SUMMARY

Aims and objectives
CN2-MSD is a typical multi-class subgroup discovery algorithm but only applicable for single-target datasets, and so the primary aim is to enhance the algorithm in order to handle multi-target data; The second aim is to design different adaption methods using the ideas of exceptional model mining; The algorithm's performances are closely related to the applied adaption methods, and so sufficient experiments should be carried out to evaluate different methods' performances.
Project type
The project is more of investigation, because it searches methods to solve the adaption problems and aims at implementing a new technique which will be based on an existing algorithm *CN2-MSD* and an idea (exceptional model mining). It is inspired by the facts that data analyst are not always only interested in unique target when doing subgroup discovery. The final outcome of the project will be a program as well as feedbacks of its experimental performances.
A summary of the implementation
The multi-target subgroup discovery algorithm is implemented through rewriting the original program of *CN2-MSD*;
More researches are carried out on fields like multi-label classification, cluster, dependency detection and statistic tests;
4 new methods are designed and totally 6 methods are implemented in the project;
Sufficient data is collected, and the data is in different type namely with numerical or nominal attributes;
Experiments are carried out to examine if the rules generated by these methods satisfy the descriptive measures and if they are meaningful in real cases;
5 key elements of the project
Totally 6 methods are implemented for doing multi-target subgroup discovery using the framework of *CN2-MSD*;
4 of the them are new and with different theoretical properties: two of the them respectively apply a correlation model and a complete independency model, and with their heuristics defined. The other two respectively incorporate mechanisms of dependency detection and cluster;
Synthetic datasets are generated artificially, and used to evaluate if the proposed methods can accurately find a intentionally included subgroup. And it is found that no matter what kind of data is provided, the method *CW* can accurately find the subgroup out;
3 real life datasets are collected from the European social survey and used to examine if the proposed methods can generate meaningful rules; Interesting patterns are found like: Compare with the general population, citizens who are younger than 68.5 with the highest degree of NVQ4/NVQ5 but not living in the Northern Ireland and not widowed have relatively poorer impressions on immigrants.
An experiment is carried out to compare the performances of different methods on multiple datasets with statistic tests.

ACKNOWLEDGEMENTS

I owe my deepest gratitude to my supervisor, Professor Peter Flach, for his guidance, encouragement and supervision. The dissertation would not have been possible without his help;
I would also like to thank Tarek Abudawood, not only because I inspired from his paper but also because he provides me essential materials that I can proceed the project;
Last but not least, I should thank for my parents for their encouragement and financial support.

Abstract

Subgroup discovery is a halfway between classification and association rule learning. It aims at generating rules denoting the most interesting patterns of a data. Related researches are mainly carried out on binary class single-target datasets. The report investigates adaption methods potentially available for multi-target subgroup discovery. Six methods which are supported by different theoretical background are considered, where four of them are new. They are implemented through adapting the multi-class subgroup discovery algorithm CN2-MSD and using the idea of exceptional model mining and mechanisms of cluster or dependency detection. Three experiments are carried out to evaluate the methods' respective performances on different aspects. Synthetic datasets are generated artificially to examine methods' abilities of finding subgroups. No matter what data is provided, method *CW* can accurately represent the subgroup. The method also finds the most unusual patterns from two of three real life datasets. The comparisons of experimental results are tested using statistic techniques. It is observed that *CW* finds the most representative rules and the incorporation of dependency detection improves the algorithm's performances.

Contents

A Unification of Subgroup Discovery and Exceptional Model Mining

1 Introduction

Classification rule learning finds rules from a given set of labeled instances. It makes use of the generated rules to solve predication or classification problems of new data [1]; *Association rule learning* identifies frequent itemsets and then generates conditional rules to uncover internal regularities and interesting patterns in a database [2]. Both classification and association rule learning are typical uses of *rule induction* which aims at interpreting knowledge of a given data set by inducing rules. Rule induction is a common form of data mining and widely used in knowledge discovery, because its induced rules can intuitively interpret interesting knowledge contained in data.

To address the differences between classification and association rule learning, three aspects should be focused on. Firstly, in classification rule learning, the training instances should have been labeled. However, database of association rule learning has no labeled attributes. Secondly, the *heuristic functions* of classification rule learning evaluate *candidates* by using criteria of classification or prediction accuracies. For association rule learning, other criteria like support or confidence are evaluated. The so called candidate can be an induced rule from classification or association rule learning. Finally, they have different goals which have been referred.

Subgroup discovery is considered as a halfway between classification and association rule learning. Associated with classification rule learning, subgroup discovery is also provided a set of labeled examples and is implemented by supervised learning algorithms. The goal of subgroup discovery is more like association rule learning that it aims at identifying interesting dependent relations between targets and partitions of attribute variables. Subgroup discovery additionally emphasizes on two more constrains. On one hand, all of its attributes except the targets are independent with each other, and on the other hand, the targets should have been described in each induced rule when subgroup discovery is finished. The heuristic function of subgroup discovery is also different from both referred rule learning tasks that it evaluates statistical or other user-defined quality criteria. An important parameter is the trade-off between generality and interestingness. Generality means sufficient instances should have been covered by an induced rule. And interestingness indicates how different the distribution of a subgroup is from the general distribution of the population.

Here is an example of a discovered subgroup. "*The unemployment rate is above average for young men with a low educational level*" (from [3] P.1). The state-

ment can be expressed in a form of rule: <*unemployment rate=above average*> ← <*age=young*>, <*gender=men*>, <*educational level=low*>. As it can be seen, subgroup discovery is well suited for finding data dependency, hence can be applied by analyst for decision support. However, in order to apply subgroup discovery to actual data analysis, issues should be considered. Firstly, most of the traditional subgroup discovery researches only focus on binary-class context (only two classes of *positive* and *negative*). Secondly, the traditional subgroup discovery only deals with database with one single target variable of nominal form. To solve these problems, different approaches have been proposed. For example, *CN2-MSD* [4] is a widely applied algorithm of subgroup discovery modified from *CN2-SD* [5], and it can be used to solve the problems of multi-class subgroup discovery. However, *CN2-MSD* is still not suitable when handling dataset with multi targets. And this leads to the primary goal of this project.

The report consists of 5 sections. The first section introduces the project's relevant contexts, motivations, aims and objectives; The second section involves the related background knowledge about the algorithm *CN2-MSD* and the idea of exceptional model mining; The problems met in the project implementation is specified in the third section. It is followed by a detailed description of the proposed methods. And this section is ended with the project design. In the fourth section, three experiments are carried out to evaluate the methods' performances; The report is finally concluded in the last section with a brief introduction of the possible future works.

Aims and Objectives

- Enhance the algorithm of *CN2-MSD*, in order to handle multi-target datasets.
- Design different methods using the framework of *CN2-MSD*.
- Implement all of the proposed methods.
- Critically evaluate their performances on sufficient datasets.

As for the first aim, the literatures related to *CN2-MSD* are reviewed, which also cover the ground knowledge of subgroup discovery. Besides, the algorithms of *CN2* [6] and *CN2-SD* are also studied; In order to design appropriate methods for doing multi-target subgroup discovery, research fields like exceptional model mining and multi-label classification are further studied; As the original algorithm of *CN2-MSD* is implemented in *java* using *Weka*, the system is designed and implemented in the same environment. And so the functions and data structures provided by *Weka* should be understood; Finally, the methods' performances are evaluated on multiple datasets using different experiments.

2 Background

2.1 Subgroup Discovery

In the following subsections, the framework of subgroup discovery is firstly introduced, followed by which is a detailed review of heuristic functions structured on numbers of targets. With a framework has been defined, a task of subgroup discovery can be conducted by an appropriate algorithm. Hence in the last subsection, algorithms of subgroup discovery are introduced.

2.1.1 The Framework of Subgroup Discovery

The framework of subgroup discovery consists of four properties: *The Targets* reflect users' interested properties; The *Subgroup Description Language* which interprets the discovered subgroups in form of rules; A *Heuristic Function* evaluates the qualities of an induced rule; A *Search Strategy* specifies the search space under which hypotheses are constructed [1].

The Target In most cases, the target is the class label of data and appeared as a single variable in nominal type. A nominal target variable is a small set of values corresponding with specific intervals or categories. Recall the example of "*The unemployment rate is above average for young men with a low educational level*" (from [3] P.1). The *unemployment rate* is a nominal target with its value matching with category "*above average*". There exists a specific subset of nominal target context, the target variable of which only contains two values that are often expressed as positive or negative. An example of such target variable could be: 1 refers to true and 0 to false. A target can also be in numerical type which often appears in other tasks of learning like regression or clustering. A numerical target variable is a range of discrete or continues values, and in cases the range can be infinitely large [7].

Although most of the current techniques are not able to handle tasks of subgroup discovery on multi-target context, the target of data is not essential to be an unique single variable. *Exceptional Model Mining* provides an idea to solve such problems [8], which takes all of the target variables as a model. Consequently, the primary task of discovering partitions with different distributions from the general population is replaced by finding subgroups whose fitted models are exceptional to a same model which fitted to the entire data. Here is an example of a subgroup derived from a multi-target context and proposed in [8]: *under conditions, the correlation between the price per square meter and the lot size will be different from the general population*. There exists two numerical target variables in the statement: the *price per square meter* and the *lot size*. Both variables are then combined together by generating a correlation model. Under the assumption of this example, a subgroup could be in form of "*<Correlation=-0.09>* ← *<drive=1>*, *<rec room=1>*, *<nbath>=2>*", compared with the correlation derived from the entire database which equals to *0.549*, it can be identified that the model fitted to the subgroup behaves exceptionally with respect to the model of the population.

The Subgroup Description Language Each subgroup is expressed as an induced rule. A rule is appeared in the form of *Head* ← *Body*, where *Body* describes the conjunction of *selectors* under which examples are covered. The *Head* is in fact a class label in data. The description of each attribute is called a selector which describes the basic test on the attribute [6]. Recall the subgroup "*The unemployment rate is above average for young men with a low educational level*" (from [3] P.1), where there exist three selectors which describe *age*, *gender* and *educational level* respectively. Change the subgroup into the form of a rule: <*unemployment rate=above average*> ← <*age=young*>, <*gender=men*>, <*educational level=low*>, where the conjunction of the last three elements represents the *Body* part, and the *unemployment rate* of "*above average*" is the head. A set of individuals covered by one unique rule are considered as belonging to one subgroup. And consequently, the rest individuals are belonging to this subgroup's complement.

The Search Strategy Subgroup hypothesis is constructed in the form of general-to-specific [9], which expands a current hypothesis by appending additional selectors. In subgroup hypothesis construction, *Brute-force* exhaustive search strategy tests all combinations of selectors [10]. However, it is not acceptable since the search space is exponentially increased when more and more selectors are concerned [3]. A more efficient search strategy should be able to qualify all of the enabled hypotheses according to specific criteria, and only expend hypotheses which satisfy users' defined standards. This is what *beam search* does. Beam search iteratively expands a fixed number of most promising hypotheses as candidate subgroups, the number of which equals to the pre-defined beam width. Algorithms which incorporate beam search strategy will be introduced in this section.

The Heuristics A heuristic function maps a subgroup into a real value which indicates a subgroup's quality. And evaluation criteria are normally determined by users' interests. Recall that a popular criteria is based on tradeoff between generality and interestingness, which has been defined in [5] as that a subgroup being found should be large enough and with a relative unusual distribution on targets from the entire database. And interestingness indicates if a subgroup is distributional unusual regarding with certain properties [11]. Generality maximizes a subgroup's size relative to the entire population. A reason to maximize the generality is that the so-called interestingness of a subgroup will not be interesting to potential users if the subgroup is too small.

2.1.2 A Review of Heuristics

In most cases of subgroup discovery, the target is a single variable in nominal type. Tasks of subgroup discovery can either be on binary-class context or multi-class context, each of which is evaluated by their respective heuristics. With the correlation between both, heuristics of multi-class context can be implemented by modifying the latter one. As for Binary-Class Contexts, values of a target variable can only be positive or negative. Typical heuristics of such problem is listed as follows.

An Exemplary Heuristic Most of the heuristics referred in this section consist of two parts, each of which evaluates one of the two criteria: generality and interestingness. An overall tradeoff is then computed by their combination. A good subgroup will be assigned with a high value. And finally the best subgroups are interpreted to the analyst. An exemplary heuristic is defined as follows [3].

$$q_{BT}(Class \leftarrow Cond) = \frac{P(Class|Cond) - P(Class)}{\sqrt{P(Class) \times (1 - P(Class))}} \sqrt{n} \sqrt{\frac{N}{N-n}} \tag{1}$$

Where $P(Class|Cond)$ is the conditional probability which describes how likely an individual would be classified into a specific class under the condition of being covered by a rule *Cond*. The estimated probability $P(Class)$ is the proportion of a class of individuals with respect to the entire population. n and N are sizes of the subgroup and the population respectively.

The Weighted Relative Accuracy Heuristic (*WRAcc*) Weighted Relative Accuracy [5] is the most commonly used heuristic in subgroup discovery. It also consists of two components and more intuitively shows the tradeoff between both criteria. *WRAcc* is defined as:

$$WRAcc(Class \leftarrow Cond) = P(Cond) \times (P(Class|Cond) - P(Class)) \tag{2}$$

As shown in Formula 2, $P(Cond)$ is the component of generality which denotes the size of a subgroup relative to the population. And the interestingness is computed by the difference between $P(Class|Cond)$ and $P(Class)$. *WRAcc* is widely used in binary-class context and has been modified in different ways to adapt to tasks on multi-class context [4].

Multi-class classification can be solved in the following two ways: compare all pairs of classes (*one-vs-one*); or compare each class with the union of all of the other classes (*one-vs-rest*) [4]. In both cases, the cost is expensive as the determination of a winner class needs iterations of comparisons. Heuristics on multi-class context aim at reducing the cost by generating an overall model. [4] proposes and experimentally evaluates six such heuristics, three of which are modified from *WRAcc* and the others are based on different theories.

One-vs-Rest Multi-Class *WRAcc* One-vs-Rest Multi-class *MWRAcc* (*WRAcc*) averages absolute values of a subgroup's *WRAcc* on each class. Its definition is defined as follows:

$$MWRAcc(b) = \frac{1}{n} \sum_{i=1}^{n} |WRAcc_i(b)| \tag{3}$$

Where n is the number of classes, b is the subgroup to be evaluated. For each class the *WRAcc* is computed with respect to the union of all of the other classes.

Weighted Multi-class *WRAcc* Weighted Multi-class *WRAcc* (*WMWRAcc*) assumes that to evaluate a subgroup's quality, each class's size should be under consideration.

WMWRAcc is defined as the followed formula and the size of a class i relative to the population is expressed as $\frac{E_i}{E}$.

$$WMWRAcc(b) = \frac{1}{n}\sum_{i=1}^{n}\frac{E_i}{E}|WRAcc_i(b)| \tag{4}$$

One-vs-One Multi-class *WRAcc* As shown in formula 5, one-vs-one *MWRAcc* firstly calculates a subgroup's *WRAcc* on each pair of classes, and then sums up them together to get an overall value which denotes the subgroup's quality.

$$\begin{aligned} MWRAcc^{1vs1}(b) &= \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1;j\neq i}^{n}|WRAcc_{i,j}(b)| \\ WRAcc_{i,j}(b) &= \frac{e_i+e_j}{E_i+E_j}\left(\frac{e_i}{e_i+e_j}-\frac{E_i}{E_i+E_j}\right) \end{aligned} \tag{5}$$

Mutual Information Mutual information refers to the mutual dependence between two variables. Having known an individual is in a subgroup, the mutual information donates how certain it can be derived that the individual also belongs to a certain class. With variable B which donates if a random instance is an individual of a certain subgroup and variable L indicates which class the instance is belonging to. The marginal and joint entropies as well as the mutual information score of subgroup b are shown in the followed formulas. The higher the score is the bigger the deviation of the subgroup from the population.

$$\begin{aligned} H(B) &= -\frac{e}{E}log\frac{e}{E}-\frac{E-e}{E}log\frac{E-e}{E} \\ H(L) &= -\sum_{i=1}^{n}\frac{E_i}{E}log\frac{E_i}{E} \\ H(B,L) &= -\sum_{i=1}^{n}\left(\frac{e_i}{E}log\frac{e_i}{E}+\frac{E-e_i}{E}log\frac{E-e_i}{E}\right) \\ MI(b) &= H(B)+H(L)-H(B,L) \end{aligned} \tag{6}$$

Chi-Squared The interestingness of a subgroup can also be indicated by the differences between observed and expected numbers of individuals of the subgroup in each class. An observed number is the actual number of individuals which are in the subgroup and assigned with this class label. And an expected number of class i is determined by the overall distribution and can be expressed by $\frac{E_i e}{E}$. Chi-square sums the squared differences between observed and expected values divided by the expected values [4]. The chi-square score is defined as:

$$Chi^2(b) = \sum_{i=1}^{n}\left(\frac{\left(e_i-\frac{E_i e}{E}\right)^2}{\frac{E_i e}{E}}+\frac{\left((E_i-e_i)-\frac{E_i(E-e)}{E}\right)^2}{\frac{E_i(E-e)}{E}}\right) \tag{7}$$

Gini-split Gini-split measures how much likely an instance would be classified mistakenly if it is randomly assigned a label only according to the class distribution. A big Gini-split score indicates an obvious deviation of the subgroup from the population. The Gini-split is defined as follows:

$$\begin{aligned} GS(b) = & \frac{1}{n}\sum_{i=1}^{n}\frac{E_i(E-E_i)}{E^2} - \frac{1}{n}\sum_{i=1}^{n}\frac{e}{E}\frac{e_i(e-e_i)}{e^2} \\ & -\frac{1}{n}\sum_{i=1}^{n}\frac{E-e}{E}\frac{(E_i-e_i)((E-E_i)-(e-e_i))}{e^2} \end{aligned} \tag{8}$$

All of the six multi-class heuristics are experimentally evaluated in [4] on 20 UCI datasets. Because their respective performances are evaluated corresponding with different *weight schemas* [5], the experimental evaluations as well as evaluations criteria are described in the next subsection of *CN2-MSD*.

2.1.3 Algorithms of Subgroup Discovery

CN2-SD [5] is an algorithm of subgroup discovery implemented by modifying a standard rule learner *CN2* [6]. Both algorithms have a same rule searching procedure, but with their own heuristics and other different mechanisms. There is a problem of *CN2-SD* that it can only handle binary-class context. To solve problems of subgroup discovery on multi-class context, [4] implements *CN2-MSD* by modifying the heuristics of *CN2-SD*. In the following content, all of the three algorithms are illustrated in detail. It first introduces the mechanisms of *CN2*, and then illustrates the modifications of both of the other two algorithms.

***CN2* Induction Algorithm** *CN2* consists of two main parts: a bottom-up procedure (Algorithm 2) of beam search which aims at generating a single rule, and a top-down procedure (Algorithm 1) which repeatedly executes the beam search until a rule set is found [6]. A rule generated by *CN2* is expressed in the form of *Class* ← *Cond*, where *Cond* is a conjunction of selectors called a *complex*.

Algorithm 1 The Procedure of $CN2(E)$ (from [6] P.262)

```
Input: E is the training set;
Let RULE_LIST be ∅
while BEST_CPX ≠ nil and E ≠ ∅ do
    BEST_CPX = Find_Best_Complex(E)
    if BEST_CPX ≠ nil then
        let E′ be examples covered by BEST_CPX
        E = E − E′
        Let C be the most common class of examples in E′
        Add the rule C ← BEST_CPX to the end of RULE_LIST
    end if
end while
return RULE_LIST
```

Algorithm 2 The Procedure of $Find\_Best\_Complex(E)$ (from [6] P.262)

Parameter: $SELECTORS$ is the set of all possible selectors
Let $STAR$ be the set containing the empty complex
Let $BEST\_CPX$ be nil
while $STAR \neq \emptyset$ **do**
  $NEWSTAR = \{x \wedge y | x \in STAR, y \in SELECTORS\}$
  $NEWSTAR = NEWSTAR$ - $\{z_1 | z_1 \in NEWSTAR, z_1 \in STAR\}$
  $NEWSTAR = NEWSTAR$ - $\{z_2 | z_2 = null\}$
  for all complex $C_i \in NEWSTAR$ **do**
    if C_i is statistically significant and better than $BEST\_CPX$ **then**
      $BEST\_CPX = C_i$.
    end if
  end for
  while $|NEWSTAR|$ > user-defined maximum **do**
    Remove the worst complex from $NEWSTAR$
  end while
  Let $STAR$ be $NEWSTAR$
end while
return $BEST\_CPX$

The bottom-up procedure exports one best complex in one execution. It works iteratively as follows: Firstly, it generates a set of complexes, each of which is a conjunction of selectors. The initial complex is an empty conjunction which covers the entire population. A new complex is constructed by appending an additional selector to an original complex. Secondly, delete two kinds of newly constructed complexes which are considered as illegal: the null complex which reflects conflictions, for example: <*age=old*> $\wedge$ <*age=young*>, and the complex which has been specialized before (*a same complex which has appeared in the previous iteration*). In the procedure of specialization, it chooses a best complex as a candidate. And then evaluate if the current candidate is better than the best complex from the former iteration. The winner is retained as the best complex of the current iteration. At the same time, the beam search also retains a fixed number of "*good*" complexes. The fixed number is pre-defined as the beam width. Finally, after the previous iteration is finished. It enters a new one which starts from generating complexes by expanding the good complexes from the last iteration. The procedure continually executes until no more complexes can be constructed.

Two heuristics are used in this procedure, one of which evaluates how good a complex is. The heuristic works as follows: Firstly, it counts the number of examples which are covered by a complex of *Cond* (*N(Cond)*) and the number of such examples which additionally belong to a specific class of *Class* (*N(Class)*). Secondly, it computes the probability distribution of each class according to $P(Class \leftarrow Cond) = N(Class|Cond)/N(Cond)$. Finally, information entropy on each complex is calculated as $Entropy = -\sum P_i log_2(P_i)$. A better complex is valued a higher score. Throughout using the information-theoretic entropy measure, the heuristic prefers a complex which has two properties: on one hand most of the covered examples are distributed in one specific class, and on the other hand all of the covered exam-

ples are concentrated in as fewer classes as possible. Provide two distributions of *P1(0.7,0.1,0.2)* and *P2(0.7,0.3,0.0)*, the latter one is in favored.

The second heuristic calculates significance of each complex. The significance indicates how unlikely a complex is occurred by chance. It is designed to maximize the difference between two distributions. Here the first distribution is the class probability distribution of examples which have been covered by a complex (f_i). And the other one reflects the class probability distribution of the general population (e_i). To calculate the significance, CN2 uses likelihood ratio statistic: $LRS = 2 \sum f_i log(f_i/e_i)$ [6]. The lower the score is, the more likely the rule is occurred by chance.

The combination of both heuristics determines the quality of a complex. It then takes the best complex as the candidate and retains a number of complexes according to the beam width. A beam search procedure finally provides a best complex. It then counts the occurrences of different classes of all of the examples which are covered by the complex. The most common class will be taken as the head of a rule with the provided complex as the rule's body.

The top-down procedure iteratively executes the beam search to generate a rule set until it covers all of the examples no more reasonable rules can be found. In each iteration, it removes all of the covered examples when a best complex is found. Finally, CN2 generates a set of rules which are listed in order. To classify an example, each rule is tried in turn until one unique rule is fired. An issue of ordered rule set is that a rule is dependent on all of the other rules which appear before it. It leads to a problem of understanding [6]. To solve the problem, one can generate unordered rule set. Its procedure is different from the former one on two aspects: Firstly, it induces rules for one class in turn. Secondly, after a beam search is finished, it only removes positive examples that is covered by the rule and belonging to the referred class [5].

***CN2-SD* Subgroup Discovery Algorithm** Four properties of subgroup discovery are: targets, subgroup description language, search strategy and heuristic, of which the heuristic is the only property which distinguish subgroup discovery from classification rule learning. As for the other three properties: the type of a target is determined by users' interests and provided data; the description language of the form $Head \leftarrow Body$ is incorporated into both subgroup discovery and classification rule learning; and a beam search strategy which efficiently constrains the search space is favored by both induction tasks. *CN2-SD* is implemented by first modifying the heuristic of *CN2*, and then being incorporated other modifications: Firstly, it updates weights of all of the covered examples after each beam search; secondly, it takes weights of examples into account while calculating their *WRAcc*; Finally, it uses probabilistic classification in both cases of ordered and unordered rule set [5].

Modification of Heuristics The beam search has an iterative procedure of specializing and evaluating complexes, during which a heuristic measures how good a complex is. The criteria of evaluating qualities of complexes are different between classification rule learning and subgroup discovery. In classification rule learning, the criteria of classification or prediction accuracy is put at the first place, because it is targeted at generating a rule set as a classifier. Differently, subgroup discovery

induces rules to uncover interesting patterns. As it is introduced before, an important parameter of users' interests is the trade-off between generality and interestingness. Hence *CN2-SD* replaces the heuristic of accuracy by *WRAcc* which maximizes the trade-off between both criteria.

Incorporation of Example Weights *CN2* makes use of covering algorithm in the top-down procedure when constructing rule set. In each iteration, either all of the examples covered by an induced rule are removed (in case of *ordered rule set* generation) or examples covered by the rule and belongs to a specific class are removed (*unordered rule set*). One issue of such mechanism is that with more rules induced, more and more examples are removed. The retained examples cause a basin in the followed subgroup discovery process. On one hand the retained examples is not sufficient enough to represent the general distribution, and on the other hand the reduced number of examples directly influence the evaluation of generality of a complex. It is the reason why only the first few generated rules are interesting. To solve the problem, [5, 12] incorporate example weights into the covering algorithm. The modification is as follows: replace the mechanism of original covering algorithm which deletes all of the covered examples by a new mechanism which weights every example corresponding with times it is covered by different rules. A small weight indicates that the example has been covered by many different rules, and they should be discarded in some level. There're two weighting schemes [5] listed as follows:

Multiplicative Weights exponentially decrease an example's weight when it is covered by more and more induced rules. The weighting scheme is defined as $w(e_j, i) = \gamma^i$, where $0 < \gamma < 1$, and i refers the number of rules which cover the example e_j. *Additive Weights* decrease examples' weights with a different scheme: $w(e_j, i) = \frac{1}{i+1}$. The weight of an example determines how much the example should be considered of in the computation of *WRAcc*.

***WRAcc* with example weights** *WRAcc* is further modified by incorporating example weights. Being assigned with different weights, the coverred examples have different contributions in evaluating subgroups. The modified *WRAcc* is defined as follows:

$$WRAcc(Class \leftarrow Cond) = \frac{n'(Cond)}{N'}\left(\frac{n'(Class.Cond)}{n'(Cond)} - \frac{n'(Class)}{N'}\right) \qquad (9)$$

As it is shown in the formula 9, N' is the sum of weights of all of the examples. $n'(Cond)$ is the weight summation of examples which are covered by an rule. And $n'(Class.Cond)$ indicates the weight summation of all of the examples which are covered by this rule and belonging to a specific class.

Probabilistic Classification The incorporation of example weights results that each example may be covered by more than one rule. *CN2-SD* applies the mechanism of Probabilistic Classification which assigns each rule with a class distribution. To illustrate the mechanism more intuitively, [5] introduce an example which is described in the following table.

Table 1: Induced Rules with Probability Distribution (from [5] P.158)

$< leg = 2 >, < feather = yes > \rightarrow < Class = bird >$ $[1, 0]$
$< break = yes > \rightarrow < Class = bird >$ $[1, 0]$
$< size = large >, < flies = no > \rightarrow < Class = elephant >$ $[0.17, 0.83]$

As shown in table 1, to classify an unseen animal with attributes *two-legged, feathered, large and non-flying* which has fired with all of the three rules, the probability distribution of each rule should be added together. It generates a final distribution of [*0.72*,*0.28*], which classifies the animal into the class of bird.

***CN2-MSD* Subgroup Discovery Algorithm** As it is discussed before, *CN2-SD* is a subgroup discovery algorithm which only focuses on binary-class context. To solve tasks on multi-class context, *CN2-MSD* modifies *CN2-SD* by changing its heuristic of *WRAcc*. [4] investigates six heuristics, among which 3 heuristics are expanded from *WRAcc* and the others are based on different theories. All of these heuristics have been introduced in the subsection of heuristic review. In [4], the heuristics are evaluated on different aspects: *descriptive measure* is evaluated on users' interested criteria, the detail of which is listed below in the subsection of criteria evaluation; *predictive measure* is evaluated on predicting ability, namely accuracy and AUC. Additionally, the generated subgroups of each heuristic are taken as features for classification algorithm naive Bayes and J48. Throughout comparing the classification results, one can conclude if the generated subgroups optimize an algorithm's classification ability. Besides, the classification ability is also evaluated on the numbers of nodes and leaves of a tree generated by J48 as well as model construction times. Their respective performance is illustrated as that: heuristics based on *WRAcc* result fewer subgroups; other heuristics show additional power of prediction; the incorporation of example weight schemas with appropriate parameter setting results more subgroups and higher predictive power.

2.2 Exceptional Model Mining

So far a general framework of subgroup discovery has been provided. As it is described before, the framework finds partitions from a dataset where the observed distributions are different from the distributions of the entire population. For traditional subgroup discovery, the differences of distributions are induced by only considering one single target. However, in cases the targets can be more complex with more attribute variables. Hence a method of *Exceptional Model Mining* [8] is provided as an extension of the current framework. Exceptional model mining expands the framework on one additional mechanism by constructing models. The models are induced on multi-attributes (normally with one or more feature attributes $x_1, x_2, ..., x_n$, and optionally with the class attribute y). Each model is identified by structural properties, by comparing which an appropriate quality measure derives the distributional unusualness of a subgroup from its complement. Hence the main idea of exceptional model mining is now becoming: a model which is fitted to a specific subgroup should be different from the same model which applied to the entire population. And such a specific subgroup is considered to be exceptional with respect to the dataset.

As shown in figure 1, the targets of a dataset is approximated by combining two kinds of populations. And the instances are plotted by considering their targets. It is quite clear that something is going on in distribution *1*, but such a dependency is unlikely existing in distribution *2*. When two populations are mixed together, it is difficult to distinguish one from another. The purpose of exceptional model mining is to distinguish the subgroup from its complement by fitting a model to this database. It still remains two questions: how to choose an appropriate model and how to derive differences from same models. As for this problem, a principal component analysis model can be constructed. Instances of the subgroup should have a very different eigenvector from the population, by comparing what one can finds the subgroup out.

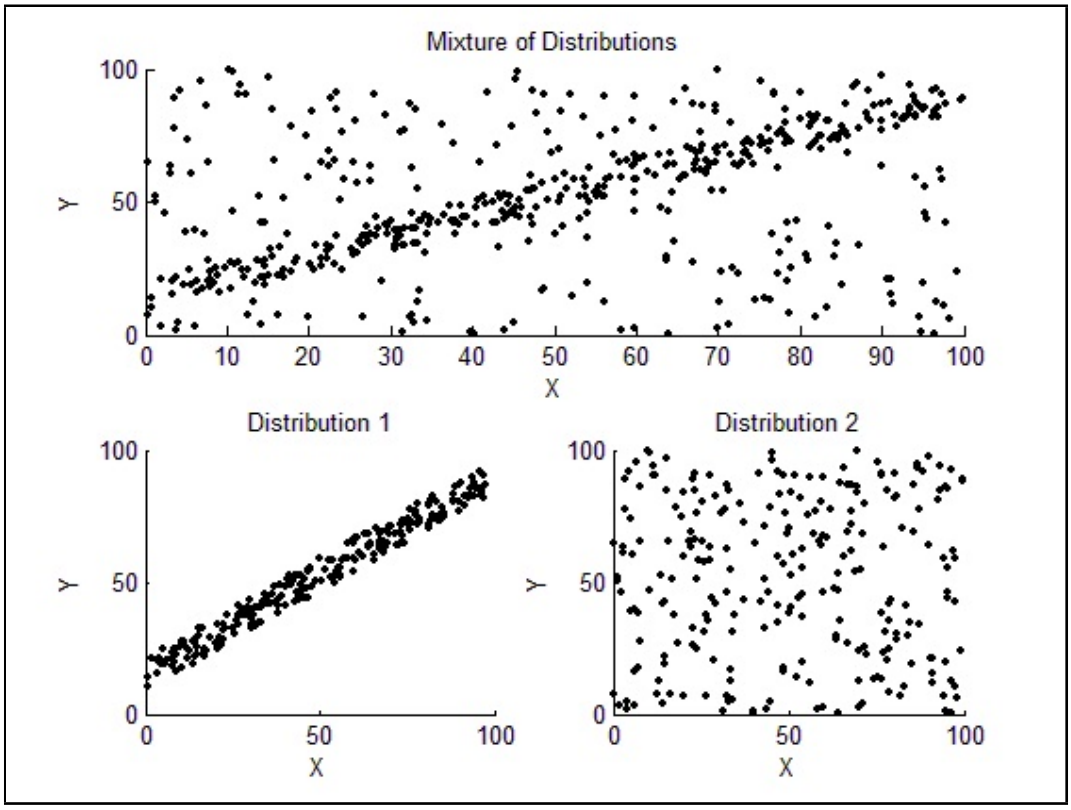

Figure 1: An Approximated Dataset with Two Distributions (reconstructed from the original idea of [13]).

2.2.1 Model Classes

An appropriate model is selected with corresponding to the targets of a database. For example, when no class attribute is provided then a correlation model can be used; And if the users' interested targets contain a numeric or normal class attribute, then a regression or classification model can be used. Different models are identified by their own structural features which should be compared by different quality measures.

Correlation Model If no class attribute is provided, exceptional model mining can construct a correlation model to represent the associations between target variables [14]. For two attributes with their respective variable x_1 and x_2, their correlation coefficients can be calculated as follows:

$$r = \frac{\sum(x_1^i - \bar{x}_1)(x_2^i - \bar{x}_2)}{\sqrt{\sum(x_1^i - \bar{x}_1)^2 \sum(x_2^i - \bar{x}_2)^2}} \tag{10}$$

Where x^i is the i^{th} example of the population, And $\bar{x}$ indicates to the mean value of all of an attribute's values. The distributional difference is reasonably being approximated by the *absolute difference* of correlation coefficients between a subgroup and its complement. To guarantee it satisfying the criteria of generality, the absolute difference is additionally weighted by its *Entropy*, by using which exceptional model mining prefers a subgroup whose size is similar to its complement. Another heuristic of *significance test* is more accurate and straight forward in meaning. The significance test is performed on the difference between the correlation coefficient of a subgroup and its complement. However, an issue is that the significance is tested on sample correlations. *Fisher Transformation* is a method which approximates a sample-based correlation to a correlation of the population [15, 8]. It is defined as $z' = \frac{1}{2}ln(\frac{1+r}{1-r})$, with its standard error equals to $\frac{1}{\sqrt{m-3}}$, where m indicates the size of a sample set. With approximated correlations, *p-values* can be defined as:

$$z* = \frac{z' - \bar{z'}}{\sqrt{\frac{1}{n-3} + \frac{1}{\bar{n}-3}}} \tag{11}$$

where z' and $\bar{z'}$ are calculated by Fisher Transformation, and respectively on a subgroup and its complement. [8] additionally takes *1* minus the *p-value*, consequently, a higher value of the final result indicates that it is more interesting.

Regression Model Regression Model is constructed on one dependent variable and one or more independent variables. It on one hand analyzes if specific relationships exist among a set of data, and on the other hand induces models for predicting. A linear regression model is normally expressed as: $y_i = a + bx_i + e_i$, where b indicates the slope of the model and a indicates its intercept. The function describes how y is changed when changing the values of x. To identify how exceptional a model is, hypothesis test can also be proposed to test how significant the slope of a subgroup's model is different from the slope of the same model of its complement. The test statistic is defined as follows:

$$t' = \frac{\hat{b}_G - \hat{b}_{\bar{G}}}{\sqrt{s_G^2 + s_{\bar{G}}^2}} \tag{12}$$

where $\hat{b}_G$ and $\hat{b}_{\bar{G}}$ respectively represent estimated slopes of a subgroup and its complement. And S represents the variance of $\hat{b}$.

Classification Model If a target contains a discrete class attribute, exceptional model mining can construct a classification model. *Decision Table Majority* (*DTM*)

is a simple classifier by calculating the relative frequencies of supports for each possible condition which is represented by a combination of attributes. If a condition is not occurred, then its frequency is calculated on the whole dataset (the relative size of the success applicants with respect to the population). An example is as follows.

Table 2: A Decision Table on Mortgage (from [8] P.8)

	Married=no	Married=yes
Age=low	0.25(20)	0.61(0)
Age=medium	0.4(15)	0.686(35)
Age=high	0.733(15)	1.0(15)

Table 2 is derived from a database of mortgage on 100 applicants. It indicates the possibilities of success for applicants of different mortgage conditions. Two attributes of *Married* and *Age* are contained in the table. Each possible attribute-values combination is filled with its relative frequencies of success. Because no applicant of low age is married, its frequency of 0.61 is calculated based on the whole database. Numbers in brackets indicate supports of each condition. This decision table concludes that *If Married=no and Age=low or medium, the output is negative, otherwise the applicant is predicated as successful*. *BDeu score* is the quality measure which estimates the performance of a classifier. It is defined as follows:

$$\prod_{x_1,...,x_k} \frac{\Gamma(\alpha/q)}{\Gamma(\alpha/q + n(x_1, ..., x_k))} \prod_{y} \frac{\Gamma(\alpha/qr + n(x_1, ..., x_k, y))}{\Gamma(\alpha/qr)} \tag{13}$$

where α indicates the equivalent sample size, q refers to the number of conditions, r is the number of different values of the output and $n(x_1, ..., x_k, y)$ indicates the number of examples with a specific condition and class label y.

Alternatively, another heuristic function of *Hellinger* estimates the distance between two distributions. It is defined as:

$$\varphi Hel(p) = \sum_{x_1,...,x_k} \sum_{y} \left(\sqrt{\hat{P}_G(y|x_1, ..., x_k)} - \sqrt{\hat{P}_{\bar{G}}(y|x_1, ..., x_k)} \right)^2 \tag{14}$$

where $P(y|x_1, ..., x_k)$ indicates the possibility of y when it is under a specific condition of $(x_1, ..., x_k)$.

2.2.2 An Approach to Exceptional Model Mining

EMDM [10] is a new approach to exceptional model mining, which on one hand maximizes models' exceptionality and on the other hand minimizes the subgroup's description. To measure the exceptionality of a subgroup, two information-theoretic measures are proposed. One is based on *Kullback-Leibler divergence* (information gain) and the other measure is on *Krimp*. On the aspect of description minimization, EMDM considers it as a binary-classification problem.

The Algorithm of EMDM EMDM focuses on two optimizations that it on one hand maximizes exceptionality in the model space and on the other hand minimizes

description complexity in the description space. EMDM is applied within an iteration procedure, during which each of both tasks is applied in turn. As shown in

Algorithm 3 The EMDM Algorithm (from [10] P.265)

Input: A database D, candidate subgroups C and exceptionality threshold ϵ
Output: A subset of all subgroups satisfying the EMM Problem.
$S \leftarrow \emptyset$
for all $G \in C$ **do**
  while G changes **do**
    $G \leftarrow$ ExceptionMaximisation(G)
    $G \leftarrow$ DescriptionMinimisation(G)
  end while
  if $exceptionality(G) \geq \epsilon$ **then**
    $S \leftarrow S \cup \{G\}$
  end if
end for
return S

Algorithm 3, the Pseudo code of EMDM starts from a set of candidate subgroups. For each subgroup, EMDM continually performs optimizations on both tasks until the subgroup cannot be further optimized. The algorithm finally exports subgroups, the models of which have exceptionality beyond the user's defined threshold.

2.3 Summary

As it is described in the section, subgroup discovery is a typical form of rule learning and has been widely used in data mining and knowledge discovery. It is a halfway between classification and association rule learning, but has a different goal. Subgroup discovery derives rules indicating interesting partitions of a dataset where the observed distribution is different from the general population. Such descriptive rules are simple and easily understood. They interpret exceptional but useful information, by using which data analyst makes management decisions. Besides, experiments proposed in [4] indicate that discovered subgroups can optimize the performances of traditional classification algorithm both in accuracy and efficiency.

Different techniques have been proposed to solve problems of subgroup discovery, but most of the current researches only concern of database of binary-class and single-target. The algorithm *CN2-MSD* is designed to handle subgroup discovery on multi-class context. It is implemented by modifying the heuristic of *CN2-SD*. Exceptional model mining provides a framework of multi-target subgroup discovery. With different kinds of targets, it constructs different models for deriving exceptionality. By using the idea of exceptional model mining, the algorithm of *CN2-MSD* is possible to be adapted to multi-target subgroup discovery. And the task of *CN2-MSD* adaption is also the main job of this project.

3 Project Development

3.1 Introduction

6 methods are designed, in order to carry out multi-target subgroup discovery. They are implemented through adapting *CN2-MSD* using ideas like exceptional model mining. This section firstly specifies the problems met in the project development. It then introduces the details of the methods and their relevant background. Designs of the project are finally described, which introduce each functional component independently.

3.2 Problem Specification

Two main problems are met in the project that how to design effective methods of multi-target subgroup discovery and how to expand the framework of *CN2-MSD* to enable that it can be applied for multi-target contexts. As for the first problem, 4 new methods are designed and totally 6 methods are implemented in the project. Details of the methods are described in the next subsection. The second problem is specified by declaring the properties of an appropriate framework.

A task of subgroup discovery can be conducted once its framework is defined. The framework appropriate for single target problems consists of four properties and is defined in the following way: Only one single target is defined for one dataset. The target denotes instances' corresponding categories; A subgroup is represented by a rule which on one hand describes the common characteristics of the covered instances, and on the other hand implies their most promising categories as well as the corresponding class distributions; A fixed number of most promising hypotheses are iteratively expanded by beam search which enables sufficient candidate subgroups can be evaluated and at the same time enhances the computation efficiency; A heuristic is used to measure each candidate's quality, and the best rule is the one of the highest score. In most cases, a heuristic is strictly under the definition of subgroup discovery that finds subsets whose distributions are different from the population with respect to the defined target.

The framework appropriate for multi-target context is defined as follows: multiple targets are defined in one dataset, and the type of targets are either in numerical or nominal. As for single target problem, the distribution's deviation between a subgroup and the population is examined by heuristics like *MWRAcc* proposed in *CN2-MSD*. However, the deviation appeared in multi-target problems should be defined as that the subgroups' distributions should be different from the population with respect to all of the targets as well as their internal interactions. And the significance computation is consequently different and need to be computed by concerning multiple targets; Subgroups are still described in the form of rules but with different style that the rules generated should intuitively represent the corresponding categories of the covered instances on different targets; Multi-target contexts bring an issue that heuristics proposed for multi-class subgroup discovery may not be applicable. In order to define the heuristic, two main approaches can be applied: The first approach considers a multi-target problem as a problem with single-target through somewhat techniques and then directly apply a single-target heuristic. The other approach di-

rectly applies heuristics on multi-target contexts, which considers all of the targets when qualifies a candidate. No matter what approaches are carried out, the dependencies between targets should always be concerned. Hence the main problem is to detect the dependencies and take them into account when doing subgroup discovery. With dependencies are detected, the first approach can selectively combines part of targets with significant dependency (like the method of *AP*). And as for the second approach, the heuristic should directly describe the internal dependencies (methods of *CM* and *IM*). Because rules generated from both single and multi target subgroup discovery have same body style, the search strategy of beam search is retained.

3.3 Methods

This section introduces the methods implemented in the project. The methods of *ABW* and *PBW* are from [16]. They are relatively simple but easily understood and implemented. Besides both methods bring a deep feeling about the contexts of multi-target subgroup discovery. 4 new methods are designed and proposed in the project. *AP* is incorporated with a dependency detection procedure and using both *ABW* and *PBW*, in order to avoid their main drawbacks: the ignorance of targets' internal dependencies of *ABW* and target values' sparsity of *PBW*. A same procedure is used in [17], where an algorithm is introduced for solving multi-label classification problems. The idea of exceptional model mining expands the traditional framework of subgroup discovery enable that framework is applicable for multi-target tasks. It is primarily defined in [8], where a correlation model is proposed with a clearly defined heuristic. However, the model is only appropriate for data with two targets. As for data with multiple targets, a new pair-wised correlation model is used in the project with the definition of a corresponding heuristic (the method of *CM*). The project also uses the complete independency model when applying the framework of exceptional model mining (the method of *IM*). The model is appeared in [18] but without defined heuristic for multi-target subgroup discovery, and so the project defines a corresponding heuristic using *Probability Density Function (Pdf)*. The last method incorporates a cluster mechanism using kernel k-means before the subgroup discovery is done, cluster is also used in [19] but in a complete different way.

Two main procedures are consisted in *CN2-MSD*: A top-down procedure which generates a rule list using weighted covering algorithm, and a bottom-up procedure which each time creates one best rule using beam search [4]. The adaption methods designed in the project update the mechanisms occurred in the bottom-up procedure. Different methods evaluate candidate subgroups in their different ways, where three of the methods are based on the heuristics proposed in *CN2-MSD*, and two of them apply the idea of exceptional model mining using correlation and complete independency models. The last method is based on both *CN2-MSD* and exceptional model mining, that it first constructs a cluster model on targets and then detects candidates' unusualness using the heuristics proposed in *CN2-MSD*.

3.3.1 *MWRAcc* Based Methods

MWRAcc qualifies the tradeoff between generality and unusualness, that an algorithm using *MWRAcc* aims at maximizing the unusualness and at the same time enable

the generality to be as big as possible. Three of the methods have applied such a heuristic but in different ways. One of the method repeatedly evaluates the quality of a candidate on each target and then statistics an overall result. The second method firstly combines targets to be a single one and then apply the heuristic for quality evaluation. The final method detects internal dependencies between each pair of targets, and combines partitions of these targets correspondingly in the same way of the second method. It then qualifies the subgroup using the same computation of the first method to get an overall result. As for these three methods, all of the other multi-class heuristics proposed in *CN2-MSD* can be applied in an exactly same way.

The Average-based Method (*ABW*) Given a candidate subgroup (for example $Att_1 \leq V_1$ and $Att_2 \neq V_2$), the bottom-up procedure tests its quality on each target and then calculates the average level. And the subgroup is qualified by heuristics like *MWRAcc* or *Chi-Squared* proposed in *CN2-MSD*. After each execution, the best rule is found and with the highest score over all of the possible complexes. Consider targets of two nominal attributes $E_1+, ..., +En = E$ and $F_1+, ..., +Fm = F = E$, where E is the size of a population and X_i indicates the i^{th} discrete value of attribute X. There is also a subgroup b which has distributions on both attributes: $e_1+, ..., +e_n = e$ and $f_1+, ..., +f_m = f = e$. Recall *MWRAcc(b;E)* qualifies subgroup b on a given target of E. When more targets are considered, the qualification can simply averages results of *MWRAcc* on each attribute. The average-based qualification measure on subgroup b is defined as: $MWRAcc(b) = (MWRAcc(b;E) + MWRAcc(b;F))/2$ [16]. An example of the computation is described by figure 2.

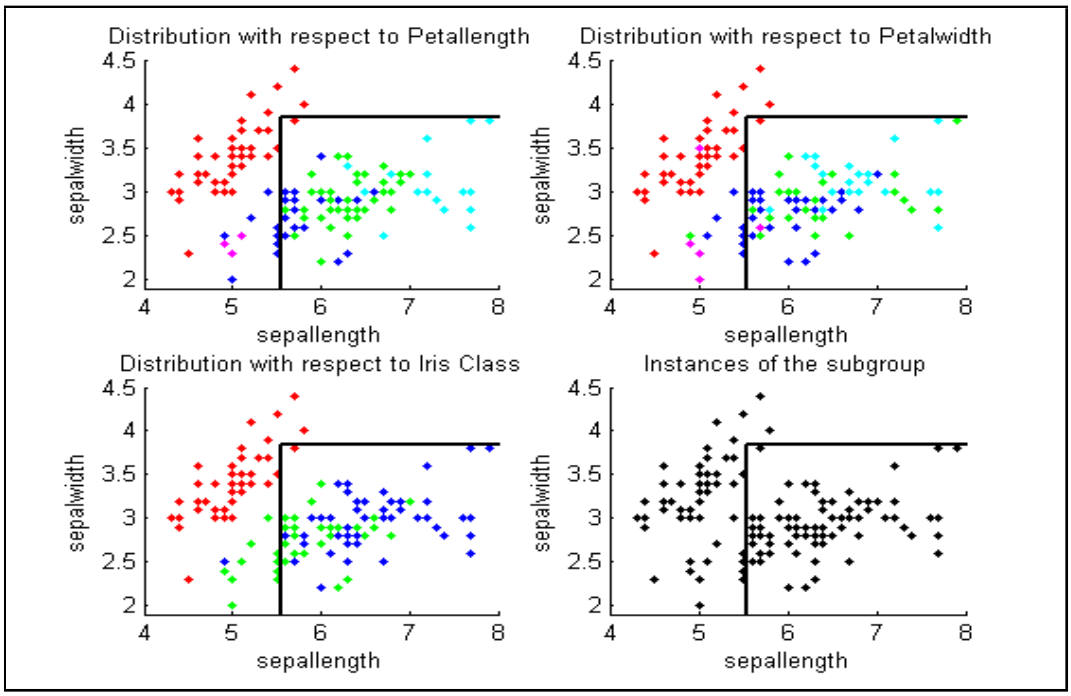

Figure 2: A Description of the Method *ABW*. Instances covered by the subgroup are plotted in each figure, where different figure is regarding with a different target, and the final one denotes an overall result.

Iris consists of 5 attributes with the last one defines instances' classes. In order to carry out multi-target subgroup discovery, the last two attributes *Petallength* and *Petalwidth* as well as the class attribute are defined as the targets. And *Sepallength* and *Sepalwidth* are respectively taken as normal attributes and represented as the X and Y axes. As for the first 3 figure, the three targets are respectively taken as the class with same normal attributes. And so each figure represents the distribution of the instances regarding with a specific target. *ABW* derives a subgroup {*Sepallength* > *5.55* and *Sepalwidth* < *3.85*} which is distinguished out by the rectangle boundary. As shown in the figures, the subgroup has different distributions when concerning with different targets. And the average level of the results denotes the subgroup's quality.

The Product-based Method (*PBW*) Average-based *MWRAcc* simply assumes there is no interactions between targets. However, it is not realistic in real cases. In order to take targets' associations into account, attributes of E and F can be considered as two orthogonal dimensions of a three-way contingency table. As shown in table 3, the left part lists distributions of a subgroup, the right part is the distributions of the entire database. Their joint distributions are listed in the table.

Table 3: A Contingency Table on Two Attributes (from [16])

e_{11}	...	e_{1m}	e_1	E_{11}	...	E_{1m}	E_1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
e_{n1}	...	e_{nm}	e_n	E_{n1}	...	E_{nm}	E_n
f_1	...	f_m	e	F_1	...	F_m	E

Recall the definition of $MWRAcc$:$MWRAcc = \frac{1}{n}\sum|WRAcc_i(b)| = \frac{1}{n}\sum|\frac{e}{E}(\frac{e_i}{e} - \frac{E_i}{E})| = \frac{1}{nE^2}\sum|e_iE - eE_i|$, according to which, the aggregation of the numbers in table 3 can be defined as: $MWRAcc = \frac{1}{nmE^2}\sum\sum|e_{ij}E - eE_{ij}|$. This is like creating a new single target by calculating *Cartesian Products* of original targets.

The implementation of *PBW* is simply a data recreation. The recreated data contains all of the normal attributes and one target which is combined by all the user defined targets. For example, given the same data *Iris* with the targets defined in the same way of *ABW* and an instance of {<*Sepallength = 5.1*>, <*Sepalwidth = 3.5*>, <*Petallength = 1.4*>, <*Petalwidth = 0.2*>, <*Class = Iris-setosa*>}. The first step is to discretize the continuous targets of *Petallength* and *Petallength* to numbers of categories. It then checks which category the instance belongs to. The attribute discretion may result a new instance like {<*Sepallength = 5.1*>, <*Sepalwidth = 3.5*>, <*Petallength* = $C_{Petallength,2}$>, <*Petalwidth* = $C_{Petalwidth,3}$>, <*Class = Iris-setosa*>}. And finally the target value of the instance is changed to a single one: <*target* = C_2_C_3_*Iris-setosa*>.

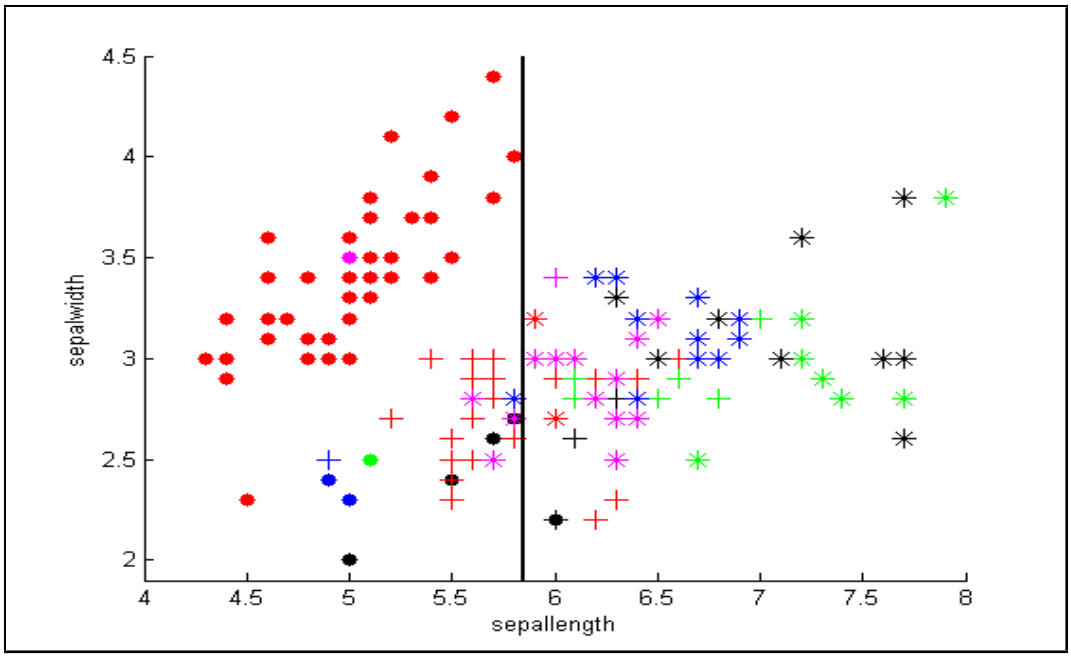

Figure 3: The Subgroup Found by *PBW*. The vertical line distinguish the subgroup from its complement, where the instances covered by the subgroup are distributed in the left part.

The subgroup derived by *PBW* is represented as $\{Sepallength < 5.85\}$ which covers instances on the left part of figure 3. The performance of *PBW* is easily influenced by data sparsity. As shown in figure, 15 distinct values are resulted and the instances with different target values are plotted in different marks and color. Except of the value plotted in red point, all of the other values only cover a small number of instances.

***MWRAcc* based method with dependencies identification (*AP*)** As for traditional classification, each example is assigned with a unique label. It aims at classifying an unseen example into their appropriate category. One of its generalizations is *multi-label classification* where each example is associated with a set of labels. Although multi-label classification is different from multi-target subgroup discovery on their respective purposes, it is still possible to adapt its ideas into subgroup discovery.

There are two main categories of the existing methods of multi-label classification: *Problem Transformation* and *Algorithm Adaptation* [20]. Problem transformation transforms problems of multi-label classification to single-label classification problems. Two kinds of methods are commonly applied for problem transformation: *Binary approaches (BR)* each time generate one classifier for one label; *Labels Power-set (LP) Approaches* take each different set of labels as one single label, and then apply single-label classification methods.

Both *BR* and *LP* have their own disadvantages that *BR* ignores internal dependencies among labels and *LP* is easily influenced by data sparsity. *ChiDep+LPBR* considers both aspects of these approaches [17]. It on one hand detects internal de-

Algorithm 4 The AP Algorithm

Input:A database D with normal attributes $X\{X_1, X_i, X_n\}$ and targets $Y\{Y_1, Y_i, Y_m\}$
Output:A list of subgroups satisfying the problems
$CurrentTarget = \{Y_1, Y_i, Y_m\}$
$BestEvaluation$=$MWRAcc$ of the best subgroup found in D(X and Y)
while $BestEvaluation$ increases **do**
  $InternalDependency = 0$
  $Pair_1 = 0$
  $Pair_2 = 0$
  $InternalDependency = 0$
  for all Pairs of $CurrentTarget(Y_p, Y_q)$ **do**
    $CurInternalDependency$=Internal dependency between Y_p and Y_q
    if $CurInternalDependency > InternalDependency$ **then**
      $InternalDependency = CurInternalDependency$
      $Pair_1 = p$
      $Pair_2 = q$
    end if
  end for
  Combine $\{Y_{Pair_1}, Y_{Pair_2}\}$ to one single target $Y_{1,2}$ using Cartesian product
  $Y' = \{Y\} - \{Y_{Pair_1}, Y_{Pair_2}\} + \{Y_{1,2}\}$
  $CurBestEvaluation$=$MWRAcc$ of the best subgroup found in D'(X and Y')
  if $CurBestEvaluation > BestEvaluation$ **then**
    $BestEvaluation = CurBestEvaluation$
    $Y = Y'$
  else
    break
  end if
end while
$RuleList$=Rules found by ABW on data D (X and Y)
return $RuleList$

pendencies of each pair of labels and on the other hand arranges appropriate classifiers of *BR* and *LP* to different labels or label sets. This is an iterative procedure with accuracy detection occurred in each step, and it aims at fixing a best arrangement or model for these labels. The whole procedure is as follows: It initially assumes complete independency between labels, and arranges *BR* classifiers to each of them. The classification accuracy from the arrangement is taken as the initial value; It then examines the dependencies between each pair of labels using *Chi-Squared Test* [18]. Details of the dependency test will be introduced in the description of the method using *complete independency model*; The pair of labels with highest dependency is taken as a label set. It then arranges *BR* and *LP* classifiers correspondingly; The current classification accuracy is compared with the accuracy from the previous model. It will accept the current arrangement and goes back to the second step, if the accuracy is improved. Or just stop, and uses the last accepted model. Finally, a data will be classified using both *LP* on label sets and *BR* on single labels.

BR is like *ABW* that each of the targets are considered independently when doing

multi-target subgroup discovery, and *PL* is like *PBW* which take target sets as a single one using Cartesian Product. It enables to applying a same procedure of dependency detection to problems of subgroup discovery. And details the adaption method *AP* is described as algorithm 4. Instead of using the classification accuracy, the method of *AP* each time compares the current *MWRAcc* with the same criteria result from the previous model, in order to fix the arrangement which can find the most unusual rules.

3.3.2 Exceptional Model Mining Based Methods

Exceptional model mining constructs an appropriate model on targets of a data and finds subsets whose fitted models are significantly different from the same model of the whole population. With the framework of *CN2-MSD*, such methods are implemented in the following way: Candidate subgroups are iteratively generated by beam search; Given the candidate, a model can be constructed on its targets; The quality of the subgroup corresponding with the model is then compared with the quality of the population. As it is described before, the quality can be correlation coefficient of a correlation model or classification accuracy of a classification model; A big difference between both qualities denotes a big deviation between the subgroup and the population. The project applies a correlation model and complete independency model, and both of them are described in the following content.

Correlation Model (*CM*) A correlation model appropriate for two targets context has been introduced in the section of background knowledge. However, the number of targets can be much bigger. As a result, it should be adapted to enable that subgroup discovery can be done when the number of targets is uncertain. As for more than two targets, the project constructs the correlation model in a pair-wised way that repeatedly constructs correlation models on each pair of targets. And the corresponding quality is computed in the way that firstly evaluates a candidate subgroup on each pair of targets using the original correlation model, and then take the average level as the final result. The heuristic is defined as the equation 15.

$$z^* = \sum_{i=1}^{|Y|} \sum_{j=i+1}^{|Y|} \left| \frac{z'_{i,j} - \overline{z'_{i,j}}}{\sqrt{\frac{1}{n-3} + \frac{1}{\bar{n}-3}}} \right| \Big/ \frac{|Y| \times (|Y| - 1)}{2} \tag{15}$$

where $z'_{i,j}$ and $\overline{z'_{i,j}}$ are calculated by Fisher Transformation from the correlation coefficient $r_{i,j}$ between target i and j ($z' = \frac{1}{2} ln(\frac{1+r_{i,j}}{1-r_{i,j}})$). They respectively denote the coefficients of the subgroup and its complement. However, as for the implementation, $\overline{z'_{i,j}}$ is directly computed from the whole population rather than the complement to speed up the computation. The heuristic based on the correlation model is defined as $z^* \times Entropy$, where the multiplication of *Entropy* aims at finding a subgroup with its size similar to its complement.

Complete Independency Model (*IM*) Interesting partitions of a data may have different dependencies among attributes, according to which a subgroup is qualified by measuring how different its internal dependencies are from the dependencies of

the entire population. The complete independency model assumes that all of the attributes are distributed independently. It then tests if the hypothesis can be rejected.

The hypothesis is tested using Pearson goodness-of-fit statistic [18] which carries out a comparison between both the observed and the expected frequencies. A high score denotes a significant dependent relation, and a low score means the dependency is not existed or not significant. [21] accesses dependencies by assuming each combination of attributes as a first order rule, in the form of ($H \leftarrow B_1 \vee B_2$). And such a rule is expressed in a multi-way contingency table (Table 4).

Table 4: Three-way Contingency Table for $H \leftarrow B_1 \vee B_2$ (from [21] P.74)

	$B_1/6$	$\bar{B}_1/14$	B_1	$\bar{B}_1$	
H	3	3	0	3	9
$\bar{H}$	3	4	0	4	11
	$B_2/13$		$\bar{B}_2/7$		20

As stated in table 4, the observed frequencies of each combination of attribute values are listed. By counting differences between the observed and expected frequencies, one can derive dependency relationships among these three attributes. The expected frequency of such a combination is calculated in the following way that product all of the marginal of the combined attribute-values and then divide by the square of the size of the population. For example $\mu\bar{H}B_1B_2 = \frac{11\times6\times13}{20^2}$.

Alternatively, the calculation of expected frequencies can also be conducted as "*initialize them all to 1, and then proportionally change them to fit each of the one-way marginal in return*" (from [21] P.74). The complete procedure is as follows: Add the first row to 9, and then each element of the first row is multiplied by 2.25; Fit the row of $\bar{H}$ to *11* according to the same way, which results each element of row $\bar{H}$ equaling to 2.75. Fit B_1, $\bar{B}_1$, B_2 and $\bar{B}_2$ in turn according to a same way. Then a final table of expected frequencies is derived. The statistic of a three way contingency table is defined as equation 16.

$$X^2 = \sum_i \sum_j \sum_k \frac{(\mu_{i,j,k} - F_{i,j,k})^2}{\mu_{i,j,k}} \tag{16}$$

Where $\mu_{i,j,k}$ and $F_{i,j,k}$ respectively denote the expected and observed frequency of a combination of $\{V_H = i, V_{B_1} = j \text{ and } V_{B_2} = k\}$. A critical value denotes whether X^2 is significant, and it is determined by the degree of freedoms K: $K = \prod |Y_i| + \sum |Y_i| - m + 1$, where m is the number of targets. The comparison between X^2 and $Chi - Squared$ Distribution denotes how significant the targets are dependent with each other. A heuristic directly measures the differences between X^2 of a candidate subgroup and the population: $log\frac{X^2}{\bar{X}^2} \times Entropy$.

However, as the candidate and the population may have different degrees of freedom, the deviation will be not accurate. The *Probability Density Function (Pdf)* measures the probability that a variable can occur at a specific point. As for this problem, it can be applied to denote how the corresponding probability densities are different between a candidate and the population. The *Pdf* of *Chi-Squared* distribu-

tion is defined as equation 17 according to [22].

$$f(x;k) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-k/2} (x > 0) \tag{17}$$

Where k denotes the degrees of freedom and $\Gamma(k/2)$ denotes the Gamma function. The heuristic of *Pdf* is defined as $(f(\sqrt{X^2}; k) - f(\sqrt{\bar{X}^2}; \bar{k})) \times Entropy$, where X^2 and $\bar{X}^2$ are respectively computed on the candidate subgroup and the population, k and $\bar{k}$ are their degrees of freedoms.

***MWRAcc* with targets clustering (*CW*)** The heuristic applied in *MR-SD* [19] firstly hierarchically clusters data instances by concerning their target attributes, and then obtain candidate subgroups by traversal of the inferred clustering tree. Each candidate subgroup is then scored by detecting if it is separable from other training instances, and the detection procedure is done on normal attributes. As it is described, the algorithm has been evaluated on real data derived meaningful rules. However, it seems that the unusualness and generality of the subgroups are not guaranteed. Compared with the methods implemented in this project, *MR-SD* should be much faster, since it only consider promising candidates. It may bring an issue that some significant rules may be out of consideration.

The method of *CW* is inspired from *MR-SD* that it firstly considers multiple targets as a single one by clustering the data instances through concerning with their targets. As a result, each instance belongs to one specific group, and this is like single-target subgroup discovery that each data is assigned with a specific target value. With the group label as the single target, the multi-target problem can be simply solved by *CN2-MSD*. The algorithm of *K-means* is firstly used to cluster the instances, and it is described as the algorithm 5.

Algorithm 5 K-means from [23]

Input: A database D with normal attributes $X\{X_1, X_i, X_n\}$ and targets $Y\{Y_1, Y_i, Y_m\}$, and the parameter K which determines the number of groups to be clustered
Output: K groups of instances
Randomly choose K instances, $\{Centers\}$=targets of K instances
while $\{Centers\}$ changes **do**
  for all Instance D_i of $D(Y_i$=targets of $D_i)$ **do**
    for all Center μ_j of $\{Centers\}(0 < j < k)$ **do**
      Compute the distance $Dis_{i,j}$ between Y_i and μ_j
    end for
    Find the minimum distance $MinDis_{i,x}$, and assign group label x to instance i
  end for
  for all Center μ_j of $\{Centers\}(0 < j < k)$ **do**
    $\mu_j = \frac{1}{\sum_{i:j(i)=j}} \sum_{i:j(i)=j} Y_i$
  end for
end while

However, the algorithm has a main drawback that it cannot be applied to non-linear problem. As a result, it is not applicable for most of the real life data. *Kernel*

K-means is enhanced by K-means through replacing the distance function by using an appropriate kernel function [24]. And both kernel functions of *RBF* (equation 18) and *Polynominal* (equation 18) are implemented in this project.

$$k_{RBF}(x_i, x_j) = exp\left(\frac{-|x_i - x_j|^2}{2\sigma^2}\right) \tag{18}$$

$$k_{polynomial,d}(x_i, x_j) = (x_i' \times x_j + 1)^d \tag{19}$$

As for K-means, the difference between an instance and a center is directly computed using euclidean distance: $|x_i - \mu_k|^2$. And the distance function is changed to $k(x_i, x_i) + \alpha_k' K \alpha_k - 2 \sum \alpha_k(j) k(x_j, x_i)$ when corresponding kernel functions are applied [23]. And as for the function, k denotes the applied kernel function, α is the parameter to be estimated where $\mu_k = \sum x_i \alpha_k(i) = X' \alpha_k$. And The whole procedure of the algorithm is illustrated as algorithm 6.

Algorithm 6 Kernel K-means from [23]

Input:A database D with normal attributes $X\{X_1, X_i, X_n\}$ and targets $Y\{Y_1, Y_i, Y_m\}$, and the parameter K which determines the number of groups to be clustered
Output:K groups of instances
Randomly initialize $\{\alpha_k\}$, each α is a vector with length equals to the data size. All of the sites of α are 0, except one site equals to 1
while $\{\alpha_k\}$ changes **do**
  for all Instance D_i of D(Y_i=targets of D_i) **do**
    for all Center μ_j of $\{Centers\}(0 < j < k)$ **do**
      Compute the distance $Dis_{i,j}$ between Y_i and μ_j using the kernel distance function: $k(x_i, x_i) + \alpha_k' K \alpha_k - 2 \sum_{j=1}^{n} \alpha_k(j) k(x_j, x_i)$
    end for
    Find the minimum distance $MinDis_{i,x}$, and assign group label x to instance i
  end for
  for all α_j of $\{\alpha_k\}$ **do**
    Assign all of the sites of α_j to 0
    for all Y_i of Y **do**
      if if Y_i belongs to group j **then**
        assign the i^{th} site of α_j to 1
      end if
    end for
    Normalize α_j by dividing all of its elements by the number of 1s it contains
  end for
end while

Both K-means and its kernel version clusters the targets of a data into k groups, namely $G\{V_{G=G_1}, V_{G=G_i}, V_{G=G_k}\}$. Each group label is taken as a target value, and the original data D with normal attributes $X\{X_1, X_i, X_n\}$ and targets $Y\{Y_1, Y_i, Y_m\}$ is replaced by a new one with X and G. Hence the multi-target problem is transformed to a single target task and can be solved by *CN2-MSD*.

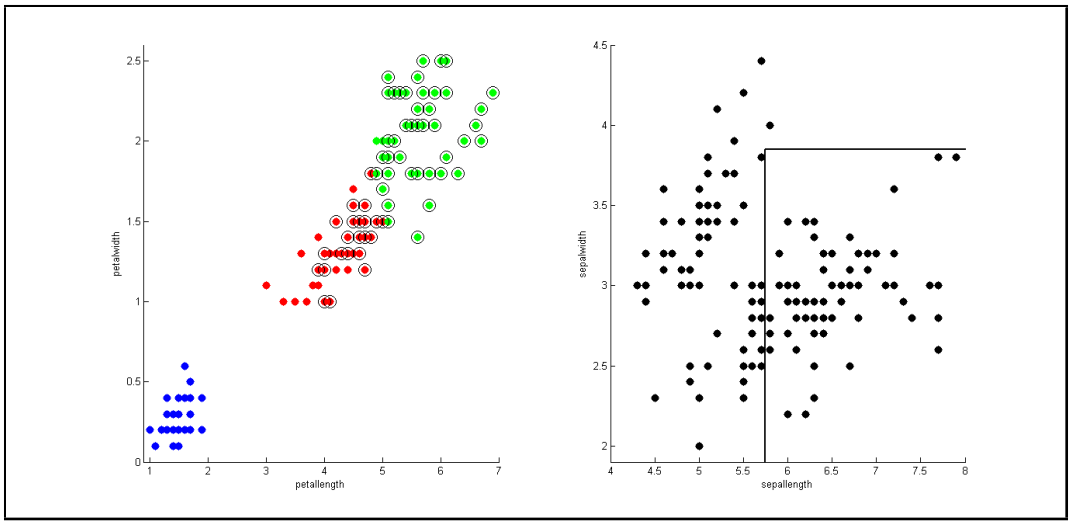

Figure 4: The Subgroup Found by *CW*. The left figure represents the grouped instances regarding with targets, and the right one show instances' distributions using normal attributes. As for the first figure, the instances are clustered into 3 groups which are shown in different color. Three targets are concerned when clustering the instances while only a 2-D figure is shown, and so a red point is appeared in the location surrounded with green points.

As shown in figure 4, data *Iris* is clustered into 3 groups by concerning attributes of *Petallength* and *Petalwidth* as well as the class attribute. With the group labels as target values, a subgroup is found with representation of {*Sepallength* > *5.75* and *Sepalwidth* < *3.85*}. Instances covered by the subgroup is circled in the first graph and distinguished out by a rectangle boundary in the second graph.

3.4 Project Design

The algorithm of *CN2-MSD* provides a framework dong multi-class single-target subgroup discovery. It is implemented by *java* as a part of *Weka* [4]. The system aims at doing multi-target subgroup discovery is developed in the same environment to enable that the framework of *CN2-MSD* can be applied and expanded. Given a data in appropriate format, the system should firstly extracts its main content using functions provided by *Weka*. It then carries out subgroup discovery using a user defined method, where all of the methods introduced in the last subsection has been implemented. The output should on one hand represents the subgroups and on the other hand evaluates them on appropriate criteria.

CN2-MSD consists of four components which work independently with each other. They are connected through taking the output of a prior component as the input of the current one. As the system is implemented in *java*, each component is represented as a *class*. The bottom most class *CN2term* defines the structure representing a selector of a rule, and a selector is in the form of {*Attribute_Name Relation Attribute_Value*} like {$age \leq 34$} or {*Educational level=PHD*}. And the top most

class *CN2* extracts all of the information defined by users, like the methods' relevant parameters and intended data. The main work is done by both components of *CN2rule* and *MakeCN2ruleList*, each of which executes one main procedure of *CN2-MSD*. A bottom-up procedure use beam search to generate a best rule each time; and a top-down procedure which repeatedly executes the beam search until a rule set is found [6]. As for this system, a class of *CN2Evaluation* is implemented to evaluate the rule list. The work of program adaption is mainly occurred in the class of *CN2rule*. It on one hand enhances the class to be applicable for multi-target datasets and on the other hand implements the referred methods. Other components are also adapted but on specific aspects. In the following content, functions of each component as well as their main work flow is described.

3.4.1 The Component of CN2term

Each subgroup is represented as a rule appeared in the form of $Head \leftarrow Body$. Given a data, the *body* implies its corresponding *head* and consequently the whole rule. Hence a rule is simply represented by its body in this system, until it is put out. *CN2* represents a rule as a conjunction of selectors, and a rule can be fixed as long as the list of selectors is determined. A selector defines a pair of attribute-value as well as their relation like $>$, $<$, $=$, $\neq$. Hence a selector is defined as a *struct* with four elements, namely *m_attIndex*, *m_equal*, *m_value* and *m_dValue*, where *m_attIndex*, *m_equal* and *m_value* are all in type of integer and respectively represent indexes of attribute, relation and the attribute values. When numerical attribute is met, *m_value* is replaced with *m_dValue* which denotes the corresponding attribute value. Given the data $Iris$, an object of *CN2term* with *CN2term.m_attIndex*$= 0$, *CN2term.m_equal=0* and *CN2term.m_dValue=2.0* represents the selector of $\{sepal\text{-}length > 2.0\}$.

3.4.2 The Component of CN2

CN2 is the main entry of the system, that it extracts all of the information required for doing multi-target subgroup discovery. Firstly, the data is analyzed in the component with its format checked. In order to apply *Weka* provided functions, all of the processed data should be formatted according to the standard limited by *ARFF*. The file name is declared in the form of *@relation FileName* and appears on top of the file contents. The name declaration should be followed by the attributes' declarations which are listed in turn. And the declarations are appeared in the form of *@attribute AttributeName Specification*, where specification is the parameter to be determined by attribute's type. The parameter is replaced with a key word of *numerical* if the attribute values are in numerical. And a nominal attribute uses a pair of curly brackets which is with the list of possible attribute values inside. Finally, the list of instances is listed as vectors and followed by the tag of *@data*.

Secondly, user defined parameters are extracted and transmitted to the main components. *CN2-MSD* defines different parameters for specific purposes: As it is described before, different heuristics are defined in *CN2-MSD* according to different theories, namely *one-vs-one WRAcc*, *weighted one-vs-one WRAcc*, *one-vs-rest WRAcc*, *weighted one-vs-rest WRAcc*, *mutual information*, *Chi-Squared* and *Gini-*

Split. The user determines which of the heuristics is chosen; The covering algorithm provided by *CN2* causes a basin that with numbers of rules induced, the new generated rules may become not as interesting as before. The reason is that with new rules are derived, all of the covered examples are removed and all of the remaining examples are not sufficient enough to find some interesting pattern. As a result, *CN2-MSD* provides mechanisms of weighting schema, and more details of these schemas have been introduced in the introduction of the algorithm. The appropriate weighting schema like *Covering Algorithm*, *Additive Weights* and *Multiplicative Weights* should be determined by the user for different purposes. In most cases, the choice of weighting schema is determined by users' interests of the size of rule list; The significance of a rule denotes its unusualness from the population and is computed by *likelihood ratio static* [6]. The result is then compared with the *chi-distribution* with a specific significant level namely 0.05 and 0.1. In order to control rules unusualness, users have to choose a suitable significant level; A beam search iteratively expands a fixed number of complexes as to reduce the memory cost. And the number is determined by the user defined beam width. As to this system, more parameters are added to fit the functional requirements of multi-target subgroup discovery. The adaption methods which have been introduced in the previous section should be chosen by the user. As it is defined that a subgroup behaves unusually with respect to user interested properties, so the indexes of the targets are also chosen by the user.

3.4.3 The Component of CN2rule

As for this system, the main function of *CN2rule* is similar with the bottom-up procedure of *CN2* that it iteratively constructs a set of complexes, and all of them are evaluated according to the heuristic determined by the corresponding adaption methods. Iteratively, a best complex is taken as the candidate and all of the other complexes which are examined to be under specific conditions are ranked according to their qualities. A fixed number of most promising complexes will be expanded and evaluated in the next iteration. The procedure will be not finished until all of the complexes have been qualified, and it will finally generate one best rule from the given data.

Different pre-process are carried out for different adaption methods. As for the *PBW*, the provided data should be recreated as a data with single target. And details of the recreation are described in the section of the method introduction. A pre-process of *AP* has also recreates the data but in a different way, and it also exports the new targets' indexes after the recreation is done. As for the correlation model based method, the pre-process records the correlation coefficients between each pair of targets on the whole population. Similarly, the method based on complete independency model also computes the dependency among the targets of the population. The method of *CW* clusters the instances into k groups with respect to their target attributes.

CN2rule constructs new complexes by expanding the current one, and it evaluates each new complex according to two conditions. In the first condition, it examines if the distributions of the new complex is same with the distributions of the current one on all of the targets. The satisfaction of the condition leads to the ignorance of the new complex and the current complex is returned. The second condition examines

if the new complex is significantly different from the current one using *likelihood ratio static*. If the second condition is satisfied, the new complex cannot be the best candidate even it is evaluated to be with better quality by specific heuristic. As for this system which concerns multiple targets, the significant test will be different for different purposes. If the users aim at finding subgroups with respect to partitions of the targets then a voting mechanism is proposed that distributions of the new complex should be significantly different on at least a half of the targets. And if the users emphasize on all of the targets, then the significance of the distributions on all targets are tested. However, in the second cases, rule may not be found when sufficient targets are concerned.

3.4.4 The Component of MakeCN2ruleList

The class of *MakeCN2ruleList* iteratively executes *CN2rule*. It examines all of the rules and collects a non-redundant rule list. A main job of *MakeCN2ruleList* is to implement the weighting schemas to ensure that new generated rules are still interesting after numbers of rules have been found. Another main work of *MakeCN2ruleList* is defined in *CN2-MSD* but not occurred in this system that it doing classification according to the rule list derived. As for this system concerns with multi-target contexts, the primary aim is to design effective subgroup discovery methods, and its performances are evaluated on *Descriptive Measures* rather than *Predictive Measures*, the classification function is ignored in this system.

3.4.5 The Component of CN2Evaluation

CN2Evaluation is not working for finding subgroups but for evaluating the performances of a final rule list. The outcome of subgroup discovery is a set of rules. By evaluating rules' qualities, one can conclude how good the discovered subgroups are. In most cases of evaluation, qualities of each rule are evaluated first. The qualities of the rule set are then averaged to indicate the qualities of the whole result. Hence the performances of a heuristic can be assessed. Descriptive measures are to evaluate subgroups' criteria like complexity and significance, which denotes rules qualities. Predictive measures assess the prediction performance of the rule set. The ultimate goal of subgroup discovery is to find interesting patterns rather than optimize the prediction performance [7, 5, 4]. The predictive measure is not considered in the implementation of the system.

Size of Models The number of rules and the number of terms in rules are main indicators of evaluating the qualities of discovered subgroups [7]. The more rules are resulted, the more information is acquired. However, a huge list of redundant rules is not interesting. As it is described before, subgroup is found regarding with users' interests, so when users are focusing on specific information, a small subgroup is better. Besides, a simple a rule with fewer terms is easily understood. Hence, a more compact set of representative rules can intuitively show useful information is preferred.

Coverage Coverage [5] of a rule is the proportion of individuals covered by this rule with respect to the population. And the commonly referred coverage is the average level of all of the rules' coverage. A subgroup's coverage reflects its generality.

Support The support of a rule is defined as the size of individuals which are covered by the rule and labeled with a certain label, and it is often represented as a proportion relative to the population. The procedure of calculating a subgroup's support is similar with the coverage computation that firstly computes supports of each rule and then takes the average level as the final result. The comparison between support and coverage can indicate how interesting the discovered subgroup is.

Significance The significance of each rule reflects how unlikely a rule occurred by chance. It is defined as the difference between two distributions: One is the classes' distribution determined by the rule. And the other one reflects the original population's distribution. To calculate the significance, CN2 uses likelihood ratio statistic [6]. The average significance of the subgroups is defined as: $Sig = \frac{1}{nR} \sum Sig(R_i)$, where nR refers the number of rules, and R_i is the i^{th} rule. *WRAcc* is another way of evaluating a subgroup's significance, besides it also reflects its coverage. *WRAcc* is an appropriate method of qualifying subgroup's interestingness and generality.

The changes of models It is believed that the internal relations among targets between a subgroup and the population are different from each other. And so the system also evaluates the relation changes on two aspects, namely correlation and dependency.

3.4.6 Overview

The 5 functional components work independently but with some links among them. Their main internal connection is the data flow that results come from one component may be taken as the another component's inputs. Besides, components may call functions or data structures defined by the others. For this system, class *CN2term* defines a structure to represent a selector which is the most basic unit composing a rule. As a result, a rule can be represented and analyzed in class *CN2rule*. The class receives input from *MakeCN2ruleList* like user defined parameters and the intended data, it then gives the best rule as the feedback. The rule is then examined in *MakeCN2ruleList*, and collected into a rule list if it is qualified. The rule list is evaluated by *CN2Evaluation* with the quality as a final output. *CN2* is the main interface between users and the system, that it transmits data and parameters to *MakeCN2ruleList* which then transfer these information to *CN2rule* for further processing. 5 components as well as their internal links are described by the *UML* chart shown in figure 5.

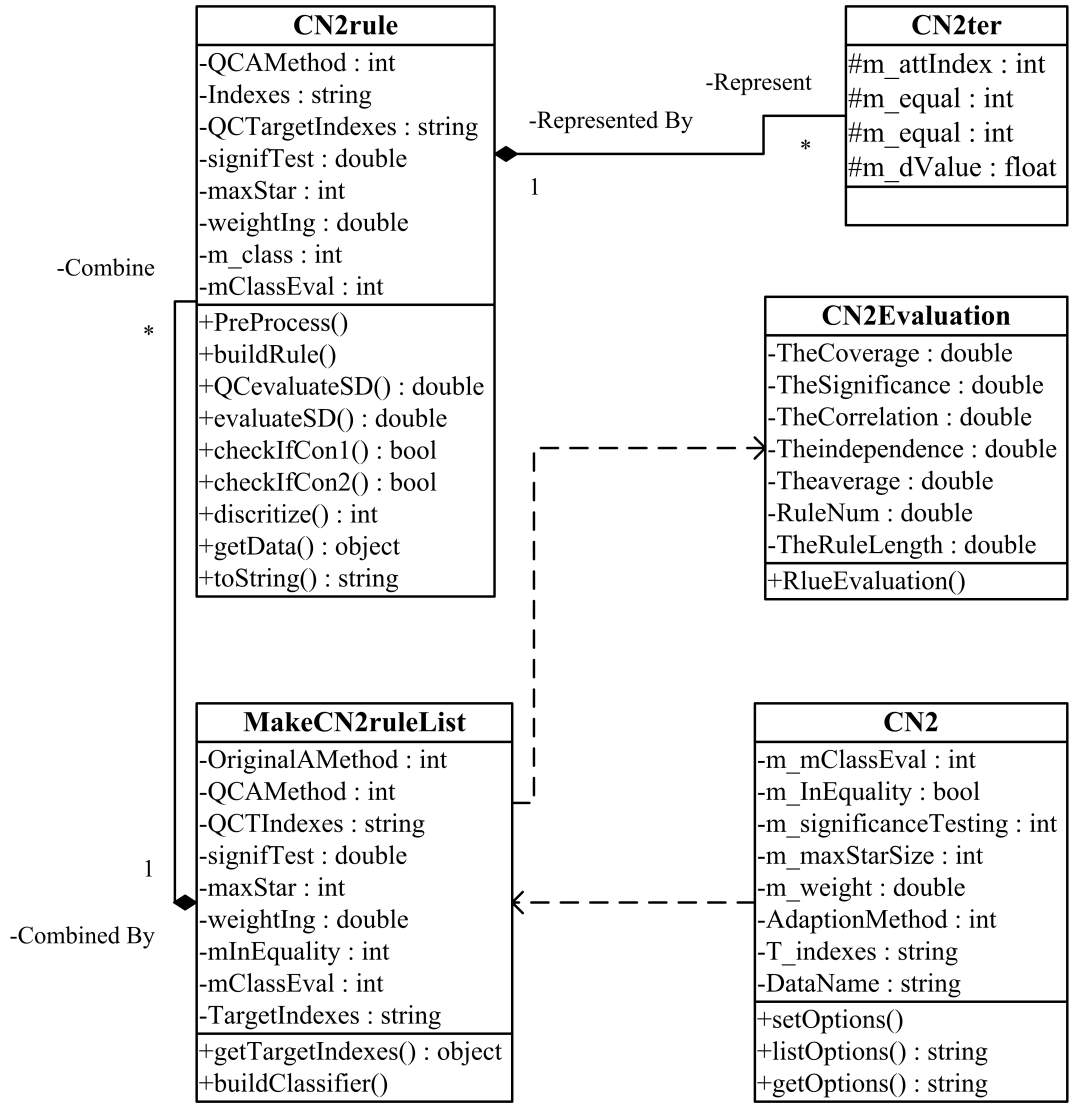

Figure 5: The UML chart of the system.

3.5 Summary

This section firstly specifies the problems met in the project through declaring the definition of an appropriate framework. 6 methods as well as their relevant theories are then introduced, where 3 of the them are based on the heuristic proposed in *CN2-MSD*, and 2 of the methods use the idea of exceptional model mining. The method *CW* is based on both of them that it firstly constructs a cluster model on the data and then uses the heuristic proposed in *CN2-MSD* to detect candidates' unusualness. The implementation of the project is finally described, where functional components as well as their internal links are refered. Among the 3 *MWRAcc* based methods, *ABW* and *PBW* are relatively easier for understanding and implementing but with their respective disadvantages. And the 4 new methods seem more appropriate for handling multi-target subgroup discovery problems. As a result, in the next section experiments are designed to evaluate these methods.

4 Experimental Results

Subgroup discovery emphasizes on the tradeoff between generality and unusualness that it maximizes subgroups' unusualness with their generality kept on a reasonable level. And so the derived subgroup is guaranteed to be with high significance and relatively big coverage; Besides multi-target subgroup discovery generates rules uncovering interesting patterns with respect to defined targets. As for real cases, a subgroup with high significance may represent meaningful knowledge; Subgroups' qualities are closely related to criteria results like rules' complexity and significance. And these criteria measure a subgroup on different aspects, according to what the proposed methods can be compared critically. Three experiments are carried out to evaluate the performances of the proposed methods on different point of views.

4.1 Experiments on Synthetic Data

Synthetic datasets are generated artificially with a subgroup included intentionally, where the subgroup is distinguishable from the population both the targets and on normal attributes. And such data is designed to outline methods' abilities of finding subgroups. Similar synthetic data is primarily used in [19], where the multi-target subgroup discovery algorithm *MR-SD* is introduced and evaluated.

4.1.1 Synthetic Data

4 datasets are used in the experiment, and distinguished between each other by attribute numbers and types. Each data consists of 200 instances, where the first *40* instances are taken as the included subgroup. In order to avoid the influence from the specific instances sequences when doing subgroup discovery, instances of the data are randomly ranked before the data is to be used. 5 binary targets are contained in each data, and the target values are initialized differently between the instances of the subgroup and its complement. As for the first *40* instances (instances covered by the subgroup), their target values are randomly assigned to be *1* and *0* with a probability distribution of (*0.9,0.1*), and for the rest instances (instances of the subgroup's complement), the probability to be *1* or *0* are both *0.5*.

Two *DB* datasets are generated with their normal attributes's numbers equal to *2* (DB_2) and *10* (DB_{10}). And the data creation of *DB* is following the same way appeared in [19]. The 2 binary attributes X_1 and X_2 can either be *1* and *0*. The attributes of the first *40* instances satisfy that $X_1 = 1$ and $X_2 = 1$. As for the other instances, X_1 and X_2 are valued randomly from *1* or *0*. However cases of $X_1 = 1$ and $X_2 = 1$ are not allowed to appear in these instances. Consequently, the subgroup is distinguishable by considering both of X_1 and X_2, but not each one of them. As for DB_{10}, the first two attributes are arranged in a same way, but with the rest attributes randomly set to *1* or *0* with equal probabilities.

Normal attributes' values of *DC* datasets are distributed according to *N(0,1)*. As for DC_2, their attributes are arranged in the following way to enable that the subgroup is distinguishable from its complement by considering both of the first two normal attributes. Decrementally rank the first *100* attribute values of X_1, and then incrementally rank its rest values of X_1. Incrementally rank the first and rest *100* val-

ues of X_2 in turn. As a result, the first *40* instances which belong to the subgroup can be represented by using both X_1 and X_2, in the form of $X_1 > V_{X_1}$ and $X_2 < V_{X_2}$. And it also guarantees that a unique attribute cannot accurately represent the subgroup. DC_{10}'s first two attributes are same with DC_2 and the other 8 attributes' values are sampled from $N(0, 1)$.

4.1.2 Results and Discussion

All of the instances of each dataset are randomly ranked before the dataset is used. As for DB_2, there are four possible subgroups represented by the combinations of Att_1 and Att_2, namely $\{Att_1{=}1$ and $Att_2{=}1\}$, $\{Att_1{=}1$ and $Att_2{=}0\}$, $\{Att_1{=}0$ and $Att_2{=}1\}$ and $\{Att_1{=}0$ and $Att_2{=}0\}$, where only the complex of $\{Att_1{=}1$ and $Att_2{=}1\}$ can accurately represent the intended subgroup. Because each target variable is defined to be binary, the representation of $Att{=}1$ equals to $Att{\neq}0$, so does the pair of $Att{=}0$ and $Att{\neq}1$. As for DB_2 and DB_{10}, the experiment examines if the intended subgroup of $\{Att_1{=}1$ and $Att_2{=}1\}$ can be found. And the results of different methods are listed in table 5.

Table 5: Rules Generated from DB_2 and DB_{10}

Methods	DB_2	DB_{10}
ABW	$Att_1{=}1$ and $Att_2{\neq}0$	$Att_1{\neq}0$ and $Att_2{\neq}0$
PBW	$Att_1{\neq}0$	$Att_5{=}0$
CM	$Att_1{\neq}0$	$Att_2{=}0$
IM	$Att_1{\neq}0$	$Att_2{=}0$
AP	$Att_1{=}1$ and $Att_2{\neq}0$	$Att_1{\neq}0$ and $Att_2{\neq}0$
CW	$Att_1{\neq}0$ and $Att_2{\neq}0$	$Att_1{=}1$ and $Att_2{\neq}0$

Table 5 lists the best rules generated from each of the proposed methods on both DB_2 and DB_{10}. Because the *head* can be implied by using the corresponding *body* if the data is given, the rules are directly represented by its *body* part. As shown in the table, except of the method *PBW*, all of the *MWRAcc* based methods can accurately find the intended subgroup both on DB_2 and DB_{10}. Sparsity caused by *PBW* limits its ability of detecting unusualness. As a result its coverage is increased as for the tradeoff. This is the reason why *PBW* is always generating shorter rules. The reason that the exceptional model mining based methods are failed in finding the subgroup out is that each of the targets is generated independently, and there is not much internal relations among them. This is also the reason why *ABW* works well.

Table 6: Rules Generated from DC_2

Methods	Best rule found	Intended instances found	Coverage
ABW	$Att_1{>}$ 0.21 and $Att_2{<}$ 0.03	40	45
PBW	$Att_2{<}$ 0.02 and $Att_1{>}$-0.96	40	84
CM	$Att_2{<}$-0.09 and $Att_1{>}$-0.48	39	63
IM	$Att_1{>}$ 0.43	37	70
AP	$Att_1{>}$ 0.10 and $Att_2{<}$ 0.07	40	51
CW	$Att_1{>}$ 0.10 and $Att_2{<}$ 0.02	40	45

Table 6 lists the best rules generated by each methods on DC_2. It also lists the coverage as well as the number of intended instances (the instances that primarily located at the first *40* places) covered by the rule. The intended subgroup has been pre-defined to be distinguishable from the whole population, so it is the subset behaves most differently from the whole population. Hence when compare two methods which find same numbers of intended instances, the smaller coverage denotes a better performance. Except the two exceptional model mining based methods, all of the other methods find the complete intended subgroup. However, *PBW* results a much bigger coverage caused by target values's sparsity. Although the rules found by *ABW* and *CW* are different, they have exactly same performances.

Table 7: Rules Generated from DC_{10}

Methods	Best rule found	Intended instances found	Coverage
ABW	Att_2 <-0.05 and Att_1 > 0.02 and Att_4 < 1.62 and Att_{10}>-2.35 and Att_8 < 1.95	40	52
PBW	Att_2 < 0.05 and Att_4 < 0.96 and Att_1 >-1.78 and Att_{10}< 1.48 and Att_7 < 2.39 and Att_8 < 2.27 and	37	85
CM	Att_1 < 0.00 and Att_6 >-1.90 and Att_{10}>-1.85 and Att_8 >-2.31 and Att_9 < 1.17	0	78
IM	Att_2 >-0.22 and Att_1 < 1.08 and Att_8 < 1.71	0	91
AP	Att_2 <-0.05 and Att_1 > 0.02 and Att_4 < 1.60 and Att_{10}>-2.35 and Att_8 < 1.95	40	52
CW	Att_2 <-0.29 and Att_1 >-0.02	40	40

Rules derived from data DC_{10} are listed in table 7. And they are compared with the rules recorded in table 6 to analyze how the methods are influenced when the number of attributes increase. As it can be seen, the exceptional model mining based methods are failed in finding the intended instances. The reason is that the internal relations among the targets of the subgroup is not much different from the population. All of the other methods find the complete subgroup, except that *PBW* misses three instances even though it results a relatively bigger coverage. It is obvious that except of the method *CW*, all of the other methods are easily influenced by the increased number of attributes. And most importantly, the rule generated from *CW* accurately covers the intended subgroup, that all of the *40* intended instances are found without other instances are covered.

CW can accurately find the subgroup out no matter what kind of data is provided. Because there is not much dependency between these targets, exceptional model mining works not very well when compare with the other methods. And this is also the reason why *AP* works similar with *ABW* on these data.

4.2 Experiments on Data from European Social Survey (ESS)

The academically-driven social survey is designed to illustrate the interactions between European citizens' institutions and their attitudes, beliefs and behaviors [25]. The survey is carried out through considering different mentality aspects and their relations, so does the experiment. The main content covered by the survey is regarding with 8 aspects like Media and social trust, Politics, Human values and Socio-demographics. This experiment aims at generating interesting rules in real cases, so rules' significance is taken as the most important criteria to evaluate the quality of a subgroup. Three data are created with same normal attributes but different targets which describe humans' habits and mentalities on different aspects. *CW* performs best on two of the data, and *ABW* finds the best rule from the rest one. Because many targets are concerned when doing subgroup discovery, it is not essential to guarantee deviation between the subgroup and the population with respect to all of the targets. Only some targets with highest significance are concerned when conclude the rules' meaning.

These datasets are about the population of the UK collected in 2006. For each data, a subgroup is represented by partitions of the normal attributes as well as the corresponding attribute values. As it is described, the experiment aims at generating rules denoting relations between human's institutions and factors like mentality states. The normal attributes are collected from variables of Socio-demographics as well as the variables of age and genders. A detail description of the attributes is listed in table 8, and each attribute is followed by its description and variable type.

Table 8: Description of the Normal Attributes of ESS Data (Edited from [25])

Attribute	Type	Attribute description
gndr	Nominal	gender
age	Numerical	age
edlvgb	Nominal	Highest educational level(UK)
emplrel	Nominal	Employment relation (employee, self-employed or family business)
wrkctra	Nominal	Employment contract type (unlimited or limited)
wkdcorga	Nominal	In what degree the citizen allowed to organize his own daily work
iorgact	Nominal	In what degree the citizen is allowed to influence policy decisions
wkhct	Numerical	Work hours per week in main job (overtime excluded)
wkhtot	Numerical	Work hours per week in main job (overtime included)
hincsrca	Nominal	Household income source
maritala	Nominal	Legal marital status
fxltph	Nominal	Dose the citizen has a fixed-line telephone in accommodation
mbltph	Nominal	Dose the citizen has a mobile telephone
inttph	Nominal	Dose the citizen use the Internet for telephone calls at home
regiongb	Nominal	Region

4.2.1 The ESS Data Regarding with Human Values

The data consists of 1779 instances. It takes Human values as the targets which describe citizens' feelings of the society. 21 targets are included in the data with their descriptions like: the feeling of importance to try new things in life and the

feeling of importance to help and care for other people. Descriptions of these targets are listed in table 9.

Table 9: Description of the targets of ESS Data (Human Values) Edited from [25]

Target	Target description
ipcrtiv	How important to be creative
imprich	How important you think of rich
ipeqopt	How important you think people should be treated equally
ipshabt	How important that others admire you
impsafe	How important do you think to live in a safe place
impdiff	How important to start new things
ipfrule	How important to follow the rules
ipudrst	How important you think that different people can be understood
ipmodst	How important to be fashion but not attract attention
ipgdtim	How important to have fun
impfree	How important to make decision for yourself
iphlppl	How important that people can help each other
ipsuces	How important to let others recognize your success
ipstrgv	How important that the safety is guranteed
ipadvnt	How important to enjoy an exiting life
ipbhprp	How important to behave properly
iprspot	How important that people can respect each other
iplylfr	How important to be loyal when touch with friends
impenv	How important to protect the environment
imptrad	How important to keep to be traditional
impfun	How important to enjoy yourself

Each target is in fact a questionnaire with citizens' answers as the corresponding values. And these target values denote how much the citizen is like the person appeared in each question. Possible values of each target could be *very much like me, like me, somewhat like me, a little like me, not like me, not like me at all*. The data is evaluated by all of the implemented methods, and the best rules generated by these methods are listed in table 10. And again, each rule is represented by its *body* part, because the *head* of the rules contains redundant information limited in meaning.

Table 10: Rules Generated from ESS Data (Human Values)

Methods	Best rule
ABW	age<51.5,wkhct<61.0,wkhtot<82.5,hincsrca≠3,maritala≠6
PBW	age>47.5
CM	age>44.5,iorgact≠8,regiongb≠12,edlvgb≠5,hincsrca≠3,maritala≠7
IM	age<48.5,regiongb≠7,wkhtot<66.5,hincsrca≠5,maritala≠4
AP	age<51.5,wkhct<61.0,wkhtot<82.5,hincsrca≠3,maritala≠6
CW	age>51.5,edlvgb≠5

As shown in table 10, attribute *age* is the most commonly occurred attribute in these rules. It can be concluded that the attribute is the most dominant factor when derive unusual patterns on human values. *PBW* generates the most general rule which is caused by target values sparsity when combine multiple targets to a

single one. As for this data with 21 targets, each of which contains 6 target values, the data recreation of *PBW* results *1772* distinct values for the final single target. With the data size equals to 1779, it seems like each instance is assigned with a distinct class from the other instances. Hence the final result has limited meaning.

The significance of each rule with respect to all of the targets are computed using likelihood static ratio and compared with the *Chi-Squared* distribution table with significant level equals to *0.05*. For each target, the degree of freedom is computed by values' number minus *1*. Because the number of values of all of the targets are 6, the critical values of all of the targets are *15.1* with degree of freedom *5*. The corresponding results are listed in table 11. And the coverage of each rule is followed by the name of the methods.

As shown in table 11, for each target, the highest significance which is also beyond the critical value of *15.1* is shown in bold. As it can be seen, the rule of *CW* has highest significance on most of the targets. And four targets with highest significance are $ipadvnt$, $imptrad$, $impenv$ and $ipsuces$. In order to analyze how different the subgroup is from the population with respect to these four targets. The distributions of both the subgroup and the population with respect to these targets are respectively shown in table 12. And the difference between two distributions are also computed.

Table 11: Targets' Significance of ESS Data(Human Values)

Target	ABW56.0%	PBW49.6%	CM46.9%	IM45.0%	AP56.0%	CW42.9%
ipcrtiv	12.99	10.04	11.81	***15.41***	12.99	11.65
imprich	34.39	41.41	34.62	30.17	34.39	***47.43***
ipeqopt	3.480	2.080	2.840	6.870	3.480	4.380
ipshabt	37.00	***40.77***	34.65	36.19	37.00	38.80
impsafe	8.290	4.910	5.290	4.700	8.290	11.90
impdiff	36.69	40.18	***42.73***	38.83	36.69	41.34
ipfrule	46.69	37.34	34.84	27.98	46.69	***54.25***
ipudrst	1.980	0.760	1.320	1.740	1.980	1.340
ipmodst	16.25	11.57	12.61	10.97	16.25	***18.17***
ipgdtim	51.54	58.49	48.77	48.17	51.54	***61.69***
impfree	5.570	3.340	2.230	5.040	5.570	4.660
iphlppl	1.850	2.190	2.200	2.110	1.850	1.960
ipsuces	46.83	60.29	64.42	52.49	46.83	***64.46***
ipstrgv	32.36	30.41	20.75	31.41	32.36	***46.53***
ipadvnt	82.80	92.79	92.08	75.03	82.80	***109.0***
ipbhprp	47.29	47.48	38.39	40.17	47.29	***60.58***
iprspot	7.200	8.620	9.260	9.720	7.200	9.790
iplylfr	4.610	1.720	1.760	4.900	4.610	3.750
impenv	53.14	52.01	51.58	41.73	53.14	***71.00***
imptrad	82.46	82.47	70.72	88.97	82.46	***104.3***
impfun	30.13	35.91	***38.03***	27.24	30.13	35.77

As shown in table 12, the difference of distributions between the subgroup and the population denotes how citizens of the subgroup are different from the population on one specific human value. As it can be seen, there is a target value of *somewhat like me* (the 2^{nd} target value) or *a little like me* (the 3^{rd} target value) splits each difference into two parts. And the target values denote the degree of agreements on

the opinion appeared in each question. For example the citizens of the subgroup agree less of the opinion denoted by *ipadvnt* but agrees more on *imptrad*. And the rule can be described as that: citizens older than 51.5 but does not have a PhD/DPhil or equivalent educational level agrees more that it is not important to have an exciting life but following life with traditions and customs, besides they are more like to care for nature but not care much to be successful.

Table 12: Distributions of the Most Significant Targets (Human Values)

Target	Subgroup/Population	Distribution
ipadvnt	Subgroup	{0.0210 0.0986 0.1327 0.1919 0.3955 0.1603}
	Population	{0.0607 0.1574 0.1832 0.2069 0.2962 0.0956}
	Subgroup-Population	{***-0.0397 -0.0588 -0.0505 -0.0150*** 0.0993 0.0648}
imptrad	Subgroup	{0.2746 0.3916 0.1380 0.1038 0.0775 0.0145}
	Population	{0.1793 0.3153 0.1877 0.1422 0.1366 0.0388}
	Subgroup-Population	{***0.0953 0.0762*** -0.0498 -0.0384 -0.0591 -0.0243}
impenv	Subgroup	{0.4008 0.4179 0.1104 0.0539 0.0145 0.0026}
	Population	{0.2979 0.3991 0.1686 0.0956 0.0349 0.0039}
	Subgroup-Population	{***0.1029 0.0188*** -0.0583 -0.0417 -0.0204 -0.0013}
ipsuces	Subgroup	{0.0289 0.1669 0.1866 0.2326 0.3180 0.0670}
	Population	{0.0658 0.2187 0.2209 0.2187 0.2310 0.0450}
	Subgroup-Population	{***-0.0369 -0.0518 -0.0343*** 0.0139 0.0870 0.0220}

4.2.2 The ESS Data Regarding with Media and Social Trust

The data contains 1807 instances with 15 normal attributes which are exactly same with the former data and 8 targets describing humans commonly uses of media and social trust. Descriptions of these targets are listed in table 13. All of the targets are in type of nominal and respectively with 7 or 11 target values, where values of the first 5 targets denote time spent and the values of the last 3 targets denote the degree that a citizen believes in one specific opinion. And a big value means more time spent or a high level of confidence.

Table 13: Description of the targets of ESS Data (Media and Social Trust) Edited from [25]

Target	Target description
tvtot	The total time of TV watching every weekday
tvpol	The time spend on politics when watching TV
rdtot	The total time of listening radio every weekday
nwsptot	The total time of reading newspaper every weekday
netuse	How often use the internet
ppltrst	How does the citizen consider most people can be trusted
pplfair	How does the citizen consider most people take advantage of him
pplhlp	How does the citizen consider most people try to be helpful

Different rules are derived from the data when use different methods, and these rules are listed in table 14 for comparison. The most general rule is found by *PBW* with a relatively bigger coverage. The rule generated by *IM* overfits to the data, and consequently results a smallest coverage.

Table 14: Rules Generated from ESS Data (Media and Social Trust)

Method	Cov	Best rule
ABW	49.5%	age<51.5,edlvgb≠0,wkhct<69.0,hincsrca≠4
PBW	49.8%	age>48.5
CM	50.6%	age<54.5,wkhtot<61.0,edlvgb≠0,wkhct>4.5, hincsrca≠8
IM	33.0%	hincsrca=1,age>26.5,iorgact≠0,emplrel≠3, wkhtot<59.5,regiongb≠8,fxltph≠2,maritala≠4
AP	36.0%	age<51.5,edlvgb≠0,wkdcorga≠1,wkhtot<97.5, hincsrca≠4,inttph≠5
CW	45.1%	age>52.5,iorgact≠6,wkdcorga≠9,wkhct>3.5

The rules are analyzed in the same way with the former data, that it first computes the significance of each rule on all of the 8 targets, and the significance are shown in table 15. The critical values denote if targets are significant are listed in the last column, and the highest significance of each target are shown in bold. The target with highest significance is *netuse* which denotes how often the citizen uses the internet. It contains 8 values describing different internet accessing frequencies from '*No access at home or work*' to '*every day*'. The highest significance on *netuse* is resulted by *ABW*, where the corresponding rule has been shown in table 14. Distributions of the subgroup and the population on target *netuse* are respectively {*0.2557*, *0.1345*, *0.0293*, *0.0138*, *0.0415*, *0.0515*, *0.1605*, *0.3132*} and {*0.0839*, *0.0660*, *0.0280*, *0.0145*, *0.0537*, *0.0570*, *0.2181*, *0.4787*}; And their difference is computed as {*0.1718*, *0.0685*, *0.0014*, *-0.0007*, *-0.0122*, *-0.0056*, *-0.0576*, *-0.1655*}. As it can be seen, the difference of both distributions is decrementally ranked with the target value increases. And the rule can be concluded that citizens who is younger than 51.5, own any kind of educational qualifications, work less than 69 hours per week and does not take pensions as the main income normally spend more time using internet.

Table 15: Targets' Significance of ESS Data(Media and Social Trust)

Target	ABW	PBW	CM	IM	AP	CW	*Threshold*
tvtot	70.87	45.41	73.62	41.27	***75.91***	62.46	*18.48*
tvpol	57.38	62.24	49.78	19.83	63.63	***78.29***	*18.48*
rdtot	23.31	15.84	26.68	31.21	***39.04***	21.50	*18.48*
nwsptot	40.55	30.22	35.60	9.370	***42.39***	40.82	*18.48*
netuse	***265.8***	180.7	249.4	131.0	253.8	208.1	*18.48*
ppltrst	30.52	17.94	34.75	***40.36***	19.63	23.71	*23.21*
pplfair	36.96	27.01	***40.80***	37.47	25.94	32.82	*23.21*
pplhlp	44.28	31.87	45.80	30.57	26.53	***44.76***	*23.21*

4.2.3 The ESS Data Regarding with Politics

1382 instances are consisted in the data, each of which is with 43 attributes where 28 of them are defined as targets describing humans' feeling of politics like: the level of interests paid on politics and opinions about immigrants. Same as the processes

occurred in the former data, this ESS data is evaluated by different methods with the corresponding rules listed in table 16.

Table 16: Rules Generated from ESS Data (Politics)

Method	Cov	Best rule
ABW	50.7%	edlvgb!=4,age>17.5,iorgact≠7,wkdcorga≠3,regiongb≠8, emplrel=1,inttph≠5,hincsrca≠2,wkhct>5.5
PBW	49.7%	age<46.5
CM	41.5%	age<50.5,wkdcorga≠5,hincsrca≠5,wkhtot<82.5,maritala≠5, edlvgb≠5,wkhct>12.5,regiongb≠1
IM	40.3%	age<45.5,wkhct<54.5,wkdcorga≠0,maritala≠7,wkhtot<71.0, hincsrca≠7,edlvgb≠0
AP	50.7%	edlvgb≠4,age>17.5,iorgact≠7,wkdcorga≠3,regiongb≠8, emplrel=1,inttph≠5,hincsrca≠2,wkhct>5.5
CW	30.9%	edlvgb=4,age<68.5,regiongb!=12,maritala!=6

Table 17: Targets' Significance of ESS Data(Politics)

Target	ABW	PBW	CM	IM	AP	CW	*Threshold*
polintr	21.49	9.240	3.180	7.500	21.49	***34.61***	*11.34*
polcmpl	26.37	1.430	2.260	3.330	26.37	***35.28***	*13.28*
poldcs	6.020	6.020	5.400	8.620	6.020	9.300	*13.28*
trstprl	24.48	14.10	14.68	25.94	24.48	***52.48***	*23.21*
trstlgl	29.95	3.580	7.630	12.49	29.95	***42.49***	*23.21*
trstplc	11.63	2.540	8.930	10.44	11.63	19.41	*23.21*
trstplt	16.75	6.420	6.330	17.40	16.75	19.14	*23.21*
trstprt	13.17	9.570	4.440	22.56	13.17	13.90	*23.21*
trstep	26.73	31.29	30.89	***42.76***	26.73	32.63	*23.21*
trstun	14.48	5.850	8.220	16.09	14.48	***35.29***	*23.21*
vote	16.71	43.45	19.38	31.02	16.71	5.510	*9.210*
stflife	17.18	10.61	26.83	26.50	17.18	***29.76***	*23.21*
stfeco	21.12	10.98	16.49	24.20	21.12	***32.38***	*23.21*
stfgov	10.29	12.72	12.76	20.02	10.29	12.16	*23.21*
stfdem	22.58	10.08	11.16	18.02	22.58	***26.79***	*23.21*
stfedu	10.60	17.92	13.06	***28.12***	10.60	20.77	*23.21*
stfhlth	18.68	29.28	19.18	32.57	18.68	***43.04***	*23.21*
gincdif	20.25	2.660	1.320	8.110	20.25	***35.96***	*13.28*
freehms	13.00	***34.24***	26.12	29.03	13.00	29.70	*13.28*
prtyban	7.290	***24.84***	11.17	22.88	7.290	16.90	*13.28*
scnsenv	1.910	***13.53***	5.880	12.50	1.910	4.650	*13.28*
euftf	16.63	31.39	19.00	46.03	16.63	***26.50***	*23.21*
imsmetn	24.70	13.51	11.10	15.45	24.70	***44.35***	*11.34*
imdfetn	47.29	21.67	20.88	22.20	47.29	***59.94***	*11.34*
impcntr	42.95	32.06	44.70	31.93	42.95	***55.77***	*11.34*
imbgeco	54.60	16.06	17.25	18.77	54.60	***66.69***	*23.21*
imueclt	74.58	34.17	31.03	33.78	74.58	***96.69***	*23.21*
imwbcnt	58.90	12.82	17.95	21.42	58.90	***86.30***	*23.21*

CW results the smallest coverage and a relatively shorter rule but with highest

significance which is shown in table 17. As described in the first experiment of synthetic data, *CW* can accurately represent a subgroup with higher significance and a relatively shorter rule. The shortest rule is found by *PBW* but with limited meaning, as the sparsity influences unusualness detection. And all of the other rules overfit to the data.

As shown in table 17, each column denotes the significance of a subgroup on one target. And the last column lists their corresponding critical values. It is obvious that *CW* results the highest significance with respect to most of the targets. One main reason is that the coverage resulted by the method is the lowest among all of the them, and a subgroup with smaller size has a bigger chance to be unusual. However, the coverage of 30.9% has been sufficient enough to denote a pattern. The rule generated by *CW* has been described in table 16. And it covers a subset of citizens who are younger than 68.5 with the highest degree of NVQ4/NVQ5 or equivalent but not living in the northern Ireland and not widowed. As for the rule found by *CW*, the last six targets have highest significance. And each of these targets describes humans' degree of agreement on specific point of views on immigrants problem. A table of the distributions of both the subgroup and the population is listed to describe their differences on each of these 6 targets.

Table 18: Distributions of the Most Significant Targets (Politics)

Target	Subgroup/Population	Distribution
imueclt	Subgroup	{0.0141 0.0164 0.0515 0.0632 0.1241 0.1311 0.1452 0.1756 0.1710 0.0515 0.0562}
	Population	{0.0716 0.0478 0.0883 0.1027 0.1346 0.1469 0.1122 0.1339 0.1049 0.0282 0.0289}
	Subgroup-Population	{0.0576 0.0314 0.0368 0.0395 0.0105 0.0157 -0.0330 -0.0418 -0.0660 -0.0233 -0.0273}
imwbcnt	Subgroup	{0.0141 0.0281 0.0609 0.0749 0.1101 0.2365 0.1405 0.1616 0.1030 0.0351 0.0351}
	Population	{0.0774 0.0507 0.1027 0.1027 0.1172 0.2424 0.0977 0.0984 0.0695 0.0232 0.0181}
	Subgroup-Population	{0.0634 0.0225 0.0419 0.0278 0.0072 0.0059 -0.0428 -0.0632 -0.0336 -0.0120 -0.0170}
imbgeco	Subgroup	{0.0304 0.0234 0.0515 0.0867 0.1077 0.1897 0.1171 0.1827 0.1452 0.0445 0.0211}
	Population	{0.0810 0.0507 0.0912 0.1071 0.1172 0.1975 0.1042 0.1194 0.0904 0.0275 0.0137}
	Subgroup-Population	{0.0506 0.0272 0.0396 0.0204 0.0095 0.0078 -0.0129 -0.0633 -0.0548 -0.0170 -0.0073}
imdfetn	Subgroup	{0.1358 0.5316 0.2857 0.0468}
	Population	{0.0803 0.4276 0.3654 0.1266}
	Subgroup-Population	{-0.0555 -0.1040 0.0797 0.0798}
impcntr	Subgroup	{0.1054 0.4941 0.3372 0.0632}
	Population	{0.0716 0.3813 0.3835 0.1635}
	Subgroup-Population	{-0.0338 -0.1128 0.0463 0.1003}
imsmetn	Subgroup	{0.1733 0.5621 0.2389 0.0257}
	Population	{0.1114 0.5014 0.3061 0.0811}
	Subgroup-Population	{0.0619 0.0607 -0.0672 -0.0554}

As shown in table 18, the distribution differences of all of the targets are split into two parts with a clear cut-off point. For example, the cut-off point of the first 3 targets are all the 6^{th} target value. For each of these targets, there exist 11 values denote the different degree of agreements about immigrants problems, where the first value denotes the highest agreement and the last value for the lowest. The 6^{th} value means that the citizen has neutral opinion on one specific problem, so does the cut-off point of the last three targets. And the subgroup can be concluded that these citizens have relatively poorer impressions on immigrants. For example that citizens under the subgroup agree more about the opinion that immigration is not good for the county's economy and cultural life.

The experiment has carried out multi-target subgroup on three data from European social survey. For each data, all of the implemented methods are used, and the best rule is found and analyzed in order to describe its meaning. Because the experiment aims at finding interesting patterns, it emphasizes more on significance, in this experiment the quality of each rule is simply evaluated using significance. However, in the next experiment, criteria like complexity and *WRAcc* are also measured on. *CW* finds the best rules from two of the data, and *ABW* finds the rule from the rest one. *PBW* generates the most generous rules on each data, which is thought as limited in meaning, as the influence of sparsity limits its ability for detecting unusualness.

4.3 Experiments on Multiple Datasets

4.3.1 Description of Datasets

Extensive experiments are carried out to critically evaluate performances of the proposed methods. 14 datasets are used, where 11 of the them are collected from *UCI* repository [26], 2 data namely *Emotions* and *Yeast* are from *Mulan* [27] and the last dataset of *ESSUK* is collected from the project of European Social Survey. A brief description of data's size and attributes arrangement is recorded in table 19.

Table 19: Data Description

Name	Instances	Nominal	Numeric	Targets
Adult	279	6	7	2
Balance Scale	625	2	0	3
Breast Cancer Wisconsin	683	8	0	2
Car Evaluation	1728	5	0	2
Chess	28056	4	0	3
Coil 1999 Competition Data	200	0	15	3
Connect	990	0	10	4
Credit Approval	653	8	6	2
Emotions	593	0	72	6
ESSUK	1752	37	3	3
Flags	194	27	10	2
Iris	150	0	2	3
Teaching Assistant Evaluation	151	3	1	2
Yeast	2417	0	103	14

Data *Adult* used in the experiment consists of a proportion of instances, as the original data contains too many instances to carry out multi-target subgroup discovery. As for the other datasets, data instances with missing attribute values are excluded. Data collected from UCI are primarily designed for classification problems, normally with single label denoting instances' classes. As for multi-target problem, the label of each data is defined as one of its targets, and the other targets are defined by interests. Targets of Mulan data have been pre-defined by providers, as these datasets are designed for problems of multi-label classification. The ESS data aims at deriving rules denoting relations between human institutions and their mentality aspects. There are three targets consisted in the data describing humans' social trusts.

4.3.2 Result Evaluation

Performances of the proposed methods are evaluated by analyzing their corresponding rule lists. And rule list's quality is indicated by 7 criteria results, namely, rule list size, average rule length, average coverage, significance, the difference of targets' internal correlation between the covered instances and the population, the different internal dependency and *WRAcc*. The first two criteria denotes rules' complexity. The criteria of coverage and significance respectively denotes subgroup's generality and unusualness. And the tradeoff between these two aspects is indicated by *WRAcc*. The criteria of correlation and dependency measures in what degree the internal relations of a subset are different from the population. The *10 fold-validated* criteria results are recorded for each data, and the average results over all of the datasets are collected in table 20 and 21. They respectively record the average level of performances for *MWRAcc* and exceptional model mining based Methods.

Table 20: Overall Performances of *MWRAcc* Based Methods

Performance	ABW	PBW	AP	CW
Size	12.6±6.0	***8.5±5.1***	10.8±5.1	11±6.0
length	5.0±3.5	***4.2±3.1***	4.5±3.2	4.7±3.5
Coverage	0.40±0.06	***0.48±0.12***	0.42±0.09	0.41±0.06
Significance	84.69±136.10	65.00±102.06	***94.81±172.04***	87.88±147.20
Correlation	0.11±0.07	0.10±0.07	0.11±0.07	***0.11±0.06***
Independency	1.83±1.04	1.79±0.96	1.84±1.01	***1.86±1.01***
WRAcc	***0.049±0.027***	0.040±0.026	0.046±0.028	***0.049±0.027***

The average performance of each method is recorded in their corresponding column with each row indicating one criteria result. The variances corresponding with the average criteria results are given in forms of $\pm Value$. A bigger variance denotes a larger fluctuation over different data. The best criteria results among these methods are shown in bold.

A more compact set of representative rules intuitively showing useful information is preferred. A shorter rule with higher significance is considered to be more representative. However, users may prefer bigger rule list which represents more interesting patterns, and this is one of the reason why weighting schema is incorporated in *CN2-MSD*.

Among all of the methods using the heuristic of *MWRAcc*, *PBW* generates the smallest rule list with shortest rules and biggest coverage but performs worst on significance and *WRAcc*. The unbalanced performance is mainly caused by target values' sparsity. Data recreation is carried out by *PBW* and leads to excessive target values, especially when big numbers of targets are defined. The extreme case is that each instance is assigned a distinct value after the recreation. Given a data of size N and an expected subgroup which covers n instances, the quality of the subgroup on this extreme case is computed as: $n \times \frac{n}{N} \times (\frac{1}{n} - \frac{1}{N}) + (N - n) \times \frac{n}{N} \times (\frac{1}{N} - \frac{0}{n}) = \frac{2n}{N} - \frac{2n^2}{N^2}$, where *one-vs-rest MWRAcc* is used as the heuristic. The maximum value of the convex function is computed through assigning its derivative function equal to 0: $\frac{2}{N} - \frac{4n}{N^2} = 0$. And the size of the corresponding subgroup with highest score is computed to be $\frac{N}{2}$. This is the reason why the average coverage of *PBW* is close to 50%.

AP balances the performances of *ABW* and *PBW*, that it results more compact rule lists and bigger coverage than *ABW* but bigger significance and *WRAcc* than *PBW*. In fact the biggest significance among all of the methods is occurred on this method, which denotes that rules generated by *AP* are more unusual. However, the followed variance value indicates the fact that the big average result is affected by some specific values in deep level. Hence the assumption that *AP* is best for detecting unusual pattern cannot be concluded. *ABW* and *CW* get the biggest criteria results of *WRAcc* which denotes the tradeoff between generality and unusualness, but *CW* performs slightly better on each of these two criteria. Besides, the methods' performances on correlation and dependency are in a similar level.

Table 21: Overall Performances of Exceptional Model Mining Based Methods

Performance	CM	IM	CW
Size	11.8±6.4	16.5±9.3	***11±6.0***
length	4.0±2.0	***3.5±2.3***	4.7±3.5
Coverage	0.40±0.09	0.38±0.07	***0.41±0.06***
Significance	54.86±75.38	31.94±35.98	***87.88±147.20***
Correlation	***0.19±0.09***	0.09±0.07	0.11±0.06
Independency	1.91±1.14	***2.33±1.53***	1.86±1.01
WRAcc	0.040±0.026	0.034±0.022	***0.049±0.027***

The criteria of correlation and dependency are measured in a same way as the heuristics used in *CM* and *IM*. This is the reason why they respectively perform best on each of the criteria. A shorter rule normally covers a bigger number of instances. However, *IM* generates the shortest rule but with the smallest coverage. It means that the big changes of targets' dependencies are likely caused by some specific attributes.

Normally, smaller coverage leads to a bigger rule list. Each time with a rule derived, all of the covered instances are removed or assigned with a smaller weight. And the more instances retained or the retained instances are with relatively bigger weights, the easier to find a new rule. This is verified by the fact that *IM* results the biggest rule list but with small coverage. Among all of these methods, *CW* performs best on both generality and unusualness which are denoted by criteria of coverage and significance respectively.

The two categories of methods emphasize on different aspects when finding subgroups. Exceptional model mining based methods aim at finding unusual pattern with respect to targets co-occurrences, while the other methods consider more on changes of targets' distributions. *PBW* is the one using *WRAcc* but consider targets' co-occurrences. However, it is easily influenced by value sparsity. *AP* considers both targets' distribution and their co-occurrences and so it performs well on both aspects. Similarly, *CW* firstly clusters instances through considering their targets, where the targets' co-occurrences are consequently accounted. In order to compare *CW* with the other methods, *wins* and *loses* of the method are recorded in table 22.

Table 22: Wins and loses of *CW*

Performance	CW vs ABW Win/Lose	CW vs PBW Win/Lose	CW vs AP Win/Lose	CW vs CM Win/Lose	CW vs IM Win/Lose
Size	***11***/3	4/***10***	8/6	6/8	***11***/3
Length	***13***/1	5/9	8/6	6/8	4/***10***
Coverage	7/7	2/***12***	4/***10***	7/7	9/5
Significance	***10***/4	9/5	9/5	***13***/1	***14***/0
Correlation	7/7	8/6	8/6	0/***14***	***10***/4
Dependency	7/7	9/5	8/6	8/6	2/***12***
WRAcc	9/5	***12***/2	***10***/4	***12***/2	***14***/0

With N datasets, the null-hypothesis assumes that the number of *wins* is distributed according to $N(N/2, \sqrt{N}/2)$. Given the distribution, the *P_value* of a specific number of *wins* can be computed. The critical value of 14 datasets at α=*0.10* is *10* according to [28], and the *wins* and *loses* in table 22 beyond the critical value are shown in bold. Because no *ties* occur when carries out comparisons, the critical value of *10* is identical to all cells. As shown in the table, *CW* performs significantly better than *ABW* on criteria of size, length and significance, which means a rule generated by *CW* is more representative that it represents more unusual pattern with less attributes. *PBW* and *AP* are significantly better to find rules with big coverage. Besides, *PBW* generates smaller rule list which is also observed in table 20. Compare with *CW*, both *CM* and *IM* perform significantly worse than *CW* but better on their respective criteria measuring internal relations, namely correlation and dependency. *CW* performs significantly better than the others on *WRAcc* except *ABW*. This is because *WRAcc* indicates a tradeoff between unusualness and generality and *CW* only performs better on the first one.

Average criteria has limited meaning, as the values cross different datasets. And the *wins* and *loses* only compare a pair of algorithms each time. In order to avoid influences from extreme values, ranks of these methods are recorded independently on each criteria. Evaluation on each data leads to the corresponding rank of the 6 methods, where the best one is ranked as *1* and the worst one is recorded as *6*. The average ranks over all of the datasets regarding with different criteria are recorded in table 23.

Table 23: Average Ranks of the Methods

Methods	Size	Length	Cov	Sig	Cor	Dep	WRAcc
ABW	4.07	4.64	4.07	2.64	3.93	4.14	2.50
PBW	***1.93***	2.71	***2.29***	3.64	4.00	4.07	4.36
CM	3.14	3.71	3.50	5.00	***1.07***	3.93	4.14
IM	4.93	***2.29***	4.36	5.36	4.79	***1.86***	5.21
AP	3.79	4.21	2.86	2.29	3.57	3.43	2.86
CW	3.14	3.43	3.93	***2.07***	3.64	3.57	***1.93***
F_F	5.46	3.69	2.90	17.24	11.02	3.39	10.82

Ranks of each method on different criteria are recorded in their corresponding rows. The best result of each criteria is shown in bold. Because the best method is ranked as *1*, smaller values in table 23 denote better performances. Differences are existed between the ranks and the former tables. As shown in the table, *AP* is not as good as *CW* on significance, and *ABW* no longer performs best on *WRAcc*. Although the average values of *WRAcc* of *ABW* and *CW* are same with each other, *CW* results the highest *WRAcc* on more datasets. The rule list generated by *PBW* is smaller than all of the other methods. Besides, it results the biggest coverage. *IM* generates the shortest rule, and results the biggest criteria value of dependency, so does the method of *CM* which results the biggest correlation.

Friedman test is used to evaluate if the ranks are significant. It is considered to be more appropriate for comparing multiple methods over multiple datasets [4]. The statistic is under the null hypothesis that all of the methods perform equally and so they should have same ranks. It is computed as formula 21.

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{20}$$

Where N and k denote the number of datasets and algorithms. R_j means the average rank of the method j over all of the datasets. Friedman's χ_F^2 is considered to be over conservative [28], and a better statistic which is proposed in [29] is used in this experiment.

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{21}$$

And the statistic is tested independently on each criteria with their F_F recording in the last row of table 23. The statistic is distributed according to F-distribution with its degrees of freedom equal to *k-1* and *(k-1)(N-1)*. As for this experiment with degrees of freedom equals to *5* and *65*, the critical value is between *1.95* ($F_{(0.10,5,60)}$) and *1.90* ($F_{(0.10,5,120)}$) at $P = 0.10$ which are checked from *StatSoft Electronic Statistic Textbook* [30]. It is smaller than all of the ranks recorded in table 23, so the null-hypotheses are rejected and it can be concluded that the methods' ranks are significant.

The statistic is then proceed with Bonferroni-Dunn post-hoc test which is proposed in [31], and against *CW* as the control learner according to the same testing method used in [4]. The performance of a method is considered to be significantly different from the control learner if their difference of average ranks are bigger than

the corresponding critical difference $CD = q_\alpha \sqrt{(k(k+1))/6N}$, where the critical values q_α for two-tailed Bonferroni-Dunn test are shown in table 24.

Table 24: Critical Values for Bonferroni-Dunn post-hoc tests (from [28] P.12)

Methods	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.241	2.394	2.498	2.576	2.638	2.690	2.724	2.773
$q_{0.10}$	1.645	1.960	2.128	2.241	2.326	2.394	2.450	2.498	2.539

With *6* methods consisted in the comparison and $q_\alpha = 0.10$, the critical value is *2.326* and the corresponding critical difference $CD = 2.326\times \sqrt{(6 \times (6+1))/(6 \times 14)} = 1.64$. The graphical illustration of the post-hoc test is shown in figure 6, where different methods are plotted in different color and the critical difference of each criteria is shown vertically.

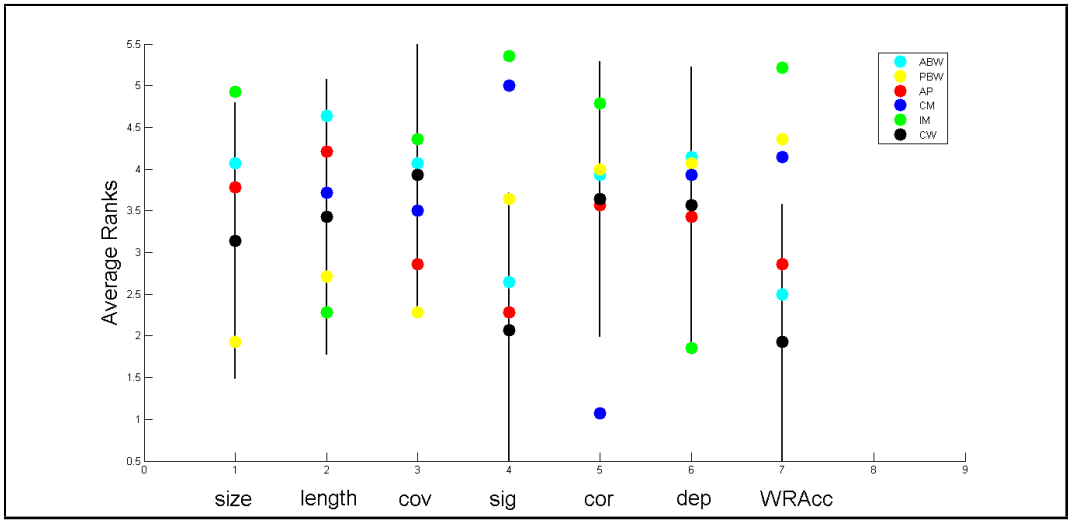

Figure 6: Bonferroni-Dunn post-hoc test against *CW* as the control learner. For each criteria, the performance of a method is considered to be significantly different from *CW*, if it is located at the vertical bar. And the method with worse performance is plotted above *CW* (the black points).

CW performs best on criteria of significance and *WRAcc* which denote its ability of detecting unusual rules. Both of the performances are significantly better than *PBW* and the exceptional model mining based methods. On the other hand, *CM* and *IM* respectively performs best on criteria of correlation and dependency, and significantly better than most of the other methods. The coverage of *CW* is significantly worse than *PBW*, but as it is described that the big coverage of *PBW* is subject to values' sparsity. For most of the criteria, the performance of *AP* is located between *ABW* and *PBW*, except on the criteria of *correlation* and *dependency* that *AP* performs better than both of them but not much significantly. Because the value of *CD* is closely related to the number of datasets, additional statistical information can be acquired when more datasets are evaluated on.

The conclusion from the experiment is that the two categories of methods perform well on different criteria. The exceptional model mining based methods perform well in detecting changes of targets' internal relations, but only on specific aspect. The *WRAcc* based methods are good at detecting unusual patterns regarding with targets' distributions. *PBW* performs bad on both aspects, even though it results the smallest rule list and biggest coverage. *CW* is the best method among all of the *WRAcc* based methods, that it generates most unusual rules (Highest significance) and greatly balances the tradeoff between generality and unusualness (Highest *WRAcc*). Besides the rules generated by *CW* are relatively shorter which means the rules are more representative.

4.4 Summary

Three experiments are carried out to evaluate the respective performances of the implemented methods. The description of each experiment starts from introducing the applied data, and then describing the corresponding results in graphs or tables. In each experiment, different methods are compared by concerning their corresponding results. Besides, the results are discussed to conclude the performances of the methods. In the first experiment, 4 synthetic datasets are created artificially with subgroups intentionally included. It aims at concluding the proposed methods' abilities of finding subgroups. Results from the 4 datasets indicate that the method *CW* can accurately find the subgroups out no matter what kind of dataset is provided. In the second experiment, three ESS data are collected to evaluate if the methods can find meaningful knowledge in real cases. As for the objective is to find interesting patterns, rules' significance is used as the evaluation criteria. *CW* finds the best rules form two of the three data. In order to evaluate methods performances by considering other criteria, empirical experiments are finally carried out and measuring the criteria results on multiple datasets. Besides the performances of different methods are critically evaluated using statistic techniques. The statistic results conclude these methods' performances. For instances, except the method of *PBW* which is influenced by target values' sparsity, the rules generated by other *MWRAcc* based methods are commonly with high significance and *WRAcc*. *CW* is considered as the best method for detecting unusual patterns. Besides, the rules generated by *CW* are relatively shorter. It means the rules from *CW* are more representative. As for the other criteria like rule list's size and coverage, *CW* is not the best but still relatively better than most of the other methods.

5 Conclusion and Future Work

5.1 Conclusion

The main objective of the project is to expand the traditional framework of subgroup discovery using the ideas like exceptional model mining, in order to solve problems with multi-target contexts. Sufficient researches are carried out on basic knowledge of subgroup discovery like the frameworks' definition, the typical heuristics and the advanced algorithms. *CN2-MSD* as well as its relevant algorithms are deeply researched, as the algorithm is applied and adapted in the project. It then researches fields like exceptional model mining and multi-label classification to conclude ideas related to the algorithm adaption. The research brings a deep feeling about the contexts of subgroup discovery and gives an initial understanding of how the corresponding algorithms work, and so how to apply exceptional model mining into these algorithms. As for the implementation, problems met in the project are specified by declaring an appropriate framework's definition, according to which the project is designed. Because the original algorithm of *CN2-MSD* is implemented in *java* using *Weka*, the project is also designed and implemented following the same environment. 4 methods are proposed in the project, where mechanisms of dependency detection and cluster are respectively applied in *AP* and *CW*. Both methods solve the multi-target problem with single-target methods. Besides, the idea of exceptional model mining is applied in both *CM* and *IM* using a correlation or a complete independency model respectively. Totally 6 methods are implemented in the project with their respective performances analyzed by sufficient experiments. And the experimental results denote improvements when comparing with *ABW* and *PBW*.

5.2 Future Work

Method *CW* performs best among the proposed methods. As shown in the experimental results, the rules generated by *CW* are more representative that the rules are with relatively higher significance and shorter length. Recall the mechanism applied in *CW* which firstly clusters the population into groups using kernel k-means, and then detects candidate subgroups' unusualness and generalities using the heuristic of *MWRAcc*. The application of kernel k-means brings some issues: Firstly, an appropriate k which denotes the number of groups is not easily determined, as the determination is deeply subject to the provided data. For instances, a data with a big number of targets is normally assigned with a bigger k. Additionally, both the number of target values and the targets' internal dependencies should be considered when determining the parameter. Secondly, an unappropriate setting of k directly affects the method's performances. A big value may lead to values' sparsity which results the rules be too generous, while a small value results the ignorance of some unusual patterns. The corresponding parameter can be determined by sufficient experiments when given a data, but it seems inefficient. In order to solve the problems, two main aspects should be further researched. Analyze the relation between data structures and the pattern function of kernel k-means, and so define an appropriate equation which maps a specific data to an appropriate k. And then carry out experiments to evaluate the formula's efficiency. The other aspect is to research more cluster al-

gorithms which can on one hand solve non-linear problems and on the other hand determine the number of clusters automatically.

Descriptive measures like rules' complexity and significance are used in the project as the evaluation criteria. The predictive measure is an additional criteria which describes rules' performance on completely different aspects. As for prediction, a normal way is to take the generated rule list as a predictive model and so to assess subgroups' predictive power. For nominal subgroup discovery, the measurement can be directly the classification accuracy which calculates the proportion of correctly classified instances or *ACC* measurement which measures the accuracy on both positive and negative examples. For numerical subgroup discovery, the regression performance is measured by *root mean squared error*. As described in [4], in order to carry out single-label classification, the rules generated by subgroup discovery can be taken as the leaves of a decision tree. The predicative performance of *CN2-MSD* is then reported that the subgroups even perform better than some classification methods (like *J48*) on classification problems. As described in [4], in order to apply the subgroup discovery algorithm for solving problems of classification, a mechanism of probabilistic classification is incorporated into *CN2-MSD*. It assigns a probability distribution of the target to each rule, and then classify an unseen instances by concerning all of the rules which cover it. However, the mechanism is only applicable for single-target subgroup discovery. The second future work is to research on the fields relevant to multi-label classification and design a mechanism for rules with multiple targets. Besides, the targets may appear in different type namely numerical or nominal, and so other aspects like regression may also need to be further researched on.

MWRAcc based methods like *CW* and *AP* apply different heuristics proposed in *CN2-MSD*. All of the heuristics have been applied in the proposed methods but only with *MWRAcc* evaluated. The reason is that *WRAcc* based heuristics result fewer subgroups in average. However, the other heuristics performs relatively better when used for prediction. The third work is to firstly evaluate the respective performances of these heuristics on multi-target subgroup discovery and then measures all of the heuristics' predication power on multi-label classification problems.

As described in the experiments, the two categories of methods perform differently on different criteria, especially on criteria of significance, *WRAcc*, correlation and dependency. The *MWRAcc* based methods perform better on the first two criteria, but not as good as the other methods on criteria which describes the changes of targets' internal relations, and vice versa. The final work is to define a criteria which describes an overall level on both aspects' performances, and then design an algorithm concerning both of them when carrying out multi-target subgroup discovery.

References

[1] P. Flach and N. Lavrac. *Rule Induction*, pages 229–267. Springer-Verlag, January 2003. http://www.cs.bris.ac.uk/Publications/pub_master.jsp?id=1000703.

[2] W. Li, M. Ogihara, S. Parthasarathy, and M.J. Zaki. New algorithms for fast discovery of association rules. In *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 283–286. AAAI Press, 1997.

[3] M. Atzmueller. Subgroup discovery. *Kunstliche Intelligenz*, 19(4):52–53, 2005.

[4] T. Abudawood and P. Flach. Evaluation measures for multi-class subgroup discovery. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*, ECML PKDD '09, pages 35–50, Berlin, Heidelberg, 2009. Springer-Verlag. http://dx.doi.org/10.1007/978-3-642-04180-8_20.

[5] P. Flach, B. Kavsek, N. Lavrac, L. Todorovski, and S. Wrobel. Subgroup discovery with cn2-sd. *Journal of Machine Learning Research*, 5:153–188, 2004.

[6] P. Clark and T. Niblett. The cn2 induction algorithm. *Mach. Learn.*, 3:261–283, March 1989. http://portal.acm.org/citation.cfm?id=637913.637942.

[7] H. Grosskreutz. Cascaded subgroups discovery with an application to regression. In *Proceedings of the 19th European Conference on Machine Learning and 12th European Symposium on Principles and Practice of Knowledge Discovery in Databases*, 2008.

[8] A. Feelders, A. Knobbe, and D. Leman. Exceptional model mining. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, ECML PKDD '08, pages 1–16, Berlin, Heidelberg, 2008. Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-87481-2_1.

[9] W. Duivesteijn, A. Knobbe, A. Feelders, and M. Leeuwen. Subgroup discovery meets bayesian networks – an exceptional model mining approach. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 158–167, Washington, DC, USA, 2010. IEEE Computer Society.

[10] M. Leeuwen. Maximal exceptions with minimal descriptions. *Data Min. Knowl. Discov.*, 21:259–276, September 2010. http://dx.doi.org/10.1007/s10618-010-0187-5.

[11] S. Wrobel. An algorithm for multi-relational discovery of subgroups. pages 78–87. Springer, 1997.

[12] D. Gamberger and N. Lavrac. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527, 2002.

[13] A. Knobbe. Exceptional model mining, 2008. http://doi.acm.org/10.1145/502512.502569.

[14] W. A. Nicewander and J. L. Rodgers. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42:59–66, 1988.

[15] R.A. Fisher. On the 'probable error' of a coefficient of correlation deduced from a small sample. *Metron*, 1:3–32, 1921. http://digital.library.adelaide.edu.au/dspace/bitstream/2440/15169/1/14.pdf.

[16] P. Flach. Personal communication, 2011.

[17] L. Tenenboim, L. Rokach, and B. Shapira. Multi-label classification by analyzing labels dependencies. In *Proceedings of the 1st International Workshop on Learning from Multi-Label Data*, ECML/PKDD MLD '09, pages 117–131, Bled, Slovenia, 2009.

[18] Thomas D. Wickens. *Multiway contingency tables analysis for the social sciences*. Lawrence Erlbaum Associates, 1989.

[19] Lan Umek and Blaz Zupan. Subgroup discovery in data sets with multi-dimensional responses. *Intell. Data Anal.*, 15:533–549, December 2011.

[20] I. Katakis and G. Tsoumakas. Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13, 2007.

[21] P. Flach and N. Lachiche. Confirmation-guided discovery of first-order rules with tertius. *Mach. Learn.*, 42:61–95, January 2001. http://portal.acm.org/citation.cfm?id=599609.599628.

[22] NIST/SEMATECH. *Chi-Square Distribution*. 2006. http://www.itl.nist.gov/div898/handbook/.

[23] Tijl De Bie. Patterns in vectors. *Chapter in Pattern Analysis and Statistical Learning*. https://patterns.enm.bris.ac.uk/files/lecture8-2010.pdf.

[24] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 551–556, New York, NY, USA, 2004. ACM.

[25] R. Jowell and the Central Co-ordinating Team. European social survey2006/2007. *Technical Report*, 2007.

[26] A. Frank and A. Asuncion. Uci machine learning repository, 2010. http://archive.ics.uci.edu/ml.

[27] G. Tsoumakas and E. S Vilcek, J.and Xioufis. Mulan:a java library for multi-label learning, 2009. http://mulan.sourceforge.net/datasets.html.

[28] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.

[29] Ronald L. Iman and James M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics - Theory and Methods*, 9(6):571–595, 1980. http://www.tandfonline.com/doi/abs/10.1080/03610928008827904.

[30] Inc StatSoft. *Electronic Statistics Textbook*. StatSoft., Tulsa, OK, 2010. http://www.statsoft.com/textbook/.

[31] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):pp. 52–64, 1961. http://www.jstor.org/stable/2282330.

6 Appendix: Original Code

Listing 1: Kernel K Means

```
/**
 * Kernel K means
 * K is the number of clusters
 */
//Declare a vector to store the K clusters
Vector Centers=new Vector();
//Firstly, initialize the K Centers
//get K random sites
int[] RandomSite = new int[KClusters];
int count = flg = 0;
while (count < KClusters){
  int rdm = (int)(Math.random() * data.numInstances());
  //check if the random integer has been saved in RandomSite
  for (int i = 0; i < count; i ++){
        if (RandomSite[i] != rdm){
          flg = 0;
          continue;
        }
        else{
          flg = 1;
          break;
        }
  }
  //If the random value is not stored
  if (flg == 0){
    RandomSite[count] = rdm;
        count ++;
  }
}
//Initialize the centers (The randoms site is 1, otherwise 0)
for (int i = 0; i < KClusters; i ++){
  double []OneCenter = new double[data.numInstances()];
  for (int j = 0; j < OneCenter.length; j ++){
    if (j == RandomSite[i])
          OneCenter[j] = 1;
        else
          OneCenter[j] = 0;
  Centers.addElement(OneCenter);
}
//Initialize the target matrix
//a column is an instance
double[][] TargetMatrix = new double[QCTargetINDEX.length][data.numInstances()];
for (int i = 0; i < QCTargetINDEX.length; i ++){
  for (int j = 0; j < data.numInstances(); j ++)        {
    TargetMatrix[i][j] = data.instance(j).value(QCTargetINDEX[i]);
  }
}
//Initialize the kernel matrix of the target matrix
int RowNum = TargetMatrix.length;
int ColNum = TargetMatrix[0].length;
double[][] KernelMatrix = new double[ColNum][ColNum];
for (int i = 0; i < ColNum; i ++){
  for (int j = 0; j < ColNum; j ++){
        double[] Vector1 = new double[RowNum];
        double[] Vector2 = new double[RowNum];
        for (int VectorIndex = 0; VectorIndex < RowNum; VectorIndex ++){
          Vector1[VectorIndex] = TargetMatrix[VectorIndex][i];
          Vector2[VectorIndex] = TargetMatrix[VectorIndex][j];
        }
        KernelMatrix[i][j] = KernelFunction(FunctionName, Vector1, Vector2, Param);
  }
}
for (int i = 0; i < ColNum; i ++)
  InsCluster.put(i,MinimalIndex (Centers, TargetMatrix, KernelMatrix, i, FunctionName, Param)
        );
int CountChangedCenters = 1;
//Continue execute, until the centers not changed
while (CountChangedCenters != 0){
  Centers.removeAllElements();
  CountChangedCenters = 0;
  double[][] CenterArrays = new double[KClusters][data.numInstances()];
  for (int i = 0; i < CenterArrays.length; i ++)
        for (int j = 0; j < CenterArrays[0].length; j ++)
          CenterArrays[i][j] = 0;
  for (int i = 0; i < ColNum; i ++)
        CenterArrays[InsCluster.get(i)][i] = 1;
  for (int i = 0; i < KClusters; i ++){
        int countOnes = 0;
    for (int j = 0; j < data.numInstances(); j ++)
          countOnes += CenterArrays[i][j];
    for (int j = 0; j < data.numInstances(); j ++)
          CenterArrays[i][j] = (double)CenterArrays[i][j] / countOnes;
    Centers.addElement(CenterArrays[i]);
  }
  for (int i = 0; i < ColNum; i ++){
    int CurrentMinimalIndex = MinimalIndex (Centers, TargetMatrix, KernelMatrix, i,
          FunctionName, Param);
          if (CurrentMinimalIndex != InsCluster.get(i)){
                InsCluster.put(i, CurrentMinimalIndex);
                CountChangedCenters ++;
          }
  }
}
```

Listing 2: Distance function of Kernel K Means

```
/**
 *  k(i) = arg minl xi*xi + alpha * X * X *alpha - 2 * alpha * X * xi
 * @param Centers
 * @param TargetMatrix
 * @param KernelMatrix
 * @param InstancesIndex
 * @return
 */
```

```
public static int MinimalIndex (Vector Centers, double[][] TargetMatrix,
double[][] KernelMatrix, int InstancesIndex, String FunctionName, double Param){
  int MinCenterIndex = 0;
  double MinCenterDistance = java.lang.Double.MAX_VALUE;
  double []TheIns = new double[TargetMatrix.length];
  for (int i = 0; i < TheIns.length; i ++)
    TheIns[i] = TargetMatrix[i][InstancesIndex];
  double Kxx = KernelFunction(FunctionName, TheIns, TheIns, Param);
  double []AL = new double[TargetMatrix[0].length];
  for (int i = 0; i < Centers.size(); i ++){
        AL = (double[])Centers.elementAt(i);
        double []ALK = new double[KernelMatrix[0].length];
        ALK = MatrixMultiply(AL, KernelMatrix);
        double ALKAL = VectorMultiply(ALK, AL);
        double ALJKXX = 0.0;
        for (int j = 0; j < AL.length; j ++){
          double []XJ = new double[TheIns.length];
          for (int count = 0; count < XJ.length; count ++)
                XJ[count] = TargetMatrix[count][j];
          ALJKXX += AL[j] * KernelFunction(FunctionName, TheIns, XJ, Param);
        }
        double TotalScore = Kxx + ALKAL - 2 * ALJKXX;
        if (TotalScore < MinCenterDistance){
          MinCenterDistance = TotalScore;
          MinCenterIndex = i;
        }
  }
  return MinCenterIndex;
}
```

Listing 3: Kernel Functions

```
/**
 * Kernel function
 * @param FunctionName: polynomia, linear, RBF
 * @param X1: the first array
 * @param X2: the second vector
 * @param Param: parameter for different kind of kernel function
 * @return
 */
public static double KernelFunction(String FunctionName,
                                    double[] X1, double[] X2, double Param){
  if (FunctionName.matches("polynomia") ||
       FunctionName.matches("POLYNOMIA")){
    double Norm_X1_X2 = 0.0;
    for (int i = 0; i < X1.length; i ++)
          Norm_X1_X2 += X1[i] * X2[i];
    return Math.pow((Norm_X1_X2 + 1), Param);
  }
  else if (FunctionName.matches("linear") ||
            FunctionName.matches("LINEAR")){
        double Norm_X1_X2 = 0.0;
        for (int i = 0; i < X1.length; i ++)
          Norm_X1_X2 += X1[i] * X2[i];
        return Norm_X1_X2;
  }
  else{
    double Norm_X1_X2 = 0.0;
    for (int i = 0; i < X1.length; i ++)
          Norm_X1_X2 += Math.pow((X1[i] -X2[i]), 2);
        return Math.exp(-(Norm_X1_X2/(2 * Math.pow(Param, 2))));
  }
}
```

Listing 4: The AP Method

```
/**
 * The method of CHi_LPBR detects targets' dependencies and correspondingly combine
 * part of the targets
 * @param data: The original Data
 * @return A new data with targets' combination
 * @throws Exception
 */
public Instances CHI_LPBR(Instances data) throws Exception{
  int[] OldTargetIndex = new int [TargetIndexes.length];
  int[] OldNormalIndex = new int [data.numAttributes() - OldTargetIndex.length];
  int[] RecordTargetIndex = new int [TargetIndexes.length];
  int[] RecordNormalIndex = new int [data.numAttributes() - OldTargetIndex.length];
  //Record the original defined targets
  System.arraycopy(TargetIndexes, 0, OldTargetIndex, 0, TargetIndexes.length);
  int countNormal = 0;
  for (int i = 0; i < data.numAttributes(); i ++){
        if (Arrays.binarySearch(OldTargetIndex, i) < 0){
      OldNormalIndex[countNormal] = i;
          countNormal ++;
        }
  }
  System.arraycopy(OldTargetIndex, 0, RecordTargetIndex, 0, OldTargetIndex.length);
  System.arraycopy(OldNormalIndex, 0, RecordNormalIndex, 0, OldNormalIndex.length);
  double oldEval = newEval = 0.0;
  //Get the initialized WRAcc
  CN2rule currentRule = new CN2rule(0,OldTargetIndex,signifTest, maxStar,
                        weightIng, mInEquality,verbose, 0,this.mClassEval);
  Instances OldData = currentRule.getData(data, data, OldTargetIndex);
  currentRule.buildRule(OldData,OldData);
  oldEval = currentRule.m_bestEval;
  newEval = oldEval * 2;
  //Stop execution, until the WRAcc is not increased
  while (newEval > oldEval && OldTargetIndex.length > 1){
        int MaxI = MaxJ = 0;
        double MaxInd = 0.0;
    //Calculate the dependency between each pair of targets
        for (int i = 0; i < OldTargetIndex.length; i ++){
          for (int j = i + 1; j < OldTargetIndex.length; j ++){
                double[] TargetIDouble = OldData.attributeToDoubleArray(OldTargetIndex[i]);
                double[] TargetJDouble = OldData.attributeToDoubleArray(OldTargetIndex[j]);
                int[] TargetIInteger = new int[TargetIDouble.length];
                int[] TargetJInteger = new int[TargetJDouble.length];
                for (int k = 0; k < TargetIInteger.length; k ++){
                  TargetIInteger[k] = (int)TargetIDouble[k];
```

```
                TargetJInteger[k] = (int)TargetJDouble[k];
              }
              double tempInd = CN2rule.PWPearsonGoodness(TargetIInteger, TargetJInteger);
              if (tempInd > MaxInd){
                MaxI = i;
                MaxJ = j;
            MaxInd = tempInd;
              }
        }
      }
      //Get a data with labels combined by labels of i and j
      Instances NewData = new Instances(OldData);
      HashMap<String,Integer> CombinedValue =new HashMap<String,Integer>();
      for (int i = 0; i < OldData.numInstances(); i ++){
        CombinedValue.put(OldData.instance(i).toString(OldTargetIndex[MaxI]) + "-" +
        OldData.instance(i).toString(OldTargetIndex[MaxJ]),1);
      }
      FastVector newAtt = new FastVector();
      Set set=CombinedValue.keySet();
      Iterator itr=set.iterator();
      while(itr.hasNext()){
        newAtt.addElement((String)itr.next());
      }
      CombinedValue.clear();
      NewData.insertAttributeAt(
      new Attribute(OldData.attribute(
    OldTargetIndex[MaxI]).name()+ "-" +
    OldData.attribute(OldTargetIndex[MaxJ]).name(), newAtt),
    OldTargetIndex[MaxI]);
      if (OldTargetIndex[MaxI] == OldData.classIndex() ||
      OldTargetIndex[MaxJ] == OldData.classIndex())
        NewData.setClassIndex(OldTargetIndex[MaxI]);
      for (int i = 0; i < NewData.numInstances(); i ++){
        NewData.instance(i).setValue(OldTargetIndex[MaxI],
          OldData.instance(i).toString(OldTargetIndex[MaxI]) + "-" +
                OldData.instance(i).toString(OldTargetIndex[MaxJ]));
      }
      NewData.deleteAttributeAt(OldTargetIndex[MaxJ] + 1);
      NewData.deleteAttributeAt(OldTargetIndex[MaxI] + 1);
      int[] NewTargetIndex = new int [OldTargetIndex.length - 1];
      int[] NewNormalIndex = new int [OldNormalIndex.length];
      for (int i = 0; i < OldTargetIndex.length; i ++)
        if (OldTargetIndex[i] != OldTargetIndex[MaxJ])
              if (OldTargetIndex[i] < OldTargetIndex[MaxJ])
                            NewTargetIndex[i] = OldTargetIndex[i];
              else
                NewTargetIndex[i - 1] = OldTargetIndex[i] - 1;
      for (int i = 0; i < OldNormalIndex.length; i ++)
        if (OldNormalIndex[i] < OldTargetIndex[MaxJ])
          NewNormalIndex[i] = OldNormalIndex[i];
        else
              NewNormalIndex[i] = OldNormalIndex[i] - 1;
      currentRule = new CN2rule
  (0,NewTargetIndex,signifTest, maxStar, weightIng,
                mInEquality,verbose, 0,this.mClassEval);
      currentRule.buildRule(NewData, NewData);
        newEval = currentRule.m_bestEval;
    //If the data's WRAcc is increased
        if (newEval > oldEval){
          oldEval = newEval;
          newEval = oldEval + 1;
          OldData = new Instances(NewData);
          OldTargetIndex = NewTargetIndex;
          OldNormalIndex = NewNormalIndex;
          TargetIndexes = NewTargetIndex;
          NormalIndexes = NewNormalIndex;
        }
  }
  QCAMethod = 0;
  for (int i = 0; i < NormalIndexes.length; i ++)
          AttIndex.put(NormalIndexes[i], RecordNormalIndex[i]);
  return OldData;
}
```

Listing 5: The Function of Heuristics

```
/**
 * The function for evaluating rules on multi-targets
 * @param QCAMETHOD         The adaption method
 * @param QCClassEVAL       The evaluating method
 * @param QCBestRULE        The rule need to be evaluated
 * @param QCTargetINDEXES   The target Indexes
 * @param def               Directly forward to evaluateSD
 * @param b                 Directly forward to evaluateSD
 * @return
 * @throws Exception
 */
public double QCevaluateSD(Instances QCData, int QCAMETHOD, int QCClassEVAL,
FastVector QCBestRULE, int[] QCTargetINDEXES, boolean def, boolean b) throws Exception{
  double SumCount = 0.0;
  //If the method of CW is chosen
  if (QCAMETHOD == QCA_Clu)
    SumCount = evaluateSDCluster(QCData, QCClassEVAL, QCBestRULE, def, b);
  //If the method of PBW is chosen
  else if (QCAMETHOD == QCA_Pro)
    SumCount = evaluateSD(QCData, QCClassEVAL, QCBestRULE, QCTargetINDEXES[0], def, b);
  //If the method of CM is chosen
  else if(QCAMETHOD == QCA_Cro){
    double Entropy = nCovered = nNotCovered = 0.0;
    Instances CoveredInstances = covered(QCBestRULE, QCData);
    for (int i = 0; i < ((double[])QCDistributionVector.elementAt(0)).length; i ++)
      nCovered += ((double[])QCDistributionVector.elementAt(0))[i];
      nNotCovered = QCData.sumOfWeights() - nCovered;
    if(nCovered > 0 && nNotCovered > 0){
      //Calculate the Entropy of the subgroup
      Entropy = -nCovered/(nCovered+nNotCovered)*
                    (Math.log(nCovered/(nCovered+nNotCovered))/Math.log(2))
                  -nNotCovered/(nCovered+nNotCovered)*
                    (Math.log(nNotCovered/(nCovered+nNotCovered))/Math.log(2));
      int countNAN = 0;
    for (int i = 0; i < QCTargetINDEXES.length; i ++){
```

```
        double[] TempLabel1 =
        CoveredInstances.attributeToDoubleArray(QCTargetINDEXES[i]);
        for (int j = i + 1; j < QCTargetINDEXES.length; j ++){
          double[] TempLabel2 =
        CoveredInstances.attributeToDoubleArray(QCTargetINDEXES[j]);
          double TempPearson = getPearsonCorrelation(TempLabel1,TempLabel2);
          if(java.lang.Double.isNaN(TempPearson))
                countNAN ++;
          else
                SumCount += Math.abs(PWiseCor.get(QCTargetINDEXES[i] + "-"
                                + QCTargetINDEXES[j]) - TempPearson);
      SumCount = SumCount/
                 ((QCTargetINDEXES.length*(QCTargetINDEXES.length-1))/2 - countNAN);
      if (java.lang.Double.isNaN(SumCount))
        SumCount = -100;
      else
        SumCount = SumCount * Entropy;
        else
          SumCount = -100;
    }
    //If the method of IM is chosen
    else if(QCAMETHOD == QCA_Ind){
      double Entropy = nCovered = nNotCovered = 0.0;
      Instances CoveredInstances = covered(QCBestRULE, QCData);
      for (int i = 0; i < ((double[])QCDistributionVector.elementAt(0)).length; i ++)
        nCovered += ((double[])QCDistributionVector.elementAt(0))[i];
        nNotCovered = QCData.sumOfWeights() - nCovered;
      if(nCovered > 0 && nNotCovered > 0){
        //Calculate the Entropy of the subgroup
        Entropy = -nCovered/(nCovered+nNotCovered)*
                        (Math.log(nCovered/(nCovered+nNotCovered))/Math.log(2))
                       -nNotCovered/(nCovered+nNotCovered)*
                        (Math.log(nNotCovered/(nCovered+nNotCovered))/Math.log(2));
            double PersonTest = Math.sqrt
                          (getPearsonGoodness(CoveredInstances, QCTargetINDEXES));
        double FreeDegree = 1.0;
        double FreeParameter = 0.0;
        for (int i = 0; i < QCTargetINDEXES.length; i ++){
          FreeDegree *= m_data.attribute(QCTargetINDEXES[i]).numValues();
          FreeParameter += m_data.attribute(QCTargetINDEXES[i]).numValues() - 1;
        }
        FreeDegree = FreeDegree -1 - FreeParameter;
        double TheIndependency = ProDF (PersonTest, FreeDegree);
        if (java.lang.Double.isNaN(TheIndependency))
          SumCount = -100;
        else
          SumCount = Math.abs(TheIndependency - PIndependency)*Entropy
          }
          else
            SumCount = -100;
    }
    //If the method ABW is chosen
    else{
      if (QCTargetINDEXES.length == 1)
        SumCount = evaluateSD(QCData, QCClassEVAL, QCBestRULE, QCTargetINDEXES[0], def, b);
      else{
        for (int count = 0; count < QCTargetINDEXES.length; count ++){
              double EvalSD =
               evaluateSD(QCData, QCClassEVAL, QCBestRULE, QCTargetINDEXES[count], def, b);
              if ((int)EvalSD == -100){
                SumCount = -100.0;
                    break;
              }
              else
                SumCount += Math.abs(EvalSD);
        }
            if (SumCount != -100)
          SumCount = SumCount / QCTargetINDEXES.length;
      }
    }
    return SumCount;
  }
  \lstset{language=Java, caption=The Computation of PDF, label=PDF}
  \lstset{breaklines}
  \lstset{extendedchars=false}
  \begin{lstlisting}
  public static double PDF(Instances data, int[] QCTargetINDEXES){
    double result = 0.0;
    //record the marginal values  <Ai-Vi>
    HashMap<String,Double> MarginalValues =new HashMap<String,Double>();
    //record the expected frequencies <AiVi-AiVi-AiVi>
    HashMap<String,Double> EFrequency =new HashMap<String,Double>();
    //record the observed frequencies <AiVi-AiVi-AiVi>
    HashMap<String,Double> OFrequency =new HashMap<String,Double>();
    //record the attribute value
    Vector AttValues = new Vector();
    //Marginal Table
    for (int i = 0; i < QCTargetINDEXES.length; i ++){
      for (int j = 0; j < data.numInstances(); j ++){
        String tempValue = QCTargetINDEXES[i] + "-" +
                           data.instance(j).toString(QCTargetINDEXES[i]);
          if (MarginalValues.containsKey(tempValue))
            MarginalValues.put(tempValue, MarginalValues.get(tempValue) + 1.0);
          else
            MarginalValues.put(tempValue, 1.0);
      }
    }
    for (int i = 0; i < data.numInstances(); i ++){
      double ExpectCell = 1.0;
      String tempAttVal = "";
      for (int j = 0; j < QCTargetINDEXES.length; j ++){
        tempAttVal += data.instance(i).toString(QCTargetINDEXES[j]) + "-";
        ExpectCell *= MarginalValues.get(QCTargetINDEXES[j] + "-" +
                      data.instance(i).toString(QCTargetINDEXES[j]));
      }
      if(!EFrequency.containsKey(tempAttVal))
        EFrequency.put(tempAttVal, ExpectCell/Math.pow(data.numInstances(),
                                     QCTargetINDEXES.length - 1));
      if(OFrequency.containsKey(tempAttVal))
        OFrequency.put(tempAttVal,OFrequency.get(tempAttVal)+1.0);
      else
        OFrequency.put(tempAttVal,1.0);
```

```
    }
    double PersonTest = 0.0;
    Set set=EFrequency.keySet();
    Iterator itr=set.iterator();
    while(itr.hasNext()){
          String tempKey = (String)itr.next();
          PersonTest += Math.pow((EFrequency.get(tempKey) - OFrequency.get(tempKey)), 2)
                    / EFrequency.get(tempKey);
    }
    MarginalValues.clear();
    EFrequency.clear();
    OFrequency.clear();
    AttValues.clear();
    double FreeDegree = 1.0;
    double FreeParameter = 0.0;
    for (int i = data.length - 1; i >= 0; i --){
      FreeDegree *= data[i].attribute(data[i].numAttributes()-1).numValues();
          FreeParameter += data[i].attribute(data[i].numAttributes()-1).numValues() - 1;
    }
    FreeDegree = FreeDegree -1 - FreeParameter;
    System.out.println(getGamma(FreeDegree/2));
    result=(1/(Math.pow(2, FreeDegree/2)*getGamma(0.5)))*
     (Math.pow(PersonTest, FreeDegree/2-1)*Math.exp(-PersonTest/2));*/
    return PersonTest;
  }
```


Listing 6: Data Recreation

```
/**
 * the function used to change the data style
 * @param data
 * @param totalData
 * @param TargetIndexes
 * @return
 */
public Instances getData(Instances data, Instances totalData, int [] TargetIndexes){
  //If the method of ABW is chosen, then combine the data targets
  if(QCAMethod == QCA_Pro){
    Instances NewData = new Instances(data);
        if (TargetIndexes.length > 1){
      FastVector newAtt = (FastVector)CombAtt.copy();
          NewData.insertAttributeAt(
                new Attribute("combinedAtt", newAtt), NewData.numAttributes());
          for (int count = 0; count < NewData.numInstances(); count ++){
            String temText = "";
            for (int count1 = 0; count1 < TargetIndexes.length; count1 ++){
          if (((totalData.attribute(TargetIndexes[count1]).isNominal())||
             (totalData.attribute(TargetIndexes[count1]).isString())))
                    temText = temText +
              (int)totalData.instance(count).value(TargetIndexes[count1]) +"-";
          else{
            int newDiscritize =
                 discritize(count, TargetIndexes[count1], totalData, totalData);
                    temText = temText + newDiscritize +"-";
              }
            }
            NewData.instance(count).setValue(NewData.numAttributes()-1, temText);
          }
          NewData.setClassIndex(NewData.numAttributes() - 1);
          for (int i = TargetIndexes.length - 1; i >= 0; i --)
            NewData.deleteAttributeAt(TargetIndexes[i]);
        }
        return NewData;
  }
  //If other methods are chosen, the data targets are discretized
  else{
        Instances myData = new Instances(data);
        for (int i = 0; i < TargetIndexes.length; i ++){
          if (!((myData.attribute(TargetIndexes[i]).isNominal())||
         (myData.attribute(TargetIndexes[i]).isString()))){
                //create a new nominal attribute corresponding to the current attribute
                //and append it at the end of the dataset
                FastVector newAtt = new FastVector();
                for (int count2 = 0; count2 < 5; count2 ++)
                  newAtt.addElement(Integer.toString(count2));
                myData.insertAttributeAt(
                new Attribute(myData.attribute(TargetIndexes[i]).name(), newAtt),
                                          TargetIndexes[i]);
        for (int count2 = 0; count2 < data.numInstances(); count2 ++){
          int newDiscritize = discritize(count2, TargetIndexes[i], data, totalData);
          myData.instance(count2).setValue(TargetIndexes[i], Integer.toString(newDiscritize))
                ;
        }
        myData.deleteAttributeAt(TargetIndexes[i] + 1);
          }
        }
        return myData;
  }
}
```