.4	94.9±3.7	94.7±4.4	94.8±4.2	94.4±2.8	95.6±4.1	94.4±2.6	92.9±2.5	93.2±2.8	90.6±2.3
Tic-Tac-Toe	69.9±2.7	74.4±4.4	85.2±2.7	94.3±3.3	94.1±3.1	94.3±2.9	100.0±0.0	87.3±1.3	91.8±1.0	92.4±1.1
Nursery	94.0±1.5	93.6±1.2	95.4±1.4	96.7±1.7	91.0±1.4	92.5±1.5	92.1±2.4	88.5±1.1	89.3±1.1	92.3±0.9
Hayes-root	75.0±7.2	53.4±8.3	78.7±8.4	73.1±9.7	77.7±8.7	74.3±7.1	75.6±10.9	79.9±5.7	77.8±5.2	82.7±6.1
Lymp	72.9±9.0	72.2±8.3	76.2±8.7	81.7±9.0	80.0±8.2	78.3±7.3	79.0±9.7	78.4±6.7	80.0±6.1	84.0±6.4
Spect.H	79.3±2.7	79.3±1.6	80.0±9.0	80.4±5.6	80.4±2.2	80.4±2.2	79.0±1.6	81.5±0.7	81.3±1.1	82.8±1.3
Abalone	62.1±1.3	61.8±1.5	62.3±1.2	62.3±1.1	61.7±1.6	60.8±0.8	61.1±1.0	61.0±1.1	60.7±1.0	60.7±1.2
Adult	73.0±4.1	82.0±2.3	82.4±4.7	82.1±4.7	75.2±3.2	80.8±2.7	81.8±3.4	81.9±2.4	82.0±2.6	82.8±3.0
Insurance	74.2±1.1	75.7±1.6	75.8±1.4	75.0±1.8	75.8±1.4	73.4±1.7	75.5±2.0	74.0±1.1	74.2±1.1	73.2±1.5
Monks	98.9±0.9	98.9±0.9	98.9±0.9	98.9±0.9	98.9±0.9	97.1±0.7	97.8±1.4	92.5±0.8	93.6±0.9	91.1±0.8
Laptop	75.7±2.6	72.9±2.9	75.3±2.3	74.5±2.9	75.4±2.1	73.2±1.8	75.4±2.0	71.6±2.1	73.8±2.3	72.6±1.7
Mushroom	98.8±0.3	100.0±0.0	100.0±0.0	100.0±0.0	99.0±0.3	100.0±0.0	92.9±1.6	95.6±0.7	97.2±1.0	96.5±0.5
Chess	93.7±3.0	97.3±3.1	98.9±3.6	98.9±3.1	96.4±2.1	95.8±3.3	95.4±2.9	97.0±2.0	97.3±2.0	97.8±2.0
Connect4	78.8±5.9	76.7±7.7	80.0±6.8	81.1±7.9	80.6±7.1	80.0±6.4	80.9±8.1	80.0±5.9	80.7±5.9	81.7±5.9
Average(%):	81.6±3.5	80.0±3.6	83.3±4.0	84.6±3.9	83.8±3.5	82.8±3.2	83.4±4.2	82.5±2.5	83.3±2.5	83.8±2.6

Table 8.3: Število KAP

Podatki	DTNB	DT	C4.5	PT	FR	RDR	CBA	SA	DC&DDC	CDC
Breast.Can	122	22	10	20	13	13	63	20	8	9
Balance	31	35	35	27	44	22	77	45	34	79
Car.Evn	144	432	123	62	100	119	72	160	32	32
Vote	270	24	11	8	17	7	22	30	6	6
Tic-Tac-Toe	258	121	88	37	21	13	23	60	24	17
Nursery	1240	804	301	172	288	141	141	175	79	80
Hayes-root	5	8	22	14	11	10	34	45	19	80
Lymp	129	19	20	10	17	11	23	60	5	7
Spect.H	145	2	9	13	17	12	4	50	8	5
Abalone	165	60	49	71	20	57	131	155	14	14
Adult	737	1571	279	571	150	175	126	130	13	88
Insurance	23	48	21	49	22	22	84	62	18	20
Monks	12	36	14	8	12	10	40	26	14	14
Laptop	101	101	72	60	28	32	41	75	19	18
Mushroom	50	50	26	12	11	8	8	20	7	11
Chess	507	101	31	23	29	30	12	120	12	17
Connect4	3826	4952	3973	3973	403	341	349	600	59	102
Povprečje(%):	457	494	300	302	71	61	74	108	22	36

Table 8.4: Skupna pokritost

Podatki	DC	DDC	CDC
Breast Cancer	65.2	72.0	72.7
Balance	74.5	82.8	86.3
Car.Evn	88.7	100.0	100.0
Vote	88.4	86.9	85.1
Tic-Tac-Toe	89.0	92.0	86.0
Nursery	90.4	98.1	100.0
Hayes-root	100.0	100.0	100.0
Lymp	81.0	90.0	88.4
Spect.H	80.9	80.7	79.4
Abalone	74.1	87.6	78.9
Adult	100.0	100.0	100.0
Insurance	81.5	89.5	100.0
Monks	82.4	86.7	90.6
Laptop	86.1	99.0	100.0
Mushroom	96.1	96.2	98.1
Chess	92.5	96.0	96.8
Connect4	91.4	98.4	98.7
Povprečje(%):	86.0	91.5	91.8

8.5.1 Diskusija

V tem poglavju v znanstvenih raziskavah obravnavamo prednosti in slabosti predlaganih asociativnih klasifikatorjev.

S programsko opremo WEKA smo ustvarili pravila o združenju razredov za vse asociativne klasifikatorje in raziskali druge algoritme klasifikacije, ker je WEKA odprtokodna programska oprema, ki vključuje vse potrebne pristope klasifikacije in je razmeroma enostavna za uporabo in programiranje. Predlagani algoritem ima nekaj omejitev:

- Da bi dobili "dovolj" pravil za zvezo razredov ("dovolj" pomeni vsaj 5-10 pravil za vsako vrednost razreda - to se v glavnem zgodi pri neuravnoteženih naborih podatkov) za vsako vrednost razreda in dosegli primerno splošno pokritost, moramo uporabiti ustrezno minimalno podporo in minimalno pragovi zaupanja. To pomeni, da bi morali upoštevati porazdelitev razredov vsakega nabora podatkov.
- Manjkajoče vrednosti ni treba odstraniti iz nabora podatkov, ker lahko vsi klasični algoritmi klasifikacije in naša metoda neposredno obravnavajo manjkajoče vrednosti.
- Numerične attribute je treba diskretizirati, ker lahko asociativni klasifikacijski modeli obdelujejo samo nominalne attribute.
- Problem prekrivanja zunanjega razreda (to pomeni, da imajo nekateri vzorci iz različnih razredov zelo podobne značilnosti) smo poskušali rešiti tako, da smo KAP razvrstili glede na vrednost njihovega razreda in problem prekrivanja med razredi (lahko zajema več pravil, ki spadajo v isti razred vzorci) z izbiro pravil glede na pokritost baze podatkov.

Za začetek algoritmi DC in DDC proizvajajo klasifikatorje, ki imajo v povprečju veliko manj pravil kot tisti, ki jih izdelajo klasične in asociativne metode učenja pravil, vključene v primerjavo. DC in DDC sta ustvarili razmeroma manjše število pravil o večjih naborih podatkov v primerjavi z vsemi drugimi metodami razvrščanja, medtem ko je CDC ustvaril nekoliko večje število pravil razvrščanja v primerjavi z DC in DDC, vendar so v povprečju vse predlagane metode dosegle najboljše rezultate glede razvrščanja pravila.

Vsi predlagani klasifikatorji so med vsemi pristopi, ki se učijo pravil, dosegli najboljšo natančnost nabora podatkov »Rak dojke« in »Spect.H«.

Natančnost metode CDC na "Hayes.R", "Adult", "Lymp" in "Connect4" je bila najboljša v primerjavi z vsemi drugimi klasifikacijskimi modeli, ki temeljijo na pravilih, vključno z DC in DDC.

Glavna pomanjkljivost metode DC je doseganje večje pokritosti nekaterih manjših naborov podatkov. To dejstvo ni presenetljivo, saj je reprezentativni CAR izbran na podlagi središča grozda in ne preverja pokritosti. DDC in CDC nimata te težave, ker je nov pristop za izboljšanje pokritosti in natančnosti predlagan v reprezentativnem postopku izbire CAR. Empirična ocena predlaganih klasifikatorjev je prikazana na Sliki 8.1.

Slika 8.1 prikazuje, da so vse tri metode dosegle podobno povprečno natančnost (DC: 82,5%; DDC: 83,3%; CDC: 83,8%) in povprečno pokritost (DC: 86,0%; DDC: 91,5%; CDC: 91,8%). Toda DC in DDC sta dosegla nekoliko boljše rezultate kot CDC

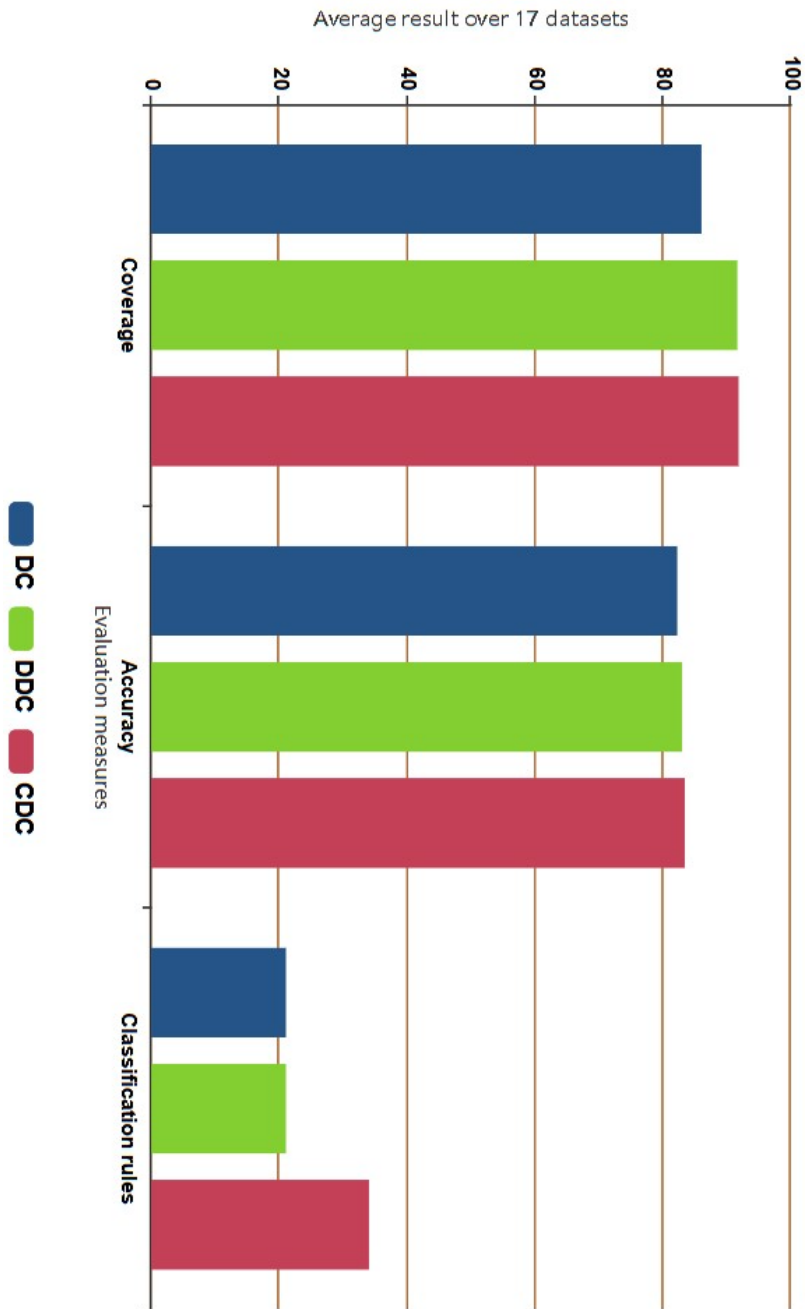


Figure 8.1: Primerjava uspešnosti našega predlaganega asociativnega klasifikatorja.

glede klasifikacijskih pravil z 22 oziroma 36. Čeprav je CDC dosegel boljšo pokritost kot DC, je dosegel slabše rezultate glede klasifikacijskih pravil kot ta metoda.

Slika 8.2 predstavlja primerjavo med predlaganimi asociativnimi klasifikatorji glede števila klasifikacijskih pravil.

Ker sta metodi DC in DDC enaki, smo jih združili v dobljeni sliki. Iz rezultata lahko razberemo, da je metoda CDC ustvarila večji klasifikator kot DC in DDC na večjih naborih podatkov. Natančneje, ko se velikost nabora podatkov poveča, CDC običajno ustvari večje klasifikatorje kot DC in DDC.

8.6 Zaključek in nadaljnje delo

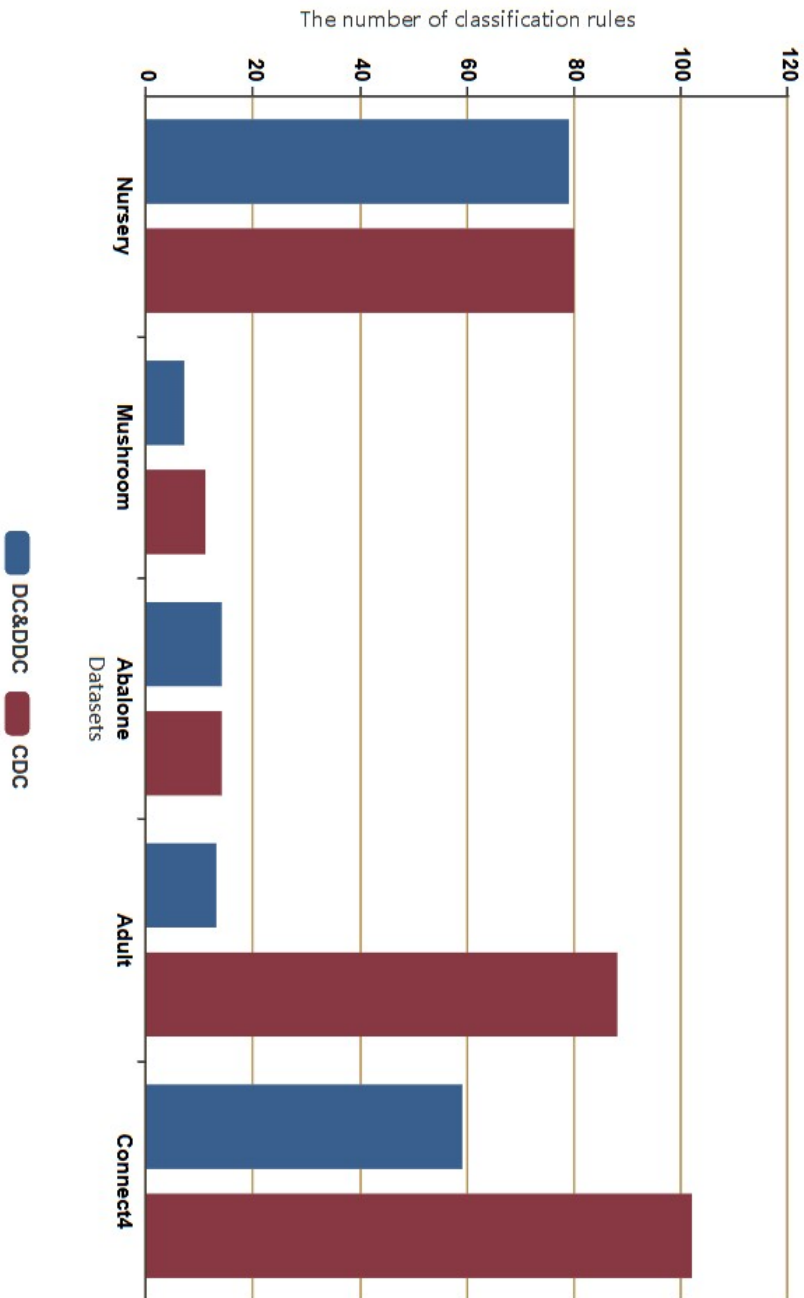
V tej disertaciji smo predstavili pet asociativnih klasifikatorjev. Prva dva - SA: enostavni asociativni klasifikator (za vsako vrednost razreda je izbrano vnaprej določeno število klasifikacijskih asociacijskih pravil) in J&B: asociativni klasifikator, ki temelji na pokritosti (pravila se izberejo na podlagi pokritosti učnih primerov) - sta bila uvedena kot del preliminarne raziskave, da bi pokazali, da zmanjšanje minimalnega zaupanja klasifikacijskih asociacijskih pravil, povečanje njihovega števila in pokritosti učnih primerov povzroči večjo klasifikacijsko točnost naučenih modelov. Z uporabo teh informacij so zadnji trije predlagani asociativni klasifikatorji, del naše glavne raziskave, nove metode, katerih cilj je bistveno zmanjšati velikost naučenih modelov, hkrati pa ohraniti njihovo klasifikacijsko točnost.

Trije novo-predlagani asociativni klasifikatorji so: DC (DC metoda temelji na neposredni mere razdalje (IBDM), metoda za identifikacijo reprezentativnega klasifikacijskega asociacijskega pravila pa temelji na središču skupine (RCC)), DDC (metoda DDC temelji na neposredni meri razdalje (IBDM), metoda za identifikacijo reprezentativnega klasifikacijskega asociacijskega pravila pa temelji na pokritosti učnih primerov (RDC)) in CDC (metoda CDC temelji na kombinirani meri razdalje (WCDM), metoda za identifikacijo reprezentativnega klasifikacijskega asociacijskega pravila pa temelji na pokritosti učnih primerov (RDC)). Vsi trije asociativni klasifikatorji temeljijo na združevanju v skupine.

Rezultati eksperimentalne evalvacije kažejo, da smo zastavljene raziskovalne cilje dosegli, saj naše predlagane metode producirajo kompaktno in smiselno, a hkrati natančne klasifikatorje z uporabo izčrpnega preiskovanja prostora učnih primerov ter uporabo omejitev in razvrščanja v skupine. Predlagane metode DC, DDC in CDC so tako znatno zmanjšale število klasifikacijskih pravil, hkrati pa ohranile klasifikacijsko točnost, ki ni bila signifikantno različna od tiste pri najsodobnejših algoritmih za klasifikacijo. Poleg tega eksperimenti kažejo, da je bilo število naučenih pravil pri naših klasifikatorjih, v primerjavi s sodobnimi klasifikacijskimi algoritmi, v povprečju 2 do 4-krat manjše. To razmerje je bilo še večje pri učnih množicah z večjim številom primerov.

Glavna pomanjkljivost predlaganih metod je njihova časovna zahtevnost. Medtem ko nekateri algoritmi za klasifikacijo (npr. C4.5 ali PART) uporabljajo požrešni pristop deli-in-vladaj, ki je zelo hiter, drugi uporabljajo pristope razveji-in-omeji, prekrivni pristop in/ali druge t.i. "izčrpane" pristope, ki so veliko počasnejši. Naše predlagane metode spadajo v slednjo kategorijo metod, ki uporabljajo izčrpano

Figure 8.2: Primerjava med našim predlaganim asociacijskim klasifikatorjem in ostalimi modeli glede velikosti na izbranih naborih podatkov.



preiskovanje prostora učnih primerov za iskanje "dobrih" klasifikacijskih asociacijskih pravil. Poleg tega pa, v fazi post-procesiranja, uporabijo še razvrščanje v skupine, da bi še dodatno zmanjšale število klasifikacijskih asociacijskih pravil, kar se dodatno pozna na časovni in prostorski zahtevnosti.

V nadaljnjem delu nameravamo predlagane metode DC, DDC in CDC optimizirati ter tako njihovo časovno zahtevnost vsaj nekoliko približali najsodobnejšim algoritmom za učenje pravil, ki temeljijo na principu "deli-in-vladaj", kar jih bo naredilo primernejše za analizo masovnih podatkov. Poleg tega je najprej potrebna analiza časovne zahtevnosti naših metod, da se opredelijo deli naših algoritmov, ki jih je mogoče optimizirati ali povzporediti.

Druga obetavna usmeritev za nadaljnje raziskave asociativnih klasifikatorjev je raziskati načine vključevanja numeričnih atributov v modele. Na tem področju že potekajo raziskave, vendar lahko uporaba razvrščanje v skupine na klasifikacijskih asociacijskih pravilih odpre nove, potencialno zanimive poglede na to področje.

V naši raziskavi smo za razvrščanje v skupine klasifikacijskih asociacijskih pravil uporabili hierarhično aglomerativno metodo razvrščanja v skupine s privzetimi parametri. Uporaba različnih sklopov parametrov razvrščanja v skupine ali celo različnih algoritmov razvrščanja v skupine in/ali različne mere razdalje so vse možne usmeritve za prihodnje raziskave glede učenja natančnih, učinkovitih, smiselnih in kompaktnih klasifikatorjev.

8.7 Prispevki k znanosti

Rezultati te doktorske disertacije bodo prispevali k znanosti na področju računalništva, natančneje na področjih podatkovnega rudarjenja in strojnega učenja. Prispevki doktorske disertacije k znanosti so sledeči:

- Predlagali smo nove (smiselne) mere razdalje (podobnosti) za izračun podobnosti med klasifikacijskimi asociacijskimi pravili. Ker v literaturi ni najti dovolj mer podobnosti (razdalje) med klasifikacijskimi asociacijskimi pravili, bo to pomembno prispevalo k znanosti na tem področju. V tem raziskovalnem delu smo predlagali dve normalizirani meri razdalje: prva, neposredna mera razdalje (ki upošteva strukturo klasifikacijskih asociacijskih pravil) in druga, kombinirana mera razdalje (ki upošteva tako strukturo pravil, kot njihovo pokritost učnih primerov) - združuje neposredno in posredno mero razdalje;
- Z uporabo normalizirane mere podobnosti (razdalje) identificiramo skupine klasifikacijskih asociacijskih pravil in samodejno določimo optimalno število identificiranih skupin;
- Definiramo dve metodi (na podlagi težišča skupin in skupne pokritosti učnih primerov za posamezno skupino) za identifikacijo reprezentativnega klasifikacijskega asociacijskega pravila znotraj vsake skupine pravil. Ta reprezentativna klasifikacijska asociacijska pravila na koncu tvorijo kompaktni in smiselni klasifikator;

- Implementiramo vse predlagane metode v novo aplikacijo, ki jo uporabimo za reševanje realnih problemov na "dobro poznanih" množicah učnih primerov.

Kazalo vsebine

Kazalo slik	viii
Kazalo algoritmov	ix
Kazalo tabel	x
1 Uvod	1
1.1 Znanstveno ozadje	1
1.1.1 Asociacijska pravila	3
1.1.2 Klasifikacijska pravila	3
1.1.3 Razvrščanje v skupine	7
1.2 Pregled literature	9
1.3 Prispevki k znanosti in metodologija	12
1.4 Pregled vsebine	14
2 Odkrivanje klasifikacijskih asociacijskih pravil	17
2.1 Algoritmi za rudarjenje pogostih postavk	17
2.1.1 (naivna) Metoda s surova silo	18
2.1.2 APRIORI (nivojski) pristop	19
2.1.3 ECLAT algoritem	20
2.1.4 Pristop pogostih dreves vzorcev: algoritem FP-Growth	21
2.2 Klasifikacijska asociacijska pravila	23
3 Asociativna klasifikacija	25
3.1 Preprost pristop k asociativni klasifikaciji (SA)	25
3.1.1 Eksperimentalna evalvacija SA pristopa	28
3.2 Asociativna klasifikacija po J&B pristopu	31
3.2.1 Eksperimentalna evalvacija J&B pristopa	39
4 Mere razdalje	47
4.1 Indirektne mere razdalje	47
4.2 Nova “direktna” mera razdalje	50
4.3 Nova “kombinirana” mera razdalje	51

5	Identifikacija skupin klasifikacijskih asociacijskih pravil (CAR)	53
5.1	Particijski algoritem razvrščanja v skupine	53
5.2	Hierarhični algoritem razvrščanja v skupine	55
6	Identifikacija reprezentativnega klasifikacijskega asociacijskega pravila znotraj skupine	59
6.1	Reprezentativno pravilo (CAR), ki temelji na težišču skupine	59
6.2	Reprezentativno pravilo (CAR), ki temelji na pokritosti primerov	59
6.3	Končni asociativni klasifikator	60
7	Eksperimentalna evalvacija in diskusija	64
7.1	Diskusija rezultatov	71
8	Zaključki in nadaljnje delo	75
	Literatura in viri	76
	Stvarno kazalo	86
	Povzetek v slovenskem jeziku	87
	Kazalo (v slovenskem jeziku)	101
	Stvarno Kazalo (v slovenskem jeziku)	103

Stvarno Kazalo

- ActOcc(R)*, 23
- Conf(R)*, 23
- Supp(R)*, 23
- d^{CMBP} , 48
- α , 51
- d^{WCDM} , 51
- d^{AMBD} , 48
- d^{CMBL} , 49
- AHCCLC*, 57
- AHCCLH*, 57
- numrules*, 27
- “razredno odvisna” diskretizacija, 25
- APRIORI, 18
- asociacijska pravila (AP), 1
- asociativna klasifikacija (AK), 1
- atribut, 3
- C-množica ID-jev, 10
- CDC, 63
- DC, 63
- DDC, 63
- direktna mera razdalje, 12
- evklidska razdalja, 47
- F-mera, 39
- FP-drevo, 22
- FP-growth, 18
- hierarhično razvrščanje v skupine, 2
- indirektne mere, 12
- izčrpno iskanje, 75
- klasifikacijska asociacijska pravila, 8
- klasifikator, 25
- kombinirana mera razdalje (WCDM), 51
- kompaktno, 26
- L, 65
- manjkajoče vrednosti, 71
- meja, 51
- mera podobnosti, 12
- mera razdalje temelječa na postavkah (IBDM), 50
- mere ustreznosti, 31
- mere zanimivosti, 11
- minimalna podpora, 2
- minimalno zaupanje, 2
- N, 65
- nadzorovano učenje, 1
- natančnost, 38
- nenadzorovano učenje, 2
- neprekrivajoča se pravila, 48
- neuravnovešene podatkovne množice, 2
- objekti, 53
- od spodaj navzgor, 2
- od zgoraj navzdol, 12
- odločitvena drevesa, 7
- optimalno število skupin, 8
- osamelci, 54
- parni T-test, 31
- podatkovno rudarjenje, 1
- podobnost znotraj skupine, 53
- pogojni del, 23
- pokritost primerov, 9

popolna povezanost (najoddaljenejši
sosed), 57
posledični del, 23
predponsko drevo, 20
prekomerno prileganje, 9
preprost asociativni klasifikator, 75
prečno preverjanje, 40
priklic, 38
problem nakupovalnih košaric, 3
problem prekrivanja znotraj razreda,
60
problem prekrivanja zunaj razreda, 59
proces odločanja, 3

razred, 12
razvrščanje v skupine, 2
RCC, 59
RDC, 60
realni podatki, 1
referenčna točka, 55
reprezentativni CAR, 13
rezanje dendrograma, 8
rudarjenje klasifikacijskih pravil, 1
rudarjenje množic pogostih postavk, 3

SA, 26
skupno pokritje, 31

statistika, 1
statistično primerljivo, 65

tehnike rezanja, 2
temelječ na pravilih, 2
tesnost, 12
testiranje statistične značilnosti, 31
testna množica, 29
točnost, 2
transakcijski, 1

umetna inteligenca , 1
upravljanje odnosov s strankami, 4
ustavitveni kriterij, 31

vertikalni format, 10
več-dimenzionalna množica podatkov,
3
večinska razredna vrednost, 69

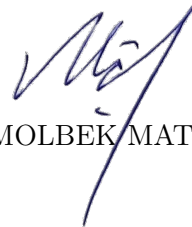
W, 65

zapis, 17
zdravstvena diagnoza, 4
zmanjšati število klasifikacijskih
pravil, 2

časovna učinkovitost, 75

Declaration

I declare that this PhD Thesis does not contain any materials previously published or written by another person except where due reference is made in the text.



JAMOLBEK MATTIEV